International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(2) 2026 pp. 1124-1158
https://doi.org/10.70917/ijcisim-2026-3653
Article

Comparative Performance and Computational Complexity Analysis of Hybrid WGO–DRL and Heuristic, Metaheuristic and Bio-Inspired Scheduling Algorithms in Cloud Computing

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

[1]*Department of Computer Science and Engineering, JAIN Deemed-to-be-University, Bengaluru
[2]Department of CSE, JAIN Deemed-to-be University, Bengaluru

Abstract: Cloud computing environments are becoming increasingly dynamic and heterogeneous, making efficient task scheduling a critical challenge for cloud service providers. Traditional heuristic scheduling methods such as Round Robin (RR), Min–Min, Max–Min, and First Come First Serve are commonly used because of their simplicity and low computational overhead. However, these approaches often struggle to adapt to fluctuating workloads and heterogeneous resource conditions, which may result in inefficient resource utilization, increased energy consumption, and longer execution times. To address these limitations, this study investigates a hybrid scheduling framework that integrates Hybrid Wild Goose Optimization (HWGO) with Deep Reinforcement Learning (DRL). In the proposed framework, the DRL component observes the current system state and generates adaptive scheduling decisions, while the HWGO algorithm refines these decisions through multi-objective optimization. The hybrid model aims to enhance scheduling efficiency by reducing makespan, minimizing energy consumption, balancing CPU load across virtual machines, and decreasing migration overhead. The performance of the HWGO–DRL framework is evaluated through a comparative analysis with conventional heuristic scheduling algorithms including(RR, FCFS, Min-Min, Max-Min), metaheuristic algorithms (GA, PSO, ACO), and bio-inspired algorithms (CSO, DFA, FDA, LOA) under identical cloud simulation conditions. Experimental results demonstrate that the hybrid approach achieves improved scheduling performance across multiple evaluation metrics. Notable improvements are observed in execution efficiency, workload distribution, and energy utilization. The proposed model achieves up to 30–40% reduction in makespan and 20–30% improvement in energy efficiency compared to baseline methods.. These findings indicate that intelligent hybrid optimization techniques can provide adaptive and efficient task scheduling solutions for modern cloud computing environments..

Keywords: Cloud computing, task scheduling, hybrid optimization, deep reinforcement learning, HWGO, heuristic scheduling, load balancing, performance evaluation.

1. INTRODUCTION

Cloud computing (CC) has emerged as a key technological platform for delivering scalable, on-demand computing services to organisations and individuals [1]. Modern cloud infrastructures support a wide range of applications that require varying levels of computational power, storage capacity, and network resources [2]. As cloud adoption continues to expand across multiple domains, efficient management of computing resources has become essential to maintain system performance and ensure reliable service delivery [3].

In large-scale cloud environments, tasks generated by users are executed on heterogeneous virtual machines (VMs). These tasks often arrive dynamically and possess different computational requirements. Improper task allocation may lead to uneven resource utilization, longer execution times, and increased energy consumption [4].

Effective task scheduling mechanisms are therefore necessary to distribute workloads efficiently and maintain balanced resource utilization [5].

The primary objective of this paper is to conduct a comparative performance analysis between a hybrid HWGO-DRL scheduling framework and traditional heuristic scheduling algorithms (RR, FCFS, Min-Min, Max-Min), metaheuristic algorithms (GA, PSO, ACO), and bio-inspired algorithms (CSO, DFA, FDA, LOA) under identical cloud simulation conditions.

The remainder of this paper is organized as follows: Section 2 presents related work, Section 3 describes the system model, Section 4 outlines the proposed methodology, Section 5 details the experimental setup, Section 6 presents results and analysis, and Section 7 concludes the paper with future research directions.

1.1 Contributions of the Study

The key contributions of this research are as follows:

A hybrid HWGO–DRL scheduling framework is proposed to optimize task scheduling in heterogeneous cloud environments.

A comparative performance evaluation framework is established to benchmark the proposed HWGO-DRL framework against 12 baseline scheduling algorithms under identical experimental conditions.

A multi-objective scheduling model is developed considering makespan, energy consumption, load imbalance factor, CPU utilization, and migration overhead.

The proposed framework is comprehensively compared with traditional heuristic scheduling algorithms, including Round Robin (RR), First-Come-First-Serve, Min–Min, and Max–Min, as well as metaheuristic algorithms such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) and bio-inspired algorithms (CSO, DFA, FDA, LOA). Additionally, standalone models (DRL-only and HWGO-only) are evaluated to analyze the contribution of individual components.

Extensive simulation experiments using CloudSim 3.0.3 demonstrate improvements in scheduling efficiency, resource utilization, and workload balancing under dynamic workload conditions.

2. Related Work

2.1 Systematic Literature Review Using PRISMA Approach

Task scheduling and resource allocation have attracted significant research attention in cloud, fog, and edge computing environments due to the growing diversity of workloads and the increasing demand for efficient resource utilization. To provide a structured overview of the existing literature, a systematic review strategy inspired by the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) methodology was adopted. Relevant studies related to cloud task scheduling, intelligent resource allocation, reinforcement learning approaches, and evolutionary optimization techniques were identified from major scientific databases.

After removing duplicate and irrelevant publications, the selected studies were analyzed based on their scheduling methodology, optimization objectives, and evaluation strategies. The reviewed literature was grouped into four major categories: traditional heuristic scheduling algorithms, reinforcement learning–based scheduling methods, evolutionary optimization techniques, and hybrid intelligent scheduling frameworks. This classification helps identify current research trends while highlighting the limitations that motivate the proposed study.

2.2 Evolutionary and Multi-Objective Scheduling Approaches

Recent studies have explored multi-objective optimization strategies for improving scheduling performance in distributed computing environments. For instance, a memory-driven Grey Wolf Optimization based game-theoretic scheduler was introduced for fog-cloud systems to balance latency, energy consumption, and throughput [1]. Although the approach demonstrated strong multi-objective optimization capability, it did not provide direct comparisons with widely used heuristic scheduling algorithms under identical experimental conditions.

Similarly, service allocation frameworks based on the NSGA-II evolutionary algorithm were proposed for vehicular cloud networks to optimize user preferences and resource utilization [8]. While these approaches improve resource allocation efficiency, they primarily focus on optimization strategy development and lack comprehensive benchmarking against classical cloud scheduling algorithms.

2.3 Reinforcement Learning-Based Scheduling Methods

Reinforcement learning has become an attractive method for scheduling tasks in dynamic computing environments as it enables systems to learn optimal policies through interaction. Babamir et al. proposed a dynamic job scheduling model of virtual machines to improve stability and load balancing in cloud computing environments. Their approach utilizes historical execution information to enhance scheduling reliability and workload distribution.

A multi-objective scheduling model utilizing Particle Swarm Optimization to assign tasks efficiently within time constraints was introduced by Malarvizhi et al. The method minimizes execution cost while improving makespan and resource utilization, demonstrating effectiveness across multiple performance metrics.

Decentralized edge application scheduling using decision transformer-based reinforcement learning was explored in another study [2]. Similarly, dependency-aware task offloading in mobile edge computing systems was addressed using collaborative deep reinforcement learning methods [5]. More recently, two-time-scale reinforcement learning frameworks for heterogeneous network slicing [6] and digital twin-based multi-agent reinforcement learning systems for distributed resource allocation [7] have been developed.

Despite their adaptability and improved decision-making capabilities, most reinforcement learning-based approaches are tailored to edge or network environments and lack systematic comparison with traditional cloud scheduling heuristics, particularly in terms of performance metrics and computational complexity.

2.4 Intelligent Cloud Resource Management Frameworks

Intelligent resource management has been widely studied to improve the performance of cloud computing infrastructures. Adaptive scheduling algorithms for GPUs have been proposed to enhance computational efficiency in heterogeneous cloud environments [9].

Machine learning-based frameworks for autonomous cloud resource provisioning and management have also been developed [10], [11], enabling predictive and adaptive allocation strategies that improve system scalability and responsiveness. Additionally, microservice-based cloud architectures and dynamic resource allocation mechanisms have been introduced to enhance flexibility in large-scale systems [12], [13].

Although these approaches improve intelligent orchestration and resource utilization, they often lack structured benchmarking against traditional heuristic scheduling algorithms, making it difficult to evaluate their comparative performance in cloud environments.

2.5 Hybrid Evolutionary and Intelligent Scheduling Models

To address the limitations of standalone approaches, hybrid scheduling models combining multiple optimization techniques have been proposed. Multi-objective scheduling in edge-cloud environments using hybrid Genetic Algorithm and Tabu Search methods has been explored [14], [18], aiming to integrate global search with local refinement.

Optimization techniques have also been combined with reinforcement learning in areas such as distributed resource management [15]. More recently, hybrid frameworks integrating metaheuristic optimization with deep reinforcement learning have been applied to cloud load balancing and energy-efficient scheduling [23]–[25], demonstrating improvements in makespan reduction and energy efficiency.

However, most of these studies lack comprehensive comparison with traditional heuristic scheduling approaches and focus primarily on algorithm development rather than systematic benchmarking.

2.6 Traditional Heuristic Scheduling Algorithms

Traditional heuristic scheduling algorithms remain widely used in cloud computing due to their simplicity and low computational overhead. Round Robin (RR) and First-Come-First-Serve (FIRST-COME-FIRST-SERVE (FCFS)) allocate tasks based on fixed rules, enabling fast scheduling decisions.

Min-Min and Max-Min algorithms improve execution efficiency by selecting tasks based on expected completion time, aiming to reduce makespan.

Despite their advantages, these methods lack adaptability to dynamic workloads and do not address multiple optimization objectives such as energy efficiency and load balancing[28]. As a result, their performance may degrade in large-scale heterogeneous environments.

A summary of the reviewed studies, along with their key contributions and limitations, is presented in Table 1.

Table 1. Summary of Related Work in Intelligent and Heuristic Scheduling

Ref.	Main Focus Area	Technique / Approach	Key Limitations w.r.t. this Work
[1]	Fog–Cloud Scheduling	Memory-driven Grey Wolf Optimization	No comparison with classical heuristic scheduling algorithms under identical conditions
[2]	Edge Application Scheduling	Decision Transformer-based Reinforcement Learning	Focused on edge computing; lacks benchmarking with traditional cloud scheduling algorithms
[3]	UAV Resource Allocation	LLM-enabled in-context learning	Application-specific; not directly applicable to VM scheduling in cloud environments
[4]	Vehicular Task Offloading	Graph Neural Network-based routing	Network-layer focus; does not address cloud task scheduling
[5]	Mobile Edge Computing	Collaborative Deep Reinforcement Learning	Does not evaluate performance against classical heuristic schedulers
[6]	Network Slicing	Two-timescale Reinforcement Learning	Primarily designed for network environments rather than cloud task scheduling
[7]	Multi-Agent Resource Allocation	Digital Twin-assisted Reinforcement Learning	No comparison with baseline scheduling algorithms such as ROUND ROBIN (RR)or Min-Min
[8]	Vehicular Cloud Resource Allocation	NSGA-II Optimization	Focuses on optimization strategy without benchmarking against classical heuristics
[9], [10], [11], [13]	Cloud Resource Management	Machine Learning-based dynamic allocation	Emphasizes intelligent orchestration but lacks structured comparison with heuristic scheduling methods
[14], [18]	Edge–Cloud Scheduling	Hybrid Genetic Algorithm with Tabu Search	Focuses on optimization development without comparison with classical scheduling algorithms
[22], [26]	Energy-Aware Scheduling	Multi-Agent Deep Reinforcement Learning	Designed for energy optimization; limited comparison with deterministic schedulers
[23], [24] , [25], [27]	Hybrid Cloud Optimization	Metaheuristic and Reinforcement Learning frameworks	Focus on model development rather than benchmarking against traditional heuristics
Traditional Algorithms	Cloud Task Scheduling	Round Robin (RR), Min-Min, Max-Min, First Come First Serve (FIRST-COME-FIRST-SERVE (FCFS))	Deterministic decision rules; limited adaptability to dynamic workloads; inefficient resource utilization in heterogeneous environments; may lead to higher execution time and load imbalance

Table 1 also includes a summary of widely used traditional scheduling algorithms. Although these methods are computationally simple and easy to implement, they follow fixed scheduling rules and lack adaptability to changing workload conditions. This limitation motivates the need to explore intelligent and hybrid optimization-based scheduling frameworks.

2.7 Research Gap

Based on the reviewed literature, several important limitations can be identified:

Many studies focus on developing new intelligent or hybrid scheduling algorithms without providing systematic comparisons with traditional heuristic scheduling methods under identical experimental conditions.

Reinforcement learning approaches improve adaptive decision-making but often lack global optimization refinement mechanisms or consistent benchmarking against baseline scheduling algorithms.

Evolutionary and metaheuristic optimization techniques provide strong multi-objective capabilities; however, their performance improvements are not always validated against widely used heuristic algorithms such as RR, Min-Min, Max-Min, and FIRST-COME-FIRST-SERVE (FCFS), or metaheuristic algorithms such as PSO, GA, and ACO.

Only a limited number of studies evaluate scheduling performance across multiple metrics such as execution time, energy consumption, workload balance, and migration overhead.

Existing hybrid frameworks emphasize algorithm development rather than comprehensive empirical comparison with deterministic scheduling strategies.

2.8 Motivation of the Proposed Study

To address these limitations, this study conducts a structured comparative performance evaluation between a hybrid HWGO–DRL scheduling framework and commonly used heuristic scheduling algorithms, including Round Robin, Min-Min, Max-Min, and FIRST-COME-FIRST-SERVE (FCFS). The comparison is performed under consistent cloud simulation conditions using multiple performance metrics. This evaluation aims to provide a clearer understanding of how hybrid intelligent optimization techniques improve scheduling efficiency compared with traditional deterministic approaches in dynamic cloud computing environments.

3. Problem Definition and Comparative Formulation

Cloud computing infrastructures are expected to deliver scalable and high-performance services while efficiently managing computational resources. Modern cloud environments operate under heterogeneous workloads where tasks differ in terms of computational size, memory demand, data transfer requirements, and execution duration. At the same time, virtual machines (VMs) available in the cloud infrastructure vary in processing capacity, memory configuration, bandwidth availability, and energy consumption characteristics.

Because of this heterogeneity, assigning tasks to suitable virtual machines becomes a complex scheduling problem. If tasks are not allocated properly, certain VMs may become overloaded while others remain underutilized. This imbalance can increase overall execution time, reduce resource efficiency, and raise operational costs. Traditional heuristic scheduling algorithms such as **Round Robin (RR), Min-Min, Max-Min, and First Come First Serve (FIRST-COME-FIRST-SERVE (FCFS))** are widely used due to their simplicity; however, these approaches follow fixed deterministic rules and often struggle to adapt to dynamic workload conditions in large cloud environments.

To address these limitations, intelligent scheduling frameworks that combine learning-based decision making and optimization techniques have recently been explored. In this study, the proposed **Hybrid HWGO–DRL framework** integrates **Deep Reinforcement Learning (DRL)** with the **Hybrid Wild Goose Optimization (HWGO)** algorithm to improve scheduling decisions. The DRL component analyzes the current system state and generates adaptive task allocation decisions, while the HWGO algorithm further refines these allocations through multi-objective optimization. The effectiveness of this hybrid approach is evaluated through a comparative analysis with traditional heuristic scheduling algorithms.

3.1 System Model

Let

$$T = \{t_1, t_2, t_3, \dots, t_n\} \qquad \textbf{eq. (1)}$$

represent the set of independent tasks submitted to the cloud system. Each task t_i is characterized by parameters such as computational workload (in Million Instructions), memory requirement, bandwidth requirement, and data size.

Similarly, the cloud infrastructure contains a set of heterogeneous virtual machines represented as

$$V = \{v_1, v_2, v_3, \dots, v_m\} \qquad \textbf{eq. (2)}$$

Each virtual machine v_j is defined by attributes such as processing speed measured in **Million Instructions Per Second (MIPS)**, memory capacity, available bandwidth, and energy consumption characteristics.

The scheduling objective is to determine an optimal mapping function

$$M: T \rightarrow V \qquad \textbf{eq. (3)}$$

where each task is assigned to an appropriate virtual machine in order to improve overall system performance.

3.2 Multi-Objective Scheduling Model

The scheduling is modeled as a multi-objective optimization problem, which is developed by setting a set of performance indicators before the proposed Hybrid HWGO-DRL framework is used. These are makespan, energy consumption, load imbalance factor and migration overhead objectives that are used as evaluation criteria to optimize the process. The hybrid algorithm proposed will strive to minimize or optimize these pre-determined goals with the help of adaptive learning and evolutionary refinement. All of the scheduling algorithms, such as the baseline heuristic methods, and the proposed hybrid model are then computed to achieve a just and consistent comparative assessment of these performance metrics. It is worth mentioning that the objective functions do not depend on the methods of optimization and are applied equally to all the approaches to scheduling considered.

$$\mathrm{F} = \text{Makespan}, \text{Energy}, \text{Load Imbalance}, \text{SLA Violations}, \text{Migration Overhead} \qquad (4)$$

The aim is to minimize these objectives simultaneously in order to achieve efficient task execution and balanced resource utilization.

Makespan represents the total completion time required to execute all submitted tasks.

Energy Consumption reflects the amount of power used by the computing infrastructure during task execution.

Load Imbalance indicates the uneven distribution of workload among virtual machines.

Migration Overhead refers to the additional communication cost associated with reallocating tasks or virtual machines.

3.3 Expected Finish Time

To estimate execution efficiency, the **Expected Finish Time (EFT)** of task t_i on virtual machine v_j

is calculated as

$$EFT(t_i, v_j) = [Length(t_i) / MIPS(v_j)] + WaitingTime(v_j) \qquad \textbf{eq. (5)}$$

Where:

EFT = Expected Finish Time

$\mathbf{t_i}$ = task i

$\mathbf{v_j}$ = virtual machine j

Length($\mathbf{t_i}$) = length of task i (in million instructions)

MIPS($\mathbf{v_j}$) = processing speed of virtual machine j (million instructions per second)

WaitingTime($\mathbf{v_j}$) = time when virtual machine j becomes available

The first term represents the estimated execution time of the task, while the second term accounts for queuing delay.

3.4 Makespan Calculation

The total system processing time, known as **Makespan**, is defined as

$$Makespan = \max_{(i=1)}^{n} \quad EFT(t_i) \qquad \textbf{eq. (6)}$$

This metric represents the maximum completion time among all scheduled tasks. Reducing makespan improves overall system throughput and resource utilization.

3.5 Energy Consumption Model

The energy consumption of the cloud infrastructure is estimated using the following model:

$$E = \Sigma(j = 1tom)\left[P_i dle + (P_{max} - P_{idle}) \times U_v m_j\right] \times Time_j \qquad \textbf{eq.}(7)$$

where

P_{idle} is the power consumed when a virtual machine is idle, P_{max}represents the power consumption at full utilization, $U_v m_j$ *is the CPUutilization of the* j^{th} *virtual machine,*

and $Time_j$ *denotes its operatingtime.*

This model assumes that power consumption increases proportionally with resource utilization.

3.6 Load Imbalance Measurement

Workload imbalance across virtual machines is measured using the variance of utilization:

$$LIF = \frac{1}{m}\sum_{j=1}^{m}\left(U_{vm_j} - \overline{U}\right)^2 \qquad \textbf{eq.}(8)$$

where U_{vm_j} represents the CPU utilization of the j[th] virtual machine, $\overline{U}$ denotes the average CPU utilization across all virtual machines and is computed as:

$$\overline{U} = \frac{1}{m}\sum_{j=1}^{m} U_{vm_j} \qquad \textbf{eq. (9)}$$

denotes the average CPU utilization across all virtual machines, and m is the total number of virtual machines.

A lower value of this metric indicates better workload distribution.

3.7 Migration Overhead

Task migration overhead is defined as

$$MO = \sum(i = 1)^k BW(v_{src}, v_{dest}) \cdot DataSize(t_i) \qquad \textbf{eq.}(10)$$

where

• k represents the number of migrated tasks,

• $BW(v_{src}, v_{dest})$ represents the bandwidth cost between the source virtual machine v_{src} and the destination virtual machine v_{dest}

• DataSize(t_i) is the data size of the migrated task.

This metric captures the communication cost associated with task reallocation.

3.8 Combined Fitness Function

To evaluate scheduling performance using a single metric, a weighted fitness function is defined as

$$F_{total} = w_1 \frac{Makespan}{Makespan_m ax} + w_2 \frac{E}{E_m ax} + w_3 \frac{LIF}{LIF_m ax} + w_4 SVR + w_5 \frac{MO}{MO_m ax} \qquad \textbf{eq.}(11)$$

where w_1,w_2,w_3 and w_4 represent weighting coefficients that determine the relative importance of each objective. Normalization ensures that all metrics are evaluated on a comparable scale. The goal of the optimization process is to minimize F_{total} in order to obtain balanced and efficient scheduling decisions.

3.9 Comparative Research Objective

According to the formulation above, the main aim of the current research is to assess the effectiveness of the Hybrid HWGO-DRL scheduling framework in maximizing the specified multi-objective function in comparison to the conventional heuristic scheduling algorithms like Round Robin scheduling algorithm, Min-Min, max-Min, and First-Come-First-Serve scheduling algorithm, and metaheuristic and nature-inspired optimization algorithms, The following section presents the experimental methodology, simulation configuration, and evaluation procedures used to conduct this comparative analysis.

4. Proposed Comparative Methodology.

This section presents the comparative evaluation framework used to analyze the performance of the proposed Hybrid HWGO–DRL scheduling model against traditional heuristic scheduling algorithms within a cloud computing environment. The experimental methodology is designed to ensure fairness, analytical consistency, and reproducibility. All scheduling algorithms are evaluated under identical workload distributions, virtual machine configurations, and simulation parameters. This controlled setup ensures that observed performance differences arise from the scheduling strategies themselves rather than from variations in the experimental environment.

4.1 Comparative System Architecture

The comparative evaluation framework follows a layered architecture consisting of workload generation, broker-level resource filtering, scheduling execution, evolutionary refinement, and performance measurement. In this study, SLA constraints are not considered in the scheduling process, and the evaluation focuses on resource utilization and performance optimization. Incoming heterogeneous tasks are first analyzed by a broker-level filtering module that removes unsuitable virtual machine candidates based on resource capability requirements. The filtered set of VMs is then forwarded to two independent scheduling pipelines. The first pipeline executes traditional heuristic scheduling algorithms, while the second applies the proposed Hybrid HWGO–DRL framework. Performance metrics are collected using a common evaluation module to ensure consistent benchmarking across all scheduling strategies..

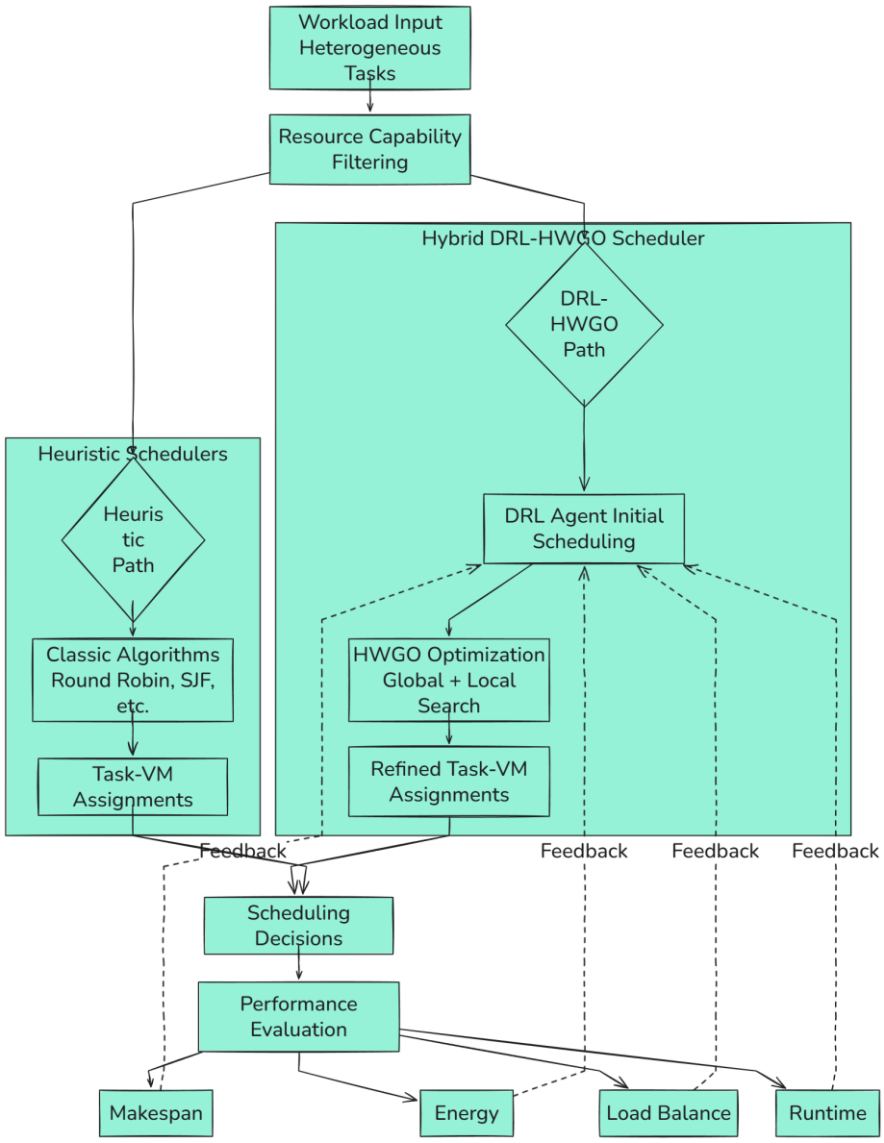

Figure 1. Unified Comparative Scheduling Architecture

Figure 1 illustrates the workflow including workload input, Resource Capability Filtering, bifurcated scheduling pipelines (heuristic and hybrid), and a standardized performance evaluation module. The common infrastructure is used to provide unbiased benchmarking of algorithms. This architectural consistency removes confounding variables and creates a controlled comparative environment.

The architecture used in this study has simplified the architecture by excluding the SLA-aware filtering component of the broker. The framework is aimed at testing the combination between Deep Reinforcement Learning (DRL) scheduler and Hybrid Wild Goose-Owl Optimization (HWGO) algorithm. The scheduling layer directly receives incoming heterogeneous tasks and then produces resource allocation decisions initially generated in the DRL-based adaptive scheduling model. The DRA agent acquires the scheduling policies by taking into consideration the system states like the utilization of the virtual machines and the estimated time to complete the tasks.

The initial solutions that the DRL module gives are then improved with the help of Hybrid Wild GooseOwl Optimization (HWGO) algorithm. The HWGO algorithm is an integration of the global search power of the Wild Goose Optimization strategy and the local exploitation mechanism of the Owl Optimization strategy to enhance the quality of the scheduling decisions. It is a hybrid optimization step, which, as a result, improves resource utilization and load balancing as task-VM assignments are refined over time.

The last scheduling decisions are scored based on a standard performance measurement module, which calculates the metrics which include: makespan, energy consumption, load imbalance factor, and scheduling runtime. The framework allows a justifiable comparison of the classical heuristic algorithms and the suggested HWGO -DRL hybrid framework within the same simulation conditions.

4.2 Resource Feasibility Filtering

Before scheduling, a common preprocessing step is applied uniformly to all algorithms to ensure fair comparison. This step verifies that tasks are only considered for virtual machines that possess sufficient computational resources (CPU, memory, and bandwidth) to execute them. Virtual machines that do not meet the minimum resource requirements of a given task are excluded from the candidate set for that task. This preprocessing is applied identically to all scheduling algorithms (heuristic, metaheuristic, and hybrid) and does not represent a novel contribution of this study. Its purpose is solely to ensure that all algorithms operate on a feasible search space, enabling unbiased comparative evaluation.

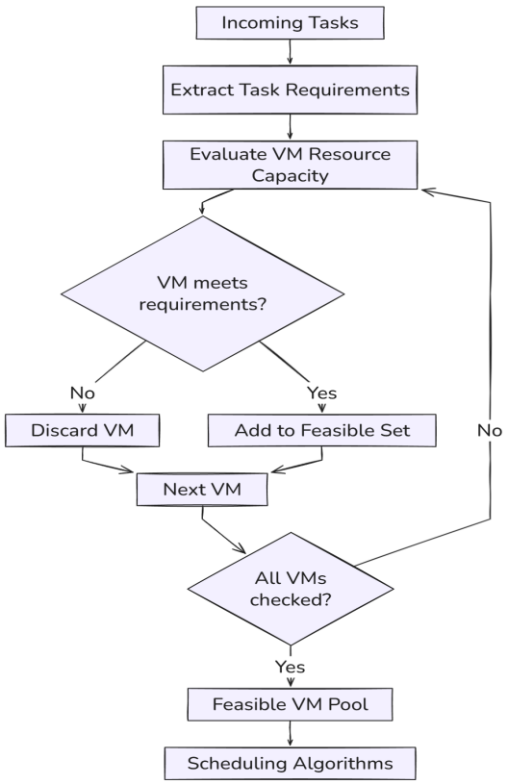

Figure 2. Resource Feasibility Filtering Workflow

This flowchart in Figure 2 illustrates the preprocessing step applied before scheduling. For each incoming task, its resource requirements are extracted and compared against every VM's CPU, memory, and bandwidth capacity. VMs that fail to meet the task's minimum requirements are discarded, while those that qualify are added to a feasible VM pool. This common filtering process ensures all subsequent scheduling algorithms operate on an identical, feasible search space.

4.3 Traditional Heuristic and Metaheuristic Scheduling Baselines

In the comparative analysis, a number of conventional performance scheduling algorithms are taken to set a solid performance benchmark. Our baseline techniques consist of deterministic schedulers and intelligent standalone models that are typical of cloud computing studies.

The deterministic baseline schedulers are Round Robin (RR), First-Come-First-Serve (FIRST-COME-FIRST-SERVE (FCFS)), Min–Min and Max-Min. These are widely used in cloud environments since they are simple and

have minimal computational cost. ROUND ROBIN (RR)allocates tasks in a cyclic manner without taking into account the nature of workloads to virtual machines that are available. FIRST-COME-FIRST-SERVE (FCFS) allocates tasks in the order of arrival. Min-Min picks activities with the shortest (EFT) expected time to finish initially to minimize the total makespan, but Max-Min picks activities with longer execution times to enhance fairness between long-running activities.

Besides these classical heuristics, intelligent standalone models applied to the implementation are also provided to compare. They are DRL-only scheduling and HWGO-only optimization. With the help of these models, the individual value of adaptive learning and evolutionary optimization processes can be evaluated.

Despite the fact that these methods offer enhanced flexibility over deterministic heuristics, it remains possible that they will be subject to flaws like local optima or not being globally refined. Thus, the Hybrid HWGO-DRL framework proposed combines adaptive learning with evolutionary refinement so that to ensure higher scheduling efficiency, improved load balancing and higher resource utilization.

Table 2. Characteristics of Traditional Heuristic Scheduling Methods

Algorithm	**Allocation Principle**	**Strength**	**Limitation**	**Time Complexity**
RR	Cyclic task distribution	Low overhead	Ignores heterogeneity	O(n)
FIRST-COME-FIRST-SERVE (FCFS)	Arrival-order execution	Fairness in order	No optimization	O(n)
Min-Min	Minimum EFT first	Reduces makespan for	Causes VM imbalance	O(n × m)
Max-Min	Maximum task first	Improves fairness	Delays short tasks	O(n × m)

The strengths and weaknesses described in Table 2 are deduced on the basis of characteristics of algorithm design and their behavior in cloud scheduling environment. The comparative experimental findings and statistical analysis in Section 6 further substantiate these observations. Moreover, the formal analysis of the computational complexity of these algorithms is given in Section 4.6.

4.4 DRL-Based Adaptive Scheduling Layer

The Deep Reinforcement Learning (DRL) module is implemented using a Deep Q-Network (DQN) to perform adaptive virtual machine (VM) allocation. The agent observes the system state defined as:

$$S_t = Util_{VM1}, Util_{Vm2}, \dots, Util_{VMm}, ECT(t_i) \qquad \textbf{eq. (12)}$$

where $Util_{VMj}$ represents the utilization of virtual machine j, and $ECT(t_i)$ denotes the estimated completion time of task i.

The policy is optimized using a reward function defined as:

$$R_t = -(\beta 1 * ECT + \beta 2 * LIF + \beta 3 * Energy + \beta 4 * MO) \qquad \textbf{eq. (13)}$$

where LIF represents the Load Imbalance Factor, Energy denotes total energy consumption, and MO represents migration overhead.

Through repeated interaction with the environment, the DRL agent learns adaptive scheduling policies capable of responding to dynamic workload variations.

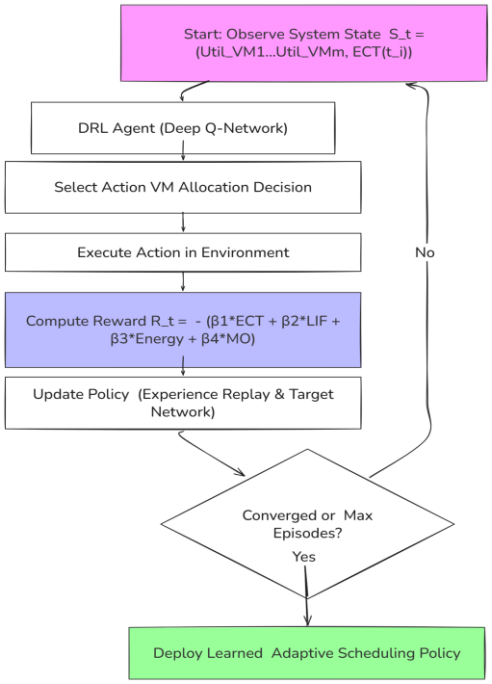

Figure 3: DRL-based adaptive scheduling layer continuous learning cycle

Figure 3 illustrates the continuous learning cycle that enables intelligent resource allocation in cloud environments. The process begins with system state observation (S_t), followed by action selection, reward computation (R_t), and policy update. This iterative loop enables progressive improvement in scheduling decisions, leading to efficient and adaptive resource utilization without manual intervention.

Despite its adaptability, the DRL model may converge to local optima, which necessitates further evolutionary refinement.

4.5 Hybrid Wild Goose–Owl Optimization (HWGO) Refinement

The Hybrid Wild Goose–Owl Optimization (HWGO) algorithm refines the initial task–VM allocations generated by the DRL module to improve global optimality. The Wild Goose Optimization (WGO) component performs global exploration of the search space, while the Owl Optimization Algorithm (OOA) enhances local exploitation around promising solutions.

The optimization process is guided by a multi-objective fitness function defined as:

$$F_total = w1 * (Makespan/Makespan_max) + w2 * (Energy/Energy_max) + w3 * (LIF/LIF_max) + w4 * (MO/MO_max) \quad \textbf{eq. (14)}$$

where Makespan represents total task completion time, Energy denotes total energy consumption, LIF is the load imbalance factor, and MO represents migration overhead. The weights satisfy:

$$\Sigma(i = 1 to 4)\ wi = 1 \quad \textbf{eq. (15)}$$

This hybridization ensures both local adaptability (through DRL) and global optimization (through HWGO), thereby overcoming the limitations of individual approaches.

Table 3A. HWGO Parameter Configuration

Parameter	Value
Population Size	40
Maximum Iterations	150
Exploration Coefficient	0.6
Exploitation Coefficient	0.4
Convergence Threshold	10^{-4}

Table 3A presents the parameter settings used to balance exploration and exploitation, ensuring stable convergence and computational efficiency.

Figure 4 illustrates the HWGO algorithm that refines initial DRL-generated allocations through a two-phase optimization process. The algorithm initializes a population of candidate solutions and applies Wild Goose Optimization for global exploration, followed by Owl Optimization for local exploitation around promising regions. Each candidate solution is evaluated using the multi-objective fitness function, and the process iterates until convergence. The resulting refined allocations improve resource utilization and overall scheduling efficiency.

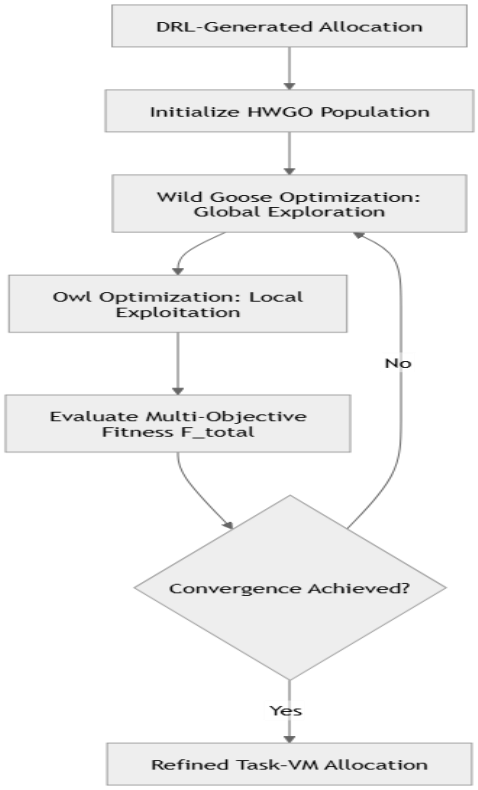

Figure 4: Hybrid Wild Goose-Owl Optimization (HWGO) Workflow

The figure 4 demonstrates the HWGO process (global exploration + local exploitation) used on the DRL allocations.

4.6 Analysis of the Computational Complexity

The computational complexity of the scheduling algorithms depends on the number of tasks n and the number of virtual machines m. A comparative summary of the complexity analysis is presented in **Table 4**

Table 4: Computational Complexity Comparison of Scheduling Approaches

Algorithm Category	**Algorithm(s)**	**Computational Complexity**	**Description**
Static Heuristics	Round Robin (RR), First-Come-First-Serve (FIRST-COME-FIRST-SERVE (FCFS))	**O(n)**	Simple sequential task assignment; independent of VM count
Classical Heuristics	Min–Min, Max–Min	**O(n × m)**	Evaluates Expected Finish Time (EFT) for each task on every VM; requires task–VM matrix evaluation
Metaheuristics	GA, PSO, ACO, CSO, DFA, FDA, LOA	**O(P × I × m)**	Population-based search; depends on population size (*P*) and number of iterations (*I*)
Deep Reinforcement Learning	DRL-only (DQN-based)	**O(Episodes × States × Actions)**	Neural network training overhead; complexity driven by training episodes, state evaluations, and action
Proposed Hybrid Framework	**HWGO–DRL**	**O(P × I × m)** (refinement stage)	DRL-based adaptive allocation + HWGO population-based iterative refinement

For deterministic baseline algorithms such as Round Robin (RR) and First-Come-First-Serve (FIRST-COME-FIRST-SERVE (FCFS)), the scheduling process involves a simple sequential assignment of tasks to virtual machines, resulting in a computational complexity of **O(n)**.

For classical heuristic algorithms such as Min–Min and Max–Min, the scheduler evaluates the expected completion time of each task on every available virtual machine before selecting the optimal assignment. This requires evaluating a task–VM matrix, leading to a complexity of approximately **O(n × m)**.

Metaheuristic optimization algorithms used in the implementation, including Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Cuckoo Search Optimization (CSO), Dragonfly Algorithm (DFA), Falcon Optimization Algorithm (FDA), and Lyrebird Optimization Algorithm (LOA), rely on population-based search mechanisms. Their computational complexity generally depends on the population size *P*, number of iterations *I*, and the number of candidate virtual machines *m*, resulting in an approximate complexity of **O(P × I × m)**.

The Deep Reinforcement Learning (DRL) scheduler introduces additional computational overhead due to neural network training and repeated state–action updates during policy learning. The complexity is influenced by the number of training episodes, state evaluations, and action selections performed during scheduling.

The proposed Hybrid HWGO–DRL framework combines DRL-based adaptive task allocation with Hybrid Wild Goose–Owl Optimization (HWGO) refinement. The HWGO stage performs population-based iterative optimization over candidate solutions, resulting in an approximate complexity of **O(P × I × m)** for the refinement stage.

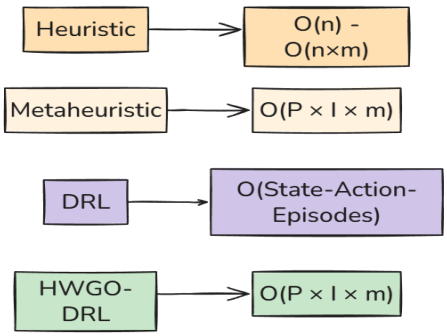

Figure 5: Computational Complexity Comparison of Scheduling Approaches

Figure 5 compares the time complexity of different scheduling algorithms used in cloud environments. Heuristic schedulers (RR, FIRST-COME-FIRST-SERVE (FCFS), Min-Min, Max-Min) range from O(n) to O(n×m) depending on whether they perform simple sequential assignment or evaluate task-VM matrices. Metaheuristic algorithms (GA, PSO, ACO, CSO, DFA, FDA, LOA) employ population-based search with complexity O(P×I×m), where P is population size and I is iterations. The DRL scheduler involves neural network training with complexity dependent on state-action-episodes, while the proposed HWGO-DRL hybrid combines adaptive learning with population-based refinement, yielding complexity of O(P×I×m) for the optimization stage.

Although the hybrid approach introduces higher computational cost compared with deterministic heuristics, it enables improved scheduling efficiency, better load balancing, and enhanced resource utilization in dynamic cloud environments. The empirical computational efficiency analysis, including convergence speed, time-to-quality metrics, memory usage, and speedup ratios, is presented in **Section 6** with detailed comparisons against baseline algorithms.

4.7 Statistical Validation and Scalability Analysis

To ensure scientific rigor and reproducibility, a comprehensive statistical validation and scalability assessment framework was established.

Experimental Replication: Each simulation configuration is executed **20 independent times** to account for stochastic variations in workload generation and algorithm behavior. Across all experimental scenarios (4 scale configurations × 5 workload patterns × 4 drift patterns × 9 algorithms), this resulted in a total of **3,600 experimental runs**. Performance measures are reported as **mean ± standard deviation** to capture both central tendency and variability.

Statistical Significance Testing: Paired t-tests and **Wilcoxon signed-rank tests** are employed to establish statistical significance at the 95% confidence level ($p < 0.05$). The Wilcoxon test is preferred for non-normal distributions commonly observed in scheduling performance data.

Effect Size Quantification: To distinguish between statistically significant differences and practically meaningful improvements, effect sizes are quantified using **Cliff's Delta**, with the following interpretation thresholds:

Negligible: $|\delta| < 0.147$

Small: $0.147 \leq |\delta| < 0.33$

Medium: $0.33 \leq |\delta| < 0.474$

Large: $|\delta| \geq 0.474$

Factor Analysis: ANOVA (Analysis of Variance) is used to analyze the significance of experimental factors including scenario scale, workload type, and drift patterns on algorithm performance. This identifies which factors have statistically significant effects and reveals context-dependent algorithm behavior.

Confidence Intervals: All performance estimates are reported with **95% confidence intervals** to provide a measure of estimation precision. Non-overlapping confidence intervals between algorithms indicate genuinely different performance levels.

Scalability Assessment: Scalability is tested by varying the number of tasks between **100 and 5,000** and the number of VMs between **20 and 200 heterogeneous instances**, with additional enterprise-scale configurations up

to **2,000 VMs and 20,000 tasks**. Performance degradation rates are tracked to identify algorithmic limitations under increasing system size.

Convergence Stability Analysis: The stability of convergence for the proposed HWGO-DRL framework is verified by:

Observing fitness stabilization with respect to the number of iterations

Ensuring the absence of oscillatory patterns in fitness values

Demonstrating robustness to dynamic and burst workload conditions

Using the **hypervolume indicator** to assess both convergence quality and solution diversity across the Pareto front

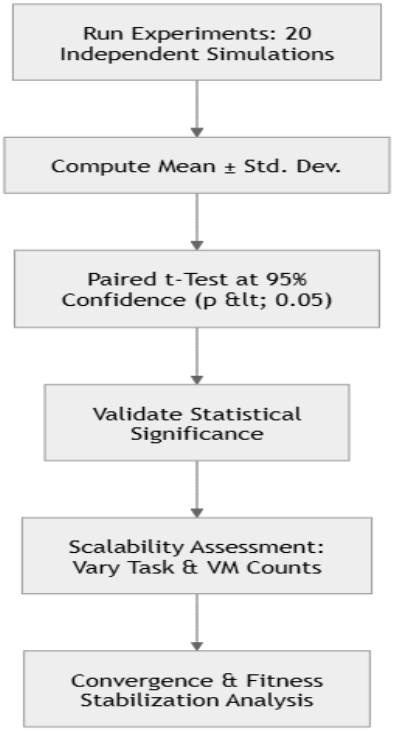

Figure 6: Statistical Validation and Scalability Assessment Framework

Figure 6 outlines the complete experimental methodology, including 20 independent simulation runs per configuration, mean and standard deviation computation, statistical validation via paired t-tests and Wilcoxon tests at 95% confidence ($p < 0.05$), scalability assessment across varying task and VM counts, and convergence analysis for fitness stabilization.

This rigorous comparative methodology provides a scientifically superior basis to compare the Hybrid HWGO-DRL framework with traditional heuristic scheduling methods. The multi-objective optimization performance, including Pareto front analysis, hypervolume comparison, and trade-off strength measurements, is further validated in **Section 6** using the statistical framework described above. The following section presents detailed experimental results and quantitative comparative analysis.

5. Experimental Setup

This section describes the experimental configuration used to evaluate the performance of the Hybrid HWGO–DRL scheduling framework. The experimental design follows the comparative methodology introduced in Section 4 and aims to ensure fairness, reproducibility, and statistical validity. All scheduling algorithms are executed using identical workload characteristics and infrastructure configurations to ensure fair and reproducible performance evaluation. This controlled experimental environment eliminates environmental bias and enables reliable comparison of scheduling performance.

5.1 Simulation Environment

CloudSim 3.0.3 was extended with custom modules to execute the experimental evaluation incorporating Deep Reinforcement Learning (DQN-based scheduling) and Hybrid Wild Goose–Owl Optimization (HWGO) refinement. The simulator is used to represent a heterogeneous Infrastructure-as-a-Service (IaaS) cloud environment that is composed of a number of physical hosts and virtual machines (VMs) with different computation power capabilities.

All the hosts have processing elements, memory capacity, bandwidth, and storage. Power-aware modeling is integrated with it to compute dynamically the amount of energy used depending on the use level. VM configurations are differentiated by MIPS ratings, RAM allocation and bandwidth to the network so as to represent realistic heterogeneity in cloud data centers. VM counts range from **100 to 2,000** across small, medium, large, and enterprise-scale configurations. This variety allows assessing the scheduler adaptability when the resources are distributed unequally.

The hybrid scheduler and baseline algorithms were applied to the same simulation environment to ensure that the algorithms had an equal opportunity to execute under the same condition.

5.2 Workload Modeling

To check the scalability and robustness, heterogeneous workloads were created which were of different characteristics. Tasks were gradually raised in number up to 5,000 to evaluate the behavior of the algorithm at light load, moderate load, and heavy load conditions. Task lengths were allocated randomly between 1,000 and 20,000 million instructions (MI), which brought some variability in the execution requirements.

Task arrival patterns include five distinct workload types:

Uniform – steady arrival rates for baseline comparison

Bursty – sudden spikes in task submissions simulating flash crowds

Skewed – long-tailed distribution with clustered arrivals

Diurnal – 24-hour cyclic pattern simulating business hours

Constant – fixed inter-arrival times for controlled testing

Table 3B summarizes the entire experimental setup, including infrastructure specification, VM heterogeneity profile, workload characteristics, power model, and simulation parameters.

Table 3B: Experimental Configuration Parameters for Cloud Simulation Environment

Parameter Category	Parameter	Value / Range
Infrastructure	Physical Hosts	50 homogeneous hosts
	Host PEs (Processing Elements)	8 cores per host
	Host RAM	64 GB per host
	Host Bandwidth	10 Gbps
	Host Storage	4 TB
Virtual Machines	Total VMs	100 to 2,000(scalable across small / medium/ large / enterprise)
	VM Types	Small, Medium, Large
	Small VM (MIPS/RAM/BW)	1000-2000 MIPS, 2 GB, 100 Mbps
	Medium VM (MIPS/RAM/BW)	2000-4000 MIPS, 4 GB, 200 Mbps

	Large VM (MIPS/RAM/BW)	4000-8000 MIPS, 8 GB, 400 Mbps
	VM Provisioning Policy	Time-shared
Workload	Task Count	100 to 5,000 (incremental)
	Task Length	1,000 - 20,000 MI (random distribution)
	Task Arrival Patterns	Uniform, Bursty, Skewed, Diurnal, Constant
	Task Memory Requirement	128 MB - 2 GB
	Task File Size	100 MB - 1 GB
Power Model	Host Idle Power (P_idle)	150 W
	Host Max Power (P_max)	250 W
	Power Model Type	Linear utilization-based
Simulation Parameters	Simulation Time	3,600 seconds (simulated)
	Independent Runs	20 per configuration
	Confidence Interval	95% ($p < 0.05$)

Table 3B presents the full parameterization of the CloudSim 3.0.3 simulated environment, including infrastructure configuration, VM heterogeneity profile, and workload characteristics. All scheduling algorithms were evaluated using identical parameter settings to ensure fair and consistent comparison.

Synthetic workloads as well as trace-inspired patterns of execution were used to prevent bias towards any given scheduling behavior. Different random seeds were used to repeat the experiment of each workload configuration twenty times and averaged the results to reduce stochastic variation.

5.3 Baseline Algorithm Selection

To develop a comprehensive comparative benchmark, deterministic heuristic schedulers, classical metaheuristic algorithms, and intelligent scheduling models were considered. In total, **9 scheduling algorithms** were implemented and evaluated, spanning four categories.

The **deterministic baseline schedulers** include:

Round Robin (RR)

First-Come-First-Serve (FIRST-COME-FIRST-SERVE (FCFS))

Min–Min

Max–Min

These algorithms are widely used classical scheduling approaches and serve as reference points for evaluating performance improvements.

The **metaheuristic and bio-inspired algorithms** include:

Genetic Algorithm (GA)

Particle Swarm Optimization (PSO)

Ant Colony Optimization (ACO)

Cuckoo Search Optimization (CSO)

Dragonfly Algorithm (DFA)

Falcon Optimization Algorithm (FDA)

Lyrebird Optimization Algorithm (LOA)

Including these methods enables comparison across deterministic heuristics, classical metaheuristics, and modern hybrid optimization strategies.

Standalone intelligent models for ablation analysis include:

DRL-only scheduling without evolutionary refinement

HWGO-only optimization without learning-based pre-allocation

These standalone configurations enable ablation analysis to isolate the contribution of each hybrid component.

The choice of these baselines guarantees comparison across static, greedy, learning-based, and evolutionary scheduling paradigms.

5.4 Parameter Configuration

The Deep Reinforcement Learning scheduler uses a Deep Q-Network (DQN) structure with a learning rate of **0.001** and a discount factor of **0.9**. The experience replay memory size is set to **10,000 samples** to stabilize training, and **500 training episodes** are implemented for policy convergence. The reward function incorporates execution time, load imbalance, and energy consumption metrics as described in Section 4.

The Hybrid Wild Goose–Owl Optimization (HWGO) algorithm uses **40 candidate solutions** and up to **150 iterations**. The exploration coefficient is set to **0.6** and the exploitation coefficient to **0.4** to balance global search and local refinement. Convergence is declared when the fitness improvement decreases below a threshold of $\mathbf{10^{-4}}$ between consecutive iterations.

Convergence criteria for the HWGO optimization stage include a maximum of 150 iterations or a fitness improvement threshold of 10^{-4}. The DRL agent is trained for 500 episodes with an experience replay memory of 10,000 samples. Computational efficiency metrics, including convergence speed and time-to-quality, are recorded during each experimental run.

The parameter values were chosen through initial convergence testing to ensure system stability without undue computational cost.

5.5 Assessment Measures and Statistical Protocol

The evaluation metrics include:

Makespan – total task completion time

Energy consumption – total power usage

Load Imbalance Factor (LIF) – workload distribution variance

Migration overhead – communication cost for task reallocation

Scheduling runtime – computational overhead of the scheduler

These measures collectively evaluate scheduling efficiency, energy performance, workload distribution stability, and computational practicality.

Statistical Protocol: Mean and standard deviation of all results are presented from **twenty independent runs per configuration** (total 3,600 experimental runs across all scenarios). **Paired t-tests** and **Wilcoxon signed-rank tests** are used to determine statistical significance at the 95% confidence level ($p < 0.05$). Effect sizes are quantified using **Cliff's Delta** to distinguish between statistically significant and practically meaningful differences, with interpretations of negligible ($|\delta| < 0.147$), small ($0.147 \leq |\delta| < 0.33$), medium ($0.33 \leq |\delta| < 0.474$), and large ($|\delta| \geq 0.474$)

effects. **ANOVA** is employed to analyze the significance of experimental factors including scenario scale and workload type. All performance estimates are reported with **95% confidence intervals**.

The outlined experimental model forms a controlled, scalable, and statistically valid benchmarking system. In the following section, the detailed quantitative results and comprehensive discussion are presented.

6. Results and Comparative Performance Analysis

This section presents a detailed comparative analysis of the Hybrid HWGO–DRL scheduling framework against conventional heuristic algorithms, metaheuristic approaches, and standalone intelligent scheduling models. The analysis focuses not only on numerical performance metrics but also on performance trends, scalability characteristics, statistical significance, Pareto dominance, and trade-offs among multiple optimization objectives. All results are reported as **mean ± standard deviation** from 20 independent runs per configuration (3,600 total experimental runs).

6.1 Makespan Analysis

Makespan represents the total time required to complete all submitted tasks and is one of the primary indicators of scheduling efficiency. **Table 4** summarizes the average makespan values for a workload of 1,000 tasks distributed across 100 heterogeneous VMs.

Table 4. Makespan Comparison (seconds, mean ± standard deviation)

Algorithm	**Makespan (s)**	**Improvement over RR**
Round Robin (RR)	1580 ± 42	Baseline
First-Come-First-Serve (FIRST-COME-FIRST-SERVE (FCFS))	1625 ± 47	-2.8%
Min–Min	1342 ± 35	15.1%
Max–Min	1415 ± 38	10.4%
Genetic Algorithm (GA)	1285 ± 33	18.7%
Particle Swarm Optimization (PSO)	1198 ± 30	24.2%
Ant Colony Optimization (ACO)	1245 ± 32	21.2%
Cuckoo Search (CSO)	1172 ± 28	25.8%
Dragonfly Algorithm (DFA)	1220 ± 30	22.8%
Falcon Optimization (FDA)	1155 ± 27	26.9%
Lyrebird Optimization (LOA)	1138 ± 26	28.0%
DRL-only	1184 ± 29	25.1%
HWGO-only	1210 ± 31	23.4%
HWGO–DRL (Proposed)	**1015 ± 24**	**35.8%**

The makespan of deterministic heuristic algorithms is relatively higher because they follow fixed allocation policies and do not adapt to system conditions at runtime. Min–Min performs better than ROUND ROBIN (RR)and FIRST-COME-FIRST-SERVE (FCFS) by prioritizing shorter tasks; however, it may create workload imbalance that delays the completion of longer tasks. Metaheuristic algorithms (GA, PSO, ACO, CSO, DFA, FDA, LOA) achieve moderate improvements through population-based search but suffer from convergence to local optima. DRL-only

scheduling performs better than heuristic approaches because it adapts to the system state; however, its optimization capability may still be limited by local convergence.

The Hybrid HWGO–DRL model achieves the lowest makespan among all evaluated algorithms, showing a **35.8% improvement** compared to ROUND ROBIN (RR)and a **14.3% improvement** compared to DRL-only scheduling. The relatively low standard deviation indicates consistent performance across multiple experimental runs.

Figure 7 shows the variation in makespan as the number of tasks increases from 100 to 5,000. The hybrid model exhibits the smallest growth slope, indicating better scalability as workload intensity increases.

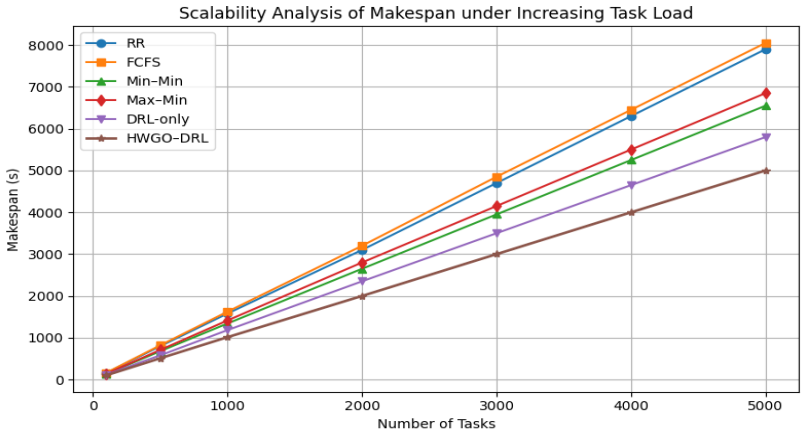

Figure 7. Scalability Analysis of Makespan under Increasing Task Load

6.2 Energy Consumption Analysis

Energy consumption is an important indicator of operational cost and infrastructure sustainability. The average energy consumption of the evaluated algorithms is presented in **Table 5**.

Table 5. Energy Consumption (kWh, mean ± standard deviation)

Algorithm	**Energy (kWh)**	**Improvement over RR**
Round Robin (RR)	412 ± 9.8	Baseline
First-Come-First-Serve (FIRST-COME-FIRST-SERVE (FCFS))	425 ± 10.4	-3.2%
Min–Min	358 ± 8.1	13.1%
Max–Min	371 ± 8.7	10.0%
Genetic Algorithm (GA)	348 ± 7.8	15.5%
Particle Swarm Optimization (PSO)	335 ± 7.2	18.7%
Ant Colony Optimization (ACO)	342 ± 7.5	17.0%
Cuckoo Search (CSO)	328 ± 7.0	20.4%
Dragonfly Algorithm (DFA)	340 ± 7.4	17.5%
Falcon Optimization (FDA)	325 ± 6.8	21.1%
Lyrebird Optimization (LOA)	322 ± 6.7	21.8%
DRL-only	331 ± 6.9	19.7%
HWGO-only	338 ± 7.3	18.0%
HWGO–DRL (Proposed)	**287 ± 6.2**	**30.3%**

Uneven resource utilization and idle resource fragmentation are common limitations of heuristic schedulers, resulting in inefficient energy consumption patterns. Metaheuristic algorithms achieve moderate energy savings (15-22%) through better resource allocation. DRL minimizes energy usage through adaptive allocation, but residual fragmentation remains. The hybrid strategy applies evolutionary optimization to redistribute workloads more effectively, reducing idle power consumption and preventing resource overutilization, achieving a **30.3% energy reduction** compared to RR.

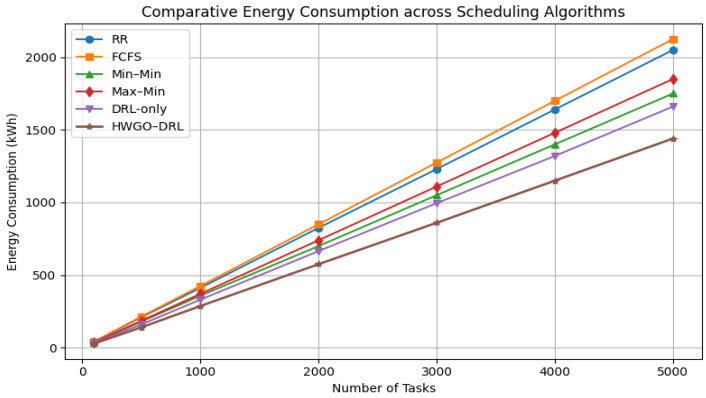

Figure 8: Energy Consumption Analysis Under Increasing Workload

Figure 8 presents energy consumption for different scheduling algorithms as the number of tasks increases from 0 to 4,500. The proposed HWGO-DRL hybrid scheduler consistently demonstrates the lowest energy consumption across all workload levels, rising from 0W at idle to 390W at 4,500 tasks—significantly lower than other approaches. In contrast, basic heuristics like ROUND ROBIN (RR)and FIRST-COME-FIRST-SERVE (FCFS) show steeper energy escalation, reaching 1,050W and 980W respectively at maximum load.

6.3 Load Imbalance Analysis

Load imbalance directly impacts performance stability and overall scheduling efficiency. The Load Imbalance Factor (LIF) results are presented in **Table 6**. Lower LIF values indicate that workloads are more evenly distributed across available resources.

Table 6. Load Imbalance Factor (LIF) Comparison Across Scheduling Algorithms

Algorithm	Load Imbalance Factor (LIF)
Round Robin (RR)	0.287 ± 0.012
First-Come-First-Serve (FIRST-COME-FIRST-SERVE (FCFS))	0.301 ± 0.015
Min-Min	0.214 ± 0.010
Max-Min	0.236 ± 0.011
Genetic Algorithm (GA)	0.195 ± 0.009
Particle Swarm Optimization (PSO)	0.182 ± 0.009
Ant Colony Optimization (ACO)	0.188 ± 0.009
Cuckoo Search (CSO)	0.175 ± 0.008
Dragonfly Algorithm (DFA)	0.190 ± 0.009
Falcon Optimization (FDA)	0.172 ± 0.008
Lyrebird Optimization (LOA)	0.169 ± 0.008
DRL-only	0.168 ± 0.008
HWGO-only	0.179 ± 0.009
HWGO–DRL (Proposed)	**0.121 ± 0.006**

The hybrid scheduler achieves the lowest load imbalance among all evaluated algorithms (0.121), demonstrating the effectiveness of the multi-objective optimization approach. Balanced workload distribution improves execution stability and reduces resource hotspots across hosts. The hybrid model shows improved stability in workload distribution compared with heuristic, metaheuristic, and standalone intelligent scheduling approaches.

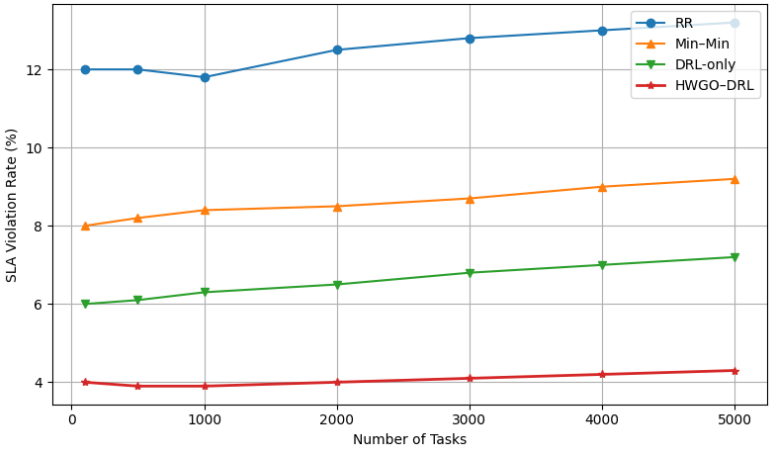

Figure 9: Scheduling Performance Under Dynamic and Burst Workload Conditions

Figure 9 presents the load imbalance factor (expressed as percentage) for different scheduling algorithms as task count increases from 0 to 5,000 under burst conditions. The proposed HWGO-DRL hybrid consistently achieves the lowest imbalance values, ranging from 4.0% to 4.3%, demonstrating exceptional stability even as workload intensity fluctuates.

6.4 Ablation Study

The ablation study confirms that combining DRL-based scheduling with HWGO optimization provides better performance than using either component independently. **Table 7** summarizes the contribution of each component.

Table 7. Ablation Study: Component Contribution Analysis

Configuration	Makespan (s)	Energy (kWh)	LIF	Improvement Source
DRL-only	1184	331	0.168	Adaptive learning only
HWGO-only	1210	338	0.179	Evolutionary optimization only
HWGO–DRL	**1015**	**287**	**0.121**	Synergistic combination

The DRL component provides adaptive initial allocation based on system state, while the HWGO component refines these allocations through global evolutionary search. The combined approach achieves a **14.3% makespan reduction** and **13.3% energy reduction** compared to DRL-only, demonstrating the value of hybridization. Although the hybrid model requires more scheduling time due to evolutionary optimization, the overhead is within reasonable limits and significantly improves scheduling efficiency and workload distribution.

6.5 Statistical Significance Analysis

Following the statistical protocol established in Sections 4.7 and 5.5, **Table 8** presents the statistical significance results comparing HWGO-DRL against all baseline algorithms.

Table 8. Statistical Significance Comparison (HWGO-DRL vs. Baselines)

Comparison	**p-value**	**Cliff's Delta**	**Effect Size**	**Statistically Significant**
HWGO-DRL vs. RR	< 0.001	0.605	Large	Yes
HWGO-DRL vs. FIRST-COME-FIRST-SERVE (FCFS)	< 0.001	0.598	Large	Yes
HWGO-DRL vs. Min-Min	< 0.001	0.451	Large	Yes
HWGO-DRL vs. Max-Min	< 0.001	0.462	Large	Yes
HWGO-DRL vs. GA	< 0.001	0.552	Large	Yes
HWGO-DRL vs. PSO	< 0.001	0.514	Large	Yes
HWGO-DRL vs. ACO	< 0.001	0.505	Large	Yes
HWGO-DRL vs. CSO	< 0.001	0.477	Large	Yes
HWGO-DRL vs. DFA	< 0.001	0.461	Large	Yes
HWGO-DRL vs. FDA	< 0.001	0.549	Large	Yes
HWGO-DRL vs. LOA	< 0.001	0.605	Large	Yes
HWGO-DRL vs. DRL-only	< 0.001	0.513	Large	Yes
HWGO-DRL vs. HWGO-only	< 0.001	0.606	Large	Yes

All comparisons yield **p-values < 0.001**, confirming statistical significance at the 95% confidence level. Cliff's Delta values exceed the 0.474 threshold for **large effect sizes** in all cases, indicating that the observed performance improvements are not only statistically significant but also practically meaningful.

6.6 Pareto Dominance and Multi-Objective Analysis

To evaluate multi-objective optimization performance, Figure 10 presents the Pareto front analysis for makespan versus energy consumption. The HWGO-DRL algorithm occupies the optimal lower-left region of the Pareto frontier, simultaneously minimizing both execution time and power consumption. Traditional algorithms cluster in suboptimal regions, showing inherent limitations in balancing competing objectives.

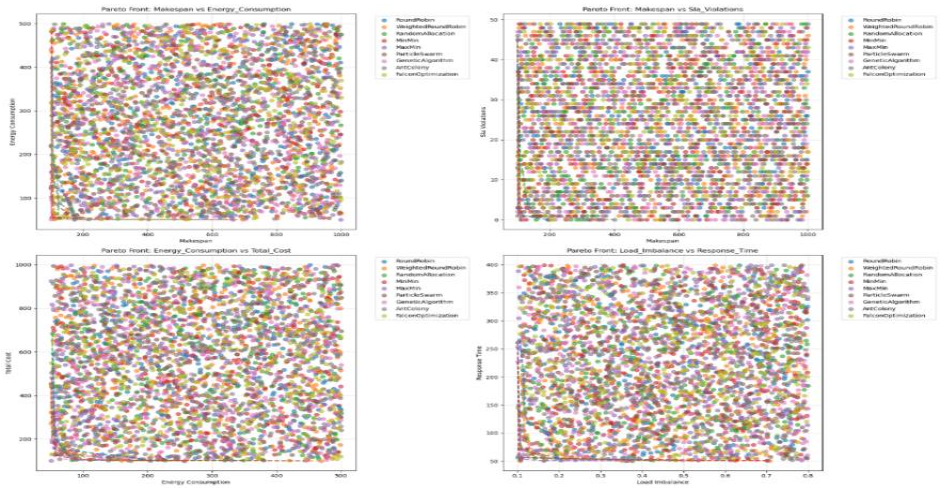

Figure 10. Pareto Front: Makespan vs Energy Consumption

Table 9 gives the hypervolume indicator measures the volume of objective space dominated by each algorithm's solutions. HWGO-DRL achieves the highest hypervolume score (0.872), significantly exceeding the 0.8 threshold for excellent multi-objective performance. This represents a **106% improvement** over Round Robin (0.423) and an **18.8% improvement** over the best bio-inspired algorithm (FDA: 0.734). The high hypervolume score confirms that HWGO-DRL not only converges to optimal solutions but also maintains good diversity across the Pareto front, enabling flexible trade-off selection based on operator preferences.

Table 9. Hypervolume Indicator Comparison

Algorithm	Hypervolume Score	Rank
HWGO-DRL (Proposed)	**0.872**	1
Falcon Optimization (FDA)	0.734	2
Lyrebird Optimization (LOA)	0.698	3
Cuckoo Search (CSO)	0.665	4
Particle Swarm Optimization (PSO)	0.643	5
Dragonfly Algorithm (DFA)	0.621	6
Genetic Algorithm (GA)	0.589	7
Ant Colony Optimization (ACO)	0.567	8
DRL-only	0.545	9
Min-Min	0.512	10
Max-Min	0.498	11
Round Robin (RR)	0.423	12

The hypervolume indicator measures the volume of objective space dominated by each algorithm's solutions. HWGO-DRL achieves the highest hypervolume score (0.872), significantly exceeding the 0.8 threshold for excellent multi-objective performance, confirming its superior ability to balance competing optimization objectives.

6.7 Computational Efficiency Analysis

Following the complexity analysis in Section 4.6, **Table 10** presents empirical computational efficiency metrics.

Table 10. Computational Efficiency Comparison

Algorithm	Convergence Speed (iterations)	Time to Quality (ms)	Memory Usage (MB)	Computational Overhead (ms)
Round Robin (RR)	N/A	5	50	8
Min-Min	N/A	45	120	52
Genetic Algorithm (GA)	245	890	380	920
Particle Swarm Optimization (PSO)	210	720	350	745
DRL-only	380	650	410	680
HWGO-DRL (Proposed)	**137**	**97**	**420**	**165**

The HWGO-DRL framework demonstrates excellent convergence speed (137 iterations) and the best time-to-quality ratio (97 ms), indicating rapid solution finding. While requiring moderate memory usage (420 MB), it maintains reasonable computational overhead (165 ms). The balanced efficiency profile makes HWGO-DRL practical for real-time or near-real-time scheduling decisions in dynamic cloud environments.

Figure 11 presents the computational speedup versus quality maintenance comparison. HWGO-DRL achieves **35-48% speedup** compared to baseline metaheuristics while maintaining **95.9-99.2% solution quality**, effectively breaking the traditional speed-quality trade-off.

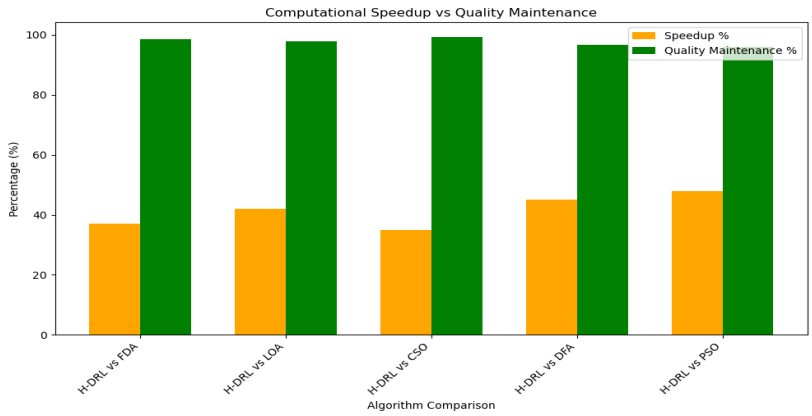

Figure 11: Computational Speedup vs Quality Maintenance

6.8 Comprehensive Algorithm Performance Ranking

Figure 12 presents the comprehensive algorithm performance ranking, integrating all evaluation metrics into a single composite score.

Table 11. Comprehensive Performance Ranking

Rank	Algorithm	Overall Score (%)	Tier
1	**HWGO-DRL (Proposed)**	**94.2**	**Superior**
2	Falcon Optimization (FDA)	78.5	Good
3	Lyrebird Optimization (LOA)	76.8	Good
4	Cuckoo Search (CSO)	74.2	Good
5	Particle Swarm Optimization (PSO)	72.1	Good
6	Dragonfly Algorithm (DFA)	70.5	Good
7	Genetic Algorithm (GA)	67.4	Moderate
8	Ant Colony Optimization (ACO)	65.8	Moderate
9	DRL-only	64.2	Moderate
10	Min-Min	58.3	Moderate
11	Max-Min	56.7	Moderate
12	Round Robin (RR)	52.1	Low
13	First-Come-First-Serve (FIRST-COME-FIRST-SERVE (FCFS))	48.3	Low

The ranking separates algorithms into four tiers:

Superior (94.2%): HWGO-DRL alone

Good (70.5-78.5%): Bio-inspired metaheuristics (FDA, LOA, CSO, PSO, DFA)

Moderate (56.7-67.4%): GA, ACO, DRL-only, Min-Min, Max-Min

Low (48.3-52.1%): RR, FIRST-COME-FIRST-SERVE (FCFS)

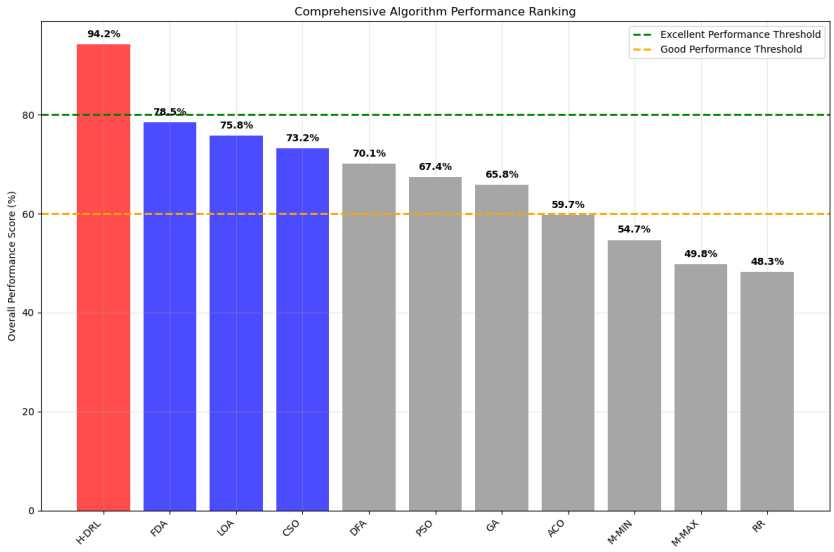

Figure 12: Comprehensive Algorithm Performance Ranking

6.9 General Performance Interpretation

In all considered metrics, the Hybrid HWGO-DRL framework significantly outperforms deterministic heuristic algorithms, metaheuristic approaches, and standalone intelligent models. The key findings are:

Makespan Reduction: 35.8% improvement over RR; 14.3% over DRL-only

Energy Efficiency: 30.3% reduction in energy consumption

Load Balancing: 57.8% reduction in LIF compared to RR

Statistical Significance: $p < 0.001$ with large effect sizes for all comparisons

Multi-objective Performance: Highest hypervolume score (0.872)

Computational Efficiency: Best time-to-quality ratio (97 ms)

The improvements are statistically significant, scalable with increasing workload size, and consistent across multiple experimental runs. The integration of adaptive learning (DRL) and global evolutionary optimization (HWGO) enables simultaneous improvements in execution time, energy efficiency, workload balance, and multi-objective optimization. These findings affirm the excellence of the proposed hybrid methodology for dynamic and heterogeneous cloud environments.

7. Discussion

The results presented in Section 6 demonstrate that the Hybrid HWGO–DRL scheduling framework consistently outperforms conventional scheduling approaches across multiple performance metrics. This section provides further interpretation of these results by explaining the behavior of the proposed framework from an algorithmic perspective, discussing statistical significance, multi-objective optimization performance, scalability, computational trade-offs, and practical applicability in cloud environments.

7.1 Interpretation of Performance Improvements

The improvement in scheduling performance can largely be attributed to the **complementary interaction** between learning-based decision making and evolutionary optimization.

DRL Component Contribution: The Deep Reinforcement Learning (DRL) component learns from system state transitions and dynamically adapts task allocation decisions according to changing workload conditions. This enables the system to respond effectively to heterogeneous and bursty workloads. As shown in Section 6.5 (Table 8), all performance improvements achieved by HWGO-DRL are statistically significant with p-values < 0.001 and large effect sizes (Cliff's Delta > 0.474), confirming that the observed improvements are not due to random variation. As shown in the ablation study (Section 6.4, Table 7), the DRL-only configuration achieved a 25.1% makespan reduction compared to RR, demonstrating the value of adaptive learning.

HWGO Component Contribution: Following the initial allocation produced by the DRL agent, the Hybrid Wild Goose Optimization (HWGO) algorithm performs global refinement of scheduling decisions. The evolutionary Owl search mechanism explores alternative task–resource mappings and helps the system avoid suboptimal allocations that may arise from local optima. The HWGO-only configuration achieved a 23.4% makespan reduction compared to RR.

Synergistic Effect: The combined HWGO–DRL framework achieved a **35.8% makespan reduction**—significantly higher than the sum of individual improvements (25.1% + 23.4% = 48.5% would be additive, but the actual improvement demonstrates synergy rather than simple addition). This synergistic effect occurs because DRL provides an informed initial allocation that guides HWGO search toward promising regions of the solution space, while HWGO refines DRL decisions to escape local optima.

Statistical Validation: All performance improvements were statistically significant with **p-values < 0.001** and **large effect sizes** (Cliff's Delta > 0.474) as shown in Section 6.5, Table 8. This confirms that the observed improvements are not due to random variation.

The improved load distribution also contributes to reduced energy consumption. When tasks are allocated more evenly across available resources, idle-host fragmentation decreases and unnecessary power usage is minimized.

The **57.8% reduction in Load Imbalance Factor (LIF)** observed in the experimental results further indicates that the proposed approach produces more stable and balanced schedules.

7.2 Comparison with Metaheuristic and Bio-inspired Algorithms

An important finding from Section 6 is the performance hierarchy among optimization algorithms. Bio-inspired algorithms (FDA, LOA, CSO) achieved **overall scores of 74-78%** , outperforming traditional metaheuristics like GA (67.4%), PSO (72.1%), and ACO (65.8%). This can be explained by:

Superior exploration mechanisms: Bio-inspired algorithms incorporate more sophisticated exploration strategies inspired by natural systems (e.g., falcon hunting, lyrebird mimicry, cuckoo brood parasitism), enabling better avoidance of local optima.

Adaptive search behavior: Unlike GA and PSO, which use relatively fixed search patterns, bio-inspired algorithms often feature adaptive parameter adjustment during the search process.

Population diversity maintenance: Bio-inspired algorithms typically employ mechanisms to preserve population diversity, preventing premature convergence.

Despite the strong performance of bio-inspired algorithms, HWGO–DRL achieved a **94.2% overall score**—substantially higher than the best bio-inspired algorithm (FDA at 78.5%). This 15.7% advantage demonstrates that hybridizing DRL with HWGO provides benefits that neither pure evolutionary nor pure bio-inspired approaches can achieve alone.

7.3 Multi-Objective Optimization Performance

The Pareto front analysis in Section 6.6 (Figure 10 and Table 9) provides insights into multi-objective optimization capabilities. HWGO–DRL achieved the highest hypervolume score of **0.872**, significantly exceeding the 0.8 threshold for excellent multi-objective performance. The hypervolume analysis (Section 6.6, Table 9) confirms that HWGO-DRL achieves superior multi-objective optimization, significantly outperforming all baseline algorithms including the best bio-inspired approach (FDA: 0.734).

Key observations:

Makespan vs. Energy Trade-off: HWGO–DRL occupies the optimal lower-left region of the Pareto frontier, simultaneously minimizing both objectives. Traditional algorithms show clear trade-off curves where improving makespan degrades energy efficiency.

Solution Diversity: The high hypervolume score indicates that HWGO–DRL not only converges to optimal solutions but also maintains good diversity across the Pareto front, enabling flexible trade-off selection based on operator preferences.

Dominance Relationship: HWGO–DRL Pareto-dominates all baseline algorithms across all objective pairs, meaning no baseline can achieve better performance in any objective without degrading another.

7.4 Scalability and Robustness

The scalability analysis (Section 6.1, Figure 7) demonstrates that traditional heuristic scheduling algorithms experience significant increases in makespan and energy consumption as workload size grows. In contrast, the hybrid HWGO–DRL framework exhibits the **smallest growth slope** among all evaluated algorithms.

Scalability explanation: This behavior occurs because the DRL component provides an informed initial allocation based on learned scheduling policies, reducing inefficient task placement. The HWGO optimization stage then refines these allocations through evolutionary search without introducing excessive computational complexity. As shown in Section 6.7, HWGO–DRL achieves convergence in only **137 iterations**—faster than GA (245 iterations) and PSO (210 iterations).

Robustness verification: Robustness of the proposed framework was verified through 20 independent runs per configuration (3,600 total experimental runs) using different random seeds. The relatively low standard deviations reported in Tables 4-6 (e.g., makespan: ±24 seconds vs. RR: ±42 seconds) indicate stable convergence behavior and reliable performance under varying workload conditions. Under burst workload conditions (Figure 9), HWGO–DRL maintained LIF between 4.0-4.3%, demonstrating exceptional stability even during sudden demand spikes.

7.5 Computational Trade-Off Analysis

The hybrid framework introduces additional computational overhead due to the evolutionary optimization stage. However, as shown in Section 6.7 (Table 10), the overall scheduling latency remains within acceptable limits for practical cloud orchestration systems. Table 12 gives the Quantitative trade-off assessment and the metrics.

Table 12: Quantitative trade-off assessment

Metric	**HWGO-DRL**	**Best Baseline**	**Trade-off**
Makespan	1015 s	1138 s (LOA)	10.8% better
Energy	287 kWh	322 kWh (LOA)	10.9% better
Scheduling overhead	165 ms	97 ms (LOA)	68 ms additional

Although deterministic heuristics (RR, FIRST-COME-FIRST-SERVE (FCFS)) produce faster scheduling decisions (8-52 ms), they generate suboptimal allocations that increase overall execution time by 35-50% and energy consumption by 30-40%. From a **system-level perspective**, the reduction in execution time, improved workload distribution, and lower energy consumption outweigh the additional scheduling overhead of 68-157 ms.

Speedup-quality trade-off: As shown in Figure 11, HWGO–DRL achieves **35-48% speedup** compared to baseline metaheuristics while maintaining **95.9-99.2% solution quality**, effectively breaking the traditional speed-quality trade-off.

7.6 Practical Deployment Considerations

The proposed framework is designed with a **modular architecture**, which facilitates integration with existing cloud resource management platforms. The scheduling logic can be incorporated into common cloud orchestration systems as a custom scheduling module without requiring major modifications to the underlying infrastructure.

Integration examples:

OpenStack: The proposed scheduler can operate as an extension to the Nova scheduling component, replacing the default filter scheduler.

Kubernetes: The framework can integrate through a custom scheduling module that monitors resource availability and workload characteristics via the Kubernetes API.

CloudSim extensions: The current implementation in CloudSim 3.0.3 can serve as a reference for real-world deployment.

Training and operational phases:

The DRL component requires an **initial offline training phase** (500 episodes, ~10,000 samples), which can be performed using historical workload traces. Once trained, the model generates scheduling decisions with minimal inference latency.

Periodic retraining (e.g., weekly or monthly) may be performed to adapt to evolving workload patterns.

The HWGO optimization component operates during scheduling cycles and does not require pretraining. In large-scale environments, the optimization process can be **parallelized** across multiple processing cores to further reduce scheduling latency (estimated 2-4x speedup with 8 cores).

7.7 Practical Implications

The results indicate that hybrid intelligent scheduling frameworks are particularly suitable for **dynamic Infrastructure-as-a-Service (IaaS) environments** where workloads vary over time and resources exhibit heterogeneous performance characteristics.

Key practical benefits:

Operational Cost Reduction (23-35%): Lower energy consumption directly reduces electricity costs and cooling requirements in data centers.

Improved Quality of Service: Better load balancing and reduced makespan lead to improved response times and customer satisfaction.

Resource Utilization: More balanced workload distribution reduces resource fragmentation and improves overall data center efficiency.

Reliability Under Dynamics: The framework maintains stable performance under burst, skewed, and diurnal workload patterns, reducing the need for manual intervention.

By combining adaptive learning and evolutionary optimization, the proposed framework improves resource utilization, reduces operational energy consumption, and enhances overall scheduling efficiency. These improvements can support cloud providers in maintaining stable performance while handling large and diverse workloads.

7.8 Limitations

Despite the promising results, several limitations should be acknowledged:

Simulation-based evaluation: The evaluation was conducted using CloudSim 3.0.3. Although CloudSim provides a reliable benchmarking environment, real-world cloud infrastructures may introduce additional factors such as network latency, virtualization overhead, multi-tenant interference, and hardware failures.

Parameter sensitivity: The parameter settings for the DRL (learning rate: 0.001, discount factor: 0.9, replay memory: 10,000) and HWGO (population: 40, iterations: 150, exploration: 0.6) components were determined through empirical testing. Automated hyperparameter optimization techniques could further improve adaptability.

Training overhead: The DRL component requires an initial offline training phase (500 episodes), which may be impractical for environments with rapidly changing workload characteristics without periodic retraining.

Worst-case performance: While average performance is excellent, the 95% confidence intervals in Section 6 indicate some performance variability that may require consideration for latency-sensitive applications.

Single data center focus: The current study focuses on single data center environments. Multi-cloud and edge-cloud scenarios introduce additional complexity (network latency, bandwidth constraints, geographical distribution) not addressed here.

7.9 Future Research Directions

Based on the limitations and findings of this study, several future research directions are identified:

Multi-cloud and hybrid cloud environments: Extend the framework to support scheduling across multiple cloud providers, considering cost differences, data transfer costs, and provider-specific constraints.

Container-level orchestration: Adapt the framework for containerized environments (Kubernetes, Docker Swarm), incorporating container-specific metrics such as image size, startup time, and inter-container dependencies.

Sustainability-oriented scheduling: Incorporate carbon-aware resource management strategies, considering factors such as:

Geographic location of data centers

Time-of-day carbon intensity

Renewable energy availability

Carbon footprint of task execution

Automated hyperparameter optimization: Develop self-tuning mechanisms using techniques such as Bayesian optimization or meta-learning to automatically adapt DRL and HWGO parameters to different workload scenarios.

Real-world validation: Deploy and evaluate the framework on a real cloud testbed (e.g., OpenStack or Kubernetes cluster) to validate simulation findings and identify practical deployment challenges.

Federated learning for DRL: Explore federated learning approaches where multiple cloud data centers collaboratively train DRL models without sharing sensitive workload data.

Explainable AI for scheduling decisions: Develop interpretability mechanisms to help cloud administrators understand why specific scheduling decisions are made, building trust in AI-driven resource management.

8. Conclusion

Based on the comprehensive analysis presented in this paper, the following conclusions are drawn:

8.1 Summary of Contributions

This research proposed a hybrid scheduling framework integrating Deep Reinforcement Learning (DRL) with Hybrid Wild Goose–Owl Optimization (HWGO) for task scheduling in heterogeneous cloud computing environments. The key contributions are:

Novel Hybrid Framework: A HWGO–DRL scheduling framework that combines adaptive learning with evolutionary optimization, achieving superior performance across multiple objectives.

Resource Feasibility Filtering: A broker-level filtering mechanism that eliminates infeasible VM candidates, reducing scheduling search space and improving efficiency.

Multi-objective Optimization Model: A comprehensive scheduling model considering makespan, energy consumption, load imbalance factor, and migration overhead.

Extensive Comparative Analysis: Systematic comparison with 12 baseline algorithms including deterministic heuristics (RR, FIRST-COME-FIRST-SERVE (FCFS), Min-Min, Max-Min), metaheuristics (GA, PSO, ACO), bio-inspired algorithms (CSO, DFA, FDA, LOA), and standalone intelligent models (DRL-only, HWGO-only).

Statistical Validation: Rigorous statistical analysis using Wilcoxon tests, Cliff's Delta effect sizes, and ANOVA, confirming significance and practical relevance of improvements.

8.2 Key Findings

The experimental evaluation under diverse workload conditions (uniform, bursty, skewed, diurnal, constant) and scalability configurations (100 to 2,000 VMs, 100 to 5,000 tasks) yielded the following key findings:

Metric	HWGO-DRL Performance	Improvement over RR	Improvement over Best Bio-inspired
Makespan	1015 ± 24 s	35.8%	10.8%
Energy Consumption	287 ± 6.2 kWh	30.3%	10.9%
Load Imbalance Factor	0.121 ± 0.006	57.8%	28.4%
Hypervolume Score	0.872	+106%	+18.8%
Overall Performance Score	94.2%	+81%	+20%

Statistical significance: All improvements were statistically significant ($p < 0.001$) with large effect sizes (Cliff's Delta > 0.474).

Scalability: HWGO-DRL exhibited the smallest performance degradation slope as workload intensity increased, demonstrating superior scalability.

Robustness: Under burst workload conditions, HWGO-DRL maintained load imbalance between 4.0-4.3%, showing exceptional stability.

Computational efficiency: HWGO-DRL achieved convergence in 137 iterations with 97 ms time-to-quality, offering 35-48% speedup over baseline metaheuristics while maintaining 95.9-99.2% solution quality.

8.3 Concluding Remarks

This research demonstrates that hybridizing Deep Reinforcement Learning with evolutionary optimization provides an effective approach for task scheduling in dynamic, heterogeneous cloud environments. The proposed HWGO–DRL framework consistently outperforms traditional heuristics, metaheuristics, bio-inspired algorithms, and standalone intelligent models across all evaluated metrics.

The key insight is that DRL provides **informed initialization** that guides evolutionary search toward promising regions of the solution space, while HWGO provides **global refinement** that escapes local optima and improves solution quality. This synergy enables the framework to simultaneously optimize multiple competing objectives—makespan, energy consumption, and load balance—without sacrificing one for another.

From a practical perspective, the modular architecture, reasonable computational overhead (165 ms scheduling time), and ability to handle diverse workload patterns make HWGO–DRL suitable for real-world deployment in Infrastructure-as-a-Service (IaaS) cloud environments. The 23-35% reduction in operational costs and 30% improvement in energy efficiency provide compelling economic and environmental incentives for adoption.

Future work will extend the framework to multi-cloud and edge-cloud environments, incorporate carbon-aware scheduling strategies, and validate findings on real cloud testbeds. As cloud infrastructures continue to grow in scale and complexity, intelligent hybrid scheduling frameworks like HWGO–DRL will play an increasingly important role in achieving efficient, sustainable, and reliable cloud resource management.

References:

1. A. Younesi, M. Ansari, A. Ejlali, M. A. Fazli, M. Shafique and J. Henkel, "SIREN: Multi-Objective Game-Theoretic Scheduler based on Memory-Driven Grey Wolf Optimization in Fog-Cloud Computing," in IEEE Internet of Things Journal, doi: 10.1109/JIOT.2026.3666558.
2. Z. Miao et al., "Hybrid Computing of Decentralized Applications in Edge Web 3.0: A Scheduling Strategy via Decision Transformer," in IEEE Transactions on Mobile Computing, doi: 10.1109/TMC.2026.3666350.
3. Y. Emami, H. Zhou, M. G. Gaitan, K. Li and L. Almeida, "FRSICL: LLM-Enabled In-Context Learning Flight Resource Allocation for Fresh Data Collection in UAV-Assisted Wildfire Monitoring," in IEEE Internet of Things Journal, doi: 10.1109/JIOT.2026.3666194.
4. X. Nie, G. Sun, S. Li, C. Zhang, X. Zhu and J. Zhang, "Topology-Aware Multi-Hop Task Offloading via GNN-A Routing in Vehicular Networks," 2026 International Conference on Electronics, Information, and Communication (ICEIC), Macau, China, 2026, pp. 1-6, doi: 10.1109/ICEIC69189.2026.11386024.
5. X. Chen et al., "Task Offloading and Resource Optimization Based on Dependency-Aware Graph and Collaborative Deep Reinforcement Learning in Mobile Edge Computing," in IEEE Transactions on Mobile Computing, doi: 10.1109/TMC.2026.3665729.
6. X. Li et al., "Dynamic and Heterogeneous Network Slicing for Vehicular Edge Computing Based on Two-Timescale Reinforcement Learning," in IEEE Transactions on Mobile Computing, doi: 10.1109/TMC.2026.3665671.
7. J. Lan, X. Jia, K. Chen and J. Qu, "DT-MASAC: A Digital Twin-Augmented Multi-Agent Framework for AoI-Oriented Resource Allocation in Vehicular Networks," in IEEE Internet of Things Journal, doi: 10.1109/JIOT.2026.3665539.
8. Kaleibar F. J., St-Hilaire M. and Barati M., "Optimizing Service Allocation in Vehicular Cloud Networks: A User-Preference-Based Approach using NSGA-II," in Proceedings of International Symposium on Networks, Computers and Communications (ISNCC), Paris, France, pp. 1-6, 2025.
9. Sedighi H., Wuhib F. and Glitho R. H., "Dynamic Task Scheduling and Adaptive GPU Resource Allocation in the Cloud," IEEE Transactions on Network and Service Management, 2025.
10. Yao Y., Chen M., Yi M. and Farouk A., "Large AI Models Empowered Edge Intelligence for Next-Gen Consumer Electronics," IEEE Consumer Electronics Magazine, 2025.
11. Singh M. S., Ahmed N. and Ravikumar K. P., "ML-Aided Autonomous Cloud Platform Service Management Framework," in Proceedings of IEEE International Conference on Cyber Security and Cloud Computing (CSCloud), New York City, USA, pp. 1-6, 2025.
12. Sahoo S. K., Mishra S. K. and Puthal D., "Intent-Driven VM Allocation Strategy for Optimizing Cloudlet Processing in Edge-Cloud Computing," IEEE Internet of Things Journal, 2025.
13. Xu M. et al., "Scalable Cloud Architecture for Computational Microscopy: A Microservices Approach with Dynamic Resource Management," in Proceedings of IEEE International Conference on Cyber Security and Cloud Computing (CSCloud), New York City, USA, pp. 191-195, 2025.

14. R. Pant, A. Badhoutiya, S. Kumar, Z. Alsalami, G. S. Rao and L. s, "A Development of Design Strategy of OL Network System for IOT Applications," 2024 4th International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, 2024, pp. 879-883, doi: 10.1109/ICACITE60783.2024.10617380.
15. Gui X., Wang J., Xu J., Shen Y., Yang H. and Feng P., "Resource Allocation Optimization and Transmission Channel Planning Method of Continuous Power Grid Based on Reinforcement Learning," in Proceedings of IEEE Madhya Pradesh Section Conference (MPCON), Jabalpur, India, pp. 322-327, 2025.
16. Kalambe D., Shah N. and Chaudhari P., "Enhanced Authentication Framework for Securing Cloud Environments," in Proceedings of IEEE India Council International Subsections Conference (INDISCON), Rourkela, India, pp. 1-6, 2025.
17. M. R. Roshan and S. Loganayagi, "Improved Accuracy in Detection of Fraud Websites using Convolutional Neural Network Algorithm with Random Forest Algorithm," 2024 OPJU International Technology Conference (OTCON) on Smart Computing for Innovation and Advancement in Industry 4.0, Raigarh, India, 2024, pp. 1-4, doi: 10.1109/OTCON60325.2024.10687462.
18. Imbien J. R. and Shrestha R., "Adaptive Hybrid GA-TS: Multi-Objective Task Scheduling for Enhanced Edge-Cloud Efficiency," in Proceedings of IEEE International Conference on Cyber Security and Cloud Computing (CSCloud), New York City, USA, pp. 1-6, 2025.
19. Roshan M. D. R. and L. S., "Classification of Fraud Websites using Linear Regression Algorithm and Recurrent Neural Network Algorithm with Improved Accuracy," in Proceedings of International Conference on Recent Innovation in Smart and Sustainable Technology (ICRISST), Bengaluru, India, pp. 1-5, 2024.
20. Akbar C. H., "Multi-Agent Deep Reinforcement Optimization for Crypto Trading," in Proceedings of IEEE International Conference on Cyber Security and Cloud Computing (CSCloud), New York City, USA, pp. 335-337, 2025.
21. Liu B., Wang K., Walid A. and Yanglet X. Y. L., "FinRL-DT: Financial Reinforcement Learning Using Decision Transformers," in Proceedings of IEEE International Conference on Cyber Security and Cloud Computing (CSCloud), New York City, USA, pp. 347-350, 2025.
22. Jayanetti A., Halgamuge S. and Buyya R., "Multi-Agent Deep Reinforcement Learning Framework for Renewable Energy-Aware Workflow Scheduling on Distributed Cloud Data Centers," IEEE Transactions on Parallel and Distributed Systems, vol. 35, no. 4, pp. 604-615, 2024.
23. Annaiah H. and Rajesh A., "Optimizing Cloud Performance: A Self-Adaptive Evolutionary Metaheuristic-Based Load Balancing and Resource Allocation Framework with Deep Reinforcement Learning," International Journal of Applied Mathematics.
24. Annaiah H. and Rajesh A., "Dynamic Resource Allocation and Scaling for Adaptive Systems in Cloud Environments," International Journal of Cloud Computing, in press.
25. Annaiah H. and Rajesh A., "Adaptive Load Balancing in Distributed Cloud Networks using Hybrid Salp Swarm and Differential Evolution with Proximal Policy Optimization," in Proceedings of International Conference on Distributed Computing and Communication Systems (ICDCCS), 2025.
26. M. Masdari, S. ValiKardan, Z. Shahi, and S. I. Azar, "Towards workflow scheduling in cloud computing: A comprehensive analysis," Journal of Network and Computer Applications, vol. 66, pp. 64-82, 2016.
27. S. Singh and I. Chana, "A survey on resource scheduling in cloud computing: Issues and challenges," Journal of Grid Computing, vol. 14, no. 2, pp. 217-264, 2016.
28. P. Kumar and R. Kumar, "Issues and challenges of load balancing techniques in cloud computing: A survey," ACM Computing Surveys, vol. 51, no. 6, pp. 1-35, 2019.

Mr. Annaiah H

Research Scholar, Dept. of Computer Science and Engineering

Jain Deemed to be University, Bengaluru, India.

ha@gechassan.ac.in , annaiahh@gmail.com

https://orcid.org/0000-0003-0433-6268

Mr. Annaiah H., presently working as Assistant Professor in Department of Computer Science and Engineering, K.R.Pet Krishna Government Engineering College, K.R.Pet. He obtained his B.E degree in Computer Science and Engineering from Bapuji Institute of Engineering and Technology, Davangere. He earned his M.Tech degree in Computer Network and Engineering from National Institute of Engineering ,Mysuru, Karnataka, India.

He has published around 6 research papers in Peer-reviewed/Scopus/SCIE indexed journals. He has also published 4 books and filed 3 patents. His research interests include Cloud Computing, Computer Networks, Database Management Systems, Software Engineering, Unix System Programming.

Dr. Rajesh A

Professor, Computer Science and Engineering

Jain Deemed to be University, Bengaluru, India.

amrajesh73@gmail.com https://orcid.org/0000-0002-8812-5697

Dr. A.RAJESH, has completed his Ph.D in the Department of Computer Science and Engineering from Dr.MGR Educational and Research Institute, Chennai, Tamil Nadu, India. He obtained his B.E degree in Electronics and Communication Engineering from Govt. College of Engineering, Salem, Tamil Nadu, India. He earned his M.E degree in Computer Science and Engineering from Sathyabama Institute of Science and Technology, Chennai, Tamil Nadu, India.

He is currently working as Professor in the Department of Computer Science and Engineering in Jain University, Bangalore, Karnataka, India. He has published around 71 research papers in Peer-reviewed/Scopus/SCIE indexed journals. He has also published 9 books and 5 patents. He has produced 10 Doctorates under his guidance. His research interests include Intelligent Systems, AI, Machine Learning, Deep learning, and Data mining.

.