Journal of Computer Science

Research Article

Masterpiece Optimization Algorithm-Based Priority-Aware Load Balancing Strategy for Cloud Data Centers

S Vijaykumar and Shanker Chandre

School of Computer Science & Artificial Intelligence, SR University, Telangana, 506371, India

Article history
Received: 11-11-2025
Revised: 23-03-2026
Accepted: 12-06-2026

Corresponding Author:
S Vijaykumar
School of Computer Science & Artificial Intelligence, SR University, Telangana, 506371, India
Email: vijays.anurag@gmail.com

Abstract: Cloud Computing (CC) is one of the widely used technologies due to its advanced features such as pay-per-use, scalability, and flexibility. The primary objective of CC is to allow users to access and purchase cloud services that are on demand through internet-based applications. Efficient load-balancing in the cloud faces challenges of high-dimensional state spaces and scalability with increasing tasks. To solve this problem, the Masterpiece Optimization Algorithm (MOA) with a priority constraint is employed for load-balancing according to the tasks efficiently. The MOA is integrated with a priority-based cost function to enhance the task scheduling process by introducing a multi-dimensional approach for load balancing. The priority-based framework helps the scheduler to dynamically recalibrate workloads. The experimental results achieve a total energy consumption of 39.8 W and an average CPU resource utilization of 99.54%, which is better than the existing algorithms, such as the hybrid Particle Swarm Grey Wolf Optimization (PSGWO) algorithm.

Keywords: Cloud Computing, Energy Efficiency, Load-Balancing, Masterpiece Japanese Pufferfish Optimization, Priority, Task Scheduling

Introduction

Cloud Computing (CC) is a widely used, cutting-edge method to compute and store huge amounts of data with minimum cost, where the data owners utilize their data from the cloud server. In dynamic environments, it is crucial to allocate resources efficiently on physical servers, which directly impacts resource utilization and the overall performance of cloud services (Nagarajan et al., 2025). However, when the user demand for cloud services increases, and data centers scale up their resources, it also increases the energy consumption significantly, leading to higher operational costs (Samha, 2024; Javaheri et al., 2024). The various resources requested by the users have diverse Input-Output (I/O) memory, network requirements, and processing. According to these constraints, the resources are allocated for each task in CC (Fang et al., 2023; Hu et al., 2023; Liu et al., 2023). Since often changes in the user demand make the resource allocation model difficult, the results lack optimal resource allocation for the specific task with high energy consumption (Shi, 2024). To solve these resource allocation issues, load-balancing is performed per task appropriately before they begin processing (Zhanuzak et al., 2024; Prathiba and Sankar, 2024).

Generally, the cloud computing efficiency highly depends on the task scheduler's ability to maximize the performance by allocating the appropriate virtual resources (Rathinam et al., 2024; Akinola et al., 2025). The inappropriate load-balancing also affects various parameters such as energy consumption and makespan, which consumes more energy and has a high execution time, leading to an impact on the quality of service (Khan, 2024; Ghafir et al., 2024). The conventional methods used for balancing workloads in cloud computing fail to consider the distributed and automated nature of the cloud, which makes the model difficult to select optimal resources. In existing research, several methods, namely, greedy approaches, approximation algorithms, reinforcement learning approaches, and bio-inspired optimization algorithms, are used for allocating resources according to the tasks (Krishna and Vali, 2025). However, these existing load-balancing algorithms have several limitations that affect the selection of optimal resources with less energy consumption. Recently, metaheuristic optimization algorithms such as Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Whale Optimization Algorithm (WOA) have been widely

utilized in CC for load-balancing (Hayyolalam and Özkasap, 2025). To ensure Energy Efficient Load Balancing (EELB) in this research, the metaheuristic optimization is integrated with a reinforcement learning model to balance the load effectively.

However, the high-dimensional search space if the number of tasks increases, it directly increases the complexity of assigning tasks to Virtual Machines (VMs), which makes it difficult to identify optimal resource allocations. As a result, load imbalance occurs where some of the VMs become overloaded while others remain underutilized, which leads to high energy consumption and increased response time. Additionally, existing conventional metaheuristic algorithms utilized for load-balancing face limitations due to premature convergence and limited search diversity, resulting in suboptimal task scheduling decisions (Tarafdar et al., 2023). This leads to a decrease in the overall system throughput, and Quality of Service (QoS) is negatively affected in dynamic cloud environments. To overcome these problems, a Masterpiece Optimization Algorithm (MOA) model is proposed to efficiently balance the workloads across the resources in the cloud. The key contributions of this research are as follows:

- The proposed load-balancing model optimizes multiple objectives, such as cost minimization, energy efficiency, and QoS, to balance the load for the specific tasks effectively in the cloud to avoid overutilization and underutilization
- The MOA algorithm is used for fine-tuning to prevent selecting suboptimal resources to ensure efficient load-balancing in the cloud with less energy consumption

Literature Review

In this section, the advantages and limitations of the existing load-balancing model are analyzed and discussed. Al Reshan et al. (2023) developed a hybrid load-balancing solution for CC by combining Grey Wolf Optimization (GWO) and PSO. This hybrid approach achieved a fast convergence and globally optimal load-balancing among virtual machines that ensured a continuous server operation. The implementation of GWO was straightforward because it relied on only a few control parameters, and it made it simple to configure. By integrating PSO and GWO, the hybrid method enhanced convergence speed and enabled a system to reach optimized results more efficiently. However, the hybrid PSO-GWO approach was likely to prematurely converge to local optima, especially in complex or dynamic cloud environments, which limited its effectiveness in handling highly variable workloads.

Kaviarasan et al. (2023) presented an optimization-inspired algorithm to enhance load-balancing. The Improved Lion Optimization Algorithm (ILOA) was found to have faster convergence for finding an optimal result in a larger search space. To address the inefficient load-balancing issue in cloud environments, an ILOA was utilized for balancing the loads efficiently and reducing latency and response time. However, the effectiveness of ILOA in handling an increasing number of virtual machines and dynamic workloads required validation for scalability.

Singhal et al. (2024) produced a Rock Hyrax Optimization (RHO) based load-balancing algorithm that addressed local maxima and power efficiency issues through QoS parameters. The RHO distributed a job between various virtual machines depending on availability and current load, which reduced total makespan time and enhanced energy efficiency. The RHO algorithm efficiently managed a resource, even in geographically scattered data centers. The RHO performed well in both jobs and virtual machines, both statically and dynamically. However, the RHO aimed to avoid local maxima that suffered from premature convergence in complex and dynamic cloud environments.

Siddesha et al. (2022) presented a Reinforcement Learning (RL) based load-balancing method that was utilized to maximize the reward mechanism and design the policy for task scheduling actions. Task scheduling helped to obtain efficient task allocation and minimize the energy consumption, and the reinforcement approach was utilized to balance load and resource efficiency for better performance of the model. However, the RL method struggled to adjust to changes in the cloud, such as varying VM capabilities, which potentially affected the efficiency of task allocation and energy consumption.

Khan et al. (2022) implemented a hybrid model, which was a combination of Particle Swarm and Grey Wolf Optimization (PSGWO) algorithms, to identify the most suitable VM for the tasks. A task scheduling model was used to calculate task execution time and monitor VM status during the execution to enhance efficient resource utilization. Task scheduling enhanced load-balancing, reduced task execution time, and increased adaptability in dynamic conditions to improve performance. However, the PSGWO method caused computation overhead due to limited resources, leading to low efficiency of the model.

Haris and Zubair (2025) developed a Battle Royale Deep Reinforcement Learning (BRDRL) based on an effective load-balancing model in a cloud system. The developed BRDRL model was a combination of BR optimization and the DRL algorithm, which was employed to enhance load-balancing. The tasks were scheduled using the round-robin scheduling algorithm, and constraints such as cost, load balance, and makespan were considered for efficient load-balancing. Additionally, the selection of optimal underutilized VM was performed by the BRO method, whereas the transferring of tasks to the VM was done by DRL methods, respectively.

Verma (2025) presented a Cost-Energy (CE)-aware load-balancing model based on an improved Spider Monkey Optimization (SMO) in a cloud environment. The presented CE-SMO model was utilized for dynamic cloud environments to adapt to the varying workloads and manage load information. However, the presented CE-SMO model had high computation overhead due to its dynamic task reassignment and constant monitoring.

From the above literature review, the existing load-balancing model is used to improve task scheduling and resource utilization and enhance energy efficiency in cloud computing environments. The hybrid metaheuristic approaches, such as PSO-GWO and RHO, face limitations such as premature convergence and scalability limitations in dynamic workloads. However, the reinforcement learning-based methods enhance adaptive task allocation and energy efficiency, but they fail to adapt to the changes in VM characteristics. Moreover, the existing load-balancing techniques face challenges related to scalability, computational overhead, and robustness in highly dynamic cloud environments.

Thus, an MOA-based load-balancing algorithm with a priority constraint is proposed to address the above limitations. By incorporating adaptive search mechanisms that efficiently maintain a dynamic balance between exploration and exploitation, the MOA is prevented from premature convergence. Moreover, the priority-based task allocation further enhances scalability by efficiently handling varying workloads and heterogeneous VM conditions. Additionally, the proposed model with a priority-based strategy reduces computational overhead through optimized decision-making, which leads to improved energy efficiency and robust performance in dynamic cloud environments. Table 1 presents a summary of existing load-balancing research works.

Table 1: Summary of existing load-balancing research works

Author and Year	Method	Optimization Type	Key Objectives	Strengths	Limitations / Research Gap
Khan and Santhosh (2022)	PSGWO	Hybrid Metaheuristic (PSO + GWO)	Efficient VM selection, reduced execution time	Improved adaptability under moderate workloads	High computational overhead; limited scalability in large-scale environments
Siddesha et al. (2022)	RL-based Scheduling	Reinforcement Learning	Energy minimization, adaptive task allocation	Reward-driven dynamic scheduling	Struggles with heterogeneous VM capabilities; performance instability
Al Reshan et al. (2023)	PSO-GWO Hybrid	Swarm-based Hybrid	Fast convergence, global load-balancing	Simple configuration; improved convergence speed	Premature convergence; limited robustness in highly dynamic workloads
Kaviarasan et al. (2023)	ILOA	Bio-inspired Optimization	Reduced latency and response time	Faster convergence in larger search spaces	Scalability concerns with increasing VMs and workload variability
Singhal et al. (2024)	Rock Hyrax Optimization	Nature-Inspired Optimization	Energy efficiency, reduced makespan	QoS-aware distribution; effective in distributed data centers	Susceptible to local optima; premature convergence in complex scenarios
Proposed Work	MOA-based Priority-Aware Load-balancing	Adaptive Bio-inspired Multi-objective Optimization	Minimize energy, make span, response time; improve QoS and scalability.	Balanced exploration–exploitation via circular search; priority-aware scheduling; overload-aware repair; statistically validated performance	— (Addresses premature convergence, scalability, and adaptability limitations)

Materials and Methods

The system model of the cloud and the information about VMs and tasks assigned to the resources are depicted in this section. Let us assume an application that is running on a cloud environment where there are multiple users simultaneously. In this application, at a time, n tasks are submitted by users, which are denoted as $\{ta_1, ta_2, \ldots, ta_n\}$, and m as Virtual Machines (VMs), which are available on the host server, which is represented as $\{Vm_1, Vm_2, \ldots, Vm_m\}$. In general, the number of tasks is greater than or equal to the number of available VMs, i.e., $n \geq m$.

The solution is one mapping where the tasks are

mapped to VMs, which is represented in Eq. 1:

$$Map_{\{i,j\}} = F(Ta_i, V_j), \exists \ (i = 1 \ to \ n \ and \ j = 1 \ to \ m) \qquad (1)$$

Where $Map_{\{i,j\}}$ denotes the mapping matrix, which is computed using the F function; and Ta_i and V_j represent tasks and VMs, respectively. This mapping equation is used to construct a mapping matrix that captures the interaction (similarity, relation, or transformation) between all input elements and reference features.

Initially, the users submit tasks to the scheduler through the application interface. Then, the load factor p of the total number of VM (m) is estimated by load balancers, assuming that the VMs are organized into groups based on their capacity and number of cores (Do et al., 2023). To provide a global system view, the load factor of each group is shared among all load balancers. If a VM within a group reaches its maximum load, which limits its ability to receive additional tasks, the tasks are assigned to another available VM in the same group. This ensures effective load-balancing within each group in the cloud environment.

Priority-Based Task Scheduling

In this research, the raw tasks are converted into a single priority score (P_i), which helps the proposed algorithm to determine the optimal placement of tasks onto VMs. The mathematical representation of the priority score for assigning tasks to VM is mentioned in Eq. 2:

$$P_i = \alpha 1 * L_{i_{norm}} + \alpha 2 * ET_{i_{norm}} + \alpha 3 * RU_{i_{norm}} \qquad (2)$$

Where $L_{i_{norm}}$ denotes the normalized length of the task, $ET_{i_{norm}}$ indicates normalized execution time; $RU_{i_{norm}}$ represents normalized resource utilization; and $\alpha 1$, $\alpha 2$, and $\alpha 3$ denote weighting factors that determine the importance of each metric. By using a priority-based cost function instead of a simple First-Come-First-Served (FCFS) approach, the proposed MOA is able to dynamically adapt to fluctuating cloud workloads. This ensures that the Service Level Agreement (SLA**)** violations are minimized and the VM resources are used efficiently. The distribution of workloads between VMs and assigning the tasks to the VMs efficiently enhances the overall system performance and stability. This workload-balancing technique reduces task completion time, increases system throughput, and enhances resource utilization, which results in a reliable CC system.

Load-balancing in the cloud refers to the process of distributing the workload optimally, which helps to utilize the resources of a VM efficiently for the scheduled tasks. Recently, load-balancing has become vital for several applications and is widely used in cloud computing services. Thus, this approach becomes significant, which enhances the cloud application performance where the number of resources is restricted in the cloud. This load-balancing is generally categorized into two kinds: Static and dynamic. Static techniques are widely used to distribute predictable loads that do not require foreknowledge of the current state of the system. At the same time, the dynamic technique distributes unpredictable loads by considering the current state of the system. However, the existing cloud load-balancing techniques have limitations such as random scheduling and dynamic reallocation, which lead to inefficient resource utilization and execution delays. Thus, an optimization-based load-balancing technique is proposed in this research to address limitations efficiently. Fig. 1 represents the proposed load-balancing framework in CC. The overall process of load-balancing is explained in this section as follows.

However, there are some limitations in existing load-balancing techniques, which are described below:

- The functions vary significantly and should not be treated equally for the cold start. Different functions have various “priorities” in terms of latency benefits, which are determined by factors including the frequency of calls, resource usage, initialization, and overhead
- Moreover, conventional scheduling strategies, such as FCFS or fair scheduling, fail to consider the priority of different functions and therefore result in suboptimal performance when scheduling the workloads

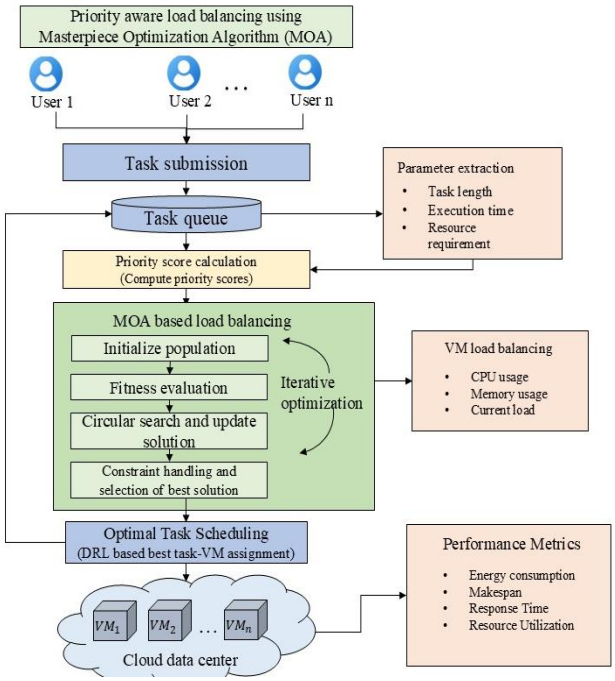

Fig. 1: Architecture of the proposed MOA-based priority-aware load-balancing framework for cloud environments

To address the aforementioned limitations, an optimization-based model is proposed in this research to enable efficient load-balancing in a cloud environment by prioritizing the tasks based on latency, execution time, and resource usage.

They formulate scheduling as an optimization problem to minimize latency and balance workloads across nodes. Thus, the dynamic approach adapts to changing workloads, ensuring high resource utilization and improved performance.

Masterpiece Optimization Algorithm (MOA)

Conventional algorithms such as PSO (Kumar et al., 2025) and Genetic Algorithms (GA), commonly used for load-balancing and resource allocation, often suffer from premature convergence, limited search diversity, and an imbalance between exploration and exploitation. Swarm-based methods, namely, ACO and GWO, may require extensive parameter tuning and struggle with constraint handling in dynamic environments. In contrast, the MOA offers a biologically inspired, adaptive search mechanism that dynamically balances exploration and exploitation using a circular search strategy. Its priority-aware evaluator function efficiently handles constraints and adapts well to real-world problems such as cloud load-balancing. MOA achieves faster convergence, better solution quality, and superior adaptability compared to conventional approaches.

The proposed MOA is a nature-inspired meta-heuristic algorithm that is developed based on the unique nest-building behavior of Japanese male pufferfish during mating. This proposed algorithm is inspired by male pufferfish behavior, which accurately builds intricate seabed nests to attract mates, and in CC, this natural strategy is applied to identify optimal solutions in load-balancing problems. The male pufferfish exhibits a remarkable behavior, spending approximately seven to nine days constructing a perfectly shaped nest on the seabed. Because of its detailed design and effort, this nest is often called a "masterpiece" or a "mysterious circle." To build this nest, the fish carefully gathers and arranges fine sand, both to create the nest's base and to cover the eggs for protection. After the eggs hatch, the nest is disturbed, and the sand is rearranged, so the male never reuses the same nest. Additionally, the fish must select an optimal spot where sufficient fine sand is available to construct and protect the nest. Once the nest is ready, the female lays her eggs inside it, and if they are well-covered and safe, they grow into baby fish.

In this MOA algorithm, the idea of finding the best spot for building a nest is turned into a strategy where the best spot with the finest sand is treated as the global optimum, the best possible solution to a problem. The algorithm then tries to find the best spot, just like the pufferfish does in real life. The proposed MOA algorithm involves three steps, namely:

Step 1 : Selecting the best point inside the nest
Step 2 : Estimation of water flow speed and direction
Step 3 : Generating and checking new directional flow points

Selecting the Best Point Inside the Nest

Since the puffer fish's masterpiece is circular, the relations produced in two consecutive dimensions follow the equation of a circle. It means that the relations in the even dimension depend on the odd dimension. During nest construction, the male pufferfish carefully gathers fine sand and deposits it in the peripheral dents/valleys, creating a structured pattern that attracts a mate. After spawning, the sand gathered in the nest's dents is moved by the water flow and covers the eggs effectively for protection. Therefore, in the algorithm provided and developed, some points are created on the imaginary circle perimeter as the valleys. Also, some search points are created around the centre of the nest inside the imaginary circle. The valleys are created using the following Eqs. 3 and 4:

$$x_{i_{new}}^{d=2m-1} = x_{i_{center}}^{d=2m-1} + r_i^d \times cos(k) \tag{3}$$

$$x_{i_{new}}^{d=2m} = x_{i_{center}}^{d=2m} + r_i^d \times sin(k) \tag{4}$$

Where $2m-1$ represents an odd direction, $2m$ denotes an even direction, k indicates the angle of the intended valley, and r_i^d represents the radius of the nest, which is reduced linearly in every iteration as per the following Eqs. 5 and 6:

$$r_0^d = 0.5\left(x_{max}^d - x_{min}^d\right) \tag{5}$$

$$r_i^d = \begin{cases} r_i^d = r_0^d - r_0^d \times \frac{i-1}{0.5\times i_{max}} \; i \le 0.5 \times i_{max} \\ r_0^d - r_0^d \times \frac{i-0.5\times i_{max}-1}{0.5\times i_{max}} \; i > 0.5 \times i_{max} \end{cases} \tag{6}$$

Where r_0^d represents the initial radius of the nest, and r_i^d denotes the circle radius in iteration i. After generating the new potential solutions ($x_{i_{new}}^d$), their fitness/objective values are calculated. The fitness of search agents is calculated based on the priority function by using Eq. 2 to assign a priority index for each nest. Based on P_i, the high-quality nest (VM) is selected for assigning tasks.

Estimation of Water Flow Speed and Direction

The water enters the nest in one half of the valleys and leaves the nest by the other half of the valleys. The fine sand on the seabed is moved by the water flow. Logically, the valleys in which the flow direction is output should have a higher volume of fine sand. Thus, the algorithm identifies 50% of adjacent valleys with the greatest volume of fine sand.

Creating and Checking New Points in the Flow Direction

Initially, the four points with the highest density of fine sand are chosen from the valleys, and then the search points are generated. Subsequently, average points on the circle perimeter that are in the outflow direction are calculated for all problem dimensions (the number of problem variables). To generate new points in each dimension of the problem, the selected point's coordinates are shifted by dx toward the stated average. If the Efficiency Factor (EF) value of the new solution is improved, the coordinates of the point measured in that dimension will be replaced with new coordinates. This process is repeated for each dimension of the examined point, and a distinct optimized point is identified at the conclusion. During this process, a type of sensitivity analysis is conducted, which greatly aids in determining the optimal answer. Moreover, for each of the four points taken, a new point is created according to the number of the problem's dimensions. The overall load-balancing process is represented in Algorithm 1.

Algorithm 1

Begin

1: Initialize population of Pop candidate solutions X_k.
(Each X_k is a task of the VM mapping matrix).

2: Normalize task parameters:
$L_{i\_norm} = L_i/L_{max}$
$ET_{i\_norm} = ET_i/ET_{max}$
$RU_{i\_norm} = RU_i/RU_{max}$

3: Compute priority score for each task:
$P_i = \alpha 1 * L_{i_{norm}} + \alpha 2 * ET_{i\_norm} + \alpha 3 * RU_{i\_norm}$

4: Evaluate fitness for each solution X_k:
For each VM_j:
Compute utilization U_j.
Compute energy E_j.
Compute total energy E.
Compute the makespan.
Compute response time RT.
Fitness $F_K = w_1 * (E/E_{max}) + w_2 * (Makespan/MS_{max}) + w_3 * (RT/RT_{max}) - w_4 * (average\ priority\ satisfied)$

5: Select the best solution X_{best} (minimum F_K).

6: For iteration t = 1 to T_{max} do

7: Update circular radius:
$R(t) - R0 * (1 - t/T_{max})$

8: Generate valley points around X_{best}.
(Create candidate mappings using circular search).

9: Evaluate the fitness of all valley solutions.

10: Select the top 50% best valley solutions.
(Based on lowest fitness)

11: From the top solutions, choose four of the best points.

12: For each selected point:
For each dimension (task assignment):
Shift assignment toward X_{best}
using:
$X_{new} = X_{current} + rand(\ ) * R(t) * (X_{best} - X_{current})$

13: Repair solution if constraints are violated:
Ensure each task is assigned to exactly one VM.
Ensure VM capacity is not exceeded.

14: Evaluate the fitness of X_{new}.

15: If $F(X_{new}) < F(X_{current})$:
Replace $X_{current}$ with X_{new}

16: Update X_{best} if a better solution is found.

17: End for

18: Return X_{best}.

End

The decision-making process in the proposed MOA-based framework is governed by a priority-aware multi-objective fitness function. Each search agent represents a complete task to VM mapping configuration, and the fitness evaluation integrates energy consumption, makespan, response time, and priority satisfaction. During each iteration, system parameters such as VM utilization and workload characteristics are dynamically updated. The adaptive circular radius mechanism enables global exploration during high workload conditions and local exploitation during steady operational states. Additionally, an overload-aware repair strategy ensures that overloaded VMs redistribute low-priority tasks, allowing the system to adapt to varying workloads and operational demands effectively.

Results and Discussion

Experimental evaluation of the proposed load-balancing in CC is described in this section. The simulation of the MOA algorithm-based load-balancing in VM resources is implemented in the CloudSim tool 3.0.3

version. Tables 2 and 3 illustrate the experimental settings and workload configuration of the proposed model utilized for load-balancing in the cloud. Quantitative and qualitative analysis of the MOA-based load-balancing approach with state-of-the-art methods and comparative analysis with existing research methods are represented in the next sections. To estimate the effectiveness of the proposed method for load-balancing, different performance measures are utilized, such as energy consumption, makespan, throughput, execution time, and resource utilization.

Table 2: System parameter settings of the proposed load-balancing framework

Experimental settings	
Tools	Description
CloudSim	3.0.3
Processor	12th Gen Intel(R) Core(TM) i7-12700K (3.60 GHz)
RAM	64.0 GB (63.8 GB usable)
System type	64-bit operating system, x64-based processor
Operating System (OS)	Windows 11 Pro
MJPO Parameters	
Parameter	value
Population size	30
Max iterations	100
Number of hosts	5
Termination condition	i_{max}
Processing capacity per host	10,000 Million Instructions Per Second (MIPS)

Table 3: Configuration of workloads, host, and VM

Workload configuration	
Parameters	Values
Number of cloudlets	50-500
Cloudlet length	1000-20000 MI
Arrival Pattern	Poisson-distributed dynamic arrival
Priority levels	Computed based on latency sensitivity, execution time, and resource demand
Task heterogeneity	High (mixed short and long jobs)
Host Configuration	
Parameters	Values
Number of hosts	5
Processing capacity	10000 MIPS
RAM	32 GB
Storage	1 TB
Bandwidth	10 Gbps
VM Configuration	
Number of VMs	10, 20, 30, and 40
VM MIPS	1000-3000
RAM	2 GB
Bandwidth	1000 Mbps
Processing elements	2

Qualitative and Quantitative Analysis

Evaluation of the proposed MOA algorithm performance in load-balancing is analyzed in this section. The state-of-the-art methods, namely the Bat Optimization Algorithm (BOA) and WOA algorithms, are considered for quantitative and qualitative analysis with the proposed load-balancing model. These algorithms are utilized for performance evaluation with the proposed MOA algorithm under an identical workload, the same VM configuration, and the same number of hosts. Figs. 2 to 4 illustrate the performance results of MOA for diverse performance metrics.

Energy consumption analysis of the MOA algorithm, which is utilized in load-balancing, is evaluated and represented in Fig. 2. State-of-the-art approaches, such as conventional optimization algorithms, are considered for the evaluation of the proposed load-balancing algorithm. From the results, it is confirmed that the MOA-based workload-balancing model efficiently reduces the energy consumption in a cloud environment.

Performance comparison of the MOA algorithm utilized for load-balancing in the cloud in terms of makespan. Fig. 3 displays the performance of the proposed MOA algorithm in terms of VMs. From Fig. 3, it is clear that the proposed MOA algorithm achieves a shorter makespan for load-balancing of the respective tasks efficiently in a cloud environment.

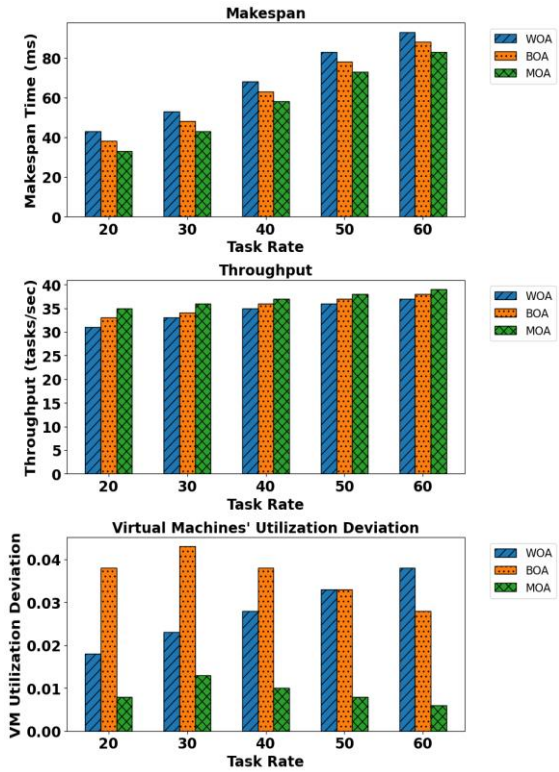

Fig. 2: Performance of the proposed MOA algorithm in terms of task rate

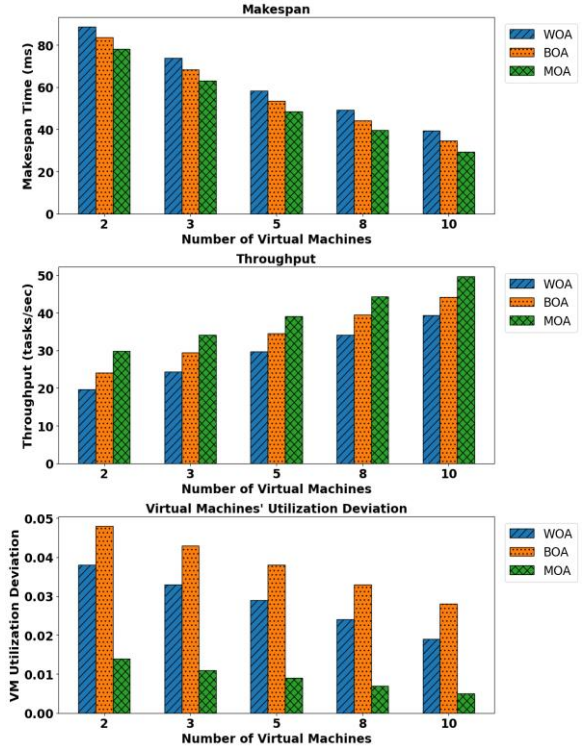

Fig. 3: Performance of the proposed MOA algorithm in terms of VMs

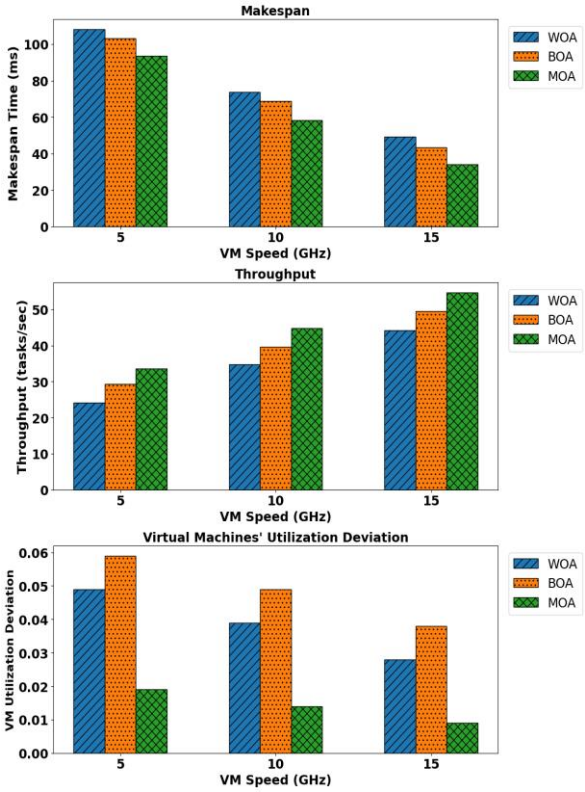

Fig. 4: Performance of the proposed MOA algorithm in terms of VM speed

Performance analysis of the MOA algorithm is utilized for load-balancing in the cloud in terms of execution time. Fig. 4 demonstrates the performance of the proposed MOA algorithm model in terms of VM speed. From Fig. 4, it is clear that the proposed MOA algorithm efficiently allocates the resource with less execution time when compared to the existing methods used for balancing workload in CC systems.

Table 4 represents the sensitivity analysis of the proposed model based on the radius reduction in terms of different parameters, β. To evaluate the impact of the radius reduction parameter β, Eq. 5 is used; the experiments are conducted by varying β from 0.5 to 3.0 while keeping other parameters constant. The sensitivity analysis of the proposed MOA algorithm is represented in Table 4, where population size = 30, max iterations = 100, and cloudlets = 1000. From the results, it is observed that the parameter radius reduction parameter β controls the balance between exploration and exploitation stages in the optimization process. When β is too small, the algorithm focuses more on exploration, which leads to slower convergence and higher makespan. If β increases, then the model focuses on exploitation, which causes premature convergence and suboptimal energy performance. At β = 1.5, the proposed algorithm achieves an optimal trade-off that results in minimized energy consumption, reduced response time, and stable convergence behavior.

To evaluate the scalability of the proposed MOA-based load-balancing algorithm, the number of cloudlets is increased from 100 to 5000 while maintaining heterogeneous task characteristics, which is represented in Table 5. Table 6 provides the energy consumption across 10 independent runs. The performance of MOA is evaluated using energy consumption, makespan, resource utilization, and execution time of the scheduling algorithm. From the results, it is clear that the proposed MOA-based load-balancing model with priority-based constraints maintains stable resource utilization above 97% and highlights the near-linear growth in scheduling time, which indicates the scalability for large-scale cloud environments.

The statistical analysis of the proposed load-balancing model using the t-test and the Wilcoxon test is represented in Tables 7, 8, and 9, respectively. In this research, the energy consumption parameter is considered for the evaluation of statistical analysis. Each experiment is repeated 10 times. An independent two-sample t-test is conducted to compare MOA with other methods at a 95% confidence level. The results confirm that the proposed method achieves statistically significant improvements in energy consumption when compared to existing algorithms. The p-values of the proposed method obtained from both the Wilcoxon test and the independent t-test are far below the significance level α = 0.05, which indicates that the performance differences are not due to random variation. This validates the reliability and consistency of

the proposed approach across multiple experimental runs. Therefore, the results provide strong statistical evidence that the proposed algorithm effectively reduces energy consumption and improves overall system efficiency.

Comparative Analysis

The comparative analysis of the proposed MOA-based load-balancing technique is presented in this section. Existing load-balancing approaches, such as the PSO-GWO model (Al Reshan et al., 2023), the rock hyrax optimization model (Singhal et al., 2024), BRDRL (Haris and Zubair, 2025), and CE-SMO (Verma, 2025), are utilized for comparison with the proposed MOA algorithm. Tables 10- 13 illustrate the comparison results of the proposed method for various scenarios and diverse performance metrics.

Table 4: Sensitivity analysis of the proposed MOA based on radius reduction rate β

β	Makespan (ms)	Energy (W-s)	Response Time (ms)
0.5	31294	138427	2598
1.0	30176	134982	2487
1.5	29864	134287	2476
2.0	30573	136912	2524
3.0	31892	141206	2637

Table 5: Scalability analysis of the proposed MOA algorithm for varying cloudlets using different performance measures

Metrics	No. of cloudlets									
	20	40	60	80	100	500	1000	2000	4000	5000
Makespan (ms)	840	1300	2150	3100	3300	15783	29864	58791	116942	146317
Energy consumption (W-s)	1180	5700	8000	11900	12500	67492	134287	267943	529618	661274
Throughput (ms)	50	150	190	12400	15000	74836	148927	294516	586732	732481
Response Time (ms)	145	360	730	930	980	1317	2476	4793	9488	11867

Table 6: Energy consumption (W-s) across 10 independent runs

Run	Proposed MJPO	Rock Hyrax (Al Reshan et al. 2023)	PSO-GWO (Khan and Santhosh, 2022)
1	134112	146203	141325
2	134893	145811	142114
3	133982	146482	141872
4	134506	145976	142603
5	134221	146318	141556
6	134774	146027	142217
7	134018	146552	141908
8	134395	145693	142345
9	134667	146109	141731
10	134302	146274	142149

Table 7: Statistical analysis of the proposed method based on energy consumption (W-s)

Algorithms	Mean (W/s)	Standard Deviation
PSGWO	141982	3265
Rock Hyrax	146145	2896
Proposed MOA	134287	2114

Table 8: Statistical analysis of the MOA-based load-balancing algorithm using the Wilcoxon and independent statistical tests

Comparison	Test Type	Test Statistic	p-value	Significance (α = 0.05)
MOA vs. PSGWO	Wilcoxon	W = 3	0.00018	Significant
MOA vs. Rock Hyrax	Wilcoxon	W = 5	0.00009	Significant
MOA vs. WOA	Independent t-test	t = 8.12	0.00001	Significant
MOA vs. PSGWO	Independent t-test	t = 4.37	0.0006	Significant
MOA vs. Rock Hyrax	Independent t-test	t = 3.02	0.0074	Significant

Table 9: Overall energy consumption analysis with existing state-of-the-art methods

Algorithms	Energy (W/s)
WOA	149638 ± 3542
PSGWO	141982 ± 3265
DQN	137914 ± 2487
MJPO	134287 ± 2114

Table 10: Comparative analysis of the proposed MOA algorithm vs. the PSO-GWO model (Al Reshan et al., 2023)-based response time (ms)

Methods	No. of Users			
	10	20	30	40
PSGWO (Al Reshan et al., 2023)	89.95	91.15	92.54	92.88
Proposed MOA algorithm	87.34	89.56	90.45	91.62

Table 11: Comparative analysis of the proposed MOA algorithm vs. the rock hyrax algorithm (Singhal et al., 2024) for VM = 10

Methods	Metrics	No. of cloudlets				
		20	40	60	80	100
Rock hyrax algorithm (Singhal et al., 2024)	Makespan (ms)	900	1500	2200	3300	3500
	Energy consumption (W-s)	2000	6000	8400	12200	14000
	Throughput (ms)	45	130	180	270	350
	Response Time (ms)	160	380	750	1000	1140
Proposed MOA algorithm	Makespan (ms)	840	1300	2150	3100	3300
	Energy consumption (W-s)	1180	5700	8000	11900	12500
	Throughput (ms)	50	150	190	12400	15000
	Response Time (ms)	145	360	730	930	980

Table 12: Comparative analysis of the proposed MOA algorithm vs. BRDRL (Mohammad Haris and Zubair, 2025) based on varying tasks

Methods	Metrics	No. of tasks				
		100	200	300	400	500
BRDRL (Mohammad Haris and Zubair, 2025)	Makespan (s)	15	31	34	37	40
	Response Time (ms)	3267	3486	4477	4675	5403
Proposed MOA algorithm	Makespan (ms)	13	27	30	33	36
	Response Time (ms)	3154	3297	4319	4487	5216

Table 13: Energy consumption of the proposed MOA algorithm vs. BRDRL (Mohammad Haris and Zubair, 2025) based on VMs

Methods	Metrics	No. of VMs						
		5	10	50	100	500	1000	2000
BRDRL (Mohammad Haris and Zubair, 2025)	Energy Consumption	0.54	1.01	2.81	3.98	6.43	7.94	8.51
Proposed MOA algorithm	(kWh)	0.51	0.97	1.56	2.82	5.23	6.45	7.76

Discussion

The proposed MOA algorithm utilized for load-balancing in the cloud attains greater performance than existing load-balancing methods. The proposed load-balancing model achieves an energy consumption of 1180 W, a makespan of 840 ms, and a response time of 145 ms, which are better than existing load-balancing works in the cloud environment. The proposed MOA method addresses the limitations of the existing algorithm and helps to enhance the load-balancing process for the tasks, respectively. The existing load balancing models contain certain limitations, as seen in the PSO-GWO model (Al Reshan et al., 2023), which frequently struggles with performance. The RHO model (Singhal et al., 2024) has some challenges, such as premature convergence, local optimal issues, and a lack of adaptability in complex cloud environments. The conventional RL method-based load balancing faces the challenge of adapting to dynamic changes in the cloud environment.

Conclusion

The proposed MOA algorithm model was employed to achieve energy-efficient load-balancing by managing tasks effectively. It combined priority-based task scheduling with adaptive load distribution to ensure efficient resource utilization. This approach improved the decision-making processes, optimizing energy usage and system performance in response to varying workloads and operational demands. The proposed MOA utilized its natural nesting behaviors, which were utilized in this research to enable adaptive and efficient load-balancing.

The experimental results of the proposed MOA method achieved energy consumption of 39.8% and resource utilization of 99.54%, which was less when compared to the existing PSO-GWO. In the future, advanced metaheuristic optimization methods will be used to enhance secure load-balancing in the cloud to ensure the confidentiality of sensitive data.

Acknowledgment

The authors express their sincere gratitude to SR University, Ananthasagar, Hasanparthy, Telangana, India, for providing the necessary support, facilities, and resources that enabled the successful completion of this research. The authors also thank colleagues and peers for their valuable suggestions and encouragement throughout the work.

Funding Information

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Author's Contributions

S. Vijaykumar: Conducted the research, performed the experiments, and drafted the manuscript.

Shanker Chandre: Contributed to data analysis, interpretation of results, and critical revision of the manuscript.

Ethics

This article does not contain any studies with human participants or animals performed by the authors. All ethical standards applicable to this type of research have been duly followed.

References

Akinola, D. G., Adetiba, E., Adetiba, E., Abayomi, A., Thakur, S., Nnaji, U., & Moyo, S. (2025). FedLoBA-1: A Load Balancing Architecture for Mitigating Resource Overloading in Federated Cloud Infrastructures. *Journal of Computer Science*, *21*(2), 432–443. https://doi.org/10.3844/jcssp.2025.432.443

Al Reshan, M. S., Syed, D., Islam, N., Shaikh, A., Hamdi, M., Elmagzoub, M. A., Muhammad, G., & Hussain Talpur, K. (2023). A Fast Converging and Globally Optimized Approach for Load Balancing in Cloud Computing. *IEEE Access*, *11*, 11390–11404. https://doi.org/10.1109/access.2023.3241279

Fang, C., Hu, Z., Meng, X., Tu, S., Wang, Z., Zeng, D., Ni, W., Guo, S., & Han, Z. (2023). DRL-Driven Joint Task Offloading and Resource Allocation for Energy-Efficient Content Delivery in Cloud-Edge Cooperation Networks. *IEEE Transactions on Vehicular Technology*, *72*(12), 16195–16207. https://doi.org/10.1109/tvt.2023.3297362

Ghafir, S., Alam, M. A., Siddiqui, F., & Naaz, S. (2024). Load balancing in cloud computing via intelligent PSO-based feedback controller. *Sustainable Computing: Informatics and Systems*, *41*, 100948. https://doi.org/10.1016/j.suscom.2023.100948

Haris, M., & Zubair, S. (2025). Battle Royale deep reinforcement learning algorithm for effective load balancing in cloud computing. *Cluster Computing*, *28*(1), 19. https://doi.org/10.1007/s10586-024-04718-7

Hayyolalam, V., & Özkasap, Ö. (2025). CBWO: A Novel Multi-objective Load Balancing Technique for Cloud Computing. *Future Generation Computer Systems*, *164*, 107561. https://doi.org/10.1016/j.future.2024.107561

Hu, H., Wu, D., Zhou, F., Zhu, X., Hu, R. Q., & Zhu, H. (2023). Intelligent Resource Allocation for Edge-Cloud Collaborative Networks: A Hybrid DDPG-D3QN Approach. *IEEE Transactions on Vehicular Technology*, *72*(8), 10696–10709. https://doi.org/10.1109/tvt.2023.3253905

Javaheri, S. D. A., Ghaemi, R., & Monshizadeh Naeen, H. (2024). An autonomous architecture based on reinforcement deep neural network for resource allocation in cloud computing. *Computing*, *106*(2), 371–403. https://doi.org/10.1007/s00607-023-01220-7

Kaviarasan, R., Balamurugan, G., Kalaiyarasan, R., & Venkata Ravindra Reddy, Y. (2023). Effective load balancing approach in cloud computing using Inspired Lion Optimization Algorithm. *E-Prime - Advances in Electrical Engineering, Electronics and Energy*, *6*, 100326. https://doi.org/10.1016/j.prime.2023.100326

Khan, A. R. (2024). Dynamic Load Balancing in Cloud Computing: Optimized RL-Based Clustering with Multi-Objective Optimized Task Scheduling. *Processes*, *12*(3), 519. https://doi.org/10.3390/pr12030519

Khan, M. S. A., & Santhosh, R. (2022). Task scheduling in cloud computing using hybrid optimization algorithm. *Soft Computing*, *26*(23), 13069–13079. https://doi.org/10.1007/s00500-021-06488-5

Krishna, M. S. R., & Vali, D. K. (2025). Meta-RHDC: Meta Reinforcement Learning Driven Hybrid Lyrebird Falcon Optimization for Dynamic Load Balancing in Cloud Computing. *IEEE Access*, *13*, 36550–36574. https://doi.org/10.1109/access.2025.3544775

Kumar, D., Pawar, P. P., Addula, S. R., Meesala, M. K., Oni, O., Cheema, Q. N., & Haq, A. U. (2025). A Smart Optimization Model for Reliable Signal Detection in Financial Markets Using ELM and Blockchain Technology. *FinTech*, *4*(4), 56. https://doi.org/10.3390/fintech4040056

Liu, H., Chen, P., Ouyang, X., Gao, H., Yan, B., Grosso, P., & Zhao, Z. (2023). Robustness challenges in Reinforcement Learning based time-critical cloud resource scheduling: A Meta-Learning based solution. *Future Generation Computer Systems*, *146*, 18–33. https://doi.org/10.1016/j.future.2023.03.029

Nagarajan, S., Rani, P. S., Vinmathi, M. S., Subba Reddy, V., Saleth, A. L. M., & Abdus Subhahan, D. (2025). Multi agent deep reinforcement learning for resource allocation in container-based clouds environments. *Expert Systems*, *42*(1), e13362. https://doi.org/10.1111/exsy.13362

Rathinam, R., Sivakumar, P., Sigamani, S., & Kothandaraman, I. (2024). SJFO: Sail Jelly Fish Optimization enabled VM migration with DRNN-based prediction for load balancing in cloud computing. *Network: Computation in Neural Systems*, *35*(4), 403–428. https://doi.org/10.1080/0954898x.2024.2359609

Samha, A. K. (2024). Strategies for efficient resource management in federated cloud environments supporting Infrastructure as a Service (IaaS). *Journal of Engineering Research*, *12*(2), 101–114. https://doi.org/10.1016/j.jer.2023.10.031

Shi, F. (2024). A genetic algorithm-based virtual machine scheduling algorithm for energy-efficient resource management in cloud computing. *Concurrency and Computation: Practice and Experience*, *36*(22), e8207. https://doi.org/10.1002/cpe.8207

Siddesha, K., Jayaramaiah, G. V., & Singh, C. (2022). A novel deep reinforcement learning scheme for task scheduling in cloud computing. *Cluster Computing*, *25*(6), 4171–4188. https://doi.org/10.1007/s10586-022-03630-2

Singhal, S., Sharma, A., Anushree, Verma, P. K., Kumar, M., Verma, S., Kavita, Kaur, M., Rodrigues, J. J. P. C., Khurma, R. A., & García-Arenas, M. (2024). Energy Efficient Load Balancing Algorithm for Cloud Computing Using Rock Hyrax Optimization. *IEEE Access*, *12*, 48737–48749. https://doi.org/10.1109/access.2024.3380159

Tarafdar, A., Sarkar, S., Das, R. K., & Khatua, S. (2023). Power Modeling for Energy-Efficient Resource Management in a Cloud Data Center. *Journal of Grid Computing*, *21*(1), 10. https://doi.org/10.1007/s10723-023-09642-5

Verma, G. (2025). Sustainable cost-energy aware load balancing in cloud environment using intelligent optimization. *Sustainable Computing: Informatics and Systems*, *46*, 101115. https://doi.org/10.1016/j.suscom.2025.101115

Zhanuzak, R., Ala'Anzy, M. A., Othman, M., & Algarni, A. (2024). Optimizing Cloud Computing Performance With an Enhanced Dynamic Load Balancing Algorithm for Superior Task Allocation. *IEEE Access*, *12*, 183117–183132. https://doi.org/10.1109/access.2024.3508793