Hindawi
Journal of Electrical and Computer Engineering
Volume 2022, Article ID 5889948, 12 pages
https://doi.org/10.1155/2022/5889948

Research Article

An Efficient Policy-Based Scheduling and Allocation of Virtual Machines in Cloud Computing Environment

S. Supreeth,[1,2] **Kirankumari Patil**,[1,2] **Shantala Devi Patil**,[2] **S. Rohith**,[3] **Y. Vishwanath**,[2] **and K. S. Venkatesh Prasad**[2]

[1]*CSE, REVA ITM, VTU Research Center, Bengaluru 560064, India*
[2]*School of CSE, REVA University, Bengaluru 560064, India*
[3]*Dept. of ECE, Nagarjuna College of Engineering and Technology, Bengaluru 562164, India*

Correspondence should be addressed to S. Supreeth; supreeth1588@gmail.com

Received 2 May 2022; Revised 11 August 2022; Accepted 29 August 2022; Published 24 September 2022

Academic Editor: Nicola Pasquino

Cloud computing has become the most challenging research field in the current information technology scenario. In this, a set of user tasks are scheduled and allocated to numerous kinds of heterogeneous virtual machines (VMs) in cloud data centers (CDCs), and these VMs are hosted by diverse types of heterogeneous physical machines (PMs). It extends several features, encompassing elasticity, safety, sustainability, and even adequate maintenance compared to traditional data centers. There are numerous techniques available for VM scheduling and allocation. However, it still requires the existence of new mechanisms that can reduce the execution time (ET) of the tasks, improve the optimization of energy usage and resource utilization (RU), and reduce time consumption. Along with optimization, VM scheduling (VMS) and VM allocation (VMA) are two-level issues that need to be considered with essential policies to govern these mechanisms. Hence, for executing optimal VMS and VMA in the data center, new optimization methodologies, such as enhanced shark smell optimization algorithm (ESSOA) at the first level and Brownian movement-centered gravitation search algorithm (BMGSA) at the second level, are proposed in this work to define the policies. Firstly, the user requests for VMs are reserved on the most appropriate PM by the proposed ESSOA, which has the lowest execution cost within deadline limits, and the proposed BMGSA decides the allocation of the chosen VM on the most appropriate PM within the resource limitations at the second level. To demonstrate the proposed algorithm's efficiency, the simulations are carried out using the Java language-based CloudSim simulator, and the proposed mechanism outcomes acquired are compared with the existing approaches. The simulation results show that the suggested algorithm is efficient in terms of the execution cost, degree of imbalance (DOI), make span (MS), and resource utilization (RU).

1. Introduction

Cloud computing [1] is the most favored option amongst users in this era to access cloud services [2–4] from any place and at any time via the internet. Cloud providers maintain these configurable computing resources, and they can be quickly provisioned and released [5]. RCloud computing is centered on a significant concept encompassing abstraction and the notion of pooling physical resources using virtualization methodology [6]. Cloud providers use virtualization technology that creates diverse kinds of cloud services on the internet, creating the potential feasibility for cloud computing. The infrastructure is given as a service, like the Amazon EC2. The runtime environment, like Google App Engine, is termed as a platform and a service. Lastly, the software is provided as a service, e.g., salesforce.com [7]. For all these services to be carried out efficiently, resources must be provisioned. There are 2 types of service provisioning that may be initiated in a cloud data center, which are as follows: (i) task scheduling (TS) and (ii) VM scheduling. In task scheduling, each task is assigned to the cloud without considering the optimization of resources at the data center, and vice-versa is carried out in VM scheduling. Figure 1 exhibits the VM scheduling architecture [8]. Every scheduler

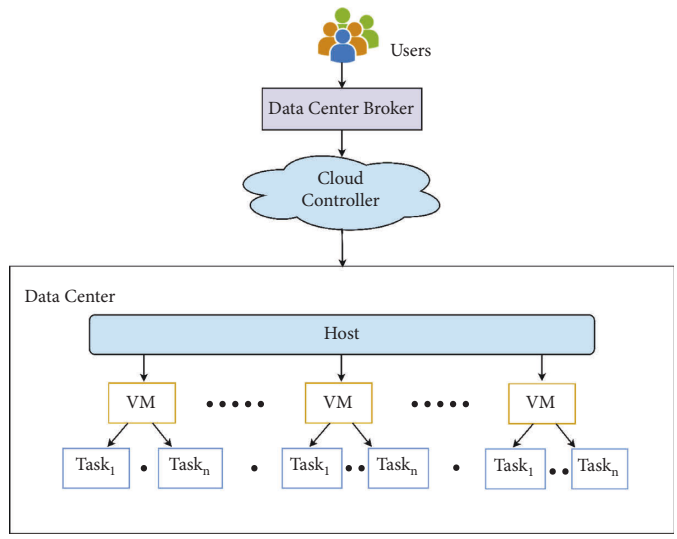

Figure 1: The architecture of the VM scheduling process.

could have features for scheduling the resources. It may be homogeneous or heterogeneous requests [9]. Hence, mapping and scheduling tasks on appropriate VMs is crucial [10]. If cloud computing accomplishes a lesser makespan, lesser execution time of tasks, and effective utilization of resources, then the mapping between the tasks and needed resources is accomplished productively. Additionally, the task must be executed quickly, and a response must be transmitted to the user [11].

Scheduling is carried out at 2 levels, which are as follows: in the 1st level, jobs produced by users are given to appropriate task schedulers, and these jobs are assigned to the VMs using cloud resources [12]. Every job/task contains numerous dependent tasks, which form directed acyclic graph (DAG) [13]. VM scheduling is the 2nd level of tasks carried out in the datacenter. The VMs are mapped to appropriate PMs, which are competent in offering the necessary resources (i.e., memory, disk space, and processor) using VM allocation [14]. A few of the PMs could be shut down if the VMs were classified as per the utilization, along with a few selectively migrated, which would decrease the cloud platforms' whole energy consumption (EC) [15, 16]. Therefore, the effective management of VMs is of greater challenge in cloud data centers and can direct the cloud service provider to satisfy their organization's objectives [17]. Through VMs, VMS and VMP are linked. Hence, these issues are mainly coupled with one another. They are identified as NP-hard optimization issues [18, 19].

Typically, it is complex to design the algorithms in cloud computing to yield optimal solutions for VM scheduling [20] and allocation. In discovering a single solution, multiobjective optimization has a massive scope of interest amongst the investigators regarding different aspects of an issue for VMS and VMA [21]. However, the algorithms are created to handle both issues separately. However, for producing an efficient solution for cloud users and providers, the issues must be managed together and integrated. To efficiently carry out VM scheduling and allocation for the cloud platform, this research adopted new optimization approaches, such as ESSOA and BMGSA.

The rest of the paper is arranged as follows: the associated literary works are exhibited in Section 2. The proposed research technique is illustrated in Section 3. The outcomes and discussion part are exhibited in the 4th section. Finally, the conclusion is given in the 5th section.

2. Literature Review

Many research scholars worked on cloud data center technology to improvise the QoS parameters for the effective utilization of the resources. However, the improvisation can be done by optimizing the resources at the virtualization layer. The latest works created in the cloud platform for VM scheduling and allocation are reviewed in this section.

Abualigah and Diabat [22] propounded a hybrid ant lion optimization (ALO) approach by leading centered differential evolution to solve multiobjective task scheduling difficulties in cloud platforms described as modified ALO (MALO). The system originated from the requirement to concurrently reduce makespan while increasing resource utilization. Utilizing the CloudSim tool kit, 2 trial series were performed on real trace datasets. The outcomes showed that the proposed MALO surpassed other renowned optimization algorithms.

Fu et al. [23] put forward a particle swarm optimization (PSO) genetic hybrid system centered on phagocytosis (PSO_PGA) to execute task scheduling in the cloud platform. Firstly, the entire population was divided into numerous subpopulations in PSO_PGA. Then, utilizing the phagocytosis approach and crossover, as well as a mutation in the genetic algorithm (GA), the particle's position in every subpopulation was updated to expand the model's search range. Finally, all subpopulations were fused to escape from local optima, which guaranteed diversity as well. At last, it was ensured that the particle population could constantly

push in the way of an excellent solution. The outcomes exhibited that the algorithm improved the cloud task's entire completion time (CT) and had the highest convergence accuracy compared to the prevailing methods. Furthermore, the findings exhibited that the proposed algorithm improved the cloud tasks' overall CTs through simulation experiments and had higher convergence accuracy than existing techniques.

Arul Xavier and Annadurai [24] suggested a swarm intelligence of societal spiders with chaotic inertia weight-centered random choice on implementing VM scheduling in the cloud environment that focused on lowering the overall makespan. In discovering the finest optimized VM for the user task amongst the VMs with the lowest makespan and balanced resource utilization, the algorithm prevented the local convergence and investigated the intelligent global searching. The outcomes significantly improved the makespan with the balanced VM distribution while utilizing the presented model.

Lu et al. [25] recommended an improved GA (I-GA) to resolve the VM allocation issue. To merge with GA, the model offered virtual hierarchy architecture. By establishing the I-GA's preliminary population generation step, the model accomplished a near-optimal solution in solving accessibility and energy consumption concerns. The outcomes have shown the considerable enrichment of the datacenter's energy efficiency and the successful preservation of its higher availability.

Tripathi et al. [26] employed an improved dragonfly method for VM allocation for predominant resource utilization in a cloud environment. In an easy dragonfly algorithm to suit it well and intended for the VM allocation issue, some alterations, like the employment of time, were made. In addition, a *V*-shaped transfer function, a solution generator function, and so on were integrated. Nevertheless, the model surpassed other prevailing methodologies because of the more extensive coordination and the competency to handle a principal balance between exploration and exploitation.

The following are the few research gaps identified based on the previous literary works.

(i) The scheduling strategies mentioned above have relied on the connections between requests and VMs. However, they do not consider the actual state of the physical nodes, which might cause the outcomes to fall short. Therefore, it is another additional consideration in scheduling virtual machines on physical nodes.

(ii) That is to say, effective VM scheduling is to be provided based on the real workloads running on them, however, most literary works did not consider the real workloads.

(iii) These studies did not adequately account for the effective balance of supply and demand between users and cloud providers since they either focused on load balancing, migration costs over specific time frames, or energy efficiency. On the other hand, most prevailing scheduling algorithms enrich the execution time and resource utilization while implementing VM scheduling.

(iv) To deal with and provide efficient IaaS services in the cloud, the prevailing works carry out VM scheduling and VM allocation independently. Hence, in a cloud platform, it is indispensable to construct an optimal plan for VM scheduling and VM allocation that plays a crucial part in enhancing resource utilization [27].

(v) Most traditional approaches considered single objectives to implement to cope with VM scheduling and allocation in the cloud environment.

To address the above-mentioned issues, new multi-objective optimization algorithms are to be developed by considering load balance, resource utilization, effective time management, and SLA violations in this research.

3. Proposed Methodology

To implement VM scheduling and allocation as a co-optimization procedure, a new optimization algorithm, such as ESSOA and BMGSA, is proposed in this paper. At first, utilizing an ESSOA execution cost is considered the critical objective of the VM scheduling. The tasks of the users are assigned to an appropriate efficient VM by considering energy, meantime, SLA Violation and the optimal chosen VMs are positioned to the suitable PMs by considering the suitable policies. Figure 2 shows the proposed research model's architecture. The fundamental definitions for the proposed model are as follows: the set of user tasks to be assigned in VMs is represented in the following:

$$T_i = T_1, T_2, \ldots, T_Q, \tag{1}$$

where the total count of tasks is Q. Every task is explicated as T_i (Z_i, D_i, S_i), where (Z_i, D_i, and S_i) signify the task size gauged by millions of instructions (MI), task deadline, and start time of a task T_i. The set of VMs for scheduling is determined from the following:

$$VM_j = VM_1, VM_2, \ldots, VM_M, \tag{2}$$

where the total number of VMs is denoted as M. Every VM_j is explicated as $VM_j(ps_j, P_{vmj})$, in which $\mathrm{ps_j}$ is the VM processing capacity, articulated as million instructions per second (MIPS), which is put as M through $\sum_{j=1}^{M} ps_j \leq Cph_k$.The amount of payment used for utilizing a VM_j per hour is indicated by P_{vmj} . Cph_k denotes the capacity of the physical machine k. The set of PMs for VM scheduling is given as follows:

$$PM_j = PM_1, PM_2, \ldots, PM_N, \tag{3}$$

where the total number of PMs is signified by N. Each PM_k is described as $\mathrm{PM}_j(ps_k)$, where the processing capacity of the PM is symbolized as ps_k, which is stated regarding MIPS.

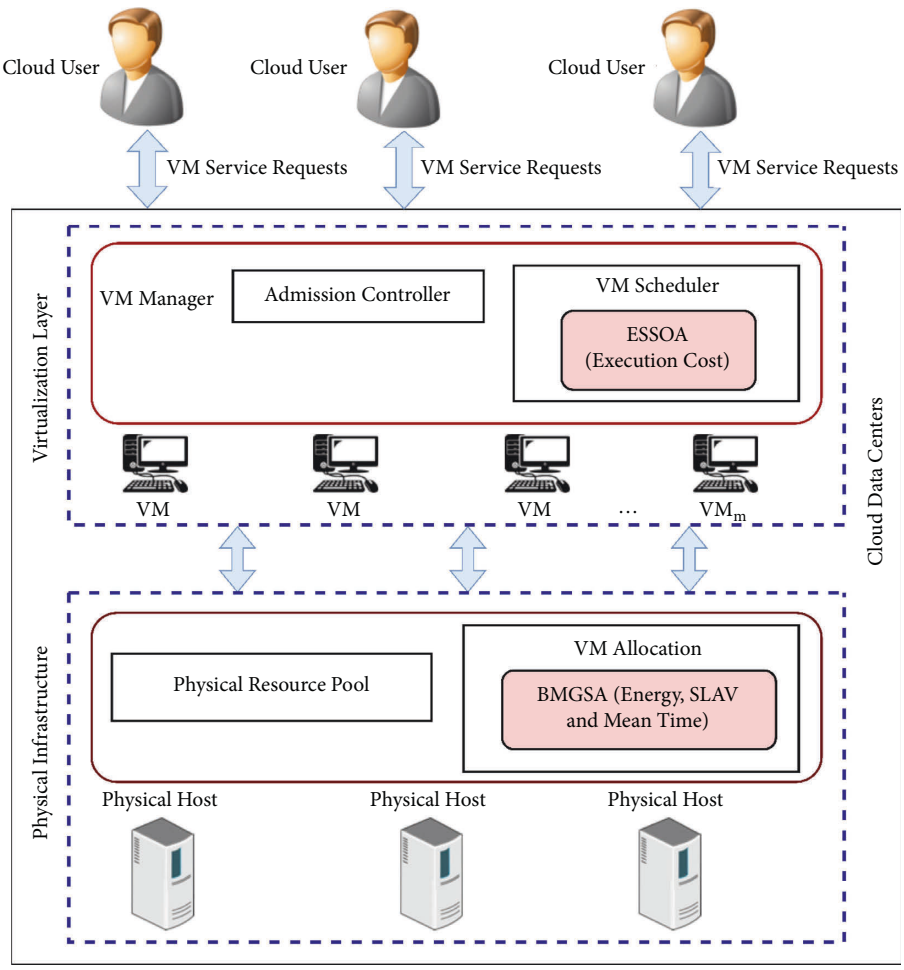

Figure 2: Proposed architecture.

3.1. Virtual Machine Scheduling. By employing the enhanced shark smell optimization algorithm (ESSOA), the VM scheduling of the proposed methodology is performed. A population-centered technique that commences with a random population is called SSOA. Imitating and simulating the shark's technique in discovering the target is the central notion of SSOA. The hunting methodology in sharks is centered on their smelling sense of competency. Accomplishing the global value of the optimization problem is the main aim of the work. In traditional SSOA, the local search carried out by an algorithm has drawbacks, such as the overflow of the search area and the disruption of random flights, because of its huge searching steps. A bidirectional search (BS) is integrated into SSOA, termed enhanced SSOA (ESSOA), which is proposed to overcome these disadvantages and enrich the local searching capability of SSOA for optimization problems. It aids in implementing the local search in the forward and backward directions. Whilst choosing the direction, greedy selection is made. If the solution ameliorates whilst traversing backwards, then backward traverse is adapted. This enhancement accelerates SSOA's convergence rate. Because of the VM's execution cost, the objective function of SSOA is calculated. The execution cost E_{ij} of T_i is outlined as the multiplication of the price of VM_j along with the CT of T_i, which is given by the following:

$$E_{ij} = P_{vmj} * \frac{C_{ij}}{3600}, \tag{4}$$

where the price of VM_j is symbolized by P_{vmj}, and the completion time of executing a task is T_i on VM_j is C_{ij}. C_{ij} can be executed if the completion time C_{ij} is within the deadline. Otherwise, C_{ij} cannot be executed. The time that VM consumes to carry out the task is called the execution time. Waiting time is the time difference between the start and end time of the VM. Therefore, the completion time that VM_j will take to perform VM_i can be computed as follows:

$$C_{ij} = S_i + ET_{ij}, \tag{5}$$

where the starting time of VM_i is S_I, and the time of carrying out T_i on VM_j at a provided time t is ET_{ij}. Centered on "2" metrics, namely the task size ST_i (number of instructions that T_i will require to implement on VM_j) and the processing speed of VMs ps_j, the ET_{ij} of the task is computed, which is calculated as follows:

$$ET_{ij} = \frac{ST_i}{ps_j}, \tag{6}$$

where,

$$ps_j = N_{Pj} * M_j, \tag{7}$$

where the number of processors in VM_j is depicted by Np_j, and M_j is the MIPS of each processor in VM_j. The constraints regarded for VMS in SSOA are mentioned below. A task should be allocated to a VM, i.e.,

$$\sum_{j=1}^{M} TV_{ij}^{i} = 1, \quad \forall i\varepsilon\{1, 2, \ldots, Q\}, \tag{8}$$

where i & j are the indexes of the task and VM, and TV_{ij}^{t} is a binary value signifying whether T_i it is designated VM_j at a given time t. It must guarantee that every task is completed before its deadline, i.e.,

$$C_{ij} \le D_i, \forall i\varepsilon\{1, 2, \ldots, Q\}, \tag{9}$$

where the deadline of task T_i is given by D_i, and C_{ij} is the CT of executing a task T_i on VM_j. The need for resources for all tasks hosted on VM should not go beyond the VM resource's utmost capacity, i.e.,

$$\begin{aligned} &\sum_{i-1}^{Q} TC_i * TV_{ij} \le VC_j, \quad \forall i\varepsilon\{1, 2, \ldots, M\}, \\ &\sum_{i-1}^{Q} TM_i * TV_{ij} \le VM_j, \quad \forall i\varepsilon\{1, 2, \ldots, M\}, \\ &\sum_{i-1}^{Q} TN_i * TV_{ij} \le VN_j, \quad \forall i\varepsilon\{1, 2, \ldots, M\}, \end{aligned} \tag{10}$$

where CPU, memory, and bandwidth demands T_i are TC_i, TM_i, and TN_i, respectively. CPU, memory, and bandwidth capacities VM_j are VC_j, VM_j, and VN_j, respectively. On any given available VM, every task is permitted to be processed, such that they fulfill the requirements of tasks, and every task must be finished without interruption once begun (non-preemptable). Thus, the task will stay in the queue until the primary task has finished its implementation if more than one task comes simultaneously.

VM_j and T_i are the input parameters of SSOA. Every individual of SSOA signifies a VM, and along with the computation of the VM is the execution cost. Therefore, the individual's dimension is equivalent to the number of VMs. A VM(VM_j) is randomly initialized when task i requires to be processed. SSOA encompasses a population of the size j of odor particles or VMs (O_{ps}), where each O_p has a dimension d. Position (p_j^g,$1 \le j \le M$) and velocity (v_j^g, $1 \le j \le M$) are the two components in each O_p. The term indicates the stage number $g(1 \le g \le g_{\max})$. The position vectors for the 1st stage of the population O_p are given in (11), and the components of the j^{th} position vector in each dimension are expressed in (12).

$$P_1^1, P_2^1, P_3^1, \ldots, P_M^1, \tag{11}$$

$$P_i = \left(P_{j,1}^1, P_{j,2}^1, P_{j,3}^1, \ldots, P_{j,M}^1\right), \quad 1 \le j \le M. \tag{12}$$

The velocity of the O_p corresponding to the position vector is given in (13), and the components of the j^{th} position vector in every dimension are symbolized in (14).

$$V_1^1, V_2^1, \ldots, V_M^1, \tag{13}$$

$$V_i = \left(V_{j,1}^1, V_{j,2}^1, \ldots, V_{j,M}^1\right), \quad 1 \le j \le M. \tag{14}$$

Each O_p is gauged by a fitness (F) function to judge the solution's quality to the issue. The fitness function for the VM scheduling process is calculated by the following:

$$F_i = P_{vmj} * C_{ij}, \tag{15}$$

where the price of VM_j is denoted by P_{vmj}, and the completion time that VM_j will take to carry out T_i is indicated by C_{ij}. From the initialized population, the individuals that have the least fitness values as the best individuals are chosen. An SSOA follows the forward and rotational movements to attain optimal global solutions. The search ensures in several stages (g), in which both movements are executed. The shark's velocity in the g^{th} stage is updated in forwarding movement utilizing the following:

$$\left|v_j^g\right| = \eta_g r_1 \nabla(F)|P_j^g + \lambda_g r_2 V_j^{g-1}, \tag{16}$$

where the gradient of the fitness function is depicted by $\nabla(F)$. A constant gradient value is indicated by η_g, and r_1 and r_2 are random numbers with uniform distribution [0, 1]. The inertia constant for the stage g is $\lambda_g \in (0, 1)g$, which restricts the shark's current velocity, such that it does not exceed a definite limit of its earlier velocity. The velocity's upper bound is articulated in the following equation:

$$\begin{aligned} \left|v_{j,h}^g\right| = \min\left(\left|\lambda_g \cdot r_1 \frac{\alpha(F)}{\alpha(p_h)}\right|_{P_{i,j}^g} + \left|\lambda_g \cdot r_2 \cdot v_{j,h}^{g-1}\right|, \left|\psi_g \cdot v_{j,h}^{g-1}\right|\right), \\ j = 1, \ldots, h = 1, \ldots, N_d, \; g = 1, \ldots, G, \end{aligned} \tag{17}$$

where the current stage velocity's upper bound is given by ψ_g. The number of positions, dimensions, and stages is denoted by the terms M, N_d, and G. Then, utilizing (18), the new position in the g^{th} stage is ascertained as follows:

$$R_j^{g+1} = p_j^g + v_j^g * \Delta t_g, \tag{18}$$

where the time duration for the g^{th} stage is signified by Δt_g, and for all stages, it is assumed to be 1. Then, the rotational movement is implemented as given below. Also, the local shark search can be devised as follows:

$$S_j^{g+1,u} = R_j^{g+1} + r_3 R_j^{g+1}, \tag{19}$$

where r_3 is the random number, and the number of points for a local search is denoted by $u = 1,....,U$. At this point, with

the aid of BS, the local search is carried out, which is computed as follows:

$$BS = \begin{cases} \text{if}\left[F_i\left(R_j^{g+1} + s_l\right) < F_i\left(R_j^{g+1}\right) \longrightarrow R_j^{g+1} = p_j^g + s_l\right], \\ \text{elseif}\left[F_i\left(R_j^{g+1} - s_l\right) < F_i\left(R_j^{g+1}\right) \longrightarrow R_j^{g+1} = p_j^g - s_l\right], \\ \text{otherwise} \longrightarrow \text{no change in the present position.} \end{cases} \quad (20)$$

The current position of sharks, objective fitness function value, and step length are symbolized by R_j^{g+1}, $F(R_j^{g+1})$, and s_l, respectively. The one with the lower cost is chosen to be the next destination among the points searched on a global and local scale. The final OP is updated as follows:

$$p_j^{g+1} = \arg\left(\min\left(F\left(R_j^{g+1}\right), F\left(S_j^{g+1,1}\right), \ldots, F\left(S_j^{g+1,U}\right)\right)\right). \quad (21)$$

As mentioned before, the movement kinds are iteratively repeated, unless a termination criterion is achieved. Algorithm 1 depicts the proposed ESSOA.

3.2. VM Allocation. By utilizing BMGSA, the VMA on the suited PMs is performed after VMS. In GSA, the bodies in the universe are expressed by agents. GSA sometimes fails to discover global optimum and is effortlessly trapped into local optima because of the deficiency of a pre-eminent balance between exploration and exploitation. A Brownian movement (BM)-centered GSA called BMGSA is proposed to conquer the premature issue and ameliorate the local searching capability of GSA. Employing the BM, a modified search equation with more helpful information from the search experiences is established in the proposed algorithm to yield a candidate solution wielded to evade trapping into local optima. Three metrics, namely the energy consumption of PM, SLAV of PM, and meantime of PM before its shutdown, are considered for BMGSA's objective for VMA. The metrics for VMA are computed as follows:

Energy consumption: while they are idle, the servers consume enormous power. It is given by the following:

$$f_e = \sum_{j=1}^{M}\left(\left(PM_k^{pm} - PM_k^{pm}\right) \times PM_{ruk} + PM_k^{pm}\right) \times W_k, \quad (22)$$

where the PM's total power consumption is given by f_e, whilst $PM_k^{p\min} = PM_k^{p\max} \times 0.6$ describes the minimum power consumption of PM_k. The utilization ratio of resources employed by PM_k at instant t is symbolized by PM_{ruk}, while $W_k \in (0, 1)$ is equivalent to 1 if PM_k is turned on. Otherwise, $W_k = 0$.

SLAV: cloud computing attempts to fulfill the quality of service (QoS) needs, which are modeled in the SLAV form to augment the response time or minimize the throughput. As stated in the subsequent equations, SLAV can be engendered by a host.

$$f_{SLAV} = A \times B, \quad (23)$$

where the average ratio of the period is symbolized by A if the host experiences CPU usage of 100%. It is stated as follows:

$$A = \frac{1}{N}\sum_{k=1}^{N}\frac{G_k}{E_k}, \quad (24)$$

where the active time of k^{th} host is E_k, and the total time when the k^{th} host experiences 100% SLAV utilization is depicted by G_k. Furthermore, the term illustrates the degradation in the performance because of VM's migration B. It is described as follows:

$$B = \frac{1}{M}\sum_{j=1}^{M}\frac{CR_j}{CP_j}, \quad (25)$$

where the total CPU utilization required by VM_j is indicated by CP_j, and the degraded performance that arises as of VM migration is indicated by CR_j.

Mean time before a host shutdown (MTBHS):

This time is gauged in seconds along with the average and is computed as follows:

$$f_{MTj} = \frac{1}{N}\sum_{k=1}^{N} Sd_k, \quad (26)$$

where Sd_k signifies the host shutdown time. The limitations considered for VMA in the proposed model are as follows:

$$\sum_{k=1}^{N} A_{jk} \le 1, \quad \forall_j \in \{1, \ldots, M\}, \quad (27)$$

$$\sum_{k=1}^{N} VM_{\text{cpuj}} \times A_{jk} \le PM_{\text{cpuk}}, \quad (28)$$

$$\sum_{k=1}^{N} VM_{\text{ramj}} \times A_{jk} \le PM_{\text{ramk}}, \quad (29)$$

$$\sum_{k=1}^{N} VM_{hdj} \times PM_{jk} \le PM_{hdk}, \quad (30)$$

where Ajk = 0 unless the VMj allocates on PM_k. VM_j should be performed on a single PM_k when SLAV_k is the least priority in constraint (27). The current physical machine (PM_k) must have adequate resources to work correctly and serve all its VM*s* at an instant t in constraints (28)–(30). The following steps explain the above metrics and constraints that are utilized in the VMA model.

Step 1: by the following equation (31), randomly generate the initial population of individuals in the search space.

$$Q_k = \left\{q_k^1, q_k^2, q_k^3, \ldots, q_k^d, \ldots, q_k^D(t)\right\}, \quad k \in 1, 2, 3, \ldots, N, \quad (31)$$

$$q_k^d(t = 0) \sim U\left(q_{\min}^d, q_{\max}^d\right), \quad (32)$$

where the position of the k^{th} individual in the d^{th} dimension is denoted by $q_k{}^d$, and the number of dimensions is depicted by D. $(q_{\min}^d, q_{\max}^d)$ symbolizes the search space's boundary for d. The velocities are initialized to zero.

$$Z_k(t = 0) = 0. \quad (33)$$

Step 2: appraise the individual's fitness by the subsequent equation:

$$F = \left(f_e, f_{SLAV}, f_{MTj}\right), \tag{34}$$

where f_e indicates energy consumption, SLAV, and f_{MT} meantime value of the PM*s* is computed as follows:

Step 3: the best and worst fitness through the population at the time t is recognized after examining the entire population's fitness. The Best (B_t) and Worst (W_t) in minimization problems are as follows:

$$\begin{aligned} W_t &= \max\left[f_1(t), f_2(t), \ldots, f_k(t), \ldots, f_N(t)\right], \\ B_t &= \max\left[f_1(t), f_2(t), \ldots, f_k(t), \ldots, f_N(t)\right], \end{aligned} \tag{35}$$

where the size of the population is articulated by N.

Step 4: the population's update phase follows the recognition of (B_t) and Worst(W_t). The masses of the agents are the 1st phases that are to be updated. As per its fitness f_k^t, the mass of agent k, M_k^t, is mapped as follows:

$$\begin{aligned} M_k^t &= \frac{m_k^t}{\sum_{j=1}^{n} m_j^t}, \\ m_i^t &= \frac{f_k^t - W_t}{B_t - W_t}. \end{aligned} \tag{36}$$

Step 5: after that, the force acting on every agent is updated. The gravitational force acting on agent k engendered by an agent j in the d^{th} dimension $W_{kj}^d(t)$ is computed utilizing agent j's mass, M_j^t, and agent k's mass, M_k^t .

$$W_{kj}^d(t) = g_t \frac{M_k^t \cdot M_j^t}{E_{kj}^t + \epsilon}\left(q_j^d(t-1) - q_k^d(t-1)\right), \tag{37}$$

where the Euclidean distance betwixt agent k and j is delineated by E_{kj}^t. The term ϵ is a lesser constant that is appended to the divisor to evade division by zero when the agents are overlapping one another, and the gravitational constant at a time t is g_t, which is computed as follows:

$$g_t = g_0 \times e^{-\Phi t / I_t}, \tag{38}$$

where the gravitational constant at the commencement of the search is g_0, and Φ is another constant. The total number of iterations is I_t.

Step 6: the total force acting on the agent i in dimension d is given by the following:

$$W_k^d(t) = \sum_{j=1, j\neq 1}^{n} rd_j^d \cdot W_{kj}^d(t), \tag{39}$$

where a random number in the interval [0, 1] is *rd*. The Brownian movement (BM) is applied to the random sequences to get a BM value rather than selecting random numbers in the range of [0, 1].

$$\begin{aligned} B_M &= h * \text{rand} * L_p, \\ h &= \sqrt{\frac{T}{G}}, \\ G &= 100 * T, \\ L_p &= \frac{1}{h\sqrt{2\pi}} \exp\left(-\frac{(D - \text{agents})^2}{2h^2}\right), \end{aligned} \tag{40}$$

where the motion period in the seconds of sharks is indicated by T. The number of sudden motions for the similar agent in proportion to time is given by G. The search space dimension is denoted by D. The agents in GSA move subjected to Newton's law of motion. The acceleration of agent k overdimension d of $\mathfrak{F}_k^d(t)$ can be computed utilizing (41) as per Newton's law.

$$\mathfrak{F}_k^d(t) = \frac{f_k^d(t)}{M_k^t}. \tag{41}$$

Step 7: with the aid of equation (42), the agent's velocities along with positions are then updated.

$$Z_k^d(t) = rd_k^d \times Z_k^d(t-1) + \mathfrak{F}_k^d(t), \tag{42}$$

$$q_k^d(t) = q_k^d(t-1) + Z_k^d(t). \tag{43}$$

The above steps are repeated to the highest iteration till it meets the stopping condition. Finally, algorithm-2 portrays the pseudocode for the proposed VMA.

Complexity analysis: a function that links the input problem's size to the algorithm's run-time may be used to define the computing complexity of an algorithm. To illustrate the computing complexity of the time usage of the suggested ESSOA and BMGSA, Big-O notation is used here as a known expression. The number of sharks/positions(n), the dimensions(d) of the presented problem, the number of iterations(z), and the cost of the function assessment have a role in the time complexity issue (c). The time complexity may be expressed in the following way:

$$\begin{aligned} O(\text{ESSOA} + \text{BMGSA}) = {} & O(\text{ProblemDef}) + O(\text{initialization}) \\ & + O(\text{SolUpdate}), \end{aligned} \tag{44}$$

where the following definitions apply to the time complexity of the elements in (44).

(1) The issue definition initialization takes $O(1)$ time.

(2) Population generation initialization takes $O(n\ X\ d)$ time.

(3) The cost function assessment takes $O(Z\ X\ c\ X\ n)$ time.

(4) Evaluating the updated solution requires $O(Z\ X\ n\ X\ d)$ time.

Thus, the following is an expression of the overall time complexity of ESSOA and BMGSA.

Input: set of VM lists, VM_j
Output: optimally selected VMs for VM Scheduling
(1) Start
(2) Initialize the population of the algorithm as $p_1^1, p_2^1, \ldots, p_M^1$ and set other parameters
(3) Compute the F of all individuals, $F_i = P_{vmj} * C_{ij}$
(4) Initialize the stage counter $z = 1$
(5) for $(z = 1: z_{\max})$
//Perform the forward movement
(6) Compute each component of the velocity vector, $|v_j^g|$
(7) Obtain the new position of shark, $R_j^{g+1} = p_j^g + v_j^g * \Delta t_g$
//Perform the rotational movement
(8) Obtain the position of the shark with the help of BS, $S_j^{g+1,u} = R_j^{g+1} + r_3 R_j^{g+1}$
(9) Select the next position of the shark based on the forward and rational movements, p_j^{g+1}
(10) end for Z
(11) Set $Z = Z + 1$
(12) Select the best position for shark in the last stage, which has the minimum fitness value
(13) End

Algorithm 1: ESSOA for optimal VM scheduling.

$$O(\text{ESSOA} + \text{BMGSA}) = 2O\left(1 + nd + Z_{cn} + Z_{nd}\right). \quad (45)$$

As was previously mentioned, the time complexity problem for ESSOA and BMGSA is polynomial in order. The suggested ESSOA and BMGSA may thus be regarded as computationally efficient optimization techniques. Briefly, it can be shown from (45) how the primary variables affect and how difficult it will be for ESSOA and BMGSA to solve an optimization problem computationally. Since we apply at two levels, constant 2 is taken into account.

Space complexity: the factors of the number of searches and the size of the issue determine how much memory space is required by ESSOA and BMGSA. It shows how much space was needed for the beginning of the planned ESSOA and BMGSA. For this, (46) may be used to express the space complexity of ESSOA and BMGSA.

$$2O(nd). \quad (46)$$

4. Results and Discussion

The proposed model's outcomes are discussed and compared with the prevailing models regarding in respect of four metrics: execution time, makespan, degree of imbalance (DOI), and resource utilization (RU) of PMs. The working platform of JAVA-based CloudSim has been used in the proposed model for the simulation. The prevailing models, such as genetic algorithm (GA), particle swarm optimization (PSO) algorithm, shark smell optimization algorithm (SSOA), and gravitation search algorithm (GSA) are considered for comparison. Google Cluster and PlanetLab workloads are the two diverse workloads contemplated for execution. Table 1 shows the different real workload traces with Resources.

In our experiments, we have compared the proposed algorithm with the existing approaches using the tools, test configuration parameters, and methodology outlined in Tables 1–3. Although this approach also allowed us to work on different workload characteristics, it led to the explosion of different test results, as shown in Figures 3–6.

The makespan's outcomes in the Google cluster and PlanetLab workloads for algorithms like ESSOA, SSOA, GA, and PSO are displayed in Figures 3(a) and 3(b). By varying the tasks from 100 to 500, the makespan for both workloads is assessed. The algorithm's makespan increases if the task counts increase. However, when analyzed with the prevailing algorithms, the proposed approach attained a very low makespan for both workloads. For 100 tasks, the proposed approach acquires the makespan of 8.32, while the prevailing SSOA, GA, and PSO attained the makespan of 12.21, 20.102, and 22.58, which are higher than ESSOA in Google Cluster workload. Similarly, the proposed approach acquires the lowest values of makespan as compared to the outcomes for the rest of the tasks of Google Cluster. Results are also well relatable to PlanetLab workload.

Figure 4 exhibits the execution cost's outcomes for algorithms in the Google Cluster and PlanetLab workloads for ESSOA, SSOA, GA, and PSO. When compared with others in both workloads, the outcomes show that the execution time accomplished by the proposed model is lower, implying that the proposed approach yields improvement in performance concerning lowering the VM scheduling cost. The task's execution cost augments over the escalating number of tasks. For the task counts like 100, 200, 300, 400, and 500, the proposed approach obtains the execution cost of 10.25, 19.63, 29.36, 36.98, and 48.25 for Google workload, which is lesser than SSOA, GA, and PSO. On the whole, extremely poor performance is acquired by PSO and the average performance is attained by GA. The ESSOA and SSOA attain excellent outcomes. However, when scrutinizing the whole performance for both workloads, the proposed algorithm outperformed others by their optimal process.

Figure 5 displays the degree of imbalance outcome amongst VMs. The execution abilities delineated the amount of load distribution amongst the VMs. As a result, the

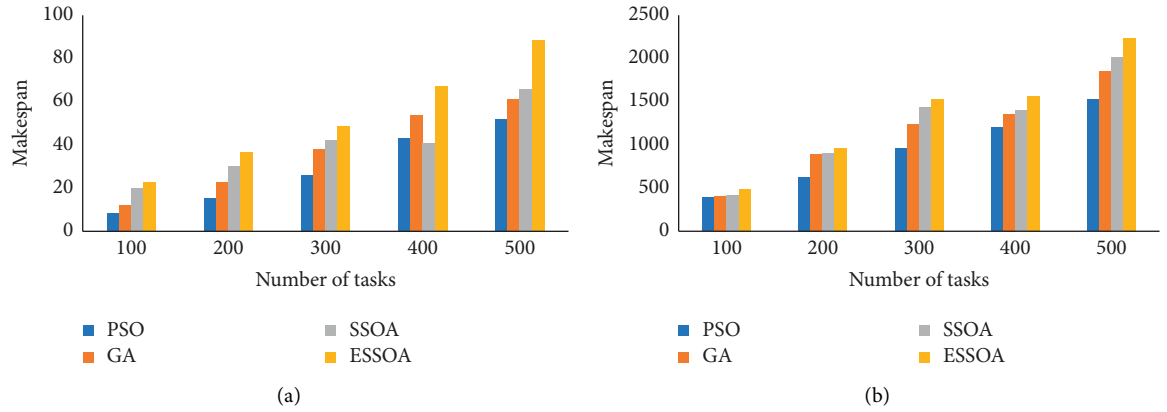

Figure 3: Makespan of the proposed and existing algorithms. (a) Makespan using Google Cluster and (b) makespan using PlanetLab.

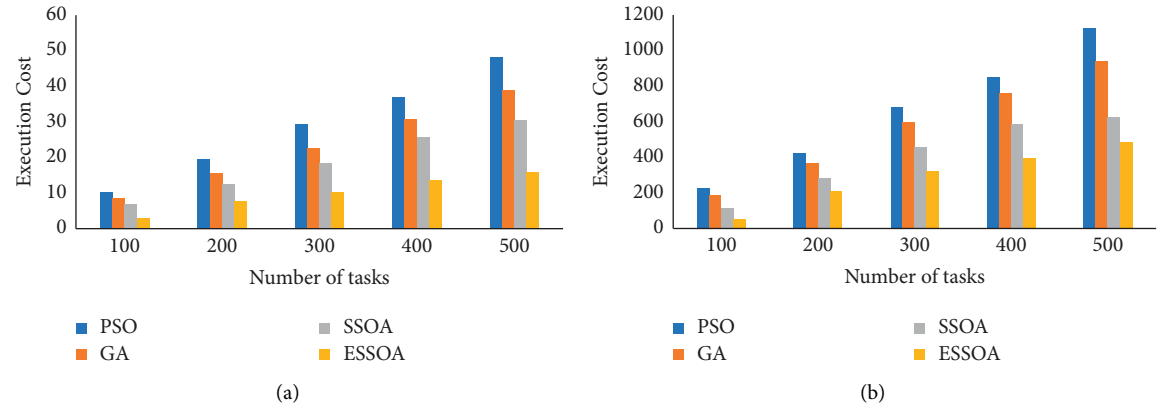

Figure 4: Execution cost of the proposed and existing techniques. (a) Execution cost using Google Cluster and (b) execution cost using PlanetLab.

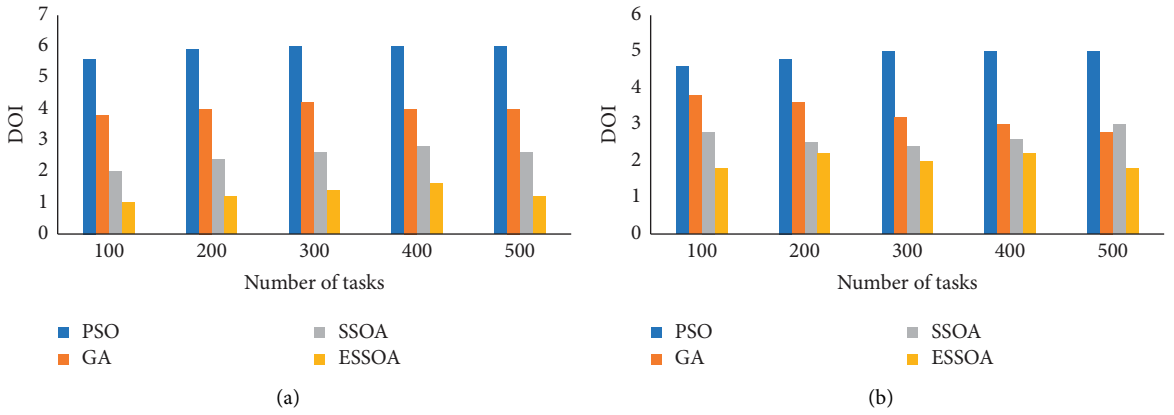

Figure 5: DOI of the proposed and existing techniques. (a) DOI using Google Cluster and (b) DOI using PlanetLab.

system's load is highly balanced and efficient, denoted by the minimum value of DOI. By varying the number of tasks from 100 to 500, the average degree of imbalance for the algorithms in Google Cluster and PlanetLab workloads is plotted. Regarding VMs' load balancing, it is perceived from the outcomes that the proposed approach creates performance enhancement. When compared to the prevailing algorithms, it achieves a minimal value of degree of imbalance for the whole varying tasks in both Google Cluster and PlanetLab workloads. For tasks like 100, 200, 300, 400, and 500, the degree of imbalance of the proposed work maintains the level of 1, 1.2, 1.4, 1.6, and 1.2 for the Google Cluster workload. Depending on the incoming tasks, the system's load may enhance or decrease. Therefore, when compared to other approaches, the proposed approach sustains a better load balance for the scheduling process.

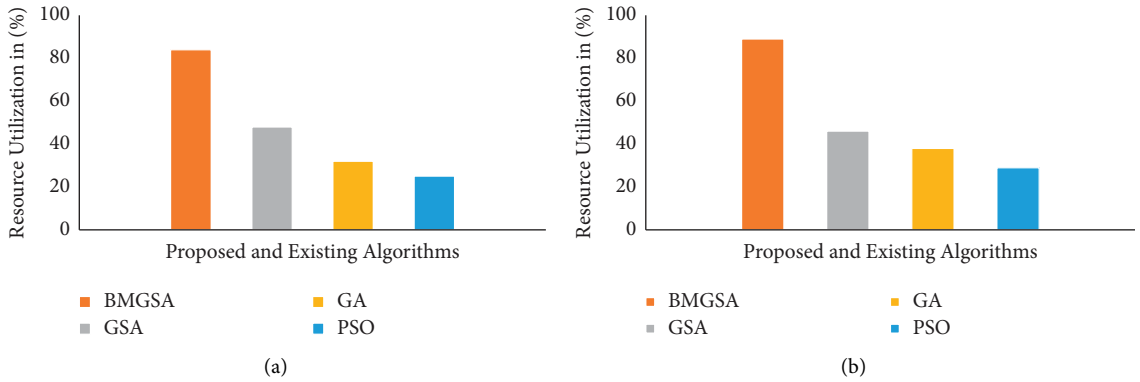

Figure 6: Resource utilization of the proposed and existing techniques. (a) Resource utilization using Google Cluster and (b) resource utilization using PlanetLab.

Input: set of PMs, PM_k
Output: optimally selected PMs for VM
(1) Start
(2) for $k = 1$ to N
(3) for $d = 1$ to D
(4) Initialize $Q_k = \{q_k^1, q_k^2, q_k^3, \ldots, q_k^d, \ldots, q_k^D(t)\}$
(5) Initialize velocity, $Z_k(t = 0)$
(6) next d
(7) Compute the fitness of each agent, $F = (f_e, f_{\text{SLAV}}, f_{MT_j})$
(8) next k
(9) end for
(10) for $t = 1$ to N
(11) Select the best agent and worst agent, B_t and W_t
(12) Update mass of each agent, m_t^i
(13) Update force acting on each agent, $W_{kj}^d(t)$
(14) Update $g_t = g_0 \times e^{-\Phi t/I_t}$
(15) Compute total force acting on an agent by Brownian Movement sequence, $W_k^d(t)$
(16) end for
(17) end for
(18) for $k = 1$ to N
(19) for $d = 1$ to D
(20) Compute acceleration, $\Im_k^d(t)$
(21) Update the position, $q_k^d(t)$
(22) Update the velocity, $Z_k^d(t)$
(23) next
(24) Compute the fitness value of the next agent
(25) next
(26) next
(27) end for
(28) end for
(29) end

Algorithm 2: BMGSA for optimal VM allocation.

Figure 6 displays the resource utilization of the proposed and existing algorithms for the proposed approach in both Google Cluster and PlanetLab workloads. This metric reveals how PMs are utilized to augment the physical hosts utilization. Comparing this to the GSA, GA, and PSO, respectively, it is observed from the figures that the proposed approach has the greater resource utilization of physical

Table 1: Different real workload traces with Resources.

Workload	Size	Dynamic VM data		
		CPU	Memory	Disk
PlanetLab	1000 VMs	Yes	No	No
Google cluster	12000 PMs	Yes	Yes	Yes

Table 2: Overview of the parameters considered for conducted experiments.

Parameters	Test cases
Workload	(i) PlanetLab (ii) Google cluster
CPU load of VMs	(i) Constant 10% of the requested capacity (ii) Constant 50% of the requested capacity (iii) Constant 100% of the requested capacity
VM size	(i) Smaller than in the baseline configuration (ii) According to the baseline configuration (iii) Bigger than in the baseline configuration
PM power characteristics	(i) All PMs have the same power characteristics (ii) Multiple different power characteristics (iii) Power characteristics with a slight slope (iv) Power characteristics with a large slope
Host overload detection	(i) Static threshold (ii) Local regression

Table 3: Configuration and simulation parameters on cloud sim.

Configurations	Value
Number of data center	10
Number of processing elements	1
Speed of processing element	10000 MIPS
Memory	2048 MB
Virtual machine monitor	Xen
Total number of tasks/cloudlets	100 to 500
Length of tasks	500–10000 million instruction
Cost of per storage	0.001 $
Cost of communication	0.001$
Cost of execution	1.0 $
Cost of memory	0.05$

hosts. When scheduling VMs to apt Hosts, the proposed algorithm considers resource utilization metrics like energy consumption, SLAV, and meantime, whilst the prevailing algorithms merely consider the execution cost within the deadline. The Resource utilization acquired by the proposed approach is 84% and 89% for Google Cluster and PlanetLab workload, which is significant, and it outweighed the GSA, GA, and PSO. Regarding maximizing the resource utilization of PMs, the proposed approach produces performance improvement.

For Google Cluster and PlanetLab workloads, results exhibit VMA algorithms that compare other metrics, such as energy consumption, SLAV, and performance degradation, because of VM migrations (PDVM). The lower values denote the algorithm's efficacy for every metric. The proposed approach acquires an energy consumption of 112.58, which is lesser than PSO (156.98), GA (145.25), and GSA (13.58) for the Google Cluster workload. Similarly, the proposed approach offers a lower energy consumption value for PlanetLab workloads. The metrics of SLAV, along with PDVM, also offer better outcomes. It acquires lower values of SLAV and PDVM for both workloads, which indicates the efficient performance of the proposed approach for VM scheduling and allocation in the cloud data center. PSO exhibits the worst performance. GA and GSA portray a moderate performance.

5. Conclusions

In cloud data centers, the policy-based VM scheduling and allocation are regarded as co-optimization problems. The proposed work provides solutions by implementing ESSOA and BMGSA at two levels for VM scheduling and allocation. Firstly, the execution time of VM scheduling is optimally reserved for suitable PMs using ESSOA. Then, the chosen VMs are hosted at the second level to appropriate PMs utilizing BMGSA. A Java-based CloudSim has been used to implement the proposed work by considering real workloads, such as Google Cluster and PlanetLab, and performance metrics, such as energy consumption, meantime, SLAVs, makespan, execution cost, degree of imbalance, and resource utilization. The findings of the proposed VM scheduling and allocation algorithms are investigated against the existing algorithms, such as SSOA, GA, PSO, and GSA. Resource utilization acquired by the proposed approach is 84% and 89% for Google Cluster and PlanetLab workload. Hence, considering the significance of the policies defined through ESSOA and BMGSA, this research deduces that scheduling and allocation can be carried out more efficiently. Because of the security limitations in both scheduling and allocation, the work will be expanded in the future with the assistance of an advanced optimization framework.

Data Availability

The labeled datasets used to support the findings of this study can be obtained from the corresponding author upon request.

Conflicts of Interest

All authors declare that they have no conflicts of interest.

References

[1] K. Karmakar, R. K. Das, and S. Khatua, "An ACO-based multi-objective optimization for Cooperating VM placement in cloud data center," *The Journal of Supercomputing*, vol. 78, no. 3, pp. 3093–3121, 2021.

[2] S. A. El-Seoud, E. A. AboGamie, and M. Salama, "Integrated education management system via cloud computing," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 11, p. 24, 2017.

[3] N. S. Hussien, S. Sulaiman, A. Aborujilah, M. Wibowo, and H. Samma, "Scalability of mobile cloud storage," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 15, p. 199, 2021.

[4] S. Supreeth and S. Biradar, "Scheduling virtual Machines for load balancing in cloud computing platform," *International Journal of Science and Research*, vol. 2, no. 6, pp. 437–441, 2013.

[5] X. Ma, H. Gao, H. Xu, and M. Bian, "An IoT-based task scheduling optimization scheme considering the deadline and cost-aware scientific workflow for cloud computing," *EURASIP Journal on Wireless Communications and Networking*, vol. 1, 2019.

[6] A. Ragmani, A. Elomri, N. Abghour, K. Moussaid, and M. Rida, "FACO: a hybrid fuzzy ant colony optimization algorithm for virtual machine scheduling in high-performance cloud computing," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 10, p. 3975, 2019.

[7] M. Ghetas, "A Multi-Objective Monarch Butterfly Algorithm for Virtual Machine Placement in Cloud Computing," *Neural Computing and Applications*, vol. 15, 2021.

[8] S. Supreeth and K. K. Patil, "Virtual machine scheduling strategies in cloud computing- A review," *International Journal on Emerging Technologies*, vol. 10, no. 3, pp. 181–188, 2019.

[9] K. R. Prasanna Kumar and K. Kousalya, "Amelioration of task scheduling in cloud computing using crow search algorithm," *Neural Computing & Applications*, vol. 32, no. 10, pp. 5901–5907, 2019.

[10] S. Ijaz and E. U. Munir, "MOPT: list-based heuristic for scheduling workflows in cloud environment," *The Journal of Supercomputing*, vol. 75, no. 7, pp. 3740–3768, 2018.

[11] H. Singh, S. Tyagi, and P. Kumar, "Crow-penguin optimizer for multiobjective task scheduling strategy in cloud computing," *International Journal of Communication Systems*, vol. 33, no. 14, 2020.

[12] F. Ramezani, J. Lu, and F. Hussain, "Task Scheduling Optimization in Cloud Computing Applying Multi-Objective Particle Swarm Optimization," *Service-Oriented Computing*, vol. 8274, pp. 237–251, 2013.

[13] S. K. Roy, R. Devaraj, A. Sarkar, K. Maji, and S. Sinha, "Contention-aware optimal scheduling of real-time precedence-constrained task graphs on heterogeneous distributed systems," *Journal of Systems Architecture*, vol. 105, Article ID 101706, 2020.

[14] H. Talebian, "Cluster computing," *Optimizing virtual machine placement in IaaS data centers: Taxonomy, Review and Open Issues*, vol. 23, pp. 837–878, 2019.

[15] L. Zhang, Y. Wang, L. Zhu, and W. Ji, "Towards energy efficient cloud: an optimized ant colony model for virtual machine placement," *Journal of Communications and Information Networks*, vol. 1, no. 4, pp. 116–132, 2016.

[16] S. Supreeth and K. Patil, "VM scheduling for efficient dynamically migrated virtual Machines (VMS-EDMVM) in cloud computing environment," *KSII Transactions on Internet and Information Systems*, vol. 16, no. 6, pp. 1892–1912, 2022.

[17] M. Masdari, S. Gharehpasha, M. Ghobaei-Arani, and V. Ghasemi, "Bio-inspired virtual machine placement schemes in cloud computing environment: taxonomy, review and future research directions," *Cluster Computing*, vol. 23, no. 4, pp. 2533–2563, 2019.

[18] D. Alboaneen, H. Tianfield, Y. Zhang, and B. Pranggono, "A metaheuristic method for joint task scheduling and virtual machine placement in cloud data centers," *Future Generation Computer Systems*, vol. 115, pp. 201–212, Feb. 2021.

[19] H. O. Salami, A. Bala, S. M. Sait, and I. Ismail, "An energy-efficient cuckoo search algorithm for virtual machine placement in cloud computing data centers," *The Journal of Supercomputing*, vol. 77, no. 11, Article ID 13330, 2021.

[20] S. Supreeth and K. Patil, "Hybrid genetic algorithm and modified-particle swarm optimization algorithm (GA-MPSO) for predicting scheduling virtual Machines in educational cloud platforms," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 17, pp. 208–225, 2022.

[21] A. Gopu and N. Venkataraman, "Optimal VM placement in distributed cloud environment using MOEA/D," *Soft Computing*, vol. 23, Article ID 11277, 21 pages, 2018.

[22] L. Abualigah and A. Diabat, "A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments," *Cluster Computing*, vol. 24, pp. 205–223, 2020.

[23] X. Fu, Y. Sun, H. Wang, and H. Li, "Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm," *Cluster Computing*, 2021.

[24] V. M. Arul Xavier and S. Annadurai, "Cluster computing," *Chaotic social spider algorithm for load balance aware task scheduling in cloud computing*, vol. 22, pp. 287–297, 2018.

[25] J. Lu, W. Zhao, H. Zhu, J. Li, Z. Cheng, and G. Xiao, "Optimal machine placement based on improved genetic algorithm in cloud computing," *The Journal of Supercomputing*, vol. 78, no. 3, pp. 3448–3476, 2021.

[26] A. Tripathi, I. Pathak, and D. P. Vidyarthi, "Modified dragonfly algorithm for optimal virtual machine placement in cloud computing," *Journal of Network and Systems Management*, vol. 28, no. 4, pp. 1316–1342, 2020.

[27] G. Shruthi, M. R. Mundada, B. J. Sowmya, and S. Supreeth, "Mayfly taylor optimisation-based scheduling algorithm with deep reinforcement learning for dynamic scheduling in fog-cloud computing," *Applied Computational Intelligence and Soft Computing*, vol. 2022, Article ID 2131699, 17 pages, 2022.