International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

OPTIMIZING CLOUD PERFORMANCE: A SELF-ADAPTIVE EVOLUTIONARY METAHEURISTIC-BASED LOAD BALANCING AND RESOURCE ALLOCATION FRAMEWORK WITH DEEP REINFORCEMENT LEARNING

[1*]Annaiah H, [2]Dr. A. Rajesh

[1*]Research Scholar, Department of CSE, JAIN Deemed-to-be-University, Bengaluru

[2]Professor, Department of CSE, JAIN Deemed-to-be University, Bengaluru

Abstract

Cloud service providers (CSPs) must manage cloud resources efficiently to optimize resource utilization and increase user application efficiency at the lowest possible cost. The increased resource reservations result in higher resource usage costs, but they improve user application needs' performance. This paper developed a hybrid evolutionary algorithm and a unique dynamic load-balancing architecture to get better load balancing (LB) and effective resource allocation (RA). Initially, compute Virtual Machine (VM) loads and cluster them using a self-adaptive evolutionary algorithm-based clustering technique. Hybrid optimization methods like Owl Optimization Algorithm (OOA) and Wild Goose Optimization (WGO) are used for load computation. Then, it introduces a multi-stage technique for optimal task allocation with Deep Reinforcement Learning (DRL) and metaheuristics. The DRL assigns tasks to underloaded VMs, and then evolutionary techniques are used to refine the assignments. The suggested approach has shown improved efficiency and scalability by providing a suitable solution to the problems associated with LB and job scheduling in Cloud Computing (CC) contexts. The developed model gained better results in makespan, energy consumption, and task prioritization.

Keywords: prioritization, computation, utilization, underloaded

1. INTRODUCTION

CC made massive applications running and storage possible by CC, which has grown in popularity as a platform for individuals and businesses [1]. The pay-as-you-use model of CC allows users to schedule tasks in an efficient and scalable manner at a lower cost than they would pay to maintain their computer systems for sporadic computing requirements [2]. CSPs must effectively manage cloud resources to maximize resource usage and boost user app efficiency with less cost. Increased resource reservations result in higher resource usage costs, but they improve user application needs' performance [3]. Efficient resource management systems are required to meet the performance needs of the applications [4]. Load is the broad term used to describe the amount of effort required by a computing unit for processing the data. The quantity of workloads transmitted to the cloud to be handled in a real-world cloud data

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

center is far greater than the number of computer nodes in the data center [5]. RA has a direct impact on the cloud system's efficiency, they become a more serious and difficult issue [6]. The process of creating an effective plan for allocating resources to tasks, or scheduling plans, is anxious with difficulties. The decision-making techniques are important in the CC resources for assigning tasks and maximizing performance parameters like throughput, makespan, reaction time, and resource utilization [7]. Effective work scheduling ensures and computes nodes as balanced nodes. Techniques for LB are essential for increasing performance [8]. Users who utilize the CSP are known as cloud users. The workload that reflects the real needs of cloud customers is called the cloud tasks list [9]. Task Manager maps the tasks on an appropriate machine based on the workload provided by the customer. This chooses a task from the task manager's list and allocates it to a VM. Finally, resource management give me the appropriate response [10]. The number of jobs and their computational demands, the number of VMs, and the processing capacity of each VM are determined in advance when using static TS methodologies [11]. Additionally, before task execution begins, these TS algorithms map every task on the VMs. Static scheduling prevents changes to the execution sequence till all user tasks are assigned to their corresponding VMs [12]. When mapped tasks are being executed, run-time modifications cannot be handled by the static scheduling algorithms [13]. There are problems with load imbalance since the static, classical scheduling methods do not make effective use of the available resources. In the same way, these methods cannot handle scheduling according to user priority [14]. Low-performance issues arise from load imbalance, underuse of cloud resources, and overloading of certain VMs. Furthermore, these methods are unable to address any run-time issues [15]. Also. they are unable to handle freshly arrived high-priority jobs in the later batches [16]. Greater user satisfaction and improved resource utilization is attained through the equitable and efficient distribution of resources [17, 18]. The only way to allocate resources fairly and effectively is to dynamically balance the local burden. It can also lessen the use of resources and prevent bottlenecks [19]. Effective TS and LB are critical for performance optimization in the CC environment [20]. Current cloud resource management approaches are hampered by inefficient utilization of resources, higher operating expenses, and lack of scalability in large-scale systems. Dynamic load balancing algorithms are incapable of responding to dynamic workloads, and as a consequence, higher makespan, latency, and power consumption are achieved. Furthermore, improper task scheduling and poor task prioritization cause decreased application performance and loss of QoS. To overcome current constraints, suggest a revolutionary three-phase architecture. First, use self-adaptive metaheuristic-based clustering to achieve balanced distribution among physical machines (PM) by grouping VMs according to load computation. Second, the TS combines metaheuristics and DRL through multi-objective hybrid optimization. Preprocessing tasks, DRL optimization, metaheuristic refinement, and dynamic feedback integration are improved for the scalability and self-healing of the CC system. The method seeks to improve makespan, optimize resource use, intelligently assign tasks, and provide fault tolerance to improve cloud performance. The main aim of the proposed approach is the improvement of cloud resource management using

Received: August 06, 2025

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

the efficiency enhancement in load balancing and work assignment. This becomes possible with the combination of a dynamic load-balancing architecture with evolutionary algorithms. The aim is to utilize the maximum available resources, enhance performance, and have excellent scalability in the face of dynamic user application requirements. Moreover, limitation factors like resource heterogeneity, energy usage, and computational complexity are major challenges. By integrating the adaptive learning in DRL with the advantages of metaheuristics, the model expects to maximize resource utilization, minimize makespan, limit energy usage, and achieve efficient task prioritization for scalable cloud performance optimization. The proposed approach presents several novel aspects in the realm of cloud resource management:

Hybrid Optimization Techniques: It incorporates the evolutionary algorithms into the dynamic load-balancing architecture. It employs the OOA and WGO for computing the VM loads. These algorithms are used to grouping VMS based on their respective loads.

Multi-Stage Task Allocation: The proposed three-phase approach that integrates DRL and metaheuristic for the task assignment is another novel contribution. DRL is employed to decide where to send tasks to underloaded VMs at the first time, so it can make smart and dynamic decisions according to the current situation. Evolutionary techniques are used to improve these initial assignments.

Integration of DRL with Evolutionary Techniques: DRL enables the system to learn to make decisions depending on the current situation while evolutionary techniques offer optimality. It enhances the effectiveness and sustainability of resource utilization.

Comprehensive Resource Management: By addressing load balancing and job scheduling in a single model, the method offers a more comprehensive answer to problems with cloud resource management.

Enhanced Performance Metrics: The reduction of the makespan, energy consumption, and the prioritization of tasks as depicted in the model shows the model's efficiency.

Section 2 details related works, section 3 describes the problem definition, section 4 explains about proposed methodology, section 5 describes the results and discussion, and section 6 ends with a conclusion and future scope.

2. RELATED WORKS

A resource-aware flexible task scheduling solution proposed by Said Nabi et al. [21] for the TS issue. The simulation experiments were conducted for resource-aware flexible task scheduling solution using Cloudsim, and the outcomes of the suggested method were then contrasted with those of alternative scheduling techniques. The achieved throughput and makespan of the described approach show significant improvements. The autonomous LB technique was created by Fatemeh et al. [22] for efficient job scheduling in CC. Here, developed autonomous load balancing (LB) technique queries are categorized as Central Processing Unit (CPU) depending on the available resources. The CloudSim tool is used to assess the performance of

Received: August 06, 2025

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

the created approach. It equally balanced the workload effectively to improve RA using CloudSim. A dynamic approach to job scheduling for VMs was presented by Babamir, et al. [23] to improve CC stability and LB. Unlike earlier research, the suggested approach improves both the equitable workload distribution between the VMs and the dependability of scheduling tasks based on the VM's prior experience. According to simulation results, the suggested approach raises system reliability as well as LB and reliability. Malarvizhi et al. [24] presented a multi-objective scheduling-based particle swarm optimization (MOSPSO) that provides the best allocation for an extensive number of tasks while adhering to time constraints in CC conditions and limiting the execution expenses of the work procedure. Each development for groups takes into account the asset's use status. MOSPSO makes a final effort to modify the loads based on the previous assignment map. The suggested strategy is successful in terms of makespan, execution time, and other parameters, according to analytical results. Using an improved LB with a Particle Swarm Optimization (PSO) task allocation model to schedule jobs over a variety of cloud resources, Arabinda et al. [25] devised an LB method that reduces makespan and improves resource consumption. This is made possible by having accurate information about the resources and tasks inside the data center. The findings from the simulation clearly show that, in terms of makespan reduction and resource utilization, the recommended scheduling method performs better than other existing approaches. Sarita Negi et al. [26] created a hybrid machine learning system that combines supervised and unsupervised learning to manage the cloud load. The VMs are first clustered using the artificial neural network-based flexible LB (ANN-LB) method. TS is facilitated by an optimization approach that makes use of several cloud criteria. The ANN-LB approach achieves about 75% resource utilization and requires 31.067% and 71.6% less time to complete. Dalia and colleagues [27] developed a successful LB model to maximize resources and improve LB. The results showed that, in contrast to the present approach, the suggested LB algorithm uses resources on average 78% more efficiently. Additionally, it performs well in terms of shorter makespan and execution times. Xuan and colleagues [28] devised a sophisticated technique known as the Improved Whale (IW) strategy for cloud task management to enhance the Whale Optimization Algorithm (WOA) -based method's capacity for finding the best possible solution. The results of the simulation-based studies demonstrate that, in comparison to the existing metaheuristic techniques, the suggested IW finds the best TS plans with a faster and more accurate convergence rate. RB Madhumala et al [29] proposed algorithm combines the Modified First Fit Decreasing Algorithm (MFFD) and Particle Swarm Optimization Algorithm to solve the optimal Virtual Machine packing in active Physical Machines to minimize energy consumption. In order to obtain the best fit solution, we first screen all Physical Machines for potential accommodations in each Physical Machine. In this study, we address how to modify the suggested efficient technique to increase the efficiency of particle swarm intelligence. The outcome demonstrates that, in comparison to the current methods, the suggested algorithm offers an optimal answer.

Received: August 06, 2025

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

RB Madhumala and Harshvardhan Tiwari [30] addressed using the Particle Swarm Optimization (PSO) technique. PSO works well for continuous data optimization; however, we must adjust a few of its parameters if we want to utilize it for discrete data, such in the case of virtual machine deployment. Our suggested methodology, Improved Particle Swarm Optimization (IM-PSO), tackles the Virtual Machine placement problem with the primary goal of optimizing cloud datacentre resource usage. When compared to the current methods, the results obtained demonstrate that the suggested algorithm offers an optimal solution. Annaiah Halappa and Rajesh [31] proposed the evolution of Information Technology had turned cloud computing into an innovative service pattern, which facilitated on-demand access to resources. A few companies had embraced this pattern through the provision of data centres because it was flexible. Researchers had, however, realized problems with maximizing resource utilization and minimizing idle time for task execution. Load balancing, especially in IaaS design, had still remained a major problem because of threats of server overload or underload. Earlier research had already investigated conventional load-balancing methods, pointing out their shortcomings and suggesting remedies to enhance efficiency, response time, makespan, and throughput. Annaiah et al [32] investigated Clouds as high-configured infrastructures providing platform and software services based on a pay-as-you-go model. Global use of cloud computing had increased because it featured a simple, service-oriented structure. Research had indicated that the increasing number of users stressed the need for load handling to be dependable using load balancing mechanisms. Researchers had endeavored to lower energy consumption and job rejection by optimizing virtual machine load allocation. Previous work had compared and evaluated current load balancing methods, such as Round Robin and Throttled algorithms, using measures like response time and data processing time.

3. PROBLEM DEFINITION

Even with all of the previous studies in the field of CC, there are still certain issues with the balance of workloads in cloud-based applications. Most methods are unable to keep up with dynamic workloads, leading to suboptimal utilization of resources and high energy usage. Moreover, the absence of proper multi-objective optimization and poor task prioritization tends to result in poor QoS. Scalability problems and the slow convergence of optimization algorithms further restrict their usability in large-scale cloud environments. Given the limited quantity of resources/VMs in CC, task allocation is an essential operation. In these models, CSP guarantees good service delivery performance by avoiding scenarios where hosts are either overloaded or underloaded, which could lead to longer execution times or machine failure, among other problems [33]. The fact that dynamic LB raises inter-VM transmission overheads (the cost of exchanging files across VMs) is one of its primary drawbacks. The primary difficulty facing CC service providers is TS. Assigning jobs to VMs is one of the scheduling's most important goals to prevent certain machines from overloading or underloaded [34]. The main focus of the current research is minimizing costs and lowering the amount of time needed to complete tasks (makespan) while still adhering to organizational requirements. However, many of them overlook important concerns like the execution diversity

Received: August 06, 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

and acquisition delay of VMs and the basic rule for CC. These approaches are improved by adaptive algorithms, hybrid optimization methods, and reinforcement learning overcome these drawbacks, providing efficient resource management and enhanced performance.

4. PROPOSED METHODOLOGY

Because of its many advantages, CC has grown quickly and become widely used in information technology (IT) contexts. Efficient work scheduling and LB are critical components of the CC landscape. A novel dynamic load-balancing framework (DLB) is presented to overcome the drawbacks of job scheduling and load-balancing techniques. The three main stages of the suggested paradigm are job allocation, TS, and LB. The design of the developed technique is shown in Figure 1.

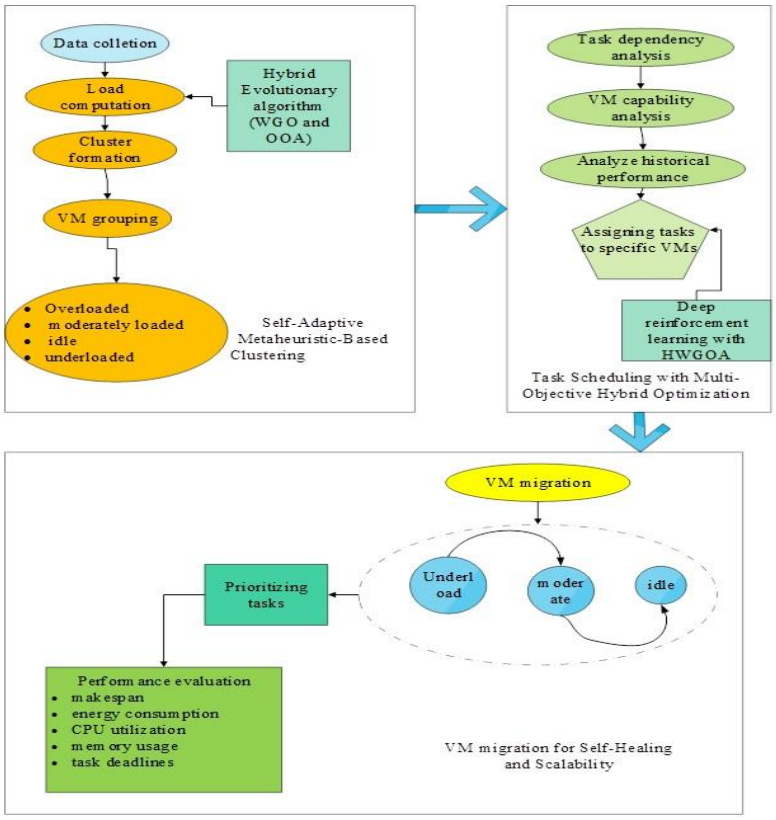


Figure 1: Process of the developed technique

4.1. Self-Adaptive Metaheuristic-Based Clustering Model

The suggested method for load computation and VM grouping in cloud environments with a hybrid optimization strategy draws inspiration from the owl and wild geese migratory and hunting patterns. Initially, generated 100 nodes and computed VM loads using the Hybrid Wild Geese and Owl (HWGO) optimization method. It blends migratory and hunting behaviors to balance the load by grouping VMs with clusters of overloaded, moderately loaded, idle, and underloaded VMs. To precisely optimize VM grouping, the clusters are generated and grouped the VMs based on the computed loads.

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

4.1.1. Data Collection

Initially, generates 100 nodes in a Python environment. One node is a PM and the remaining nodes are VMs.

4.1.2 Proposed HWGO Algorithm

The HWGO method is a metaheuristic optimization method created through the hybridization of WGO and OOA. WGO [35] imitates the long-distance migratory pattern of geese, encouraging global exploration of the solution space through retention of group movement and direction. OOA [36] simulates the sharp hunting prowess of owls, emphasizing local exploitation through adaptive search in local areas. By combining these two approaches, HWGO optimally balances exploration and exploitation and improves optimization performance. It finds extensive applications in feature selection, image segmentation, engineering design, and hyperparameter tuning in machine learning for its effectiveness in solving complex, nonlinear, and high-dimensional problems.

4.1.3 Load Computation with HWGO

To calculate VM loads, use an HWGO strategy. Group VMs into overloaded, moderately loaded, idle, and underloaded clusters depending on the computed loads to attain equal distribution between PM. A hybrid algorithm is used to calculate the load of cloud virtual machines under PMs. The loads are computed using hybrid optimization HWGO (wild geese, and owl) algorithm. The HWGO evolutionary algorithm's customized method for RA optimization and workload management makes it an attractive option for LB and cluster building in cloud computing. By merging the owl and wild geese models, this method leverages the power of evolutionary concepts to address the intricate problems present in CC settings and balance the load by utilizing clusters to group virtual machines. It's capacity to adapt to changing workloads and resource needs. Furthermore, WGO and OOA's hybrid design combines the best features of both evolutionary models to provide a strong solution that can divide work among clusters effectively while reducing latency and optimizing resource use. This combination makes LB easier and guarantees that computing resources are used as efficiently as possible to fulfill performance goals and reduce bottlenecks. In the end, cloud providers improve the overall quality of service for end customers by utilizing evolutionary algorithms called the Hybrid WGO and OOA algorithms. Also improve the scalability, dependability, and cost-effectiveness of their network.

The created model computes the VMs based on how wild geese and owls migrate and hunt. Initially, a wild goose population is established so that the vector of the position of the i^{th} wild goose is comparable to y_i. The greatest position $\vec{P}_i$ as well as migration speed $\vec{V}_i$are decided.

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Phases of displacement and motion velocity

The goal of the wild geese movement is to reach the front and neighboring members in the sorted population. It is a team's coordinated, ordered, and controlled movement. Equations (16) and (17) provide the velocity and displacement based on the synchronized velocity of the geese.

$$\vec{V}_{i,n}^{Iter+1} = \left(s_{1,n} \times \vec{V}_{i,n}^{Iter} + s_{2,n} \times \left(\vec{V}_{i,n}^{Iter} - \vec{V}_{i-1,n}^{Iter}\right)\right) + s_{3,n} \times \left(\vec{P}_{i,n}^{Iter} - y_{i-1,n}^{Iter}\right) + s_{4,n} \times \left(\vec{P}_{i+1,n}^{Iter} - y_{i,n}^{Iter}\right) + s_{5,n} \times \left(\vec{P}_{i+2,n}^{Iter} - y_{i+1,n}^{Iter}\right) - s_{6,n} \times \left(\vec{P}_{i-1,n}^{Iter} - y_{i+2,n}^{Iter}\right) \quad (16)$$

Let, $y_{i,d}$, $\vec{P}_{i,d}$, and $\vec{V}_{i,d}$ are the d^{th} dimensions of the current velocity, optimum position, and position at this moment of the i^{th} wild goose, respectively. $\vec{R}_{k,n}$; $k = 1,2,\ldots 11$are random numbers with a uniform distribution between 0 and 1. Every wild goose's velocity of its front and rear components determines the variations in position and velocity, as seen in Eqn. (16). i.e. the locations of its surrounding members. The wild geese use information from nearby individuals in the classified population to guide their movement and navigation, and they typically reach these individuals using Equation (16), which simulates the wild geese migratory process. Moreover, Eqn. (17) illustrates the global best member to provide flock motions with extra direction. This position is shifted in an organized manner and coordinated with the front members to portray the movements of all members in a planned sequence.

$$y_{i,n}^{v} = \vec{P}_{i,n}^{Iter} + \vec{R}_{7,n} \times \vec{R}_{8,n} \times \left(\left(\vec{G}_{n}^{Iter} + \vec{P}_{i+1,n}^{Iter} - 2 \times \vec{P}_{i,n}^{Iter}\right) + \vec{V}_{i,n}^{Iter+1}\right) \quad (17)$$

Let, $\vec{G}_n$ is the greatest standing among all members globally.

- **Wild geese strolling and looking for food**

The way this phrase is modeled ensures that the i^{th} wild goose approaches the member in the front. i.e. the $(i+1)^{th}$ goose $\left(\vec{P}_{i}^{Iter} \rightarrow \vec{P}_{i+1}^{Iter}\right)$. Stated differently, the i^{th}goose reaches out to $\left(\vec{P}_{i+1}^{Iter} - \vec{P}_{i}^{Iter}\right)$. The formula for the wild goose's foraging and walking y_i^w is as follows eqn. (18)

$$y_i^w = \vec{P}_{i,n}^{Iter} + \vec{R}_{9,n} \times \vec{R}_{10,n} \times \left(\vec{P}_{i+1,n}^{Iter} - \vec{P}_{i,n}^{Iter}\right) \quad (18)$$

- **Reproduction and evolution of wild geese**

Another aspect of a wild geese's existence is its reproductive and evolutionary stages. Additionally, modeling is done in such a way that a mix of migration (y_i^v) and strolling while trying to find food (y_i^w) is used. In this update the β and Ic_i Equation (19) explains the modified formula that is used to increase the load balancing performance of VMs to compute the loads based on the overloaded, moderately loaded, idle, and underloaded variables of the owl paradigm to the wild geese framework.

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

$$y_{i,n}^{Iter+1} = \begin{cases} \vec{P}_{i,n}^{Iter} + \beta \times Ic_i + \vec{R}_{7,n} \times \vec{R}_{8,n} \times \left(\left(\vec{G}_n^{Iter} + \vec{P}_{i+1,n}^{Iter} - 2 \times \vec{P}_{i,n}^{Iter} \right) + \vec{V}_{i,n}^{Iter+1} \right) & if \vec{R}_{7,n} < 0.5 \\ \vec{P}_{i,n}^{Iter} - \beta \times Ic_i + \vec{R}_{9,n} \times \vec{R}_{10,n} \times \left(\vec{P}_{i+1,n}^{Iter} - \vec{P}_{i,n}^{Iter} \right) & if \vec{R}_{7,n} \geq 0.5 \end{cases}$$

(19)

Moreover, β is a constant that decreases linearly from 1.9 to 0. β encourages the investigation of the search space, and first makes significant alterations. These variances are lessened as the algorithm develops to promote exploitation. Moreover, Ic_i is the variation in strength for i^{th}owl and is calculated using equation (20).

$$Ic_i = \frac{\vec{I}_i}{\vec{r}_i^2} + Rs \qquad (20)$$

Let, $\vec{I}_i$ is the information on the normalized intensity of i^{th}owl i, $\vec{r}_i^2$, Rs is denoted as random noise. In the end, the established hybrid optimization method creates clusters based on four parameters: overloaded, moderately loaded, idle, and underloaded for the virtual machines. It improves the efficiency of the cloud LB procedure by precisely grouping the virtual machines (VMs) according to their computed load. The pseudocode format of the evolutionary HWGO algorithm is detailed in algorithm.1.

Algorithm:1 pseudocode for HWGO evolutionary algorithm

Start
{
Initialization
{
Set parameters: , , and
}
Generate an initial population of solutions randomly
{
Calculate the fitness of each solution in the population
Sort the population based on fitness
Update the global best solution
}
end for
{
Generate a random number
Calculate the inertia weight and velocity
Update the position of the solution
pseudocode for Update owl parameters , and
Evaluate the fitness of the new solution
{
Update the local best solution
}
Else
{

Received: August 06, 2025

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

```
        Update the global best solution
        }
End if
Compute VM loads
Generate cluster
VM grouping
Effective LB
end
```

4.1.4 Cluster-based VM grouping

The HWGO algorithm calculates the loads of VMs and generates a cluster for grouping the VMs based on the loads. Each virtual machine's burden is represented by a weight value, which is $\widetilde{W} = \{\widetilde{W}_{v1}, \widetilde{W}_2, \ldots \widetilde{W}_{vl}\}$. Based on the weight value of each VM, the clusters are generated to group the VMs based on underloaded VMs with $\tilde{v}_{mo} = \{\tilde{v}_{mo_1}, \tilde{v}_{mo_2}, \ldots, \tilde{v}_{mo_n}\}$, moderately loaded VMs with $\tilde{v}_{mm} = \{\tilde{v}_{mm_1}, \tilde{v}_{mm_2}, \ldots, \tilde{v}_{mm_n}\}$, idle loaded VMs with $\tilde{v}_{mi} = \{\tilde{v}_{mi_1}, \tilde{v}_{mi_2}, \ldots, \tilde{v}_{mi_n}\}$ and overloaded VMs with $\tilde{v}_{mu} = \{\tilde{v}_{mu_1}, \tilde{v}_{mu_2}, \ldots, \tilde{v}_{mu_n}\}$. The developed model forms clusters $c_1, c_2, c_3$and c_4 for overloaded, moderate, idle, and underloaded.

4.2 TS with Multi-Objective Hybrid Optimization:

This stage expands on the clusters' discovered underloaded virtual machines and established work scheduling goals [42]. It presents a multi-phase method for optimum work allocation that makes use of evolutionary algorithms and DRL.

4.2.3 Context-Aware Task Preprocessing

To maximize task allocation in cloud data centres, the procedure entails job dependencies analysis, VM capability profile, and past outcomes integration. The order in which tasks are executed is determined by identifying task dependencies, and virtual machines are profiled according to their CPU architecture, memory, storage, and network bandwidth. Task allocation decisions are guided by past VM performance data, with the goals of minimizing makespan and maximizing throughput. Load distribution is tracked to guarantee effective use of resources, which leads to best-in-class task execution on chosen virtual machines. The overall goal of the method is to improve CC systems' efficiency and performance by allocating tasks to virtual machines strategically.

Task Dependency Analysis:

Examine incoming tasks to see if there are any relationships between them. This entails realizing which chores must be finished before moving on to others.

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Makespan: Makespan, as determined by Equation (21) is the length of time required to complete the jobs on the mapped virtual machines in the cloud data center, where M_s is denoted as makespan, n is the number of VMs and $c_j[t]$displays the finishing time of $\tilde{v}_{mj}$

$$M_s = max\{c_j[t]\} = max\{c_1[t], c_2[t], \ldots, c_n[t]\} \quad (21)$$

ARUR: Equation (22) is used to calculate the average resource utilization (ARUR) for each algorithm during the full execution of all jobs on the available VMs, where $avgM_s$ is computed using equation (23).

$$arur = \frac{avgM_s}{M_s} \quad (22)$$

$$avgM_s = \frac{\sum_{j=1}^{n} M_s(j)}{n} \quad (23)$$

Throughput: The quantity of jobs finished in a certain amount of time, as stated in Equation (24), is the definition of the throughput metric's quantity, Th is denoted as throughput, and $T(s)$ is considered as total tasks.

$$Th = \frac{T(s)}{M_s} \quad (24)$$

VM Capability Profiling:

Examine each VM in the cluster, taking into account its CPU architecture, memory size, storage accessibility, and network bandwidth.

Weight: Using Equation (25) the weight number concerning the load is computed as follows, where $\widetilde{W}_l$ and $\tilde{l}_l$ denote the weight and load on l^{th} VM, while $\tilde{e}_j$ symbolizing the time of execution of s^{th} given assignments on VM.

$$\widetilde{W}_i = \tilde{l}_i \frac{\frac{1}{\tilde{e}_j}\sum \tilde{e}_j \forall assigned\_task\_\tilde{v}_l}{\tilde{c}(\tilde{v}_l)} \quad (25)$$

Capacity: $\tilde{\boldsymbol{c}}(\tilde{v}_l)$ gives the VM's capacity, which is calculated using equation (26).

$$\tilde{c}(\tilde{v}_l) = (n_l(p) \times n_l(mips) + b^l{}_w) \times \frac{1}{100} \quad (26)$$

The ability of every l^{th} VM $\tilde{c}(\tilde{v}_l)$is determined by taking the processor count into account $n_l(p)$, the million instructions per second value $n_l(mips)$, and the bandwidth ($b^l{}_w$) of l^{th} the VM.

CPU utilization: Similarly, eqn. (27) is used to determine the CPU consumption of VMs.

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

$$cpu_u = \frac{T(cpu_u)}{n_l(p)} \quad (27)$$

$T(cpu_u)$ is expressed as the total CPU utilization process.

Historical Performance Integration:

Examine past data on VM performance indicators for various job kinds, such as execution time, resource usage, and failure rates.

Execution time: To calculate execution time, find the proportion of the length of thes^{th}task l_s and $mips$of l^{th} VM. VM Optimal is suggested for s^{th} the task that reduces $e(t)$. The execution time ($e(t)$) of s^{th} task on l^{th}. Equation (28) provides VM, where l_s is denoted as s^{th} length.

$$e(t) = \frac{l_s}{mips_l} \quad (28)$$

Transmission rate: The duration of the transmission of the s^{th} task on a certain VM is found by dividing the task size by $S(t)$ and bandwidth of the VM ${b^l}_w$. The transmission time (T_t) for a task $t(s)$ on a VM $\tilde{v}_l$ is computed as eqn. (29)

$$T_t = \frac{S(t)}{{b^l}_w} \quad (29)$$

Load distribution: Equation (30) is used to calculate the clusters' current load distribution.

$$Dist_l = \sum_{j=1}^{c} \sum_{i=1}^{n} v_{ml(i)}^{j} - l_j^2 \quad (30)$$

To assign VMs to complete the task faster, the created technique evaluates the task, VM, and cluster capacity.

4.3 Task allocation using multi-objective Optimization with DRL

First, a DRL [41] model trained on past data and a predetermined reward function assigns the first job. To each cluster's underloaded VMs, this agent assigns jobs. This allocation is further refined using metaheuristic methods HWG. Using the Q-learning method, the DRL agent learns an optimal scheduling policy while taking into account input from the cloud architecture and future decisions. TS on virtual machines is represented by the state space while scheduling decisions are represented by actions. Based on implementation costs, the reward function assesses effective TS. Q-learning assesses input from the surroundings to maximize subsequent decision-making. LB is achieved through VM migration, whereby the VM manager the task and migrates VMs as needed. Tasks are prioritized based on user-defined criteria, and continuous performance monitoring expands the DRL agent's database of information to facilitate future scheduling decisions that are better informed.

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

4.3.3 Initial Task Allocation by DRL:

The DRL agent distributes tasks to underloaded VMs in each cluster using a specified reward function. Then use HWGO evolutionary algorithms to enhance the job allocation that the DRL agent suggests. These methods thoroughly investigate the solution space and find even more optimal task-to-VM mappings that maximize the specified goals.

Teaching an agent how to act and adjust to changes in its surroundings is the primary goal of reinforcement learning. In particular, use the methods of Q-learning to learn an optimal scheduling policy by assessing feedback from the cloud infrastructure and taking future decisions into account.

Let $T = \{t_1, t_3, \ldots., t_n\}$be a collection of jobs that several users have submitted and $vm = \{vm_1, vm_2, \ldots. vm_n\}$be a collection of virtual machines. Moreover, $S(t)$ is the current state of the cloud scheduler at that moment t in the state space s and $A(t)$be a move within the action space$a$ at time moment t with the possibility of using eqn. (31)

$$\rho\left(\frac{s\prime}{a,s}\right) = \rho\left(\frac{S(t+1)=s\prime}{S(t)} = s, A(t) = a\right) \qquad (31)$$

Let, $\sum_{s\prime \in S} \rho\left(\frac{s\prime}{a,s}\right) = 1$. The policy for cloud schedulers is $\pi\left(\frac{s}{a}\right)$, that they assign each assignment and link states to actions t_i to a VM vm_j. The benefits of acting right away $A(t)$ in state $S(t)$ is $R(t)$. The objective of the cloud scheduler is to determine the optimal scheduling strategy that minimizes the overall reward value for each job and VM that is taken into account. The following are RL solution's states, actions, and reward functions.

State Space: At each time t, the state indicates the jobs' current scheduling on the VMs, and each VM vm_j. A task t_i is allocated to any VM vm_j.

Action Space: The action $A(t)$ serves as a representation of the scheduled action at that moment t of every task that is taken into consideration on the accessible VMs. The action for every job can be displayed as either 0 or 1. When a task t_i is assumed to a vm_j , The task action space is shown as a size vector n, e.g.: (0, 1, 0, 0, … , 0), of task t_i affixed to the second VM.

Reward: The effectiveness of the job scheduling procedure is represented by the reward function. If the task t_i is allocated to VM vm_j , specify the cost of execution $\psi_{ij}$is the instantaneous personal gain. The total benefit at that moment $tR(t)$ is the total amount of expenses. Equation (32) defines the individual reward based on the bandwidth, RAM, CPU, and disk storage quantities, where $\vartheta_{i,j}$ is the waiting time t_i and is allocated to vm_j , U_{pj} is every unit's price vm_j , and $\xi_{i,j}$ are the execution time of t_i on vm_j .

$$\psi_{ij} = \left(\xi_{i,j} + \vartheta_{i,j}\right) \times U_{pj} \qquad (32)$$

Use the Q-learning approach in this phase to assess the cloud system environment's input to maximize decision-making going forward. Following the receipt of each incentive, the average Q-value of action a on the state s resulting in the policy π is signified $q_\pi(s, a)$and Equation

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

(33) has the optimal value distribution and the Bellman the greatest efficiency for optimal value distribution can be stacked in equation (34).

$$q_*(s,a) \tag{33}$$

$$q_*(s,a) = \sum_{s'} \Gamma(s'/a,s)\left[r + \delta \min_{a'}\left(q_*(s',a')\right)\right] \tag{34}$$

Let, δ is the discount factor, which measures how much the previous actions have an impact on the future reward, and Γ indicates the likelihood that the current condition change s to the next state s' under action a. At the time of arrival, every task's t_i resource requirements are known and assigned to VM vm_j using eqn. (35)

$$p_i^{cpu} \le cpu_j^t; p_i^{ram} \le ram_j^t, p_i^{bw} \le bw_j^t; p_i^{ds} \le ds_j^t \tag{35}$$

Let, $p_i^{cpu}, p_i^{ram}, p_i^{bw}, p_i^{ds}$ are the needs for the task's CPU, RAM, bandwidth, and disk storage, and cpu_j^t, ram_j^t, bw_j^t, and ds_j^t are the specifications for the present resource amounts available. Next, the cloud scheduler assesses $q_\pi(s,a)$ for the current policy π. After that, eqn. (36) is used to modify the policy. Moreover, this model's primary goal is to identify the best course of action π^* diminishing the benefit of any state s. In this phase update the hybrid behavior of HWGO $y_{i,n}^{Iter+1}$ for enhancing the assigning process of the VMs based on the load condition using eqn. (37). Lastly, the constructed model gives assignments to the $\tilde{v}_{mo}$, if the $\tilde{v}_{mo}$ is busy means it migrates the tasks to $\tilde{v}_{mi}$, and $\tilde{v}_{mm}$.

$$\pi' = arg\min_a q_\pi(s,a) \tag{36}$$

$$\min_{\pi^*} e[q_{\pi^*}(s,a)] + y_{i,n}^{Iter+1} \tag{37}$$

4.3.4 Migrating tasks between VMs to balance load

Migrating tasks between VMs for load balancing is employed to optimize overall system performance by avoiding resource bottlenecks and providing even workload distribution among available VMs. This helps in optimizing resource usage, minimizing response time and latency, and promoting energy efficiency by task consolidation and powering off idle VMs. It also enhances the reliability and scalability of the system, making it extremely efficient for dynamic and large-scale cloud computing environments.While LB between PMs is facilitated by VM migration, LB among VMs is facilitated by both TS and VM clustering. The VM manager keeps track of each PM's load and updates PM data. The load on the VMs included in that PM is used to determine the PM load, as shown in equation (38).

$$l(p_m) = \sum_{i=1}^{l} l_i \tag{38}$$

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Equation (38) measures the load on m^{th} PM concerning the load l of all VMs (l_i). In addition to LB, VM migration allows energy efficiency. If any VM gets overloaded, the VM manager chooses whether to migrate the VM. Additionally, the VM is moved to the best destination VM. Consider p_m with $\{\tilde{v}_{m1}, \tilde{v}_{m2}, \tilde{v}_{m3}, \ldots, \tilde{v}_{mi}, \ldots \tilde{v}_{ml}\}$and fill virtual machines as $\{l_1, l_2, l_3, \ldots, l_i, \ldots l_l\}$. If p_m is overloaded, the best VM is selected and moved from that PM to a different PM. The best VM for migration is chosen using proposed HWGO, and it is calculated for l^{th} VM as eqn. (39), where $ram(\tilde{v}_{ml})$ is the RAM of $\tilde{v}_{ml}$ and bw_l is bandwidth of $\tilde{v}_{ml}$. The criteria for a VM are selected based on the best VM for migration and are expressed as eqn. (40).

$$M(t)\tilde{v}_{ml} = \frac{ram(\tilde{v}_{ml})}{bw_l} \quad (39)$$

$$Opt(\tilde{v}_m) = min\{M(t)\tilde{v}_{ml}\} \quad (40)$$

A VM that meets the prerequisite is designated as and is chosen for migration. The created model moves tasks from one VM to another to improve efficiency and balance load. Lastly, a method for job prioritization based on user-defined dates or criteria is devised. The task execution performance is regularly monitored by the system. The DRL agent uses this input to improve its decision-making process and refresh its knowledge for upcoming TS choices. The step-by-step procedure of proposed evolutionary dynamic load-balancing framework for CC algorithm is detailed in algorithm.2.

Algorithm 2 : Evolutionary Dynamic Load-Balancing-CC Algorithm	
Step 1	1.1 Objective: Efficiently allocate tasks to VMs to optimize resource utilization. 1.2 Process: • Task Submission: Users submit tasks to the cloud data center. • Task Preprocessing: Perform task dependency analysis to identify independent and dependent tasks. • Task Classification: Classify tasks based on size, priority, and resource requirements. 1.3 Initial allocation using DRL and Prioritize underloaded VMs for task assignment. 1.4 Evaluation: Evaluate the task using historical performance data for better decision-making.
Step 2	2.1 Objective: Optimize task execution time and minimize makespan using multi-objective optimization. 2.2 Process:

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

	• VM Profiling: Analyze VMs' CPU, memory, storage, and bandwidth. • Weight Calculation: Compute the VM's weight using load and execution time. • Task Assignment: Assign tasks using a HWGO algorithm. • Makespan Calculation: Calculate the time required to complete all tasks using: • Performance Monitoring: Track task progress and adjust VM assignments if necessary.
Step 3	3.1 Objective: Ensure equitable distribution of tasks to prevent resource over-utilization or under-utilization. 3.2 Process: • Cluster Formation: Group VMs into clusters using their load values. • Task Migration to Identify overloaded VMs using their resource consumption levels and Perform task migration from overloaded VMs to underloaded or idle VMs. • VM Migration: Evaluate the best VM for migration • Resource Utilization Optimization: Calculate the Average Resource Utilization Rate (ARUR) • Continuous Learning: DRL agent updates its knowledge based on performance feedback for future decision-making.

5 EXPERIMENTAL SETUP

The proposed framework requires a robust system configuration for efficient load balancing and RA in a CC environment. A high-performance setup with an Intel Core i7 or AMD Ryzen 7 processor, 32 GB RAM, and a 1 TB SSD is recommended, along with an NVIDIA RTX 3090 or A100 GPU for Deep Reinforcement Learning (DRL) training. Ubuntu 22.04 LTS or Windows 11 serves as the preferred OS, while tools like CloudSim or iFogSim can simulate cloud environments. The self-adaptive evolutionary clustering algorithm operates with a population size of 100 and 500 iterations, while Owl Optimization Algorithm (OOA) and Wild Goose Optimization (WGO) use 50 populations for 300 iterations. The DRL model, implemented using PyTorch or TensorFlow, employs a Dueling Double Deep Q Network (D3QN) with a learning rate of 0.0001 and a discount factor of 0.99. Key performance metrics such as makespan, energy consumption, and load balance index are evaluated to ensure the effectiveness of the proposed framework.

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

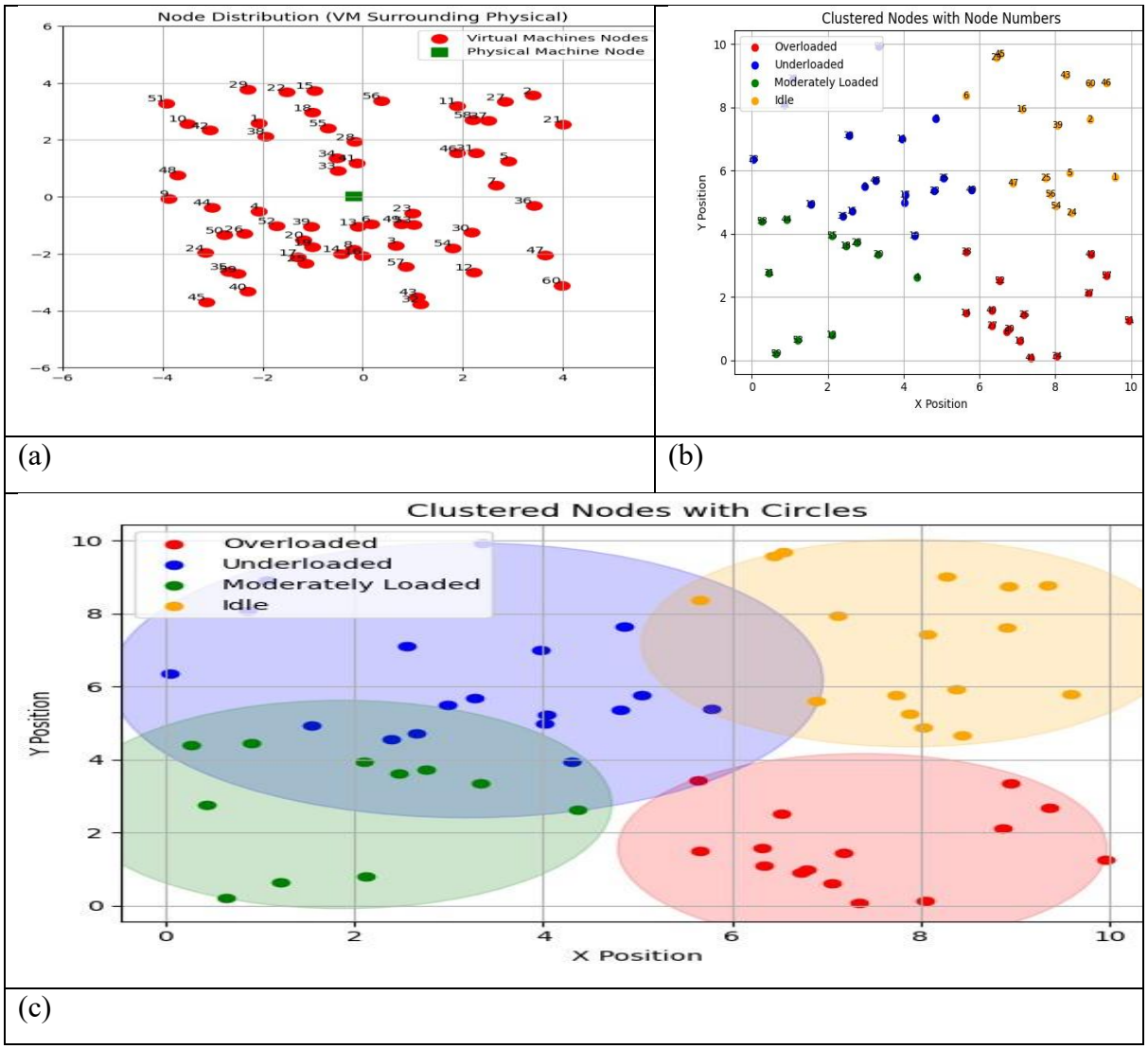


Fig.2. depicts the performance models (a) Node distribution, (b) Clustered node with node number, (c) Clustered node with circles.

Figure.2. represents the initial distribution of nodes, with VM nodes depicted as red circles encircling a central physical machine node in green. This design shows a single centralized physical machine controlling several virtual machines. The second graph shows the nodes divided into various load states through clustering with red indicating overloaded nodes, blue for underloaded nodes, green for moderately loaded nodes, and yellow for idle nodes. The third graph represents the clusters using elliptical borders, well marking the various node states. This grouped illustration is beneficial for load balancing, which optimizes the use of resources by redistributing tasks across nodes to avoid overload and maintain efficient system performance.

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

5.1 Performance comparison

The gained performance results of the developed model are validated with the existing evolutionary algorithm used for LB and RA techniques. The evolutionary algorithms used for the performance comparisons are Falcon Optimization (FOA) [37], Lyrebird Optimization (LOA) [38], Cuckoo Search optimization (CSO) [39], and Dragonfly Optimization (DFA) [40]. The formula used to measure the performance metrics is detailed below. The makespan is measured using eqn. (41), where $t_{end}(i)$ is denoted as task end time. The energy consumption is measured using eqn. (42), $p(i)$is considered as task power consumption, and $t(i)$ is denoted as task time duration. The balanced CPU utilization is measured using eqn. (43), u_{avg} is denoted as average CPU utilization, and $u(i)$ is considered as task CPU utilization. The optimized memory usage is measured using eqn. (44), T_a is considered as allocated total memory, and T_r is represented as required task memory. The task prioritization is calculated using eqn. (45), and d_i is denoted as task deadline, I is considered as important task, T_i is considered as task execution time, and w is considered as weight.

$$makspan = max\left(t_{end}(i)\right) \quad (41)$$

$$E_c = \sum_{i=1}^{n} p(i) \times t(i) \quad (42)$$

$$B_{cpu} = \frac{1}{n}\sum_{i=1}^{n}\left|u_{avg} - u(i)\right| \quad (43)$$

$$O_m = \frac{T_a}{T_r} \times 100\% \quad (44)$$

$$t_p = w(1) \times \frac{1}{d_i} + w(2) \times I + w(3) \times \frac{1}{T_i} \quad (45)$$

Choosing the right evolutionary algorithms is essential to getting the best results. Because of its system for striking a balance between exploration and exploitation, FDA is especially well-suited for dynamic contexts where RA must be regularly and instantly changed. LOA's exceptional capacity to adjust to several factors at once guarantees effective load distribution among heterogeneous systems. A comprehensive investigation of viable options is ensured by CSO's reliable technique of producing new solutions through Lévy flights, which helps to prevent local optima and achieve a balanced load distribution across resources. Effective search space exploration and exploitation are made possible by DFA's dual attraction and repulsion mechanism. These evolutionary algorithms work together to offer a complete toolkit for handling LB and RA challenges. Because each algorithm plays to its strengths, the framework can effectively manage resources, reduce response times, and guarantee stability and high system throughput. The Comparative Analysis of Tasks with different evolutionary techniques is detailed in Table 1.

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Table 1: Comparative analysis of Task with different evolutionary techniques

Model	Task	Makespan	Energy Consumption	Balanced CPU Utilization	Optimized Memory Usage	Task Prioritization
FDA [33]	100	468	49.61574	0.0269	5313.6104	5940
	200	1045	65.61824	0.025866	5404.4675	25800
	300	1648	80.97394	0.024316	5573.8971	45850
	400	2163	81.13317	0.0935	5719.417494	50800
	500	2364	93.98643	0.02456	5821.105324	53750
LOA [34]	100	499	65.15811	0.03424	6029.2364	5950
	200	1086	68.63142	0.01674	6380.0291	30900
	300	1684	72.8653	0.02396	6573.920833	45850
	400	2164	75.98326	0.04896	6879.920833	50697
	500	2445	74.73126	0.07096	6943.920833	53954
CSO [35]	100	499	65.15811	0.03424	6029.2364	5950
	200	1086	68.63142	0.01674	6380.0291	30900
	300	1684	72.8653	0.02396	6573.920833	45850
	400	2164	75.98326	0.04896	6879.920833	50697

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

	500	2445	74.73126	0.07096	6943.920833	53954
DFA [36]	100	449	44.72476	0.04276	5513.0531	5550
	200	1086	56.1718	0.04516	4615.504375	47600
	300	1389	59.27371	0.011362222	5785.768622	50850
	400	2052	61.41229	0.051235	5741.274244	50800
	500	2085	63.97825	0.09694	5883.952204	50950
Proposed	100	349	43.72476	0.003276	4713.0531	6650
	200	986	55.1718	0.004516	4415.504375	55900
	300	1346	58.27371	0.001136222	5085.768622	72950
	400	1972	60.41229	0.011235	5241.274244	81950
	500	1985	62.97825	0.008344	5383.952204	82690

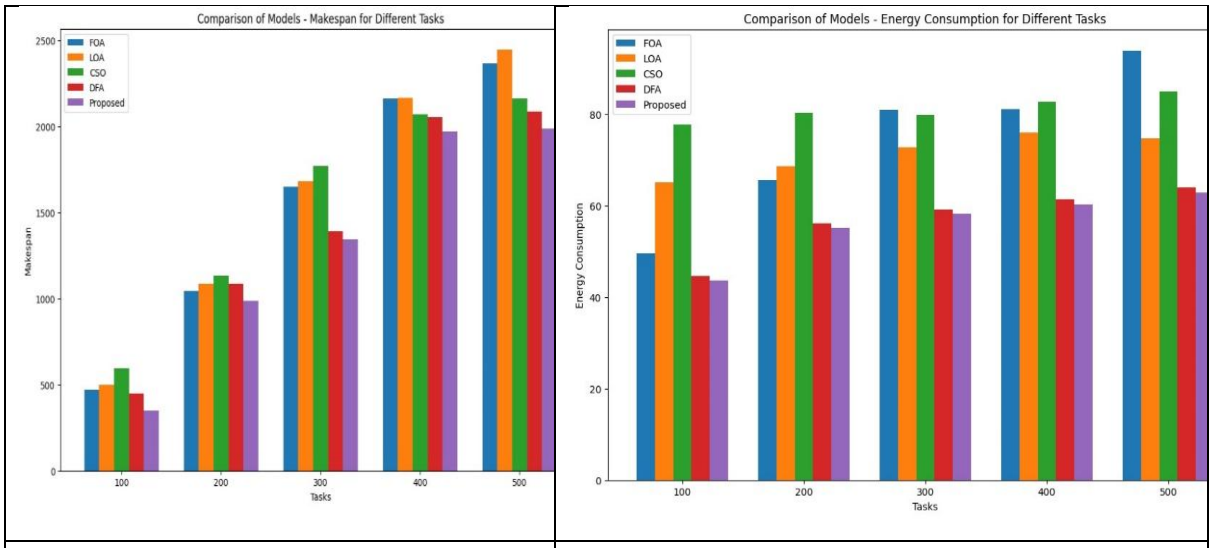
Table 1 provides the comparative study of task scheduling and RA methods in CC based on parameters such as makespan, energy usage, balanced CPU utilization, optimized memory usage, and task prioritization. The model always performs better than FDA, LOA, CSO, and DFA for all task sizes, providing the minimum makespan (349 to 1985) and reducing energy usage (43.72 to 62.97), which will save operational costs. It keeps balanced CPU usage (0.003 to 0.008) and effective memory management (4713.05 to 5383.95), conserving resources. Further, the model is effective in task prioritization, reaching 82690 at 500 tasks, indicating that it can effectively process critical workloads. The following results demonstrate the efficiency and effectiveness of the proposed model in task scheduling problems in CC environments. Fig.3 illustrates the comparison models for different task (a) Makespan, (b) Energy Consumption, (c) Balanced CPU Utilization, (d) Optimized Memory usage, (e) Task Prioritization.

Received: August 06, 2025

International Journal of Applied Mathematics

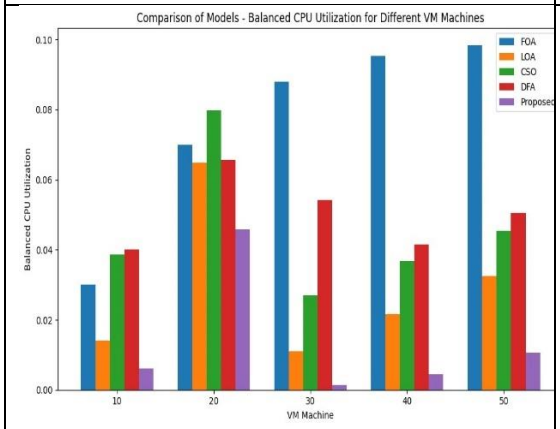
Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

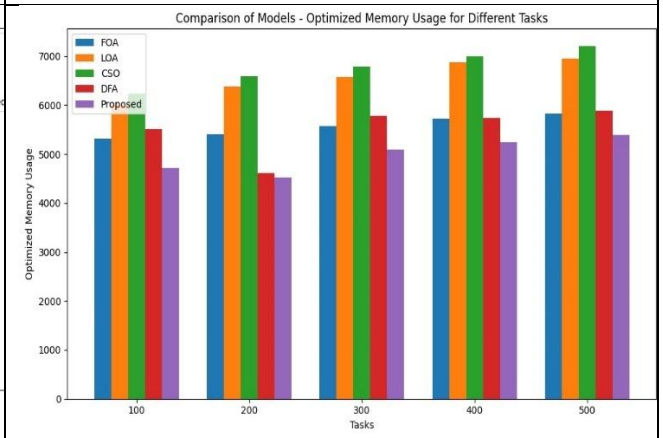


(a)

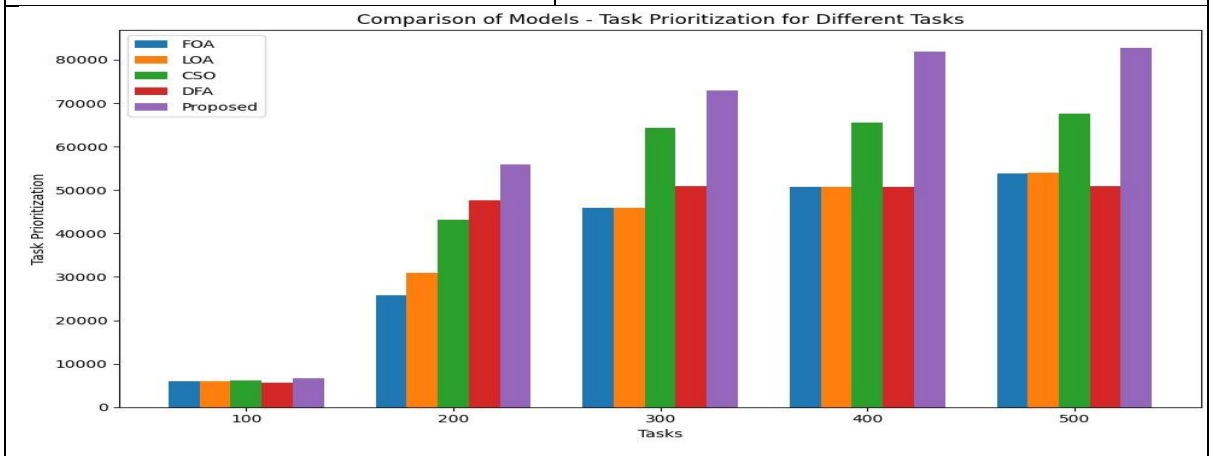
(b)



(c)



(d)



(e)

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Fig.3 illustrates the comparison models for different task (a) Makespan, (b) Energy Consumption, (c) Balanced CPU Utilization, (d) Optimized Memory usage, (e) Task Prioritization.

Figure.3 compare the performance of various models, namely FOA, LOA, CSO, DFA, and a proposed model, on various parameters such as makespan, energy usage, CPU usage, memory usage, and task scheduling. The first figure indicates the comparison in makespan, where the proposed model has a lower makespan in all cases, meaning that tasks are completed faster. The second plot depicts energy consumption, with the most efficient proposed model consuming the least amount of energy. The third plot compares CPU utilization, with the proposed model showing balanced utilization over other models. Optimized memory usage is compared in the fourth plot, with the proposed model having the lower memory usage. Finally, the fifth figure illustrates task prioritization, in which the proposed model is able to successfully prioritize tasks, leading to superior overall performance. These results prove the efficiency of the proposed model in handling computational resources. Table 2: Comparative analysis of VM machines with different evolutionary techniques.

Table 2: Comparative analysis of VM machines with different evolutionary techniques

Model	VM machine	Make span	Energy Consumption	Balanced CPU Utilization	Optimized Memory Usage	Task Prioritization
FDA [33]	10	59	7.845058	0.03	6169.16	48
	20	64	8.91672	0.07	6370.69	55
	30	67	9.659801	0.08	6466.24	60
	40	70	11.66201	0.09525	6554.25	64
	50	72	12.96201	0.09834	6825.12	65
LOA [34]	10	60	8.045058	0.014	6123.24	56
	20	66	9.325058	0.06475	6463.31	66
	30	68	10.11506	0.01096	6886.52	69
	40	71	11.21506	0.02163	6992.34	71
	50	73	13.29506	0.032564	7094.62	73
CSO [35]	10	63	9.669854	0.03856	6436.85	74

Received: August 06, 2025

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

	20	65	10.65487	0.079856	6858.36	78
	30	66	11.85746	0.0269	6894.64	80
	40	67	13.96858	0.03684	6936.98	82
	50	68	14.86955	0.045489	7098.36	85
DFA [36]	10	61	11.38761	0.04	6012.29	60
	20	64	12.84371	0.06575	6218.76	62
	30	66	13.37326	0.0541	6319.01	64
	40	68	13.59393	0.041475	6423.84	66
	50	70	14.59842	0.0505	6542.64	67
Proposed	10	49	3.387614	0.006	5912.29	82
	20	55	4.843707	0.04575	4918.76	88
	30	56	5.373257	0.00141	4719.01	94
	40	59	5.593933	0.004475	3623.84	96
	50	60	6.89842	0.0105	1742.64	97

Table 2 provides a comparative analysis of four techniques for RA and work scheduling in cloud computing. The proposed model consistently outperforms FDA, LOA, CSO, and DFA across various VM configurations, achieving a significantly lower makespan (60 at 50 VMs) and reduced energy consumption (6.89842 units). It ensures balanced CPU utilization (0.0105) and optimized memory usage (1742.64 units), leading to efficient resource management. Additionally, the proposed approach excels in task prioritization (97 at 50 VMs), ensuring effective execution of high-priority tasks. While FDA, LOA, CSO, and DFA also demonstrate competitive results, the proposed model's consistent superiority confirms its effectiveness in addressing work scheduling challenges in cloud environments. Fig. 4. Illustrates the comparison model for VM machines (a) Makespan, (b) Energy Consumption, (c) Balanced CPU Utilization, (d) Optimized Memory usage, (e) Task Prioritization.

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

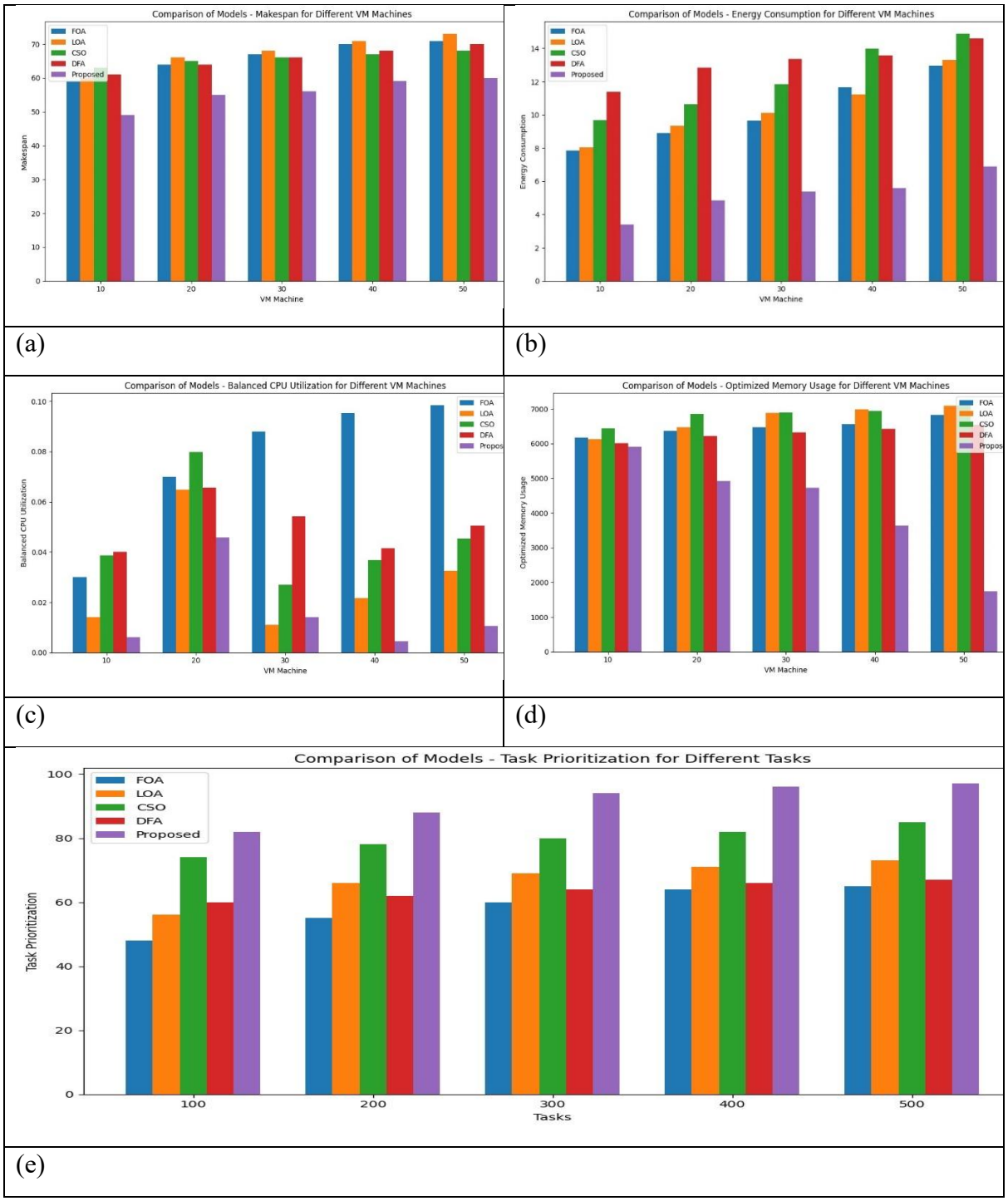


Fig. 4. Illustrates the comparison model for VM machines (a) Makespan, (b) Energy Consumption, (c) Balanced CPU Utilization, (d) Optimized Memory usage, (e) Task Prioritization.

Fig.4 compared across different tasks and virtual machines (VMs) in the performance of FOA, LOA, CSO, DFA, and the proposed model. From the first figure, the proposed model constantly

Received: August 06, 2025

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

produces a reduced makespan, meaning quicker task completion. For the second figure, the proposed model decreases the energy consumption largely, indicating improved energy efficiency. The third bar exhibits balanced CPU use, whereby the proposed model delivers more evenly allocated processing tasks. Memory efficiency-wise, as is evident in the fourth figure, the proposed model consumes less memory than other models. Finally, the fifth figure illustrates the improved task priority exhibited by the proposed model in controlling task processing. Overall, the proposed model compares favorably to current models with regard to resource efficiency and computation performance.

5.2 Discussion

The proposed HWGO model with DRL significantly enhanced the performance of load balancing and RA in cloud environments. The results demonstrated notable improvements in reducing makespan, energy consumption, and CPU utilization, while optimizing memory usage and task prioritization. This study confirms that the hybrid approach is an effective solution for dynamic cloud environments requiring efficient resource management.

This work employs a novel hybrid metaheuristic evolutionary algorithm known as HWGO Optimization to introduce an efficient LB. The dynamic traffic model of the multicast service is constructed in space-time through an effective reinforcement learning model named DRL, which is used as a basis for further network RA. The work shows that the introduced hybrid evolutionary algorithm and DRL system substantially enhance load balancing and RA in cloud computing. The important finding is that makespan and energy consumption were reduced, while the model ensured improved task scheduling over conventional practices. Evidence provided indicates that the framework recorded significant makespan reduction and energy savings, confirming the effectiveness of the framework in dynamic cloud environments. The findings prove the ability of the model to improve cloud service efficiency without loss of scalability.

The suggested framework provides load balancing and RA with a hybrid approach of OOA and WGO, which ensures minimized resource congestion and optimized VM utilization. Its self-adaptive evolutionary clustering algorithm allows dynamic load management, and DRL optimizes task assignments further, minimizing makespan and energy usage. It also exhibits scalability and efficient task prioritization within large-scale cloud environments. But the model's dependency on precise data, its high computational complexity, and the risk of convergence in DRL can be challenging. In addition, its poor generalizability to other cloud platforms and the necessity for metaheuristics and DRL expertise add to implementation complexity, with higher operational expenses.

The research seeks to improve cloud resource management through the introduction of a hybrid evolutionary algorithm and DRL model for enhanced load balancing and resource allocation. It responds to the urgent need for resource utilization optimization, energy consumption minimization, and makespan minimization. The significance is in its ability to improve cloud service provider (CSP) efficiency in dynamic environments. Nevertheless, challenges such as

Received: August 06, 2025

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

model convergence, computational complexity, and generalizability exist. There can be further research in adaptive learning methods, cross-platform portability, and optimization to boost scalability and robustness.

6 CONCLUSION

This research presents a significant contribution to the field of CC by introducing a robust framework that integrates DRL with hybrid evolutionary algorithms like OOA and WGO for LB and RA. Furthermore, combining DRL with the evolutionary HGWO model provides a strong way to investigate intricate solution spaces and fine-tune job allocation choices, enhancing the overall scalability and efficiency of the system. Moreover, the VM migration process is developed to guarantee the system's resilience against unforeseen circumstances, enhancing its dependability and accessibility. The proposed model effectively optimizes task scheduling, minimizes makespan, reduces energy consumption, and enhances CPU utilization, demonstrating its superiority over conventional evolutionary algorithms. For the research community, this approach offers a scalable and resilient solution for dynamic cloud environments, ensuring better system reliability and performance under varying workloads. The gained results of the developed model are validated with existing evolutionary algorithms and attained better CPU utilization, less energy consumption, and less execution time. In the future, hybrid evolutionary algorithms with artificial intelligence-based attack detection models can improve the performance of LB and resource scheduling techniques. Additionally, enhancing energy-aware optimization, improving fault tolerance, incorporating adaptive learning models, ensuring security, and conducting large-scale comparative analyses can further refine the framework for cost-effective and efficient resource management.

DECLARATIONS:

FUNDING

On Behalf of all authors the corresponding author states that they did not receive any funds for this project.

CONFLICTS OF INTEREST

The authors declare that we have no conflict of interest.

COMPETING INTERESTS

The authors declare that we have no competing interest.

DATA AVAILABILITY STATEMENT

All the data is collected from the simulation reports of the software and tools used by the authors. Authors are working on implementing the same using real world data with appropriate permissions.

Received: August 06, 2025

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

AUTHOR'S CONTRIBUTIONS MODEL

Author 1: Annaiah H

He participated in the methodology, Conceptualization, Data collection and writing the study

Author 2: Dr. A. Rajesh

He Performed the Analysis the overall concept, writing and editing

ETHICS APPROVAL

No ethics approval is required.

REFERENCES

[1] Ramezani Shahidani, F., Ghasemi, A., Toroghi Haghighat, A. and Keshavarzi, A., 2023. Task scheduling in edge-fog-cloud architecture: a multi-objective load balancing approach using reinforcement learning algorithm. Computing, 105(6), pp.1337-1359.

[2] Liu, L., Feng, J., Mu, X., Pei, Q., Lan, D. and Xiao, M., 2023. Asynchronous deep reinforcement learning for collaborative task computing and on-demand resource allocation in vehicular edge computing. IEEE Transactions on Intelligent Transportation Systems.

[3] Mangalampalli, S., Karri, G.R., Kumar, M., Khalaf, O.I., Romero, C.A.T. and Sahib, G.A., 2024. DRLBTSA: Deep reinforcement learning based task-scheduling algorithm in cloud computing. Multimedia Tools and Applications, 83(3), pp.8359-8387.

[4] Cui, T., Yang, R., Fang, C. and Yu, S., 2023. Deep reinforcement learning-based resource allocation for content distribution in IoT-edge-cloud computing environments. Symmetry, 15(1), p.217.

[5] Aghapour, Z., Sharifian, S. and Taheri, H., 2023. Task offloading and resource allocation algorithm based on deep reinforcement learning for distributed AI execution tasks in IoT edge computing environments. Computer Networks, 223, p.109577.

[6] Abdulazeez, D.H. and Askar, S.K., 2023. Offloading mechanisms based on reinforcement learning and deep learning algorithms in the fog computing environment. Ieee Access, 11, pp.12555-12586.

[7] Kruekaew, B. and Kimpan, W., 2022. Multi-objective task scheduling optimization for load balancing in a cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning. IEEE Access, 10, pp.17803-17818.

[8] Zhu, R., Li, G., Zhang, Y., Fang, Z. and Wang, J., 2023. Load-balanced virtual network embedding based on deep reinforcement learning for 6G regional satellite networks. IEEE Transactions on Vehicular Technology.

[9] Huang, J., Wan, J., Lv, B., Ye, Q. and Chen, Y., 2023. Joint computation offloading and resource allocation for edge-cloud collaboration in internet of vehicles via deep reinforcement learning. IEEE Systems Journal.

Received: August 06, 2025

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[10] Lin, L., Zhou, W.A., Yang, Z. and Liu, J., 2023. Deep reinforcement learning-based task scheduling and resource allocation for NOMA-MEC in Industrial Internet of Things. Peer-to-Peer Networking and Applications, 16(1), pp.170-188.

[11] Jangra, A. and Mangla, N., 2023. An efficient load balancing framework for deploying resource schedulingin cloud-based communication in healthcare. Measurement: Sensors, 25, p.100584.

[12] Zhu, X., Luo, Y., Liu, A., Xiong, N.N., Dong, M. and Zhang, S., 2021. A deep reinforcement learning-based resource management game in vehicular edge computing. IEEE Transactions on Intelligent Transportation Systems, 23(3), pp.2422-2433.

[13] Shuaib, M., Bhatia, S., Alam, S., Masih, R.K., Alqahtani, N., Basheer, S. and Alam, M.S., 2023. An optimized, dynamic, and efficient load-balancing framework for resource management in the internet of things (iot) environment. Electronics, 12(5), p.1104.

[14] Naderializadeh, N., Sydir, J.J., Simsek, M. and Nikopour, H., 2021. Resource management in wireless networks via multi-agent deep reinforcement learning. IEEE Transactions on Wireless Communications, 20(6), pp.3507-3523.

[15] Elgendy, I.A., Meshoul, S. and Hammad, M., 2023. Joint task offloading, resource allocation, and load-balancing optimization in multi-uav-aided mec systems. Applied Sciences, 13(4), p.2625.

[16] Cheng, Y., Cao, Z., Zhang, X., Cao, Q. and Zhang, D., 2024. Multi objective dynamic task scheduling optimization algorithm based on deep reinforcement learning. The Journal of Supercomputing, 80(5), pp.6917-6945.

[17] Gong, Y., Huang, J., Liu, B., Xu, J., Wu, B. and Zhang, Y., 2024. Dynamic resource allocation for virtual machine migration optimization using machine learning. arXiv preprint arXiv:2403.13619.

[18] Kardani-Moghaddam, S., Buyya, R. and Ramamohanarao, K., 2020. ADRL: A hybrid anomaly-aware deep reinforcement learning-based resource scaling in clouds. IEEE Transactions on Parallel and Distributed Systems, 32(3), pp.514-526.

[19] Seid, A.M., Boateng, G.O., Anokye, S., Kwantwi, T., Sun, G. and Liu, G., 2021. Collaborative computation offloading and resource allocation in multi-UAV-assisted IoT networks: A deep reinforcement learning approach. IEEE Internet of Things Journal, 8(15), pp.12203-12218.

[20] Peng, Z., Lin, J., Cui, D., Li, Q. and He, J., 2020. A multi-objective trade-off framework for cloud resource scheduling based on the deep Q-network algorithm. Cluster Computing, 23, pp.2753-2767.

[21] Nabi, S., Ibrahim, M. and Jimenez, J.M., 2021. DRALBA: Dynamic and resource aware load balanced scheduling approach for cloud computing. IEEE Access, 9, pp.61283-61297.

Received: August 06, 2025

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[22] Ebadifard, F. and Babamir, S.M., 2021. Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment. Cluster Computing, 24, pp.1075-1101.

[23] Ebadifard, F., Babamir, S.M. and Barani, S., 2020, April. A dynamic task scheduling algorithm improved by load balancing in cloud computing. In 2020 6th International Conference on Web Research (ICWR) (pp. 177-183). IEEE.

[24] Malarvizhi, N., Aswini, J., Sasikala, S., Chakravarthy, M.H. and Neeba, E.A., 2021. Multi-parameter optimization for load balancing with effective task scheduling and resource sharing. Journal of Ambient Intelligence and Humanized Computing, pp.1-9.

[25] Pradhan, A. and Bisoy, S.K., 2022. A novel load balancing technique for cloud computing platform based on PSO. Journal of King Saud University-Computer and Information Sciences, 34(7), pp.3988-3995.

[26] Negi, S., Rauthan, M.M.S., Vaisla, K.S. and Panwar, N., 2021. CMODLB: an efficient load balancing approach in cloud computing environment. The Journal of Supercomputing, 77(8), pp.8787-8839.

[27] Shafiq, D.A., Jhanjhi, N.Z., Abdullah, A. and Alzain, M.A., 2021. A load balancing algorithm for the data centres to optimize cloud computing applications. IEEE Access, 9, pp.41731-41744.

[28] Chen, X., Cheng, L., Liu, C., Liu, Q., Liu, J., Mao, Y. and Murphy, J., 2020. A WOA-based optimization approach for task scheduling in cloud computing systems. IEEE Systems journal, 14(3), pp.3117-3128.

[29] RB Madhumala, Harshvardhan Tiwari, Verma C Devaraj (2021).Virtual machine placement using energy efficient particle swarm optimization in cloud datacenter. Cybernetics and Information Technologies p 62-72

[30] RB Madhumala, Harshvardhan Tiwari (2022). Resource optimization in cloud data centers using particle swarm optimization. International Journal of Cloud Applications and Computing (IJCAC) p 1-12

[31] Annaiah Halappa and Rajesh A. “Performance Analysis of Load Balancing Mechanism in Cloud Computing”, International Conference on Applied Intelligence and Sustainable Computing (ICAISC) 2023, DOI: 10.1109/ICAISC58445.2023.10199269

[32] Annaiah H. , Rajesh A. , Bharani, “Survey on Load Balancing Algorithms in Cloud Environment” ,International Conference on Advances in Engineering and Technology (ICAET2023) 12 and 13, April, 2023

[33] Bouflous, Z., Ouzzif, M. and Bouragba, K., 2022, October. Analysis of load balancing algorithms used in the cloud computing environment: advantages and limitations. In Proceedings of the Future Technologies Conference (pp. 206-226). Cham: Springer International Publishing.

[34] Gong, Y., 2016. The quest for low-latency at both network edges: design, analysis, simulation and experiments (Doctoral dissertation, Télécom ParisTech).

Received: August 06, 2025

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[35] Ghasemi, M., Rahimnejad, A., Hemmati, R., Akbari, E. and Gadsden, S.A., 2021. Wild Geese Algorithm: A novel algorithm for large scale optimization based on the natural life and death of wild geese. Array, 11, p.100074.

[36] Cao, Y., Wang, Q., Wang, Z., Jermsittiparsert, K. and Shafiee, M., 2020. A new optimized configuration for capacity and operation improvement of CCHP system based on developed owl search algorithm. Energy Reports, 6, pp.315-324.

[37] Bai, G., 2024. Enhancing Harris Hawks Optimization Algorithm for Resource Allocation in Cloud Computing Environments. International Journal of Advanced Computer Science & Applications, 15(3).

[38] Khan, A.R., 2024. Dynamic Load Balancing in Cloud Computing: Optimized RL-Based Clustering with Multi-Objective Optimized Task Scheduling. Processes, 12(3), p.519.

[39] Alexander, A. A., & Joseph, D. L. (2016). An efficient resource management for prioritized users in cloud environment using cuckoo search algorithm. Procedia Technology, 25, 341-348.

[40] Amini, Z., Maeen, M. and Jahangir, M.R., 2018. Providing a load balancing method based on dragonfly optimization algorithm for resource allocation in cloud computing. International Journal of Networked and Distributed Computing, 6(1), pp.35-42.

[41] Mangalampalli, S., Karri, G. R., Kumar, M., Khalaf, O. I., Romero, C. A. T., & Sahib, G. A. (2024). DRLBTSA: Deep reinforcement learning based task-scheduling algorithm in cloud computing. *Multimedia tools and applications*, *83*(3), 8359-8387.

[42] Mao, L., Chen, R., Cheng, H., Lin, W., Liu, B., & Wang, J. Z. (2023). A resource scheduling method for cloud data centers based on thermal management. *Journal of Cloud Computing*, *12*(1), 84.

Received: August 06, 2025