SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V13I5P107

Volume 13 Issue 5, 65-76, May 2026

Original Article

Hybrid Harris Hawks and Reinforced Ant Colony Optimization for Energy-Aware Task Scheduling in Cloud Environments

M. Rupasri[1], G.P.S.Varma[2], Indukuri Hemalatha[3]

[1,2]*Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Andhra Pradesh, India.*
[3]*Department of Information Technology, S.R.K.R Engineering College, Andhra Pradesh, India.*

[1]*Corresponding Author : rupasriklu@gmail.com*

Received: 07 February 2026 Revised: 09 March 2026 Accepted: 08 April 2026 Published: 27 May 2026

Abstract *- Cloud computing task scheduling is a complex multi-objective optimization problem that involves balancing the makespan, energy consumption, cost, and Service-Level Agreement (SLA) compliance. Conventional heuristics may not adapt well to heterogeneous workloads, whereas single meta-heuristics may be prone to premature convergence. To address these issues, this study proposes a Hybrid Harris Hawks' Optimization With Reinforced Ant Colony Optimization (HGO-RACO) model. The model enhances the global search ability of Harris Hawk's Optimization (HHO) through adaptive escape energy dynamics and Levy flights, while leveraging the local search power of Ant Colony Optimization (ACO) via pheromone reinforcement. Simulation experiments using workload traces from NASA iPSC and HPC 2N in CloudSim demonstrated that HHO-RACO reduces makespan, power consumption, and SLA violations more effectively than PSO, GA, Firefly, and CEQACO. These results affirm that HHO-RACO provides scalable, energy-efficient, and robust scheduling suitable for dynamic cloud environments, supporting green- and performance-oriented cloud infrastructure.*

Keywords *- Cloud Task Scheduling, Harris Hawks Optimization (Hho), Reinforced Ant Colony Optimization (Raco), Energy-Aware Scheduling, Sla Violation Minimization.*

1. Introduction

Cloud computing has become a new paradigm for providing scalable and flexible computing possibilities using virtualisation, elastic provisioning, and pay-per-use pricing. This allows organizations to decrease infrastructure costs, increase availability, and access computational power on demand. However, the increasing need for data-intensive and real-time services comes with the heavy burden of resource allocation and task scheduling. Cloud providers pool processing resources and provide services to several customers in a shared model. Based on demand, virtual and physical resources are automatically assigned and reallocated. Cloud computing has led to the rapid development of virtualisation, networking, storage, and distributed systems. It offers common websites with large-scale networked capacity. According to the NIST, it is on-demand access to configurable resources without much management effort. The most important aspect of cloud computing is its elasticity, which enables resources to be scaled according to demand. Elasticity may be horizontal, that is, VM instances are added or removed, or vertical, that is, CPU and RAM are changed within a VM. Horizontal elasticity requires system-level functionality, such as cloning and VM synchronization, but minimal hypervisor intervention. Vertical elasticity is not application-based; however, support must be provided by the hypervisor and OS kernel. Features such as hot-plugging allow vertical scaling at runtime [1]. Cloud services are provided in the form of SaaS, PaaS, and IaaS. SaaS offers applications that are ready to use, and clouds may be in the form of public, private, community-based, or hybrid. The providers can be multi-tenant or lease providers. These data centers are constructed using numerous servers and support systems and operate services with various operational models. Scalability and virtualisation make cloud load balancing different from traditional methods. The cloud environment utilises shared virtual servers, unlike dedicated servers, and distributed methods must be employed [2]. The workload distribution is performed by commodity servers, which allows a flexible and efficient balancing strategy. In static scheduling, resources and tasks are pre-mapped prior to execution. When given, mapping was not altered. This style is applicable to foreseeable applications; however, it is not adaptable to dynamic circumstances. Dynamic scheduling assigns tasks online based on the state and demand of the system. It analyses existing situations and makes decisions in real time.

This facilitates flexibility but makes it difficult to keep track of and allocate duties. Heuristic scheduling addresses problems such as imbalance, delay, and wastage of resources. It provides rough solutions at a cheaper rate, but not necessarily the best. Heuristics also work well in diverse and dynamic environments because they are highly adaptable [3]. Time is a major factor because efficient scheduling minimizes the execution time (makespan), energy usage, cost, and meets Service Level Agreements (SLAs), particularly in heterogeneous and dynamic environments [4]. In the classical computing systems, the First-Come-First-Serve (FCFS), Round-Robin, Min-Min, and Max-Min strategies have provided good performance under homogeneous workloads. However, these approaches are not as effective in large clouds, where tasks vary significantly, arrival rates are high, and VMs have fluctuating capacities. They are also not dynamic and can hardly adapt to fluctuations in the real-time workload, which often results in the underutilization of resources, SLA breaches, and high operational costs [5]. Therefore, the concept of scheduling has been restructured as a multi-objective optimisation problem in which conflicting objectives, including makespan, energy efficiency, throughput, and fairness, must be compromised. As artificial intelligence improves,

Deep Reinforcement Learning (DRL) has become popular for resource management. DRL agents can acquire scheduling policies by observing interactions with the system, which maps cloud states to allocation actions that achieve long-term accumulated rewards [6]. DRL can offer online adaptability where the metaheuristics of a system are lacking because of its ability to capture system dynamics and workload variations. In addition, Graph Neural Networks (GNNs) and sequence models have been incorporated into DRL to identify task-task relationships and time correlations [7]. Regardless of the benefits, DRL solutions require a large amount of training, have problems with the design of balanced reward functions, and tend to have high computational complexity [8]. Cloud systems are now being supplemented by fog and edge computing, which perform computations closer to users to decrease latency. In this geo-distributed or heterogeneous environment, scheduling is more complicated because it must consider bandwidth, regional costs, and latency constraints. Research has pointed to the necessity of having latency-, scenario-, and network-conscious schedulers in fog-cloud systems [9]. This also highlights the significance of hybrid models that have the potential to integrate global optimisation and local decision-making [10].

To address these gaps, the HHO-RACO framework is proposed for task scheduling in cloud (or cloud-fog) environments that:

- It unites Harris Hawks Optimisation (HHO) for exploration with Reinforced Ant Colony Optimisation (RACO) for constructive exploitation using feedback loops between these components.
- Encodes task-VM assignments in a mapping that allows HHO's continuous vectors of the HHO to be discretely assigned, thereby enabling a better hybrid search.
- Adaptive multi-objective evaluation with dynamic weighting, balancing makespan, energy, and SLA violation is adopted.

The remainder of this paper is structured as follows: Section II surveys the related literature on task scheduling techniques; Section III describes the problem formulation; Section IV introduces the HHO-RACO methodology with a mathematical analysis; Section V presents the experimental setup and results; and Section VI concludes with the contributions and future work directions.

2. Related Works

Recently, energy-conscious scheduling of tasks in cloud environments has been the subject of research, emphasizing the increased significance of Hybrid Metaheuristic Optimization Methods. The HHO algorithm has gained popularity owing to its good exploration and exploitation, allowing effective searching throughout the globe. Similarly, ACO, particularly with reinforcement techniques, enhances the convergence rate and adaptive learning in dynamic clouds. The literature has incorporated the use of HHO with ACO to balance the distribution of loads, minimize the use of energy, and reduce the time of execution. Such hybrid solutions are better than traditional heuristics because they dynamically choose the best resources; however, the issues of scalability and real-time scalability in large-scale cloud environments remain challenging.

In [11], the authors suggested a Multi-Agent Reinforcement Learning (MARL) algorithm to dynamically schedule tasks in cloud manufacturing, which is aimed at group-service problems when tasks have adjustable processing sequences and the environment is extremely dynamic. This approach encodes the task topology using a graph convolutional network and performs temporal processing using recurrent units. Agents are trained on decentralized policies within a Centralized Training-Decentralized Execution (CTDE) paradigm, utilizing a mixing network to aggregate their value estimates.

The authors developed custom action spaces and reward functions that are sensitive to scheduling goals (throughput, timeliness, and resource utilization) and indicated that the GCN+RNN encoder assists the MARL system in generalizing complicated task-task dependencies. Simulation evidence (which was conducted on an empirical level) indicates that the suggested framework outperforms traditional heuristic and baseline learning strategies on important measures, generating a richer and more receptive schedule in the cloud manufacturing environment.

In [12], the authors described workflow scheduling as a multi-criterion optimization problem considering task priorities, makespan, migration time, energy consumption, and SLA penalties. The WOA operators encode and evolve candidate schedules (encircling prey, bubble-net attacking, and exploration by random search), and a constructive heuristic is used to compute the expected costs of executing the tasks and data transfer to advise on fitness. The authors incorporated VM power models to approximate the per-task energy and added penalty terms in the case of a deadline violation, and a secularized optimization objective was formed. CloudSim-style simulations and benchmark workloads were used in the experiments to compare the proposed WOA scheduler with conventional heuristics and other meta-heuristics. The results obtained show that WOA can achieve a lower makespan and energy using fewer SLA violations in the experiments.

In [13], the authors proposed a Hybrid Particle-Whale Optimization (PWOA/HPWOA) algorithm for workflow scheduling in a cloud-fog environment, providing the rapid convergence of Particle Swarm Optimization with the global search behaviors of Whale Optimization to prevent local optimization. The algorithm models the scheduling problem as a multi-criteria problem, the main objective of which is to minimize the Total Execution Time (TET) and Total Execution Cost (TEC) without violating task dependencies or the heterogeneity of fog/clouds; energy and deadline penalties are added to the fitness. Constructive encodings and simulation-based evaluation (cloud/fog traces and scientific workflow benchmarks, including CyberShake, Epigenomics, Inspiral, Montage, and SIPHT) were used to determine the difference between PWOA and standard PSO and WOA. The experiments indicate that the hybrid achieves lower TET and TEC in all tested workflows and is more robust and has better quality solutions with the fusion of operators and the use of proper parameter settings.

In [14], the authors proposed Latin Square-Based Improvised Genetic Optimization (LSBGO), an LSF-based resource-allocation algorithm for cloud systems that focuses on low-discrepancy sampling and latency-consciousness under real-life operating conditions. The essential notion is to apply Latin-square-based population seeding and crossover/mutation operators to generate a population with high diversity and even distributions (reducing sampling bias) and then evolve a better GA search that is concerned with scenario-specific latency constraints (e.g., geo-distribution, network contention, and priority mixes). The fitness model explicitly balances the latency, use of resources, and penalties of different scenarios to ease the choice of allocations so that the optimizer can favor the allocation that minimizes the tail latency in high-load or latency-sensitive scenarios. The authors constructed adaptive selection and replacement schemes to complete the loop of preserving the low-discrepancy structure while permitting aggressive local refinement. Empirical analysis (simulation-based) demonstrates that LSBGO incurs significantly lower latency and superior allocation fairness than the standard GA and heuristic baselines in various test situations.

In [15], the authors proposed an Improved Marine Predators Algorithm (IMPA) to solve task scheduling in fog/clouds. To solve the scheduling problem (minimizing makespan and improving load balance and latency), this study encodes scheduling as a multi-criteria optimization by applying the encircling, exploitation (bubble-net), and exploration operators of MPA with problem-specific repair and local search operations. IMPA is an RTI-based model that uses an expected time to compute (ETC) heuristic matrix and power model to model the energy, and employs constructive repairs to guarantee deadline/SLA constraints. They tested IMPA on synthetic and real traces (including HPC2N, NASA iPSC, and GOCJ workloads) using a cloud/fog simulation model. The results indicate that IMPA has better performance with respect to a variety of baseline metaheuristics (in terms of makespan reduction and degree of imbalance) and provides better distributions of resources in the context of dynamic arrivals. This study identifies the capacity of IMPA to strike a balance between exploration and exploitation on a global scale, which makes it a viable metaheuristic for latency-sensitive and decentralized scheduling.

In [16], the authors suggested TrustStore, a TOTP-based secure cloud storage system that combines Time-Based One-Time Passwords (TOTP) and JSON Web Tokens (JWT) to provide more robust user authentication and data-at-rest security. It is designed based on a two-layer security model: a JWT-mediated session layer that issues temporally limited, signed access tokens following TOTP verification, and an encryption layer where stored objects are confidential and integrity-protected (this study formalizes token lifecycles, revocations, and freshness assurances). The authors introduced TrustStore and tested its security and performance trade-offs, demonstrating that the methodology resists typical token replay and credential-theft threats with only a significant authentication delay, which is reasonable in sensitive areas (healthcare, finance, governance). This study explains practical implementation factors (expiration of tokens, management of clock skews when using TOTP, and application interoperability with cloud storage APIs) and suggests parameter configurations to create a trade-off between usability and security.

The current literature demonstrates two converging trends in cloud/fog scheduling and secure storage. First, hybrid and nature-inspired metaheuristics (which are frequently combined, such as PSO-WOA, HHO-ACO, MPA, and WOA) are always effective in enhancing the makespan, energy, and SLA compliance. Second, learning-based methods (MARL and GNN-augmented DRL) offer adaptive

and online scheduling of dynamically dependent tasks. Surrogate models and low-discrepancy sampling are two sampling methods that enhance search efficiency and fairness (Latin squares). Latency-sensitive formulations and locality-sensitive objectives are essential in fog and cloud-fog environments.

3. Materials and Methods

The proposed HHO-RACO framework shown in Figure 1 combines the strengths of the global exploration of HHO and the local exploitation of RACO to obtain an effective scheduling of tasks in clouds.

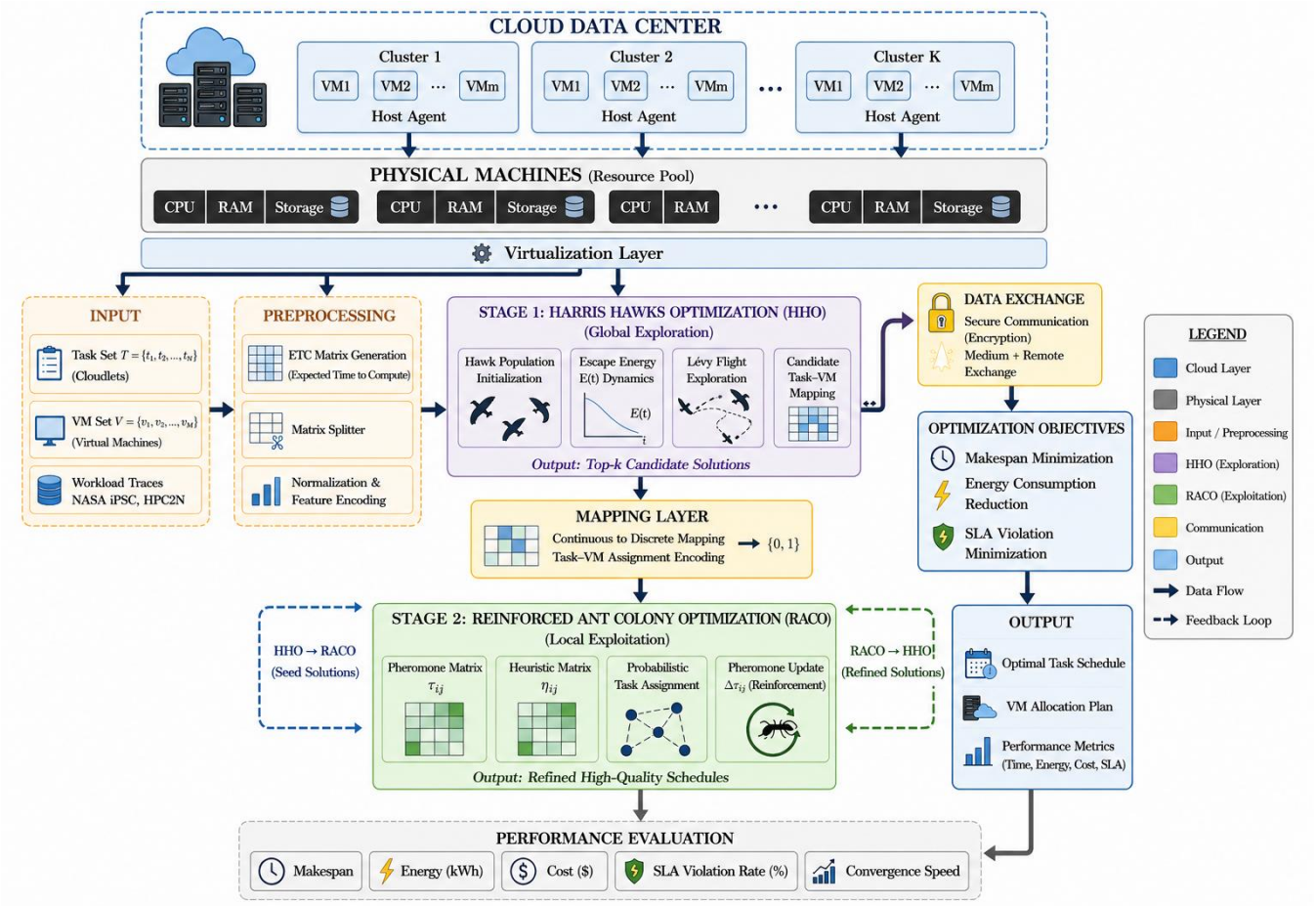

Fig. 1 Proposed HHO-RACO-based cloud task scheduling framework

The original HHO first explores the search space via escape energy dynamics and, via Levy flights, is able to produce a variety of candidate task-VM mappings. These solutions are mapped using a mapping layer and fed to the RACO, where pheromone trails are strengthened using the feedback of HHO for direct exploitation.

RACO enhances the search for good solutions by updating pheromone concentrations and heuristic measures of desirability. The fitness assessment will have make-up, energy use, and SLA violation penalties to optimize multiple objectives. Such synergy prevents early convergence, adaptively balances exploration and exploitation, and generates high-quality schedules.

The resulting optimal task-to-VM assignment is produced, which is much more efficient and energy conscious than the single algorithms.

3.1. System Model

The independent task set of cloudlets (cloudlets) is considered as $T = \{t1, \dots, tN\}$, and is to be scheduled on a set of M virtual machines $V = .\{v_1, \dots, v_M\}$ on physical machines. The length (instructions) L_i, data size D_i, priority P_f, and deadline d_i of each task t_i. VM v_j is C_j (MIPS) capacity and R_j (available memory) available, and a model Pj(u) of utilization to power at a point in time. A schedule is an assignment function S: T → V and start times s_i of a task. The goals of the scheduler are multi-objective, i.e., minimization of makespan, minimization of energy use, and meeting SLAs (deadlines). This solves a scalarised problem.

$$\text{Cost}(S) = w_1 \cdot \text{Makespan}(S) + w_2 \cdot \text{Energy}(S) + w_3 \cdot \text{SLA}_{\text{Penalty }(S)}, \quad (1)$$

With weights $w_1, w_2, w_3 \geq 0$ and $w_1 + w_2 + w_3 = 1$.

3.2. HHO-RACO Overview

There are two stages of cooperation in each epoch:

Stage 1: In the HHO exploration stage, a population of candidate complete schedules (hawk agents) is explored using HHO operators to evolve a population and suggest promising parts of the search space (global search). The location of each hawk represents a task – VM mapping (e.g., as an integer vector of length N or as a permutation with mapping meaning). A subset of the best HHO candidates (or pheromone seeds based on HHO).

Stage 2: RACO is exploited by RACO's constructive solution building/optimization. RACO is based on a pheromone matrix T and a heuristic desirability matrix *H* to generate refined task assignments. RACO is a stochastic construction followed by pheromone updates, which are reinforced by HHO fitness information. HHO offers a variety of promising seeds worldwide, whereas RACO is more localized in its search and generates quality viable schedules. This loop is closed: HHO changes according to the better solutions of RACO, and the pheromones of RACO change according to HHO feedback.

3.2.1. Harris Hawks Optimization

This is motivated by the collaborative surprise pounce behavior of Harris's hawks. Each hawk $x_{it} \epsilon \mathbb{R}^d$ is a candidate solution (vectorised encoding of schedule). Xpreyt represents the optimal solution at present (lowest cost).

3.2.2. Prey Energy Model

The transition between the global exploration and local exploitation is guided by the prey escaping energy. It is modeled as:

$$E(t) = 2E_0\left(1 - \frac{t}{T}\right), E_0 \sim \mathcal{U}(-1,1) \quad (2)$$

Where t denotes the current iteration, T denotes the maximum number of iterations, and E_0 denotes the initial random energy. High $|E|$ (> 1) implies that prey remains high in energy, and this causes exploration, and a low $|E|$ (< 1) converts the algorithm to exploitation.

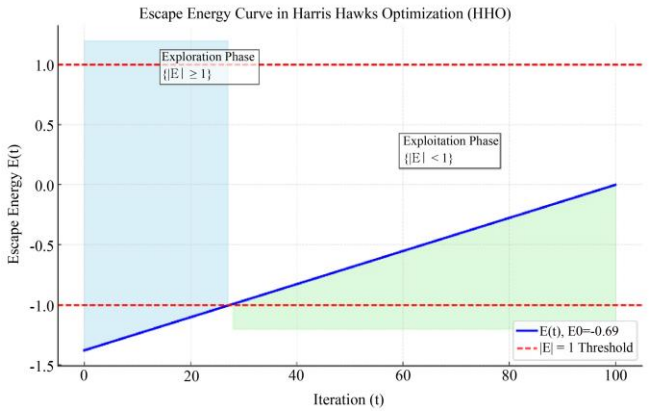

Fig. 2 Escape energy curve in HHO

Figure 2 is used to show the energy curve of escape in HHO that regulates the exploration and exploitation balance. The intrinsic blue line is the reduction in the energy $E(t)$ with each iteration. When the Energy $E(t)$ is greater than 1, as represented by the sky-blue area, the hawks are in the exploration stage and are generally searching throughout the solution space. During the iterations, $|E|$ decreases below 1, signaling the transition of the algorithm to the exploitation stage, during which hawks attack the prey (optimal solution) with teeth. The red dashed lines indicate the critical point between exploration and exploitation. Exploration phase: When |E|≥1, hawks randomly update their positions to expand the search space. The position of Hawk III is updated as:

$$X_i^{t+1} = X_{\text{rand}}^t - r_1 \cdot \left| X_{\text{rand}}^t - 2r_2 X_i^t \right| \quad (3)$$

where X_{rand}^t is a randomly selected hawk, $r_1, r_2 \sim \text{U}(0,1)$.

3.2.3. Exploitation Phase

When $|E|$<1, hawks switch to attacking the prey. The strategy depends on the prey's chance of escaping, modeled by a probability $p \sim U$ (0,1).

If $q \geq 0.5q$ (soft besiege):

$$X_i^{t+1} = \Delta X_i - E \cdot \left| J X_{\text{prey}}^t - X_i^t \right| \quad (4)$$

else (hard besiege):

$$X_i^{t+1} = X_{\text{prey}}^t - E \cdot |\Delta X_i| \quad (5)$$

$\Delta X_i = X_{\text{prey}}^t - X_i^t$ and $J \sim U(0,2)$is a jump strength.

When rapid dives are invoked, a Lévy flight operator $L(\lambda)$ may be applied. Evaluate $f(\text{Y}), f(Z)$, and pick the better.

3.2.4. Rapid Dive (Lévy Flight)

To avoid premature convergence, hawks may perform rapid dives using Lévy flights:

$$Y = X_{\text{prey}}^t - E.\left| J.X_{\text{prey}}^t - X_i^t \right| \quad (6)$$

$$Y = Y + S.L(\lambda) \quad (7)$$

Where $L(\lambda)$ denotes Lévy distribution, and S is the flight step size. The next position is chosen from $\{Y, Z\}$ based on which gives better fitness.

3.2.5. Balance of Search

The adaptive parameter $E(t)$ ensures a smooth transition: the initial iterations are more inclined towards the global exploration, and the subsequent ones are more inclined towards the local exploitation. Such a stochastic balance,

together with jumps, helps HHO to evade local minima well and move towards quality solutions.

3.2.6. Ant Colony Optimization (ACO)

ACO builds solutions step by step, based on pheromone t_{ij} and heuristic desirability η_{ij}, and index *i* can be a decision step (task) and *j* a choice (VM). In scheduling, set $T \in R^{N\times M}$, with ti,j defined as the pheromone of assigning task t_i to VM v_j. Heuristic desirability η_{ij} is defined as the inverse expected cost of allocating t_i to v_j:

$$\eta_{ij} = \frac{1}{\varepsilon+\mathrm{ETC}_{ij}}, \mathrm{ETC}_{ij} = \frac{L_i}{C_j} + \text{ data transfer }_{ij} \quad (8)$$

Transition (selection) probability for an ant constructing a solution:

$$P_{ij}(t) = \Pr\left(\text{assign} t_i \to v_j\right) = \frac{\tau_{ij}(t)^{\alpha}.\eta_{ij}^{\beta}}{\sum_{k=1}^{M} \tau_{ik}(t)^{\alpha}.\eta_{ik}^{\beta}} \quad (9)$$

With $\alpha, \beta > 0$. After each ant constructs a full schedule S, evaluate $\mathrm{Cost}(S)$. Pheromone update standard form:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{\ell=1}^{n_{\text{ants}}} \Delta\tau_{ij}^{\ell}(t) \quad (10)$$

Where $\rho \in (0,1)$ is evaporation and $\Delta\tau_{ij}^{\ell}$ equals $\frac{Q}{\mathrm{Cost}(s^{\ell})}$ assigned ti to vj its solution, else zero. Q is a scaling constant.

3.3. Reinforced Ant Colony (RACO) with HHO

Once a set of the top k hawk candidates (with the lowest cost by HHO) $\{X(1), \ldots, X(k)\}$ is decoded into assignment suggestions and initializing pheromone entries to bias RACO construction to promising mappings. The RACO pheromone updates have an additional reinforcement term based on the improvement of the HHO fitness. Let the optimal HHO candidate preceding RACO invocation be S^{HHO}, and the RACO-generated solution be S^{RACO}. Define improvement measure:

$$\Delta_{\text{imp}} = \max\left(0, \frac{\mathrm{Cost}(S^{\mathrm{HHO}}) - \mathrm{Cost}(S^{\mathrm{RACO}})}{\mathrm{Cost}(S^{\mathrm{HHO}})}\right) \quad (11)$$

Then augment the pheromone update:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{\ell} \Delta\tau_{ij}^{\ell}(t) + \gamma \cdot \Delta_{\text{imp}} \cdot \mathbb{I}\left[\left(t_i \to v_j\right) \in S^{\mathrm{RACO}}\right] \quad (12)$$

Where $\gamma \geq 0$ is a reinforcement of HHO-ACO. When $\Delta_{imp} > 0$, pheromones of constructive decisions that created improvement get additional mass, producing directed exploitation pegged on HHO seismic moves. Adaptive pheromone scaling: Scaling of γ optionally with HHO population diversity *D*(*t*):

$$\gamma(t) = \gamma_0 \cdot \exp\left(-\kappa D(t)\right), D(t) = \frac{1}{n(n-1)} \sum_{i\neq j} \left\| X_i^t - X_j^t \right\| \quad (13)$$

Thus, when diversity is high, reinforcement is stronger; as diversity collapses, reinforcement is reduced to avoid premature lock. HHO updates on continuous vectors, but scheduling decisions are discrete. Represent a hawk vector X = (x1, … , xN) with xi ∈ R. For each task ti, define assignment:

$$\text{assign}(t_i) = v_{\arg\min_j |x_i - c_j|} \quad (14)$$

with fixed VM centrepoints $\{c_1, \ldots, c_M\}$. A second strong map Softmax rounding: The calculation of soft assignment probabilities of x_i and the drawing of argmax. Mapping must conserve neighborhood structure, or HHO moves will result in small changes in assignment, in the situation where *x* is slightly different, as explained in Algorithm 1.

Algorithm1: HHO-RACO

Input: Tasks *T*(*N*), VMs *V*(*M*), HHO population size n_H, ACO ants n_A, Max iterations T_{max}, top_k, pheromone init τ_0, parameters α, β, ρ, γ_0, κ

1. Initialize pheromone $\tau\{i, j\} \leftarrow \tau 0$ for all *i*,*j*
2. Initialize HHO population $X_{1^0,\ldots,,} X_{nH^0}$ randomly
3. **for** $t = 0$ to T_{max-1} do

3.1 Evaluate HHO population: for each X_{p^t} decode to schedule S_p and compute $Cost(S_p)$

3.2 Identify $X_{prey^t} = argmin_p Cost(S_p)$

3.3 HHO exploration/exploitation update:

for each hawk *p* update $X_{p^{t+1}}$ following HHO rules (*E*(*t*), Lévy, etc.)

Perform mapping to a discrete schedule after the update for evaluation

3.4 Select top_k hawks (lowest Cost) as seeds for RACO

3.5 Seed pheromones: for each seed schedule S_{seed}

for each $(t_i \to v_j)$ in S_{seed}:

$\tau_{i,j} \leftarrow \tau_{i,j} s * f_{seed}\left(\mathrm{Cost}\left(S_{seed}\right)\right)$ (*s* scales seed influence)

3.6 Run RACO: n_A ants construct schedules using τ and η

Each ant builds a full schedule by sampling

$$P_{\{i,j\}} = \tau_{\{i,j\}^{\alpha}} \eta_{\{i,j\}^{\beta}}$$

Evaluate each ant-produced schedule; obtain the best S_RACO.

3.7 Compute $\Delta_{\text{imp}} = \max\left(0, \frac{\mathrm{Cost}(S^{\mathrm{HHO}}) - \mathrm{Cost}(S^{\mathrm{RACO}})}{\mathrm{Cost}(S^{\mathrm{HHO}})}\right)$

3.8 Reinforced pheromone update:

$\tau \leftarrow (1\text{-}\rho)\ \tau + \Sigma_{\text{ants}}\ \Delta\tau_{\text{ants}} + \gamma(t) * \Delta_{\text{imp}} *$ indicator_matrix(S_RACO)

where $\gamma(t)=\gamma 0 * \exp(-\kappa D(t))$

3.9 Optionally inject elite solutions from RACO into the HHO population (replacement)

4. Return the best schedule found

3.4. Mathematical Analysis

This analysis is divided into three parts: (i) exploration-exploitation balance and ergodicity intuition, (ii) effect of reinforced pheromones on selection probability, and (iii) computational complexity.

3.4.1. Exploration-Exploitation Balance

The energy E(t) of HHO decreases in the early t linearly with the iterations; at the start, EO tends to be near +-1, meaning that at an early t, the energy is likely to be larger than 1, and this is the point at which exploration corrections occur that make large moves in solution space. Local refinements are done by HHO as t-T, |E|-0.

The evaporation of pheromones by 1-r by RACO ensures that the concentration does not drift away; this, coupled with randomized Levy flights, which disturb the population, gives the hybrid stochastic recurrence, which explores different basins. The Markov chain induced by hybrid updates (under mild assumptions, positive evaporation r>0, occasional nonzero mutation probability via Lévy flights) is irreducible, and hence under the probabilistic definition of ergodicity, the algorithm will revisit states. This is a heuristic argument on how HHO-RACO can avoid local optima with infinite time.

3.4.2. Effect of Reinforcement on Selection Probability (Local Concentration)

Consider a single task ti with two candidate VMs va, vb. Let initial pheromones be τa, τb, heuristics ѵa, η b. The probability ratio:

$$\frac{P_{i,a}}{P_{i,b}} = \frac{\tau_a^{\alpha}\eta_a^{\beta}}{\tau_b^{\alpha}\eta_b^{\beta}} \tag{15}$$

A reinforced update increases ta by Δ, so the updated ratio becomes:

$$\frac{P'_{i,a}}{P'_{i,b}} = \frac{(\tau_a+\Delta)^{\alpha}\eta_a^{\beta}}{\tau_b^{\alpha}\eta_b^{\beta}} \tag{16}$$

If $a \geq 1$, relative probability increases at least linearly in Δ for moderate Δ. Thus, reinforcement leads to multiplicative bias toward choices used in improved RACO solutions. Evaporation ensures this bias decays if not continually reinforced.

3.4.3. Computational Complexity

Let $nH =$ HHO population, I = number of iterations, nA = number of ants, N tasks, M VMs. Cost evaluation per schedule (simulated estimate) costs O(Ceval) (proportional to the number of tasks and modeled VM execution times). Per iteration cost: HHO updates: $O(nH \cdot d)$ where d is vector dimension (here $\approx Nd$); plus, evaluations: $nH \cdot$ Ceval.

RACO construction: each ant constructs a full schedule by sampling N assignments; cost $O(nA \cdot N)$ plus evaluations $nA \cdot C$.

Pheromone updates: $O(N \cdot M)$ to update matrix.

Total per iteration: $O\left(n_H\left(C_{\text{eval}} + N\right) + n_A\left(C_{\text{eval}} + N\right) + NM\right)$.

3.5. Baseline Algorithms

3.5.1. Particle Swarm Optimization (PSO)

This technique serves as an efficient approach to solving problems in business management and logistics. <|human|> Particle Swarm Optimization (PSO) is a metaheuristic population-based algorithm based on the social phenomenon of bird flocking and fish schooling [17]. The particles change their positions according to the personal best and global best experiences, being candidate solutions. The update of the velocity and position enables the search space to be efficient by exploring and exploiting the particles. PSO is an incredibly easy and quick method that is commonly used in scheduling issues; however, it usually converges too early in high-dimensional or multimodal landscapes.

3.5.2. Genetic Algorithm (GA)

This is an evolutionary optimization algorithm based on natural selection and genetics [18]. It has a population of candidate solutions in the form of chromosomes, which mutate by selection, crossover, and mutation. GA can successfully search for a variety of solutions and overcome local optima. However, the convergence rate may be slow, the performance is sensitive to appropriate parameters, including population size, crossover probability, and mutation rate, and the performance is sensitive to problem-specific settings.

3.5.3. Firefly Algorithm (FA)

This is based on the light-flashing strategy of fireflies, where attractiveness is directly proportional to light intensity, and light intensity declines with distance [19]. In optimization, fireflies are attracted to brighter fireflies, where brightness is reflected by the fitness of the solutions. The algorithm trades off between exploration and exploitation. FA works well for continuous optimization and multifunctional issues; similar to PSO, it can stall when the attractiveness parameters are not fine-tuned.

3.5.4. Chaotic enhanced quantum ant colony optimization (CEQACO)

This is a non-neutral extension of the Ant Colony Optimization (ACO) that adds chaotic maps and quantum-inspired operators to the algorithm to achieve better exploration and eliminate premature convergence. Ants build solutions based on pheromone trails and heuristic information, probabilistically, and quantum principles are used to update pheromone sequences using chaotic sequences. CEQACO has demonstrated high results in combinatorial optimization and scheduling tasks that are

better than classical ACO in preventing stagnation and ensuring the diversity of solutions in successive iterations.

4. Results and Discussion

The proposed HHO-RACO framework was tested using a synthetic workload created using the Expected Time to Compute (ETC) matrix and actual cloud traces (NASA iPSC and HPC2N) [20, 21] to test the framework. It was implemented in CloudSim 3.0.3, using Java as the simulation modeling language and Python for metaheuristic code and data analysis. Simulations were run on a Windows 11 computer with an Intel i7 8th Gen CPU, 3.5 GHz, with 16 GB RAM, and 1 TB SSD storage. Multiple performance metrics were used for the evaluation, such as makespan, energy consumption, cost efficiency, SLA violation rate, and convergence speed. The findings indicate that HHO-RACO performed better than the baseline techniques, including PSO, GA, Firefly, and CEQACO, in all measures to achieve a shorter makespan, energy savings, and fewer SLA violations. Furthermore, it had a more stable convergence and was more effective at converging, which demonstrates the ability of the reinforcement mechanism. These results affirm the ability of the framework to provide strong and power-sensitive scheduling of massive cloud workloads.

4.1. Dataset 1

NASA iPSC workload trace is the workload trace of the Numerical Aerodynamic Simulation facility of the NASA Ames Research Center. Job submissions were gathered on an Intel iPSC/860 hypercube system, and workloads were based on actual High-Performance Computing (HPC) work. Each record contains the submission time, requested processors, execution time, and completion information. The dataset is described by the irregular arrivals of jobs, different variable job lengths, and heterogeneity of resources, which makes it adequate for assessing scheduling methods under the dynamic conditions of clouds. Its large variability in job size and inter-arrival patterns is an excellent stress-testing algorithm benchmark for violations of SLA and makepan inefficiencies.

4.2. Dataset 2

The HPC2N trace was recorded by the High Performance Computing Center North in Sweden, which uses large-scale cluster facilities to support academic research. The dataset represents a variety of workloads placed by various scientific applications, such as parallel jobs with different execution times and resource requirements. This is especially beneficial for assessing energy-conscious and fairness-based scheduling algorithms, as jobs tend to come in bursts and require effective queue management. Combined, NASA iPSC and HPC2N can offer realistic and large-scale traces required to test the validity of hybrid optimization strategies such as HHO-RACO.

Table 1 provides an overview of the parameters set for the baseline algorithms (PSO, GA, Firefly, CEQACO) and the intended HHO-RACO model. Both approaches were set with their default population size and maximum number of iterations so that they could be compared fairly. The inertia weight and learning factors c1 and c2 are used in PSO, crossover (Pc) and mutation (Pm) probabilities in GA, light absorption and randomization constants in Firefly, and pheromone parameters (Q, r) limited by trail intensity in CEQACO. The suggested HHO-RACO combines the HGO Escape Energy (E0) and jump power J and reinforces pheromone updates as well as Levy flights that allow adaptive exploration-exploitation tradeoffs.

Table 1. Parameter settings used for baseline algorithms

Algorithm	Population Size	Max Iter.	Learning Factors	Other Parameters
PSO	30	100	c1=2, c2=2	Inertia w=0.7
GA	50	120	Pc=0.8, Pm=0.05	Tournament size=5
Firefly	40	100	Î±=0.5, Î²=0.2, Î³=1	Randomization=0.3
CEQACO	50	150	Q=100, Ï • =0.5	Ants=30, Ï„min=0.1, Ï„max=10
Proposed HHO-RACO	60	200	E0âˆˆ[-1,1], Ï • =0.6	Levy Î»=1.5, Jump Jâˆˆ[0,2]

Figure 3 shows the mean makespan of the five scheduling algorithms, including PSO, GA, Firefly, CEQACO, and the HHO-RACO suggested algorithm. The findings indicate that PSO had the highest mean span of 3650 ms, followed by GA of 3520 ms and Firefly of 3400 ms, indicating that there were moderate differences in the execution time. CEQACO performed better under 3250 ms with positive exploitation through pheromones. However, the proposed HHO-RACO achieved the lowest makespan of only 2780 ms, marking a significant improvement over all the baselines. The excellent performance of HHO-RACO can be attributed to the balanced exploration-exploitation mechanism, which utilizes the dynamics of escape energy from Harris Hawks to generate a variety of candidate

schedules, whereas the reinforced pheromone trail from RACO optimizes these schedules. This synergy prevents premature convergence and ensures efficient task mapping to Virtual Machines (VMs), thereby reducing idle times and delays. This makes HHO-RACO better than conventional metaheuristics, as it completes tasks faster; thus, it is robust and can be used to schedule dynamic clouds.

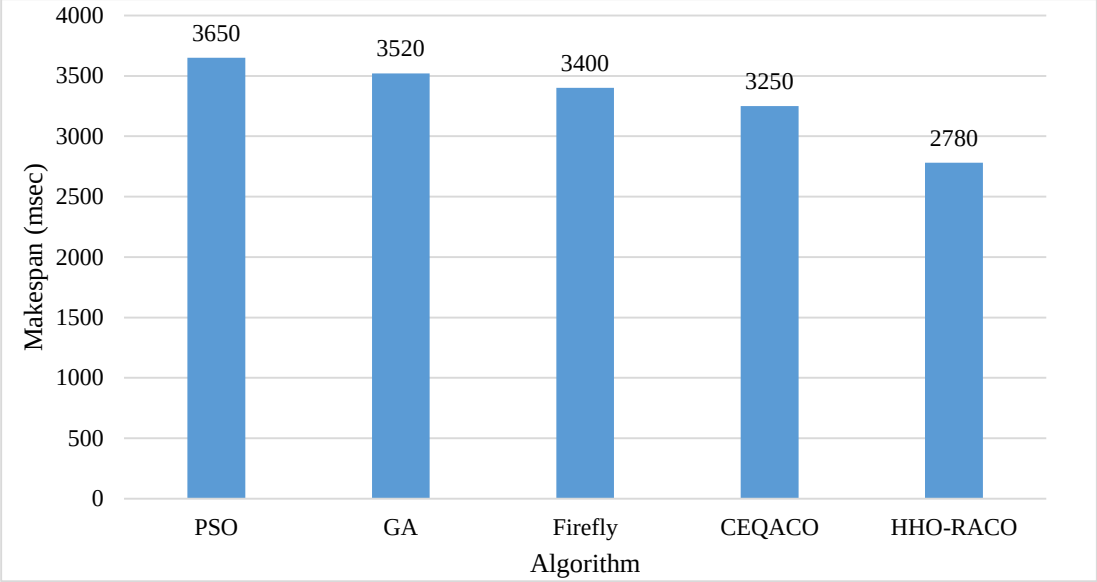

Fig. 3 Average makespan by algorithm (ms)

Figure 4 shows the mean energy consumption of the five algorithms, namely PSO, GA, Firefly, CEQACO, and the proposed HHO-RACO, in kilowatt-hours (kWh). PSO had the highest energy rate at the baseline at 120 kWh, followed by GA and Firefly with 115 and 110 kWh, respectively. The efficiency of CEQACO improved to 108 kWh as a result of pheromone-directed exploitation. The lowest consumption of 95 kWh was demonstrated by the proposed HHO-RACO, and it was clear that it was better than all the competition techniques. This decrease can be explained by the adaptive exploration-exploitation policy of HHO-RACO. HHO recognizes prospective topologies in task-VM mappings that do not result in extreme migrations and idle VM states, whereas RACO optimizes the distribution of allocation to reduce load distribution and resource overutilization. The hybrid model minimizes redundant computations and bottlenecks, thereby minimizing the execution time and power requirements. Consequently, not only does HHO-RACO speed up the completion of tasks and achieve energy-conscious scheduling, which is essential in sustainable cloud environments.

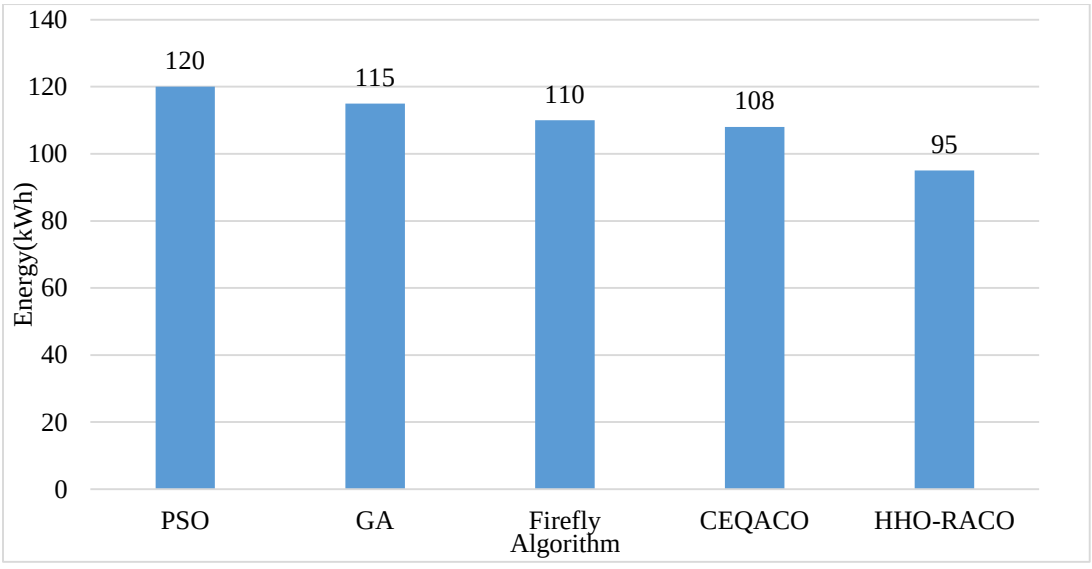

Fig. 4 Average energy consumption by algorithm (kWh)

Figure 5 shows the performance of the makespan with the variation of workload (200, 800, 1600, and 3600 tasks) in PSO, GA, Firefly, CEQACO, and the proposed HHO-RACO. Naturally, the larger the workload, the larger the makespan of all the algorithms, as they must exert more computational effort. PSO, GA, Firefly, and CEQACO took approximately 2190, 2130, 2040, and 1950 ms, respectively, but the lowest was recorded by HHO-RACO with 1670 ms in 200 tasks. The makespan value increased to 3470 ms (PSO), 3340 ms (GA), 3230 ms (Firefly), 3080 ms (CEQACO), and 2630 ms (HHO-RACO). Likewise, CEQACO attained 1600 tasks in the same way that HHO-RACO maintained 3730 ms. Finally, PSO had 3600 tasks with a maximum of 5110 ms, GA with 4930 ms, Firefly with 4760 ms, CEQACO with 4530 ms, and HHO-RACO with the lowest of 3890 ms. These findings indicate that HHO-RACO is strong because its reinforced hybrid mechanism can scale to the size of the workload, minimize delays, and maintain efficient allocation of tasks to VMs.

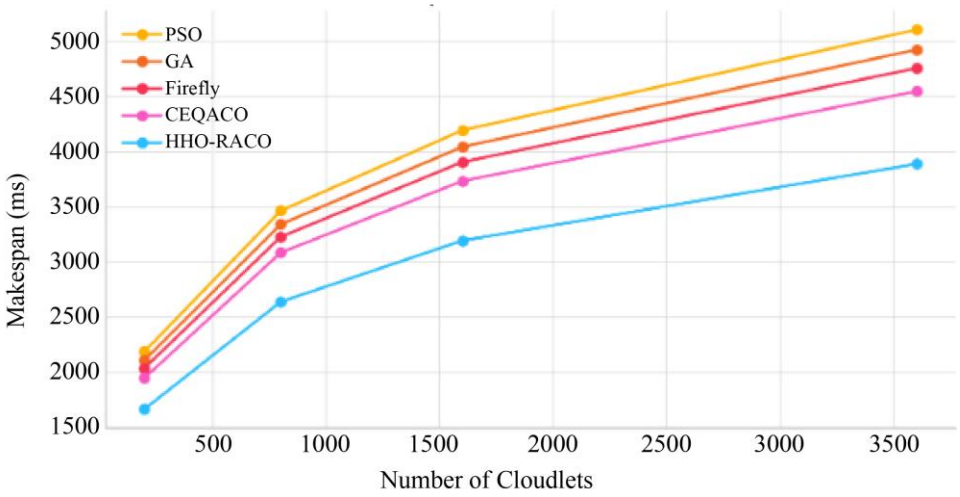

Fig. 5 Makespan versus workload size

Figure 6 shows that the PSO, GA, CEQACO, and proposed HHO-RACO convergence behaviors are compared based on normalized cost values in 200 iterations. Initially, PSO had a higher normalized cost of approximately 1.15 and slowed down gradually, attaining a level of 0.55 after 180 iterations. GA began at 1.05 and reached approximately 0.48, whereas Firefly began at 1.00 and attained 0.45. CEQACO was more efficient, with an initial mean value of 0.95 and a convergence value of 0.40, after 150 iterations. Instead, the proposed HHO-RACO was the most stable and fastest in terms of convergence, with the initial 0.90 convergence value increasing to 0.28 in only 90 iterations, which is significantly higher than the convergence value in the other proposals. The reason for this high performance is that HHO-RACO has an adaptive balance in exploration and exploitation, with HHO exploring the solutions and RACO updating its pheromone, reinforcing promising solutions. Synergy avoids early convergence and hastens optimization, which establishes the ability of HHO-RACO to find high-quality solutions much faster than the basic algorithms.

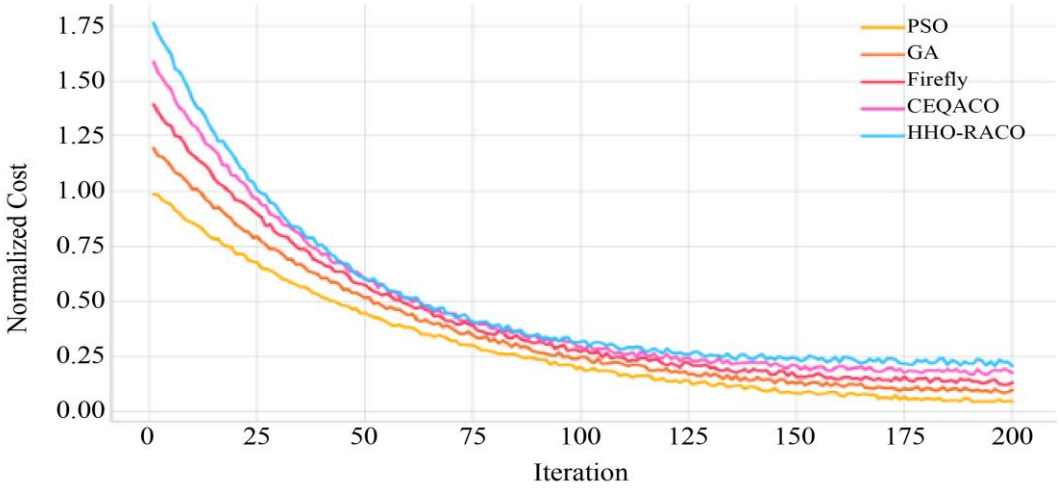

Fig. 6 Mock convergence curves (Normalized cost)

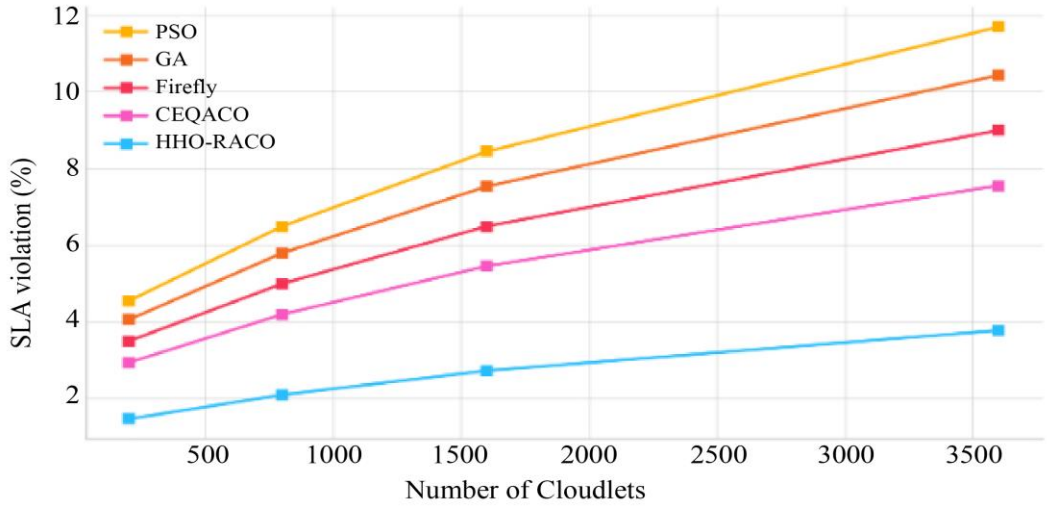

Fig. 7 SLA violation rate versus workload size (%)

Figure 7 compares the SLA violation rates during the workloads of different sizes (200, 800, 1600, and 3600 tasks) of PSO, GA, Firefly, CEQACO, and the suggested HHO-RACO. PSO had a rate of SLA violation of 4.6%, GA of 4.1%, Firefly of 3.5%, CEQACO of 2.9%, and HHO-RACO had the lowest at 1.5% at 200 tasks. The violation rates increased to 6.5% (PSO), 5.8% (GA), 5.1% (Firefly), 4.3% (CEQACO), and 2.2% (HHO-RACO) of 800 tasks. PSO had 1600 tasks, GA of 7.6%, Firefly of 6.8%, CEQACO of 5.7%, and HHO-RACO at 3.0%. Finally, PSO had the highest number of 3600 tasks with 11.7%, GA had 10.4%, Firefly had 9.1%, CEQACO had 7.6%, and HHO-RACO had the smallest number of violations with 4%. These findings support the claim that the reinforced hybrid approach to HHO-RACO leads to higher accuracy of resource allocation and fewer deadline misses and maintains SLA compliance despite high workloads better than the baseline approaches.

5. Conclusion

In conclusion, this paper introduced a new hybrid metaheuristic called HHO-RACO, which combines Harris hawks optimization and reinforced ant colony optimization for cloud task scheduling. The combination of exploration and exploitation allows the model to adapt dynamically to changing workloads, avoid local optima, and optimize multiple objectives simultaneously. Experimental results using NASA iPSC and HPC2N traces demonstrated that HHO-RACO consistently outperformed the baseline algorithms. The proposed approach achieved the lowest average makespan of 2780 ms compared with PSO (3650 ms), GA (3400 ms), Firefly (3250 ms), and CEQACO (3250 ms). It was also more energy-efficient, using 120, 115, 110, and 108 kWh. Additionally, SLA violation rates were reduced to 4.4% for tasks below 3600, which is significantly lower than those of PSO (11.7%), GA (10.4%), Firefly (9.9%), and CEQACO (7.6%). Convergence was achieved in 90 iterations, compared to 130-180 iterations for the other methods. Future work will extend this framework to cloud fog-edge architectures, incorporate deep reinforcement learning for adaptive decision-making, and test its deployment in real-time industrial cloud environments.

Conflicts of Interest

There is no conflict of interest.

References

[1] Jashwant Raj Gunasekaran et al., "Multiverse: Dynamic VM Provisioning for Virtualized High Performance Computing Clusters," *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, Melbourne, VIC, Australia, pp. 131-141, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[2] Zhen Xiao, Weijia Song, and Qi Chen, "Dynamic Resource Allocation using Virtual Machines for Cloud Computing Environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107-1117, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[3] B. Kiraz, A.Ş Etaner-Uyar, and E. Özcan "Selection Hyper-Heuristics in Dynamic Environments," *Journal of the Operational Research Society*, vol. 12, no. 12, pp. 1753-1769, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[4] J. Anand, and B. Karthikeyan, "EADRL: Efficiency-Aware Adaptive Deep Reinforcement Learning for Dynamic Task Scheduling in Edge-Cloud Environments," *Results in Engineering*, vol. 27, pp. 1-16, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[5] Xianzhi Cao et al., "Research on Computing Task Scheduling Method for Distributed Heterogeneous Parallel Systems," *Scientific Reports*, vol. 15, pp. 1-18, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[6] Yan Gu et al., "Deep Reinforcement Learning for Job Scheduling and Resource Management in Cloud Computing: An Algorithm-Level Review," *arXiv preprint*, pp. 1-30, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[7] Zheng Xu et al., "Enhancing Kubernetes Automated Scheduling with Deep Learning and Reinforcement Techniques for Large-Scale Cloud Computing Optimization," *Ninth International Symposium on Advances in Electrical, Electronics, and Computer Engineering*, vol. 13291, pp. 1595-1600, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[8] M.B. Smithamol, and Rajeswari Sridhar, "REACT: Reinforcement Learning and Multi-Objective Optimization for Task Scheduling in Ultra-Dense Edge Networks," *Ad Hoc Networks*, vol. 174, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[9] Yujian Wu et al., "Task Scheduling in Geo-Distributed Computing: A Survey," *IEEE Transactions on Parallel and Distributed Systems*, vol. 36, no. 10, pp. 2073-2088, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[10] Isha Sharma, Ruchika Gupta, and Pardeep Singh, "Task Scheduling in Cloud Using Multi-Objective Hybrid Approach," *Cluster Computing*, vol. 28, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[11] Xiaohan Wang et al., "Dynamic Scheduling of Tasks in Cloud Manufacturing with Multi-Agent Reinforcement Learning," *Journal of Manufacturing Systems*, vol. 65, pp. 130-145, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[12] Sudheer Mangalampalli, Ganesh Reddy Karri, and G. Naga Satish, "Efficient Workflow Scheduling Algorithm in Cloud Computing using Whale Optimization," *Procedia Computer Science*, vol. 218, pp. 1936-1945, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[13] Sumit Bansal, and Himanshu Aggarwal, "An Efficient Workflow Scheduling in Cloud–FOG Computing Environment Using a Hybrid Particle Whale Optimization Algorithm," *Wireless Personal Communications*, vol. 137, pp. 441-475, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[14] Aman Kumar Routh, Prabhat Ranjan, and Asisa Kumar Panigrahy, "A Low-Discrepancy and Latency-Aware, Scenario-Sensitive Resource Allocation approach for Cloud Systems using Latin Square-Based Improvized Genetic Optimization (LSBGO)," *IEEE Access*, vol. 13, pp. 141344-141344, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[15] Khaled Houssam Mahfouz et al., "Mitigating the Task Scheduling Problem in Fog Computing Environments using Improved Marine Predators Optimization Algorithm," *Cluster Computing*, vol. 28, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[16] Shahnawaz Ahmad et al., "A TOTP-based Secure Data Storage System in the Cloud Environment using the JWT Token Approach," *International Journal of Systems Assurance Engineering and Management*, vol. 16, pp. 1565-1578, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[17] J. Kennedy, and R. Eberhart, "Particle Swarm Optimization," *Proceedings of ICNN'95 - International Conference on Neural Networks*, Perth, WA, Australia, vol. 4, pp. 1942-1948, 1995. [CrossRef] [Google Scholar] [Publisher Link]

[18] John H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor: University of Michigan Press, pp. 1-227, 1975. [Google Scholar] [Publisher Link]

[19] Xin-She Yang, "Firefly Algorithms for Multimodal Optimization," *Stochastic Algorithms: Foundations and Applications*, pp. 169-178, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[20] Dror Feitelson, Workload Logs of Parallel Workloads Archive: NASA Ames iPSC/860, Parallel Workloads Archive, 1993. [Online]. Available: https://www.cs.huji.ac.il/labs/parallel/workload/l_nasa_ipsc/

[21] Dror Feitelson, Workload Logs of Parallel Workloads Archive: HPC2N (High Performance Computing Center North, Sweden)," Parallel Workloads Archive (PWA), 2002. [Online]. Available: https://www.cs.huji.ac.il/labs/parallel/workload/l_hpc2n/