IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 15, No. 2, April 2026, pp. 1286~1299
ISSN: 2252-8938, DOI: 10.11591/ijai.v15.i2.pp1286-1299

Energy-efficient virtual machine allocation using directional and boundary-aware bobcat optimization

Nida Kousar Gouse, Gopala Krishnan Chandrasekaran
Department of Computer Science and Engineering, GITAM School of Technology, Bengaluru, India

Article Info

Article history:

Received Sep 22, 2025
Revised Jan 8, 2026
Accepted Jan 25, 2026

Keywords:

Bobcat optimization algorithm
boundary-aware strategy
Cloud computing
Directional movement
Energy-efficient virtual machine allocation
Physical machines

ABSTRACT

Cloud computing (CC) has gained significant traction due to its ability to deliver services in a scalable and adaptable manner, catering to diverse user requirements. However, in virtualization technology, one of the primary challenges is managing the energy consumption required to maintain service quality, as it directly impacts the operational expenses of data centers. To address this challenge, this research proposes a directional movement and boundary-aware strategy-based bobcat optimization algorithm (DMBABOA) for energy-efficient virtual machine (VM) allocation aimed at minimizing energy consumption in cloud environments. The directional search and boundary-aware correction enhance convergence and ensure feasible resource distribution. This ensures effective utilization of resources, improved virtualization management, and substantial energy savings. The experimental findings establish that the proposed DMBABOA optimizer reaches a minimum execution time of 134.48 s when the number of VMs is equal to 1,200 with 200 users, compared to existing methods such as the metaheuristic VM allocation approach to power efficiency of sustainable cloud environment (MV-PESC).

Corresponding Author:

Nida Kousar Gouse
Department of Computer Science and Engineering, GITAM School of Technology
Bengaluru, India
Email: nkousar@gitam.in

1. INTRODUCTION

Cloud computing (CC) has emerged as a dominant archetype for efficient data storage and processing, offering end-users on-demand services supported by virtualized resources [1]. Through virtualization, multiple virtual machines (VMs) can be created and assigned to a physical machine (PM), significantly reducing the cost of acquiring additional server infrastructure [2]. VMs provide benefits including mobility, agility, scalability, and elasticity [3] by virtualizing PM resources like storage, CPU, and RAM, and implementing user tasks [4]. This model has transformed the delivery of technological services, enabling service providers to offer users a wide range of facilities [5]. These systems basically contain diverse types: infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS) [6]. In IaaS, providers deliver virtualized computing resources like servers, networks, and storage, allowing customers to run applications on existing infrastructure without purchasing hardware. PaaS enables developers to organize and maintain applications without the need to install or oversee the underlying infrastructure [7], [8]. SaaS provides software across the internet, enabling users for performing the devices using a web browser lack of maintenance or deployment requirements [9], [10]. In cloud environments, service requests from users are highly dynamic, making resource allocation an ongoing challenge for cloud service providers (CSPs) [11]. Because of limited resources, CSPs must manage allocation while considering

multiple factors such as service quality, pricing, fairness, profitability, and load balancing [12], [13]. As both CSPs and consumers seek to maximize their benefits, this process becomes complex [14], [15]. Poor resource management can lead to substantial resource wastage, making efficient allocation crucial for enhancing utilization rates and improving power efficiency [16]. Energy-aware VM allocation and migration play vital roles in balancing energy consumption and maintaining service quality [17]. However, achieving both security and energy efficiency in CC remains a significant challenge [18]. Reducing energy consumption supports sustainable computing and lowers the operational expenses of organizational data centers [19], [20]. Sharma *et al.* [21] developed a multidimensional virtual machine model (MDVMM) and a branch-and-price-assisted energy-efficient VM approach at the data center. This branch-based VM approach minimizes energy consumption and resource wastage through selecting the ideal PM counts at the cloud datacenter. However, MDVMM struggled to adapt to dynamic or unpredictable workloads in cloud environments, leading to suboptimal migration, enhanced overhead, and service level agreement (SLA) violations. Yao *et al.* [22] implemented a load-balancing mechanism-driven virtual machine consolidation (LBVMC), targeted to minimize SLA violations and energy utilization through balanced multidimensional resource utilization across PMs. However, LBVMC requires frequent VM migrations to maintain balanced resource utilization across PM, which leads to temporary service distribution.

Cao *et al.* [23] introduced a secure and energy-efficient VM allocation mechanism to prevent co-residence attacks and quantified three key factors: security risks due to co-residence of VMs from different users, comprehensive energy consumption, and workload inequality among PMs. Although these objectives were minimized simultaneously, a random number of VM from various users reached a cloud that required an optimization solution to dynamically relate to previous allocation and unknown allocation requests. However, defending against co-residence attacks led to underutilization of physical resources, thereby reducing energy consumption due to the trade-off between higher security and resource utilization. Madireddy and Ravindranath [24] introduced dynamic virtual machine relocation (DVMR) for confidential data centers to understand virtualization and minimize energy consumption in CC. DVMR system generated an approach for PM load conditions by identifying overloaded and underloaded PMs to manage virtualization and minimize energy consumption efficiently. However, frequent VM relocations introduce significant migration overhead, which reduces application performance and causes service distributions, especially under high-load conditions.

Ajmera and Tewari [25] proposed the green particle swarm optimization (GPSO) approach for VM allocation on energy-efficient green systems. These systems, referred to as green particles, are designed to consume less power while performing a global search to determine an optimal VM scheduling plan that minimizes the number of active servers. This approach effectively reduces overall power consumption in data centers while maintaining SLA. Singh *et al.* [26] presented the metaheuristic VM placement framework for power efficiency in a sustainable cloud environment (MV-PESC). This method employed an extended flower pollination optimization approach incorporating the principles of the random fit approach along with the standard flower pollination optimization technique. An efficiency of the framework was validated using workload traces from the Google cluster dataset, demonstrating optimization capability in real-world cloud environments. Swain *et al.* [27] implemented a resource-prediction-assisted VM allocation strategy aimed at reducing the energy consumption whereas improving system consistency.

The key contribution presented in enhancing a feed-forward neural network (FFNN) through a self-adaptive differential evolution approach that combines multidimensional learning and global search capabilities. By accurately forecasting future resource demands, this approach enables proactive and fault-tolerant VM management, minimizes failure impact, and enhances performance. Mahmoodabadi and Baygi [28] introduced an energy-effective virtual machine placement (VMP) technique utilized the vector bin packing approach to reduce energy usage in datacenters. Their work focused on a bin packing with linear usage cost (BPLUC) model, which described for both fixed and variable operational costs, thereby achieving optimized resource allocation with reduced power consumption. Although CC enables dynamic resource sharing, it also exposes systems to co-residence attacks, where malicious VMs exploit shared hardware with target VMs. Existing solutions, such as hypervisor controls and side-channel defenses, are often resource-intensive or inflexible. Prior VM allocation methods lacked real-time adaptability, workload balancing, and energy efficiency. Figure 1 illustrates the taxonomy of existing VM allocation strategies, providing a clearer synthesis of the related works.

The crucial notes of this research are: an innovative energy-efficient and consistent VM allocation approach named directional movement and boundary-aware strategy-based bobcat optimization algorithm (DMBABOA) is proposed for cloud environments, ensuring optimal placement on energy-efficient and consistent PMs. To ensure load balance of destination PMs later to VM allocation, this study designs a VM allocation approach in terms of resource fitness and VM load relationship. The performance of the approach is validated through simulation using the CloudSim toolkit by considering different performance metrics.

This paper is formatted as trails: section 2 specifies the proposed methodology. Section 3 signifies the VM allocation using the bobcat optimization algorithm (BOA). Section 4 shows the results and discussion. Section 5 gives the conclusion.

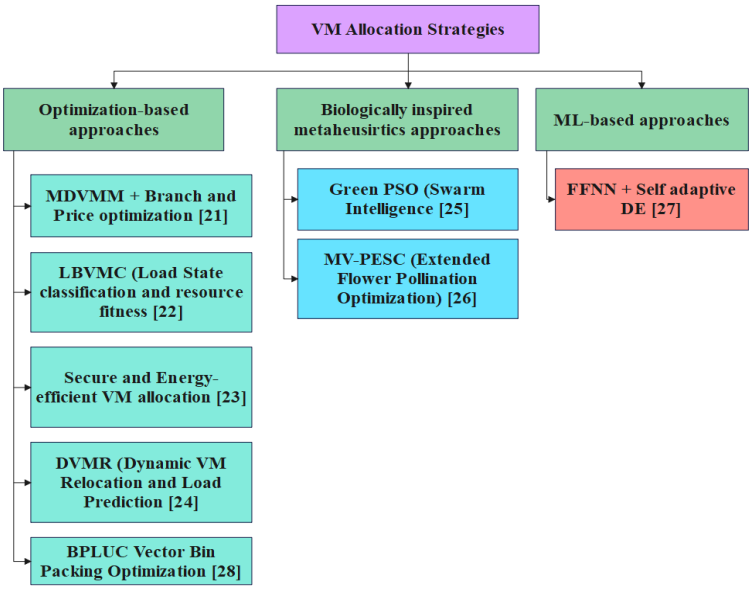

Figure 1. Taxonomy of existing VM allocation strategies

2. PROPOSED METHOD

This section outlines the proposed secure VM allocation strategy in detail. It begins with an overview of the system model, describing the core components and interactions within the CC environment. Next, the concept of energy efficiency is formulated as an optimization problem, defining the objectives and constraints governing the allocation process. Finally, the application of the DMBABOA is presented to efficiently solve the formulated problem while ensuring both security and optimal resource utilization. Figure 2 outlines an overall system architecture of the proposed method, representing the key components for energy-efficient VM allocation.

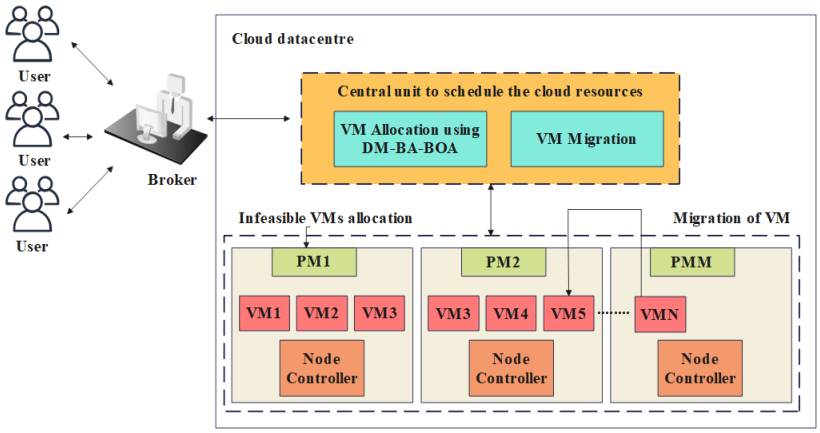

Figure 2. Overall system architecture of the proposed method

2.1. System model

The proposed system model is designed for an IaaS environment, where a data center typically comprises application controllers, cloud administrators, and local administrators. In this architecture, an application controller manages incoming user requests using built-in software. A cloud administrator

oversees cloud services at the application programming interface (API) level, ensuring that user requests are directed to the appropriate application. At the physical infrastructure level, a local administrator handles the internal resources of PMs, assigns VMs to end users, and executes their tasks. As a program supervisor, the local administrator determines whether a new VM request can be fulfilled while adhering to quality of service (QoS) requirements.

The cloud manager utilizes information from the local manager to coordinate and manage VM migration between physical hosts. This includes deciding whether to relocate a VM, enable virtualization features, or use storage drives for VM operations. The local manager is responsible for continuous maintenance of each VM on every host and manages the allocation of physical resources among them. Here, the BOA is introduced for achieving the energy-efficient and secure VM allocation. The model represents a virtual data center containing multiple cloud clients consisting of N physical nodes and an associated list of PMs, expressed as: $P = \{PM_1, PM_2, \dots, PM_n\}$. Each PM is homogeneous and consists of V VMs, represented as: $V = \{VM_1, VM_2, \dots, VM_v\}$. VMs are dynamically allocated to minimize energy consumption, and their resource demands fluctuate, particularly in terms of CPU utilization. Each VM can be assigned a maximum CPU capability, denoted as C_v. Given that each PM has CPU capacity C_p, VM counts, which are hosted on a single PM is determined by the ratio C_p/C_v. As these VMs can execute a wide range of applications with distinct behaviors, they may vary significantly in function and often run concurrently.

Based on real-time workload demands, VMs can be dynamically activated or deactivated on a PM, enabling efficient utilization of the physical system's resources. Each VM can operate different operating systems and applications simultaneously on the same PM, ensuring flexibility and efficient resource usage for cloud users. Services are provided through applications distributed across several VMs within the underlying cloud infrastructure. Given the highly variable nature of workloads, the resource requirements of each VM can differ significantly; therefore, VMs may migrate across PMs to optimize resource utilization, reduce redundancy, and release unused capacity. Consequently, PMs can be turned off or moved to an idle state to reduce power consumption.

2.2. Energy efficiency model

Most contemporary mainframes are processed through dynamic voltage and frequency scaling (DVFS) advancement, which enables real-time adjustment of operating frequency to reduce energy consumption. Since CPU utilization typically reflects workload intensity, the power usage of a PM is largely affected by its CPU load. DVFS is broadly adopted as an effective mechanism to balance system performance and energy consumption. By adjusting the processor's operating frequency and voltage, DVFS influences overall energy consumption, as lower frequencies may extend execution time despite reducing power. An efficient DVFS-based scheduling strategy aims to minimize total energy consumption while satisfying QoS requirements such as execution deadlines. DVFS allows the processor to operate at different voltage-frequency combinations during active and idle communication periods, contributing to optimized power usage.

Processor allocation strategies generally assign clusters to individual processing units (PUs). A sum of power utilized through PM depends on several components, including network, CPU, storage, and memory (RAM) utilization. Prior studies have shown a strong linear relationship between CPU usage and total energy consumption in servers. For instance, the processor is the dominant contributor to host power consumption, indicating that CPU utilization is directly linked to the power draw. Accordingly, the power model of a server can be expressed as a function of CPU usage, represented mathematically in (1).

$$P_j = \left(P_j^{busy} - P_j^{idle}\right) \times U_j^P + P_j^{idle} \quad (1)$$

Here P_j^{idle} means average power in an indolent state, P_j^{busy} denotes the average power at full utilization, and U_j^P represents the resource consumption of the host j in the current power state. The total energy usage for hosts is estimated as (2).

$$Min \sum_{j=1}^{n} P_j = \sum_{j=1}^{n} \left[Y_j \times \left(\left(P_j^{busy} - P_j^{idle}\right) \times \sum_{i=1}^{m} \left(X_{ij} \times R_i^{VM}\right) + P_j^{idle}\right)\right] \quad (2)$$

Where R is the set of VM processors, Y_j specifies the operation update of the PM with respect to 0 and 1, X_{ij} denotes the task of VM to PM. Even when a PM is idle (i.e., 0% utilization), it still consumes a significant amount of its power. Let α represent the proportion of power consumed by PM in the idle state relative to when it is fully utilized and β represent the power utilization at the current operating level. The total power consumption of PM_i is defined as (3). Where P_i^{max} denotes the power consumption of PM_i.

$$PM_i = \alpha \times P_i^{max} + (1 - \alpha) \times \beta \times P_i^{max} \quad (3)$$

2.3. Security quantification

In particular, assume a comparable distribution of malicious users. The security quantification considers the likelihood of malicious VMs being co-located with legitimate users, which directly influences system vulnerability. This study models the probability of malicious VMs being co-located with legitimate users, thereby capturing potential security risks, as formulated in (4).

$$R_{sec} = P_{mal} \times \frac{\sum_{i=1}^{n_s}\left(n_{co-loc}^{i}-1\right)}{n_s*(n_u-1)} \tag{4}$$

Where P_{mal} represents the evaluated malicious user percentage, and n_s and n_u denote the number of PMs and users, respectively. n_{co-loc}^{i} is a count of co-located users at PM_i. This study assumes that a PM with only one user is secure. Notably, a PM hosting a single user is considered secure, as no co-location exists to enable malicious interactions.

3. VIRTUAL MACHINE ALLOCATION USING BOBCAT OPTIMIZATION ALGORITHM

In the design of the BOA, the population update mechanism within the solution space is encouraged through natural hunting strategies of wild bobcats. In this mechanism, a bobcat initially trails a location of prey and travels towards it. Subsequently, it traps and attacks a prey at an opportune moment and eventually catches the prey later to hurtling procedure. The updated stages of the BOA are described.

3.1. Initialization

BOA is a population-driven optimization approach which iteratively explores the solution space, leveraging the collective search capability of its agents to effectively solve optimization problems. Based on the design inspiration of the BOA, the solution space is analogous to the natural habitat of bobcats, and the position of each bobcat within this habitat represents a location of a BOA member in the solution space. Thus, in the BOA, each bobcat in the population represents a potential solution within the problem-solving space, where its position corresponds to specific values assigned to the decision variables. Mathematically, the bobcat position can be expressed as a vector, in that, every constituent denotes the target component. Collectively, all bobcats constitute an algorithm's population, denoted as a matrix, as represented in (5). The initial position of each bobcat is arbitrarily generated in the solution space using (6).

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix}_{N\times m} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i,m} & \cdots & x_{i,d} & \cdots & x_{i,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,d} & \cdots & x_{N,m} \end{bmatrix}_{N\times m} \tag{5}$$

$$x_{i,d} = lb_d + r.(ub_d - lb_d) \tag{6}$$

Where X denotes the population matrix of the BOA, X_i denotes the i an individual solution, and $x_{i,d}$ denotes the dth dimension in the solution area. N denotes bobcat counts, m specifies the count of target components, and r is an arbitrary count in the range [0, 1]. ub_d and lb_d correspond to the upper and lower boundaries of the dth dimension, individually. As discussed earlier, a location of every bobcat denotes an individual problem-solving, according to that an aim value is evaluated. The values of this function corresponding to these solutions are expressed as a vector, as shown in (7).

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix}_{N\times m} = \begin{bmatrix} F(X_1) \\ \vdots \\ F(X_i) \\ \vdots \\ F(X_N) \end{bmatrix}_{N\times m} \tag{7}$$

Where F denotes a vector of the estimated aim value, and F_i denotes aim value corresponding to ith bobcat. X represents the group of all candidate solutions. An aim value $F(X_i)$ evaluates every solution. Collectively, these evaluations form the matrix F, which holds fitness values of all N candidate solutions across objectives.

3.2. Exploration: tracking and moving towards prey

In an initial stage of the BOA, a location of population members in a solution space is updated according to bobcat's tracking and movement toward prey at the time of hunting. Capturing the bobcat's movement toward its prey results in greater modification in locations of population members in a solution

space and enhances the exploration capability of the BOA with respect to maintaining a global search. In the BOA deployment, for every bobcat, the other population member's locations with superior aim value are identified as prey positions. An individual prey group for every bobcat is identified through (8).

$$CP_i = \{X_k : F_k < F_i \; and \; k \neq 1\}, where \; i = 1,2,\dots,N \; and \; k \in \{1,2,\dots,N\} \quad (8)$$

Where CP_i represents the group of individual prey positions for the ith bobcat, X_k denotes a population member through the optimal aim value in accordance with ith bobcat, and F_k denotes the aim value. In the BOA, every bobcat arbitrarily chooses the identified prey items and initiates an attack. According to the modeled movement of the bobcat toward the prey, a new position is estimated for every BOA individual through (9). If an unknown location attains an enhanced aim value, it exchanges a prior location of the respective member, as defined in (10).

$$x_{i,j}^{P1} = x_{i,j} + \left(1 - 2r_{i,j}\right).\left(SP_{i,j} - I_{i,j}.x_{i,j}\right) \quad (9)$$

$$X_i = \begin{cases} x_{i,j}^{P1}, F_{i,j}^{P1} \leq F_i \\ X_i, else \end{cases} \quad (10)$$

Where $SP_{i,j}$ denotes the prey selected by ith bobcat in jth dimension, and $x_{i,j}^{P1}$ denotes the new location estimated for the ith bobcat in j th dimension of the exploration stage of the proposed BOA. $r_{i,j}$ represents a random number in a range between 0 and 1, and $I_{i,j}$ refers to a number randomly chosen as either one or two.

3.3. Exploitation: chasing to catch prey

In this stage, positions of population members in a solution space are reorganized with respect to simulated chase among bobcat and its prey at the time of hunting. This chase occurs near the prey's location and eventually results in the capture of the prey. The modeled movement of bobcats during this phase introduces small, localized changes to the positions of individuals in the population, thus enhancing the exploitation capability of the BOA and improving its local search performance. An unknown location for each BOA member is estimated close to the prey's location using (11), based on the modeled positional adjustments during the chase. If this updated position attains an optimal function value, it replaces the previous respective individual, as defined in (12).

$$x_{i,j}^{P2} = x_{i,j} + \frac{1-2r_{i,j}}{1+t} \times x_{i,j} \quad (11)$$

$$X_i = \begin{cases} x_{i,j}^{P2}, F_{i,j}^{P2} \leq F_i \\ X_i, else \end{cases} \quad (12)$$

Where $x_{i,j}^{P2}$ denotes an unknown location estimated for the i th bobcat with respect to the exploitation stage of BOA. The conventional BOA uses stochastic search mechanisms to simulate the hunting behavior of bobcats; however, such randomness can lead to inefficient exploration of high-dimensional or constrained problems. To address this limitation, two different strategies are applied in the BOA, discussed in detail as follows.

3.4. Directional movement and boundary aware strategy

To enhance precision, convergence speed, and boundary handling, this research integrates a directional movement strategy guided by the location of the prey and applies boundary-aware adjustments to ensure feasible solutions. This approach draws inspiration from the prey-predator interaction operator in the zoological search optimization (ZSO) [29] model while maintaining the solitary nature of the bobcat. The enhanced BOA combines a directional movement mechanism with a boundary-aware adjustment strategy inspired by ZSO. The functions of the introduced strategy are explained as follows:

i) Directional vector-based movement: each bobcat (solution) intelligently moves toward promising regions using adaptive directional vectors calculated relative to better-performing solutions (prey). Assume the following mathematical modeling: location of bobcat i $x_i \in R^d$; location of chosen prey $x^* \in R^d$; consistently dispersed random number $r \sim U(0,1)$; current iteration t; maximum number of iterations T; directional vector from bobcat to prey $\vec{D}_i$; and exploration control parameter α. The directional vector and position update of the bobcat are defined as (13) and (14).

$$\vec{D}_i = x^* - x_i \quad (13)$$

$$x_i^{t+1} = x_i^t + \alpha.r.\vec{D}_i \quad (14)$$

Where $\alpha = \left(1 - \frac{t}{T}\right)$. This ensures broader exploration in earlier iterations and more focused movement in later ones. The strategy emphasizes movement toward promising solutions, resulting in optimal outcomes. This dynamic approach helps the algorithm avoid premature convergence and increases the likelihood of identifying global optima.

ii) Boundary-aware adjustment: solutions are dynamically adjusted to stay within the legal bounds of server capacities (CPU and memory), ensuring feasibility and preventing random resets or clipping. The boundary-aware adjustment mechanism corrects solutions that violate problem constraints such as CPU or memory capacity in cloud environments. The boundary-aware adjustment is calculated using (15).

$$x_{i,j}^{t+1} = \begin{cases} LB_j + r.\left(UB_j - LB_j\right), if\ x_{i,j}^{t+1} < LB_j \\ UB_j + r.\left(UB_j - LB_j\right), if\ x_{i,j}^{t+1} > UB_j \\ x_{i,j}^{t+1}, otherwise \end{cases} \quad (15)$$

This ensures that no solution exceeds the problem's feasible space.

iii) Adaptive exploration coefficient (α): the coefficient dynamically decreases over iterations, enabling broad early exploration and refined exploitation in later stages. Algorithm 1 presents the pseudocode for the DMBABOA approach for better reproducibility. Because engineering problems involve constraints, the actual metaheuristic approach BOA does not handle the optimization issues. Therefore, this study prepares all metaheuristic approaches using a static penalty function. The fitness function considers energy consumption and is defined in (16).

$$F(R_i) = f(R_i) + w.\sum_{i=1}^{m}\left(max\left(0, g_i(R_i)\right)\right) \quad (16)$$

Where $F(.)$ denotes fitness function, $f(.)$ and $g_i(.)$ denote objective and restriction functions, individually, and w is a constant set. This dynamic adjustment prevents premature convergence while maintaining the balance between global and local search. By integrating this strategy, the algorithm naturally adapts to the optimization process and ensures robust performance across different problem scales and complexities.

Algorithm 1. Pseudocode of the DMBABOA approach

```
BEGIN
1. Initialize system parameters:
      - Population size (N)
      - Maximum iterations (Max_Iter)
      - Lower and upper bounds (LB, UB) for server resources (CPU, Memory)
      - Adaptive exploration coefficient α ∈ [1,0]
      - VM and host configuration data (CPU, Memory, Security Level)
2. Initialize population of bobcats:
      For each bobcat i = 1 to N
          Randomly initialize position X_i within bounds [LB, UB]
          Evaluate fitness (FX_i) using:
              F(X_i) = w_1 * (1/Energy(X_i)) + w_2 * (Utilization(X_i)) + w_3 * (Security(X_i))
3. Identify best solution:
       X* = argmax(F(X_i)) over all i
4. Iteration loop (for t = 1 to {Max}_{Iter}):
      Update α = α_0 * (1 − t / Max_Iter)    // Gradually reduce exploration
      For each bobcat i in population:
          - Select prey (X^\ast) from better individuals (elitism-based selection)
          - Compute directional vector:
              D_i = X* − X_i
Update position with directional movement:
       X_i_new = X_i + α * rand() * D_i
          - Apply boundary-aware adjustment:
                For each dimension j:
       X_i_new[j] < LB[j]
       X_i_new[j] = LB[j] + rand() * (UB[j] − LB[j])
Else X_i_new[j] > UB[j]
       X_i_new[j] = UB[j] − rand() * (UB[j] − LB[j])
          - Evaluate new fitness:
              F(X_i_new) = w_1 * (1/Energy(X_i_new)) + w_2 * (Utilization(X_i_new)) + w_3 * (Security(X_i_new))
          - Update X_i F(X_i_new) > F(X_i)
- Update global best:
```

```
If any F(X_i) > F(X_*), set X_* = X_i
5. END LOOP        X*
6. Output X* as the optimal VM allocation.
7. Evaluate results:
Energy Consumption (kWh), Makespan (s), Execution Time (s), SLA Violation Rate (%), Latency
(ms), and Fitness Convergence over iterations
END
```

Where the parameter α controls the exploration-to-exploitation balance; r is a uniform random factor that promotes diversification, and the fitness function integrates both energy consumption from the DVFS-based modeling and co-residence security metrics. Solutions violating CPU or memory limits are corrected before evaluation to ensure feasibility. Figure 3 illustrates the flowchart of the proposed DMBABOA approach for energy-efficient VM allocation, which involves both exploration and exploitation phases.

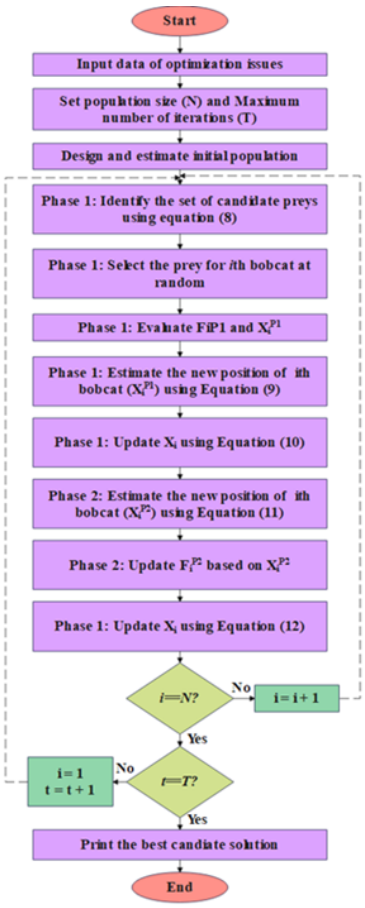

Figure 3. Flowchart of proposed DMBABOA approach for energy-efficient VM allocation

4. RESULTS AND DISCUSSION

A sequence of implementation was performed for validating the performance and effectiveness of the proposed container scheduling framework. The model implementation was performed using Python 3.3 on a system configured using Intel Core i5 CPU and 6 GB RAM under Windows 10 OS. In this study, a synthetic workload was generated using the CloudSim simulation toolkit to evaluate the performance of the proposed energy-efficient, security-aware VM-allocation algorithm. The workload was designed to mimic real-world cloud environments with the following characteristics.

- Task size: each task was assigned a random instruction length uniformly distributed from 1,000 to 10,000 million instructions (MI), representing diverse computational complexities.
- Data size: input and output data sizes for the tasks varied between 10 MB and 500 MB, reflecting a range of cloud-service demands.
- Task arrival pattern: a Poisson distribution was used to simulate task-arrival times, capturing the stochastic and bursty nature of real-world user requests.

- Simulation scale: experiments were conducted with workloads ranging from 10 to 1,000 tasks scheduled across a simulated infrastructure comprising 10 data centers and 15 VMs.
- Synthetic nature: although real-world datasets were used, the synthetic workload accurately captured realistic variability in task arrival, size, and resource demands.

This simulation setup ensured a controlled yet realistic environment for evaluating the energy efficiency of the proposed optimization algorithm, security-aware placement capability, and constraint-handling performance. Table 1 lists the key configurations of VM used in simulations such as experimental constraints, size, and resource limits. The processing VM dynamically adjusts according to application-request demands. Clouds vary randomly from lowest to highest based on demand and are arranged as consistently distributed variables that simulate independent changes in various program applications. Moreover, VM changes during the experiment occur intermittently at a fixed 60 s time interval.

Table 1. Key configuration of VMs used in simulations, including experimental constraints, size, and resource limits

Parameters	Default value
MIPS	2000
Bandwidth	1GB
Size of VM	2.5GB
Processing element	2
Utilization of CPU	0% to 100%

4.1. Performance analysis

This section signifies an analysis of the simulation findings for estimating the performance of the proposed DMBABOA. A comparative analysis of four scheduling algorithms was conducted to determine the significance of the proposed approach. The DMBABOA approach was compared with the coyote optimization algorithm (COA), fox optimization algorithm (FOA), grey wolf optimization algorithm (GOA), and conventional BOA.

Table 2 presents the performance analysis of energy consumption for 100-1,000 tasks across the five scheduling algorithms. The proposed DMBABOA exhibits the lowest energy consumption in each scenario, outperforming COA, FOA, GOA, and conventional BOA. This improvement results from its directional movement and boundary-aware mechanisms, which reduce the number of active PMs by effectively consolidating VMs. As the task count increases, energy consumption naturally increases; however, DMBABOA manages this increase more efficiently. For example, for 1,000 tasks, DMBABOA consumes 2,732.99 KWh, whereas COA consumes 3,621.35 KWh. The significant energy savings highlight the effectiveness of this method for power-aware VM placement, contributing to both cost reduction and environmental sustainability in datacenters.

Table 2. Performance analysis of energy consumption (Kwh) across number of tasks comparing the proposed method with existing methods

Number of tasks	COA	FOA	GOA	BOA	Proposed DMBABOA
100	588.76	543.27	516.43	487.32	441.88
300	1387.41	1292.15	1216.52	1140.79	1056.72
500	2183.92	2024.18	1912.63	1796.54	1648.38
700	2926.17	2724.55	2578.92	2412.17	2216.47
1,000	3621.35	3365.91	3184.47	2973.93	2732.99

Figure 4 shows the performance analysis of the makespan for task loads ranging from 100 to 1,000. The makespan refers to the total time required to complete all scheduled tasks. The graph demonstrates that DMBABOA consistently achieved the lowest makespan values across all task sizes. This reduction indicates efficient utilization of computing resources and minimized scheduling delays. The steep increase in makespan for traditional methods (such as COA and FOA) highlights their limitations in handling large workloads. Meanwhile, DMBABOA maintains low overhead even under pressure, reflecting superior dynamic scheduling and load-balancing capabilities. This performance enhancement ensures faster response and improved QoS in real-time cloud environments.

Table 3 presents the performance analysis of the average execution time across different tasks using the five algorithms. Execution time is critical for determining cloud system responsiveness. The DMBABOA consistently outperforms others, achieving the lowest execution times across all task sizes from 23.02 s at 100 tasks to 142.67 s at 1,000 tasks. This efficiency is attributed to its intelligent scheduling mechanism that

prioritizes faster VM allocation and resource availability. The table also shows that as the count tasks enhances, a gap between DMBABOA and the other algorithms becomes more pronounced. This highlights the scalability and ability to maintain performance under heavy loads. The reduced execution time also contributed to the enhanced user satisfaction and overall system throughput.

Figure 5 presents a graphical representation of the SLA violation rate versus the number of tasks, emphasizing the improved reliability achieved using DMBABOA. The figure illustrates the SLA violation rate under increasing task loads, where an SLA violation occurs when tasks are not completed within a stipulated response or execution time. DMBABOA maintained the lowest SLA violation rate among all algorithms tested, reflecting its efficient and adaptive resource-management strategy. With traditional algorithms, violations increase significantly as workload increases, indicating poor scalability and unpredictable task handling. The proposed method ensures balanced load distribution and strategic VM placement, reducing latency and missed deadlines. This consistency in SLA adherence makes DMBABOA highly suitable for service-critical and time-sensitive cloud applications.

Table 4 presents the performance analysis of the latency across different task counts, comparing the proposed method with existing methods. The results show that DMBABOA achieves the lowest latency in all workloads, ranging from 29.17 ms at 100 tasks to 105.28 ms at 1,000 tasks. Latency represents the time delay between a user's request and the system's response. Lower latency values indicate faster and more reliable service provision. The other algorithms exhibit a sharper increase in latency as task count rises, demonstrating poor adaptability and bottlenecks in communication and computation. In contrast, the proposed method effectively reduces queue time and ensures better task-to-resource mapping. Directional- and boundary-aware strategies contribute significantly to reducing processing delays and improving response efficiency, which are crucial for interactive and real-time cloud services.

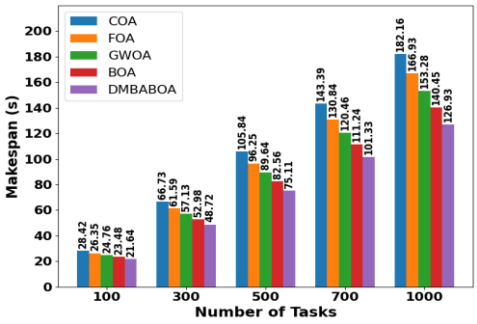

Figure 4. Makespan (s) across the number of tasks, comparing the proposed method with the existing methods

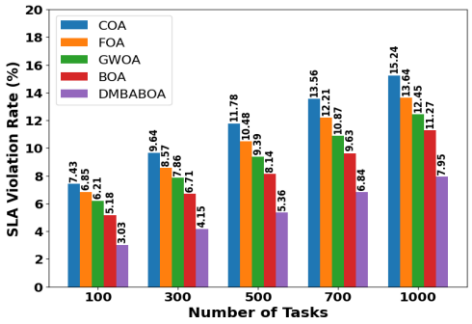

Figure 5. SLA violation rate versus number of tasks, emphasizing improved reliability achieved using DMBABOA

Table 3. Performance analysis of average execution time(s) across different tasks for comparison of the proposed method with existing methods

Number of tasks	COA	FOA	GOA	BOA	Proposed DMBABOA
100	30.21	28.47	26.89	25.14	23.02
300	73.65	69.84	64.15	58.93	54.26
500	118.36	110.27	102.86	95.41	87.65
700	160.48	149.16	137.94	126.03	115.42
1000	198.92	183.57	170.42	156.78	142.67

Table 4. Performance analysis of latency (ms) across different tasks for proposed method with existing methods

Number of tasks	COA	FOA	GOA	BOA	Proposed DMBABOA
100	38.54	36.17	34.26	32.45	29.17
300	64.28	60.92	56.14	52.47	48.36
500	91.43	85.16	78.72	73.28	67.49
700	115.89	108.36	100.27	94.38	86.45
1,000	142.36	133.29	123.16	115.34	105.28

Table 5 presents the convergence behavior of the optimization approaches by tracking their fitness values over 50 iterations. DMBABOA exhibited rapid and most reliable convergence, accomplishing a fitness value of 0.97, whereas the others converged more slowly and to lower values. This indicates that DMBABOA quickly identifies near-optimal solutions with fewer iterations, thereby improving

computational efficiency. Faster convergence is essential in real-time applications where decisions must be made rapidly. The table also validates the strength of the proposed method in escaping local optima and achieving global optimization through hybrid strategies. The steady improvement in fitness values across iterations shows that the model adapts dynamically to workload fluctuations.

Table 6 summarizes the statistical results, including execution time, energy consumption, SLA violations, and the mean and standard deviation of the proposed DMBABOA approach compared with existing approaches. The low variance values indicate that the performance of the proposed method remains stable across multiple simulation runs. Narrow confidence intervals confirm the reliability and repeatability of the results. For example, the mean energy consumption was 2,462.73 KWh with a tight confidence interval of ±5.83, demonstrating that the outcomes consistently align with expected values. This robustness under varying workload scenarios makes DMBABOA a dependable choice for energy-efficient and SLA-compliant VM allocation in real-world cloud infrastructure.

Figure 6 illustrates the graphical representation of fitness versus number of iterations, demonstrating DMBABOA's superior convergence stability and consistently improved performance. Although COA achieves the highest fitness, it exhibits noticeable fluctuation, indicating possible instability. FOA and GWOA maintained higher fitness values but showed moderate variation. Notably, DMBABOA demonstrates the most stable and smooth convergence, gradually improving without abrupt jumps. Despite achieving a comparatively lower fitness value, its consistent progression highlights robustness and reliability. This stability makes DMBABOA well-suited for energy-aware and constraint-sensitive environments, such as secure VM allocation in CC.

Table 5. Performance analysis of fitness-value convergence across different iteration numbers, comparing the proposed method with existing methods

Iteration	COA	FOA	GOA	BOA	Proposed DMBABOA
10	0.72	0.76	0.78	0.82	0.87
20	0.79	0.81	0.84	0.87	0.91
30	0.83	0.85	0.87	0.89	0.94
40	0.85	0.87	0.89	0.91	0.96
50	0.86	0.88	0.90	0.92	0.97

Table 6. Statistical results including execution time, energy consumption, SLA violations, and mean and standard deivation of the proposed DMBABOA method compared with existing methods

Method	Execution time (s)	Energy consumption (kWh)	SLA violations (%)	Mean ± Std. dev (Fitness)
MV-PESC	148.62	61,520	3.41	0.87 ± 0.02
GPSO	142.17	59,870	2.96	0.89 ± 0.02
FFNN	139.48	58,630	2.75	0.91 ± 0.01
BOA	134.92	57,220	2.41	0.94 ± 0.01
Proposed DMBABOA	128.36	53,200	1.87	0.97 ± 0.01

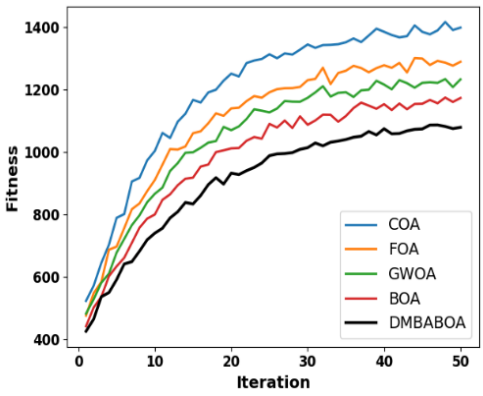

Figure 6. Graphical representation of fitness versus number of iterations, demonstrating DMBABOA's superior convergence stability and consistently improved performance

4.2. Comparative analysis

Table 7 specifies the comparative analysis of the proposed with existing approaches using power in Watts ((P) (W)), active physical machines (APM), and execution time (s), based on the number of VMs and

users. The existing methods, such as MV-PESC [26] and energy efficiency ratio (EER)-VMP [27], were evaluated and compared with the proposed method. The comparative analysis was validated using various performance metrics, including power (W), APM, and execution time, based on the number of VMs and users.

Table 7. Comparative analysis of proposed with existing approaches using P (W), APM, and execution time (s) based on the number of VMs and users

Method	VM	Users	P (W)	APM	Execution time (s)
MV-PESC [26]	100	20	7.62E+3	25	2.09
	400	80	2.42E+4	45	6.44
	800	160	4.12E+4	74	12.40
	1,200	200	5.78E+4	104	19.24
EER-VMP [27]	100	20	7.77E+3	31	3.12
	200	40	1.41E+4	41	17.69
	400	80	2.83E+4	80	20.28
	600	120	3.57E+4	100	107.59
	800	160	4.26E+4	109	123.17
	1,000	200	5.58E+4	219	145.74
Proposed DMBABOA	100	20	7.57E+4	25	2.04
	200	40	1.32E+4	36	13.49
	400	80	2.64E+4	73	16.38
	600	120	3.42E+4	92	101.38
	800	160	4.12E+4	101	112.38
	1,200	200	5.32E+4	210	134.48

4.3. Discussion

Conventional approaches to VM allocation in cloud environments are often constrained by excessive energy usage, poor load distribution, and frequent violations of SLA, particularly under fluctuating workloads. A key limitation of these methods is their inability to adapt to changing demands or respect boundary conditions related to server capacity. The proposed DMBABOA addresses these challenges by integrating directional movement and boundary-aware mechanisms, enabling intelligent, constraint-aware scheduling decisions. This leads to more balanced resource utilization and a notable reduction in unnecessary VM migrations. In contrast to methods that rely solely on CPU metrics, DMBABOA enables flexible adjustments while maintaining resource usage within permissible limits. Experimental analysis shows that DMBABOA consistently delivers improved performance across all evaluation metrics, including reduced energy consumption, faster execution times, lower latency, and shorter makespan, when compared with optimization techniques such as COA, FOA, GOA, and conventional BOA. The algorithm also demonstrated strong convergence behavior and minimal performance variation across multiple trials, confirming its suitability for real-time, scalable cloud infrastructures. Overall, the DMBABOA offers an effective and dependable solution for energy-aware and secure VM scheduling in modern data centers. From an electrical engineering standpoint, the inclusion of DVFS-based power modeling directly supports hardware-level energy optimization. By correlating CPU utilization with power draw, DMBABOA facilitates adaptive scheduling aligned with processor frequency scaling. This has practical implications for edge-computing systems, smart grids, and embedded platforms, where real-time energy awareness and thermal stability are essential. The proposed framework can also be extended to heterogeneous edge devices to enable energy-regulated workload distribution.

5. CONCLUSION

VM allocation in an IaaS cloud environment is classified as a non-deterministic polynomial (NP)-hard problem, requiring advanced optimization strategies for efficient resource allocation. To address this, DMBABOA is proposed in this research to explore the global solution space and identify an optimal VM scheduling that minimizes energy consumption and reduces SLA violations. The fitness function in DMBABOA make sures that VMs are allocated to energy-efficient PMs capable of managing higher workloads with minimal power usage. By incorporating directional-movement and boundary-aware strategies, the algorithm enhances convergence precision while maintaining feasibility within constrained environments. Simulation results demonstrate that DMBABOA significantly lowers average energy consumption and SLA violations compared with existing optimization algorithms. The current implementation primarily considers workload variance based on CPU million instructions per second (MIPS) utilization. The proposed approach contributes to broader green-computing and energy-aware scheduling initiatives. By optimizing VM placement through directional search and boundary-aware mechanisms, DMBABOA aligns with sustainable computing objectives applicable to edge AI, distributed cloud, and environmentally conscious data-center operations. Future extensions of this research will incorporate

additional resource metrics, such as memory and bandwidth, while also evaluating the communication overhead introduced by VM migration and inter-data center coordination.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Nida Kousar Gouse	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓	✓
Gopala Krishnan Chandrasekaran		✓				✓		✓	✓	✓	✓	✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study. Annotated simulation scripts or CloudSim configurations can be seen in GitHub at https://github.com/nidakousar11/VmAllocation/blob/main/VMAllocation.zip.

REFERENCES

[1] N. K. A. Nemirajaiah and C. K. Raju, "Securing virtual machines using cloning in cloud services," *Engineering, Technology and Applied Science Research*, vol. 15, no. 2, pp. 20770–20775, Apr. 2025, doi: 10.48084/etasr.9391.
[2] M. Radi, A. A. Alwan, and Y. Gulzar, "Genetic-based virtual machines consolidation strategy with efficient energy consumption in cloud environment," *IEEE Access*, vol. 11, pp. 48022–48032, May 2023, doi: 10.1109/ACCESS.2023.3276292.
[3] R. Chen, B. Liu, W. W. Lin, J. P. Lin, H. W. Cheng, and K. Q. Li, "Power and thermal-aware virtual machine scheduling optimization in cloud data center," *Future Generation Computer Systems*, vol. 145, pp. 578–589, Aug. 2023, doi: 10.1016/j.future.2023.03.049.
[4] S. Singhal, N. Gupta, P. Berwal, Q. N. Naveed, A. Lasisi, and A. W. Wodajo, "Energy efficient resource allocation in cloud environment using metaheuristic algorithm," *IEEE Access*, vol. 11, pp. 126135–126146, Nov. 2023, doi: 10.1109/ACCESS.2023.3330434.
[5] S. Durairaj and R. Sridhar, "Coherent virtual machine provisioning based on balanced optimization using entropy-based conjectured scheduling in cloud environment," *Engineering Applications of Artificial Intelligence*, vol. 132, Jun. 2024, doi: 10.1016/j.engappai.2024.108423.
[6] E. Suganthi and F. K. M. Selvi, "Cloud computing by implementing state and random-based virtual machine load balancing model," *International Journal of Intelligent Engineering and Systems*, vol. 17, no. 3, pp. 92–101, May 2024, doi: 10.22266/ijies2024.0630.08.
[7] P. Udayasankaran and S. J. J. Thangaraj, "Energy efficient resource utilization and load balancing in virtual machines using prediction algorithms," *International Journal of Cognitive Computing in Engineering*, vol. 4, pp. 127–134, Jun. 2023, doi: 10.1016/j.ijcce.2023.02.005.
[8] E. Al-Masri, A. Souri, H. Mohamed, W. Yang, J. Olmsted, and O. Kotevska, "Energy-efficient cooperative resource allocation and task scheduling for internet of things environments," *Internet of Things*, vol. 23, Oct. 2023, doi: 10.1016/j.iot.2023.100832.
[9] Z. Khodaverdian, H. Sadr, S. A. Edalatpanah, and M. Nazari, "An energy aware resource allocation based on combination of CNN and GRU for virtual machine selection," *Multimedia Tools and Applications*, vol. 83, no. 9, pp. 25769–25796, Aug. 2024, doi: 10.1007/s11042-023-16488-2.
[10] A. Ghasemi and A. Keshavarzi, "Energy-efficient virtual machine placement in heterogeneous cloud data centers: a clustering-enhanced multi-objective, multi-reward reinforcement learning approach," *Cluster Computing*, vol. 27, no. 10, pp. 14149–14166, Jul. 2024, doi: 10.1007/s10586-024-04657-3.
[11] A. Naik and K. Sooda, "Efficient job scheduling in cloud environments using reinforcement learning actor-critic models," *Engineering, Technology and Applied Science Research*, vol. 14, no. 5, pp. 16559–16564, Oct. 2024, doi: 10.48084/etasr.8104.
[12] A. Gopu *et al.*, "Energy-efficient virtual machine placement in distributed cloud using NSGA-III algorithm," *Journal of Cloud Computing*, vol. 12, no. 1, 2023, doi: 10.1186/s13677-023-00501-y.

[13] B. Magotra, D. Malhotra, and A. K. Dogra, "Adaptive computational solutions to energy efficiency in cloud computing environment using VM consolidation," *Archives of Computational Methods in Engineering*, vol. 30, no. 3, pp. 1789–1818, Nov. 2023, doi: 10.1007/s11831-022-09852-2.
[14] S. Singh and R. Kumar, "Energy efficient optimization with threshold based workflow scheduling and virtual machine consolidation in cloud environment," *Wireless Personal Communications*, vol. 128, no. 4, pp. 2419–2440, Oct. 2023, doi: 10.1007/s11277-022-10049-w.
[15] W. Hua, P. Liu, and L. Huang, "Energy-efficient resource allocation for heterogeneous edge-cloud computing," *IEEE Internet of Things Journal*, vol. 11, no. 2, pp. 2808–2818, Jan. 2024, doi: 10.1109/JIOT.2023.3293164.
[16] R. Keshri and D. P. Vidyarthi, "Energy-efficient communication-aware VM placement in cloud datacenter using hybrid ACO–GWO," *Cluster Computing*, vol. 27, no. 9, pp. 13047–13074, Jun. 2024, doi: 10.1007/s10586-024-04623-z.
[17] A. Tarafdar, S. Sarkar, R. K. Das, and S. Khatua, "Power modeling for energy-efficient resource management in a cloud data center," *Journal of Grid Computing*, vol. 21, no. 1, Feb. 2023, doi: 10.1007/s10723-023-09642-5.
[18] R. Mandal *et al.*, "MECpVmS: an SLA aware energy-efficient virtual machine selection policy for green cloud computing," *Cluster Computing*, vol. 26, no. 1, pp. 651–665, Jul. 2023, doi: 10.1007/s10586-022-03684-2.
[19] Z. Ma, D. Ma, M. Lv, and Y. Liu, "Virtual machine migration techniques for optimizing energy consumption in cloud data centers," *IEEE Access*, vol. 11, pp. 86739–86753, Aug. 2023, doi: 10.1109/ACCESS.2023.3305268.
[20] S. Sunil and S. Patel, "Energy-efficient virtual machine placement algorithm based on power usage," *Computing*, vol. 105, no. 7, pp. 1597–1621, Feb. 2023, doi: 10.1007/s00607-023-01152-2.
[21] N. K. Sharma, S. Bojjagani, Y. C. A. P. Reddy, M. Vivekanandan, J. Srinivasan, and A. K. Maurya, "A novel energy efficient multi-dimensional virtual machines allocation and migration at the cloud data center," *IEEE Access*, vol. 11, pp. 107480–107495, Sep. 2023, doi: 10.1109/ACCESS.2023.3320729.
[22] W. Yao, Z. Wang, Y. Hou, X. Zhu, X. Li, and Y. Xia, "An energy-efficient load balance strategy based on virtual machine consolidation in cloud environment," *Future Generation Computer Systems*, vol. 146, pp. 222–233, Sep. 2023, doi: 10.1016/j.future.2023.04.014.
[23] L. Cao, R. Li, X. Ruan, and Y. Liu, "Defending against co-residence attack in energy-efficient cloud: an optimization based real-time secure VM Allocation Strategy," *IEEE Access*, vol. 10, pp. 98549–98561, Sep. 2022, doi: 10.1109/ACCESS.2022.3206021.
[24] A. R. Madireddy and K. Ravindranath, "Dynamic virtual machine relocation system for energy-efficient resource management in the cloud," *Concurrency and Computation: Practice and Experience*, vol. 35, no. 3, Nov. 2023, doi: 10.1002/cpe.7520.
[25] K. Ajmera and T. K. Tewari, "Energy-efficient virtual machine scheduling in IaaS cloud environment using energy-aware green-particle swarm optimization," *International Journal of Information Technology*, vol. 15, no. 4, pp. 1927–1935, Mar. 2023, doi: 10.1007/s41870-023-01227-5.
[26] A. K. Singh, S. R. Swain, and C. N. Lee, "A metaheuristic virtual machine placement framework toward power efficiency of sustainable cloud environment," *Soft Computing*, vol. 27, no. 7, pp. 3817–3828, Apr. 2023, doi: 10.1007/s00500-022-07578-8.
[27] S. R. Swain, A. Parashar, A. K. Singh, and C. N. Lee, "An intelligent virtual machine allocation optimization model for energy-efficient and reliable cloud environment," *Journal of Supercomputing*, vol. 81, no. 1, Dec. 2025, doi: 10.1007/s11227-024-06734-1.
[28] Z. Mahmoodabadi and M. N.-Baygi, "An approximation algorithm for virtual machine placement in cloud data centers," *Journal of Supercomputing*, vol. 80, no. 1, pp. 915–941, Jul. 2024, doi: 10.1007/s11227-023-05505-8.
[29] R. Zhong, Y. Xu, C. Zhang, and J. Yu, "Leveraging large language model to generate a novel metaheuristic algorithm with CRISPE framework," *Cluster Computing*, vol. 27, no. 10, pp. 13835–13869, Jul. 2024, doi: 10.1007/s10586-024-04654-6.

BIOGRAPHIES OF AUTHORS

Nida Kousar Gouse from Bengaluru, Karnataka, holds an M.Tech. in Computer Science and Engineering. She is currently pursuing her Ph.D. at GITAM University, with her research focused on cloud computing and cybersecurity. Her work delves into advanced methodologies and innovative technologies to drive transformative solutions in these critical areas. Accomplished author, she has written a book on cloud computing, contributing to the academic and professional understanding of this ever-evolving field. She can be contacted at email: nkousar@gitam.in.

Gopala Krishnan Chandrasekaran received his Ph.D. in Computer Science and Engineering from St. Peter's Institute of Higher Education and Research, Chennai, India. He received his M.E. degree in Computer science and Engineering from Anna University Tirunelveli, India and B.E. degree in Computer Science and Engineering from Madurai Kamaraj University, India. His areas of interest are mobile computing, operating systems, computer networks, computer graphics and multi core architecture. He has published many papers in international conferences and international journals in various fields of advanced technologies. He is a life time member of ISTE. He can be contacted at email: gchandra@gitam.edu.