Journal of Computer Science

Research Article

Optimized Cloud Load Balancing Using Deep Q-Learning With Inverted S-Shaped Hierarchical Reverse Superb Fairy-Wren Optimization

B. R. Yogeetha and S. P. Anandaraj

Department of Computer Science and Engineering, Presidency University, Yelahanka, Bengaluru, Karnataka, India

Article history
Received: 06-10-2025
Revised: 02-03-2026
Accepted: 04-05-2026

Corresponding Author:
B. R. Yogeetha
Department of Computer Science and Engineering, Presidency University, Yelahanka, Bengaluru, Karnataka, India
Email: yogeetha.20223CSE0029@presidencyuniversity.in

Abstract: Cloud computing offers a scalable and cost-effective platform by providing on-demand access to shared computational resources. However, effective load balancing is essential to maintain optimal performance and maximize resource utilization for ensuring even distribution of network traffic across servers, preventing overload, improving response time and enhancing the system reliability. This manuscript proposes the Deep Q-Network with Inverted S-shaped Hierarchical Reverse Superb Fairy-wren Optimization (DQN with ISHR-SFO) model for adaptive load balancing on cloud computing. DQN utilizes Reinforcement Learning (RL) to predict optimal task-to-VM allocations based on states and rewards. The traditional SFO algorithm improved global exploration and local exploitation by incorporating inverted S-shaped escape energy and hierarchical reverse learning to prevent premature convergence and improve the search diversity. The proposed model obtained lower execution time of 1443s, makespan of 8495s, energy consumption of 35.25 J, high throughput of 120 kbps and 99.12% resource utilization. Experimental evaluation using CloudSim determines that the proposed model effectively minimizes energy consumption, makespan and execution time, while increasing throughput and resource utilization compared to traditional algorithms.

Keywords: Cloud Computing, Deep Q-Network, Inverted S-Shaped Hierarchical Reverse, Load Balancing, Reinforcement Learning and Superb Fairy-Wren Optimization

Introduction

With the availability of the internet and the growing demand for high-performance computation, cloud computing has evolved rapidly. Cloud computing offers both hardware and software resources to process tasks (Rajawat et al., 2024; Choppara and Lokesh, 2025; Li et al., 2024). It is defined as an internet-based method that provides shareable resources such as memory, networks, and applications to end users (Zhanuzak et al., 2024). Cloud services dynamically assign tasks to Virtual Machines (VMs) that are made available to end users (Khan, 2024). Cloud computing involves scheduling and allocating available sources for the tasks submitted by end users. The outcome of this procedure result in skewed source utilization, where certain sources are overutilized while others remain underutilized (Zhang, 2024). If errors such as machine failures occur, then the resource utilization becomes inefficient; therefore, load balancing among resources is essential (Brahmam and Reddy 2024). Since cloud load balancing significantly affects overall system performance, concerns arise regarding compromise between the financial interests of cloud service providers and the level of end-user satisfaction (Kumar et al., 2025; Chraibi et al., 2021). The agreements between cloud service providers and end users, known as Service Level Agreements (SLAs), are considered in load balancing. Many cloud data centers are developed to handle multiple requests. However, allocation requests result in high energy consumption and lower resource utilization. As resource utilization directly impacts energy consumption, optimizing load balancing is essential in cloud computing environments.

Load balancing is employed to balance the load among physical and virtual machines as jobs are allocated within cloud data centers, ensuring that nodes maintain similar

loads (Verma, 2024). Mapping of tasks for various resources is performed to optimize energy consumption. An effective load-balancing model enables efficient resource utilization and helps minimize carbon footprint of data centers, and reduces the requirements of machine cooling (Menaka and Kumar, 2024). Each physical machine operating in a data center generates heat, and prolonged utilization of physical machines lead to the formation of hot spots. Load balancing issues are resolved in various phases by grouping them into heuristic scheduling, metaheuristic-depended scheduling and hybrid metaheuristics (Singal and Verma, 2024; Ullah et al., 2024; Alruwais et al., 2024). Each of these models has its own advantages and disadvantages. Swarm Intelligence (SI) has emerged as one of the most prominent meta-heuristic approaches and has been widely utilized in various ways over the past few years. The primary motivation behind SI idea is the observation that various social swarms exhibit coordinated group behavior (Akerele et al., 2024). Individuals within these swarms operate collectively through coordinated and interactive devices to reach various types of goals. The natural collective behavior of each swarm among swarms enhances the process efficiency of interactive device. Recently, SI approaches have been developed to resolve various optimization issues, particularly in resource managing tasks such as load balancing, which require robustness and flexibility (Abdulghani, 2024).

Problem Statement

Cloud computing environment has difficulties in obtaining effective load balancing, especially under dynamic and large-scale task loads. Traditional models suffer from higher energy consumption, increased makespan and suboptimal resource utilization because of static scheduling policies and limited adaptability to changing workloads. As the number of tasks and Virtual Machines (VMs) scales up, these models struggle to handle high throughput and responsiveness. Additionally, they fall into local optimal and ineffective exploration patterns that causes underutilization of available resources and increased operational costs.

Objective

The main objective of this manuscript is to develop a load balancing model by integrating Deep Q-Network (DQN) with Superb Fairy-wren Optimization (SFO) to improve task scheduling efficacy in the cloud. The model aims to reduce makespan and execution time, while minimizing the energy consumption and maximizing the throughput and resource utilization. The decision-making ability of Reinforcement Learning (RL) and an adaptive exploration-exploitation behavior of SFO includes inverted S-shaped escape energy and hierarchical reverse learning. The proposed model dynamically optimizes task to VM allocation, thereby ensuring scalability and robustness under different workloads and VM configurations.

Contribution

The primary contribution of this manuscript is described below:

1) The integration of Deep-Q-Network (DQN) with Superb Fairy-wren Optimization (SFO) is developed to enable an intelligent and adaptive task scheduling for efficiently balancing exploitation and exploration
2) An inverted S-shaped escape energy mechanism was incorporated within SFO to maintain high exploration in early iterations and high exploitation in later phases, thereby minimizing the premature convergence and enhancing global search efficiency
3) A hierarchical reverse learning was included to preserve the algorithm from the local optima stagnation, which improves solution diversity and accelerates convergence towards optimum task to VM mappings
4) DQN's Q-value-based policy learning was used to dynamically predict the optimal task-to-VM allocations, while enabling scheduling decisions that adapts to changing model decisions and workload patterns

Literature Review

Kalaskar and Thangam (2024) presented an innovative strategy to improve fault tolerance and load balancing abilities of cloud computing, which integrated Graph Neural Networks (GNNs) with Dynamic Multiqueue Optimization Scheduling (DMQOS). The presented model used DMQOS, which was adapted to dynamic nature of cloud workloads. This dynamic model improved response time and resource consumption that enhanced load balancing and the efficacy of the model. GNNs were employed for predicting and mitigating faults also to safeguard the accessibility of service. The presented model failed to adapt to unpredictable and highly variable task loads, which led to VM overloading and underutilization.

Krishna and Vali (2025) developed Meta Reinforcement Learning Driven Hybrid Lyrebird Falcon Optimization for Dynamic Load Balancing in Cloud computing (Meta-RHDC), a novel method for dynamic load balancing on cloud environments. The developed method used convolutional and recurrent neural network for predicting loads of VMs and dynamically classified them into overloaded and underloaded classes. The developed approach unoptimized task allocation led to excessive energy utilization, thereby maximizing operation costs and carbon footprint.

Singhal et al. (2024) used metaheuristic-based algorithm to dynamically adjust the load distribution, while ensuring resilience and sensitivity for modifying the workloads also handling energy consumption. Here, the presented Rock Hyrax-based load balancing model addressed local maxima and energy efficacy problems by utilizing QoS parameters. The model's performance was validated qualitatively and statistically considered the static and dynamic modes of works and VMs. The model had long makespan and task execution time that degraded the model performance and violated SLAs.

Ghorbani et al. (2025) introduced load balancing algorithm that used Learning Automate (LA) to distribute real-time tasks among edge and cloud servers dynamically. Through continuous learning from prior experiences, the model adapted to changing workloads and network conditions, thereby ensuring optimum task allocation. The introduced model applied Service Time Measurement (STM) metric for validating the server's performance and made informed decisions about task distribution. The introduced model efficiently balanced workload among edge and cloud servers by taking parameters such as task complexity, server capacity, and network delay. The introduced model left a huge portion of resources underused due to poor load distribution.

Simaiya et al. (2024) presented Deep Learning with Particle Swarm Optimization and Genetic Algorithm (DPSO-GA) for dynamic workload provisioning on cloud computing. An initial stage of the presented model was employed to address the prediction drawback by integrating the advantages of these two models in fine-tuning hyperparameters. In next stage, Convolutional Neural Network Long Short-Term Memory (CNN-LSTM) was used for forecasting resource consumption and PSO-GA was also utilized to train it. One dimensional CNN and LSTM were utilized for forecasting cloud resource usage in different subsequent time steps. LSTM models simulate the temporal data that predicted the upcoming VM workload, while CNN model captured complicated differences from the features that were collected from VM workload statistics. The traditional algorithms often got trapped in local optima because of poor exploration and exploitation balance.

Nagarajan et al. (2025) developed Deep Kronecker ResNetXt Forward Fractional Network (DK-ResNeXt FF Net), which simulated cloud environment and includes multiple Physical Machines (PMs), Virtual Machines (VMs), cloud managers and service providers. Initially, round-robin strategy was used for task distribution by VMs. The VMs classification was performed by Deep Fuzzy Clustering (DFC) approach based on parameters such as Central Processing Unit (CPU), memory usage, frequency scaling factor, Million Instructions Per Second (MIPS), and VM bandwidth. The Gated Recurrent Unit (GRU) was used for precise load prediction and overloaded VMs relocate tasks to under load VMs in DK-ResNetXt FF-Net that was developed by the combination of Deep Kronecker Networks (DKN), ResNeXt and Fractional Calculus (FC).

Sahu and Verma (2026) introduced a hybrid AI-based model that combined Multi-Agent Collaborative Reinforcement Learning (MACRL), Salp Swarm Algorithm (SSA), Neuroevolution of Augmenting Topologies (NEAAT), Energy-Aware Task Scheduling with Deadline (EATSD), Merging Multiple Fused Learning Data Elements (MMFLDE), and Zeroth-Order Optimization (ZOO) for optimizing energy consumption and resource allocation. EATSD mechanism integrated with NEAT ensured adaptive and energy-effective task scheduling by using neural network models.

The proposed model resolved significant challenges in cloud load balancing by integrating DQN for policy learning and SFO for adaptive global optimization. This model dynamically allocated tasks to VMs, minimized energy and time costs, increased resource utilization and maintained performance in scale. The improved mechanisms such as inverted S-shaped escape strategy and hierarchical reverse learning ensured robust convergence, which made the mode efficient and scalable for cloud environments.

The proposed model follows interaction between DQN and ISHR-SFO. Initially, DQN performs policy learning by observing cloud state (task load, VM capacity, resource utilization) and produces initial task-to-VM allocation based on learned Q-values. The cloud environment returns reward based on makespan, energy consumption, and resource utilization. The generated allocation policy is provided to ISHR-SFO module as initial population. ISHO-SFO refines theses task-to-VM mappings by global exploration, local exploitation and diversity preservation. The inverted S-shaped escape energy controls exploration-exploitation balance dynamically. The refined optimal mapping acquired from ISHR-SFO is re-evaluated in environment and updated state-action-reward is stored in replay buffer. This refined experience enhances DQN training stability and accelerates convergence towards optimal scheduling policies. Hence, DQN provides intelligent policy learning, when ISHR-SFO improves solution refinement and prevents premature convergence, thereby developing cooperative closed-loop optimization algorithm.

AI-based automation in production systems determines how ML and intelligent decision-making models effectively improve process efficacy accordingly. Such advancements ensure the significance of adopting AI-based optimization algorithms in cloud resource allocation and load balancing, where the dynamic decision-making for scalable service delivery.

Proposed Section

In this manuscript, the proposed DQN is integrated with ISHR-SFO for dynamic load balancing on cloud computing. DQN component learns optimum task-to-VM allocation policies via Reinforcement Learning (RL) by maximizing reward function that represents energy efficacy, resource utilization, and reduced makespan. To improve global search and avoid local optima, SFO algorithm refines task mappings for an adaptive exploration and exploitation. Additionally, inverted S-shaped escape energy mechanism is presented to maintain exploration in early stages that enables strong convergence, whereas hierarchical reverse learning prevents stagnation and preserves population diversity. The integration of DQN and ISHR-SFO enables the model to dynamically adjust for varying workloads and VM availability ensures optimum load distribution, minimized energy consumption, maximized throughput, and maximized resource utilization across cloud environments. Fig. 1 represents the overall process of DQN and ISHR with SFO model.

Reinforcement Learning (RL) Formulation

RL models are decision-making models in dynamic environments with states, actions, and rewards to optimize the strategy of load balancing. Its mathematical expression is given in Eq. 1:

$$Q(s,a) = E_n \int_{t_{io}}^{\infty} e^{-\alpha(t_i - t_{io})} r(t_i) dt \,|S_0 = S, A_0 = A \qquad (1)$$

The above Eq. 1 represents the function of Q-value in RL, and estimates the expected reward for taking action a in state s. $Q(s,a)$ is calculated as expected value E_n of integral over future rewards $r(t_i)$.

The $e^{-\alpha(t_i - t_{io})}$ term represents the discount factor and ensures that early rewards have higher impact than later ones, thereby controlling the impact of future rewards.

Deep Q Network

DQN utilizes Q-learning approach, which is one of the mostly used Reinforcement Learning (RL) models. DQN architecture contains convolution, dropout, MaxPooling, flatten, and dense layers. The predicted discounted cumulative reward is determined as action values. This is the process of RL and their mathematical expression is given in Eq. 2:

$$Q_l^{\pi}(A_l, w_l) = J\left[\sum_{p=0}^{\infty} \eta^p x_{l+p} | A_l = A, w_l = w; \pi\right] \qquad (2)$$

In the above Eq. 2, the J is an expectation, optimal action-value process is defined as $J^*(A,W) = max_{\pi} Q^{\pi}(A,w)$ that obtains Bellman optimum expression.

Q-Learning Update Rule

Q-learning update rule updates Q-values based on Bellman equation that supports an agent to learn optimum actions by exploration and exploitation, and their mathematical expression is given in Eq. 3:

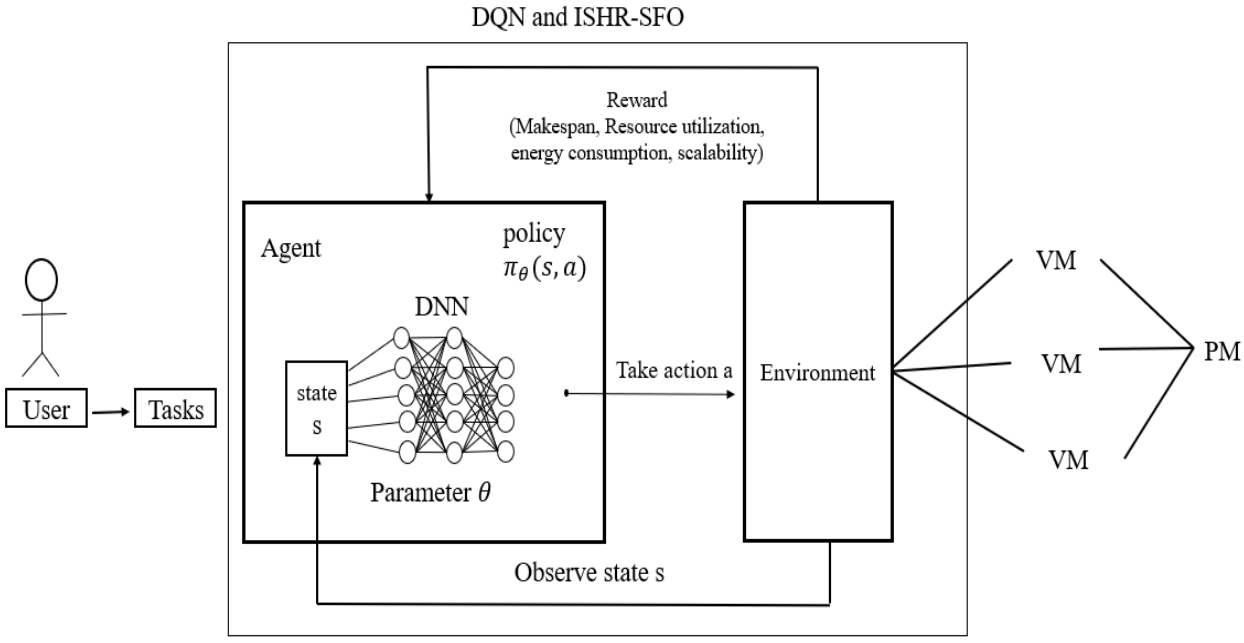

Fig. 1: Overall process of DQN and ISHR with SFO model

$$Q^{(n+1)}(S_n, A_n) \leftarrow Q^{(n)}(S_n, A_n) + \beta \left[\frac{1-e^{\alpha T_k}}{\alpha} r(S_n, A_n) + \max_{a^{-1}}^{\alpha T_k} Q^{(n)}(S_{n+1}, A_n) - Q^{(n)}(S_n, A_n)\right] \quad (3)$$

In the above, Eq. 3 is a Q-learning update rule that determines updated Q-value function in reinforcement learning that incorporates an exponential reward adjustment. The $Q^{(n+1)}(S_n, A_n)$ is an updated Q-value to state S_n and action A_n. The β is a learning rate and reward adjustment is influenced by the exponential decay factor $e^{\alpha T_k}$ to emphasize time-sensitive scheduling decisions, where T_k represents present scheduling time step and α controls growth rate. This mechanism acts as a reward shaping strategy to prioritize faster task completion and accelerate convergence. For stability, α is selected within small bound range to prevent divergence in training. This also includes maximum value function across probable actions in the next state, which ensures optimum action selection. Subtraction of $Q^{(n)}(S_n, A_n)$ ensures that the update is relative to the prior value, and maintains balance between new data and prior knowledge.

Reward Function

Reward function determines the feedback mechanism in RL, which ensures effective resource allocation by increasing the model's performance and their mathematical expression is given in Eq. 4:

$$R = \sum_{r=0}^{T} \gamma^t r_t \quad (4)$$

The above equation is a discounted cumulative reward in RL. The R represents the total reward which is acquired through adding immediate rewards r_t across time, and weighted through discount factor γ^t, which determines how much future rewards contributes to total reward with little values while prioritizing immediate rewards.

Minimization of Makespan

Makespan defines the total time requirement to complete whole tasks. The lower makespan represents the best scheduling efficacy and their mathematical expression is given in Eq. 5:

$$M_{opt} = \max_{i \in Tasks}(C_i) \quad (5)$$

The above Eq. 5 defines makespan, which is the maximum completion time among all tasks. The M_{opt} represents completed as latest completing time among all tasks that defines the overall time requirement to finish all tasks.

Resource Utilization

It calculates the ratio of original resource utilization to total available resources. The higher utilization enables an effective workload distribution and their mathematical expression is given in Eq. 6:

$$R_{util} = \frac{\sum_{i=1}^{N} R_i}{\sum_{i=1}^{N} R_{max}} \quad (6)$$

In the above Eq. 6, the R_{util} calculates the ratio of actual resource utilization to total available resources. The high R_{util} value represents the effective resource utilization and ensures balance load distribution in the network.

Energy Consumption

Energy consumption calculates the total energy consumed by calculating resources in task execution. Less energy consumption enhances the network sustainability and their mathematical expression is given in Eq. 7:

$$E_{con} = \frac{\sum_{i=1}^{N} P_i \times U_i \times T}{\sum_{i=1}^{N} P_{max} \times T} \quad (7)$$

In the above Eq. 7, the E_{con} calculates an energy efficiency of the network, the higher E_{con} values represent better energy efficacy and optimized resource utilization.

Scalability Efficiency

Scalability efficiency calculates how efficiently the network adapts to increased workloads. The higher scalability efficacy ensures optimum performance under varying loads and their mathematical expression is given as Eq. 8:

$$S_{eff} = \frac{\sum_{i=1}^{N} S_i \times U_i}{\sum_{i=1}^{N} S_i} \quad (8)$$

In the above Eq. 8, the S_{eff} calculates how effectively the network scales by estimating resource usage across various load conditions. The higher S_{eff} value represents better scalability, and ensures optimum performance across maximizing workloads.

Superb Fairy-Wren Optimization (SFO) Algorithm

Superb Fairy-Wren Optimization (SFO) algorithm is a new metaheuristic approach that mimics various behaviors of Superb Fairy-wrens that includes growth of young birds, breeding, feeding and avoiding natural enemies. Specific phases are described below:

Random Initialization

The SFO randomly produces initial population in search space. Population includes of N members and first location of ith member X_i^{ini}, its mathematical expression is given in Eq. 9:

$$X_i^{ini} = lb + rand(1, D) \times (ub - lb) \quad (9)$$

In the above Eq. 9, the ub and lb represents upper and

DOI: 10.3844/jcssp.2026.1797.1810

lower bounds of search area, $rand(1, D)$ is a vector of D-dimensional random vectors, where every component is uniformly distributed among 0 and 1.

Growth Phase of Young Birds

The initial phase of SFO is the development phase of a young bird. In this stage, the population constantly updates the location of young birds as their development enables global search. The mathematical expression for updating the position of each member is given in Eq. 10:

$$X_i^{new} = X_i^{old} + rand(1, D) \times (ub - lb) + lb \quad (10)$$

In the above Eq. 10, the X_i^{old} represents the present position of ith member and the X_i^{new} represents the updated position.

Breeding and Feeding Stage

This is a second phase of SFO, where SFO updates the position of population members by developing a teaching mechanism of superb Fairy-wren in breeding and feeding, and their mathematical expression is given as Eq. 11:

$$X_i^{new} = a \times X_{best} + \left(X_{best} - X_i^{old}\right) \times \sin\left((ub = lb) \times \left(2 + \frac{2FEs}{MaxFEs}\right)\right) \quad (11)$$

In the above Eq. 11, a represents the constant value of 0.8 in accordance with source literature, the X_{best} represents the location of a global best member. FEs is the count of fitness function, the $MaxFEs$ is a maximum count of fitness function calculations.

Avoiding Natural Enemies Phase

This is the last phase of SFO, the simulation is performed on defense mechanism of Superb Fairy-wren to avoid predation by natural enemies. SFO updates the population location, whose phase improves the search range of the population in a search area and their capability to use local search, and their mathematical expression is given in Eq. 12:

$$X_i^{new} = X_{best} + X_i^{old} \times l \times b \times \sin\left(\frac{\pi}{2} \times \left(1 - \frac{FEs}{MaxFEs}\right)\right) \quad (12)$$

In the above Eq. 12, the l represents D-dimensional random vector considers Levy distribution, the b represents constant score of 0.2.

SFO Implementation

In every iteration, SFO selects one of the three phases for updating population. The growth phase of young birds is primarily utilized for global exploration. Breeding and feeding phase preferred local exploitation and "avoiding natural enemies" phase is utilized for maximizing the randomness of SFO exploration on search area. In SFO, the probability of computing growth phase of young birds is similar to probability of computing breeding and feeding phase as well as "avoiding natural enemies" phase, which decides the phase to compute through the judging size of r_1 and 0.5 in every iteration. While $r_1 > 0.5$, SFO executes the growth phase of young birds, the SFO executes one of the remaining two phases. Next, to breeding and feeding phase as well as avoids natural enemies phase, SFO chooses in accordance with a magnitude of the threshold s and their mathematical expression is given in Eq. 13. While $s > 20$, avoids natural enemies phase is computed, then breeding and feeding phase is employed. Mathematical expression for execution mechanism of SFO is given in Eq. 14:

$$s = r_2 \times 20 + r_3 \times 20 \quad (13)$$

$$X_i^{new} = \begin{cases} X_i^{old} + rand(1, D) \times (ub - lb) + lb, r_1 > 0.5 \\ a \times X_{best} + \left(X_{best} - X_i^{old}\right) \times \sin\left((ub - lb) \times \left(2 + \frac{2FEs}{MaxFEs}\right)\right), r_1 \le 0.5 \; and \; s \le 20 \\ X_{best} + X_i^{old} \times l \times b \times \sin\left(\frac{\pi}{2} \times \left(1 - \frac{FEs}{MaxFEs}\right)\right), r_1 \le 0.5 \; and \; s > 20 \end{cases} \quad (14)$$

In the above Eq. 14, the r_1 represents random number on [0,1], the r_2 and r_3 represents random numbers to standard normal distribution.

Inverted S-Shaped Escape Energy

Search space is defined by the escape energy of a prey within SFO algorithm. Generally, while absolute score of escape energy is represented as $|E|$ is huge, superb fairy-wren is much motivated to enter an exploration stage. Though, the escape strategy of prey shows linear decreasing, which represents constant rate of energy reduction. Fast decline on escape strategy in early exploration stage resulted in the population diversity loss and premature convergence in the following exploitation stage. Therefore, an inverted S-shaped process was used to describe the dynamics of prey escape strategy that exploits on characteristics of slow decline rates in early and final phases decline rate in middle phase. The mathematical expression for determining an inverted S-shaped escape energy is given in Eq. 15:

$$E = 2E_0 \times \left(1 - \frac{1}{1 + e^{a - bt}}\right) \quad (15)$$

In the above Eq. 15, the a, b represents parameters which manage the shape of an inverted S-shaped function that are set to 5 and $15/T_{max}$, t represents the present count of iterations. This strategy ensures the escape energy of prey to maintain a huge value of prolonged time on initial phase, facilitated by global exploration. This indicates that the escape strategy remains small value to

long time in the final phase, thereby enhancing local exploitation.

Hierarchical Reverse Learning

The individuals lead to gradually gathered in certain local spaces in iteration process, which makes the algorithm fall into local optima. To avoid premature convergence, the SFO is integrated with hierarchical reverse learning. This strategy is required to measure each individual's fitness value in population and rank it as Rank 1, Rank 2, …, Rank N from small to enormous. Next, N individuals are separated to NL levels, each with LS individuals. The L_1 shows best grade of individuals, the L_{NL} represents worst grade. The quality of reverse individual is not better than that of present individual. There are still certain individuals whose quality declines after reverse learning. The acquired individual $X_{opposite}$ and prior phase optimum individual X_j^{best} integrates to acquire reverse individual towards better. The mathematical expression to this process is given in Eq. 16:

$$X_j' = \lambda \times X_{opposite} + (1 - \lambda) \times X_j^{best} \quad (16)$$

Algorithm 1 – DQN and ISHR-SFO for load balancing in cloud

Input – Number of tasks $T = \{t1, t2, \dots, tn\}$, Number of VMs $V = \{v1, v2, \dots, vm\}$, $max_{episodes}$, $max_{iterations}$, ε exploration rate, α learning rate, γ discount factor and Replay Buffer
Initialize
    Q-network weights θ randomly
    Target Q-network weights $\theta' \leftarrow \theta$
    Initialize SFO population with random task-VM mappings
    Initialize inverted S-escape energy E
    Set episode to 1
While $episode \leq max_{episodes}$ do:
    Initialize cloud environment in Cloud Sim
    Observe initial state S_0 from environment
For every time step t do:
    With probability ε
        Choose random action a_t
Else
    Choose action $a_t = argmax_a\, Q(s_t, a;\, \theta)$
    Compute action $a_t \rightarrow allocate\ task\ to\ VM$
        Observe reward r_t and next state $s_{\{t+1\}}$
        Store experience $(s_t, a_t, r_t, s_{\{t+\}})$ on Replay Buffer
Sample mini-batch from Replay Buffer
For every sample (s, a, r, s')
$y = r + \gamma^* max_{a'}\, Q(s', a'; \theta')$
Compute gradient descent on $\left(y - Q(s, a;\, \theta)\right)^2$
Every step:
    Update target network: $\theta' \leftarrow \theta$
    Update current state: $s_t \leftarrow s_{\{t+1\}}$
End for
Initialize population P with present task-VM allocations
For iteration = 1 to $max_{iterations}$ do:
    For every individual i in P:
        Calculate fitness $f(i)$ using:
        Makespan, energy consumption, resource utilization and scalability
Update escape energy E using inverted S-shaped strategy
If $rand < 0.33$:
    Apply growth phase of young birds (global exploration)
Else if $rand < 0.66$
    Apply breeding/feeding phase (local exploitation)
Else:
    Apply avoiding enemies phase (random search)
    Apply Hierarchical Reverse Learning to top-ranked individuals
    Update population P with better solutions
End for
    Select best task-VM mapping from SFO and update DQN replay buffer
    $episode \leftarrow episode + 1$
End while
Return – Optimal task-to-VM allocation policy reducing energy, makespan, increasing throughput and resource utilization

Complexity Analysis

The computational overhead of the proposed model arises from DQN training and ISHR-SFO optimization. The time complexity of DQN training per episode is $O(E \times B \times |A|)$, where E represents the number of episodes, B represents batch size and $|A|$ represents action space size. The ISHR-SFO algorithm introduced additional complexity of $O(I \times P \times D)$, where I is the number of iterations, P represents population size and D represents dimensionality of task-VM mapping space. Hence, the overall complexity of a hybrid model is $O(E \times B \times |A| + I \times P \times D)$. Though this increased computational cost while comparing individual scheduling methods, the improved convergence and solution show additional overhead for large-scale cloud environments.

Experimental Analysis

The performance of the proposed DQN and ISHR-SFO model is evaluated using CloudSim 5.0 toolkit for simulating scalable and heterogeneous cloud computing environment. The system configuration for the model is described below. The below Table 1 represents the simulation parameters of the proposed model. The Table 2 presents hyperparameters of DQN and ISHR with SFO model.

Table 1: Simulation parameters of proposed model

Parameter	Values
Number of Physical Machine (PMs)	1
Number of VMs	40
Number of tasks	200 to 1000
VM allocation policy	Time-shared
RAM per VM	2048 MB
Bandwidth	1000 Mbps

Table 2: Hyperparameters of DQN and ISHR with SFO model

Parameter	Values
Episodes	200 to 500
Learning rate α	0.001
Discount factor γ	0.95
Exploration rate ε	1.0 to 0.01
Replay buffer size	10,000
Batch size	64
Optimizer	Adam
Population size	30
Maximum iterations	100
Inverted S-escape energy parameters	β=5, E_0=2

The simulation setup and parameter selection are developed to ensure stability and fair comparison. CloudSim 5.0 is selected because of its wide adoption and its ability to model dynamic cloud environments, VM scheduling, energy consumption, and scalability. The number of tasks (200 to 1000) and VMs (40) are chosen to represent varying workload intensities from moderate to heavy load conditions. VM configurations and allocations policy follow standard allocation policy. DQN hyperparameters include learning rate, discount factor, batch size, and replay buffer size are adopted from the established reinforcement learning and validated to ensure stable convergence. SFO population size and iteration count are selected to balance solution quality and computational cost, when inverted S-shaped escape energy and hierarchical reverse learning parameters are selected to maintain efficient exploration and exploitation trade-offs:

- CPU – Intel Core i7, 2.8 GHz
- RAM – 16 GB
- OS – Windows 10 (64 bit)

All baseline models are reimplemented and evaluated under same CloudSim 5.0 simulation settings to ensure fair and unbiased comparison. The baseline models include metaheuristic-based optimization algorithms such as CSO, BMO, LOA and SBO, existing reinforcement learning-based models such as Vanilla DQN, Distributional DQN, and Rainbow DQN, which represents state-of-the-art optimization and learning-based cloud load balancing strategies. The hyperparameters of DQN and ISHR-SFO includes learning rate, exploration rate, discount factor, population size, and iteration count that are empirically tuned by preliminary experiments. These parameters are tunable and adjusted to adapt the model for various workload scales and cloud configurations.

Performance Analysis Based on Number of Tasks

The below Figs. 2 to 6 represents a performance of the proposed DQN-SFO model across increasing task loads from 200 to 1000 tasks. As number of task increases, the energy consumption rises gradually as presented in the Fig. 2, but remains optimized because of an energy-efficient scheduling through RL-based reward function. Makespan increases proportionally as presented in Fig. 3, which remains lesser than compared models because of DQN's predictive task allocation and SFO's exploration and exploitation balance. Execution time presented in Fig. 4, increases slightly but reduces through model's dynamic decision-making and adaptive learning strategies. Throughput enhances consistently as presented in Fig. 5 and determines that the model efficiently scales with workload, thereby increasing task completion. The resource utilization as presented in Fig. 6 increases toward near-saturation, representing the model's capability to maintain optimum load distribution across VMs under high tasks. These results demonstrate the model's superior scalability and energy efficacy in cloud environments.

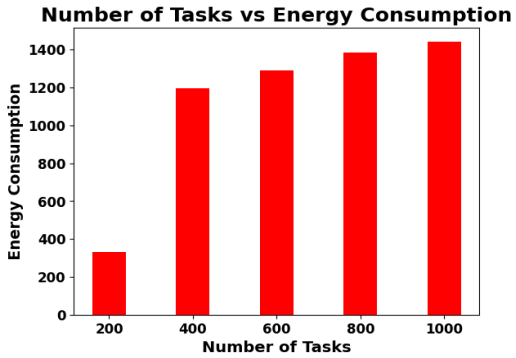

Fig. 2: Number of tasks vs Energy Consumption (J)

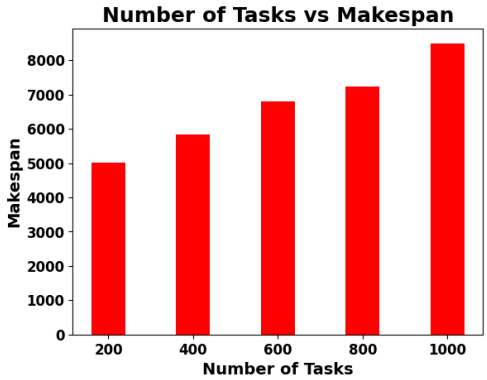

Fig. 3: Number of tasks vs Makespan (s)

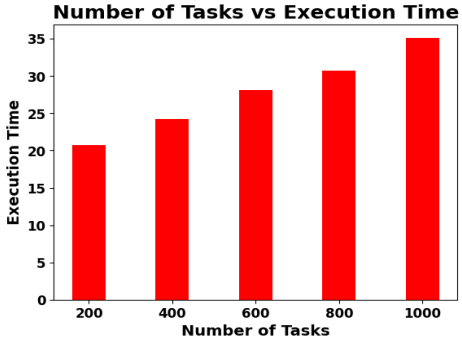

Fig. 4: Number of tasks vs Execution time (s)

DOI: 10.3844/jcssp.2026.1797.1810

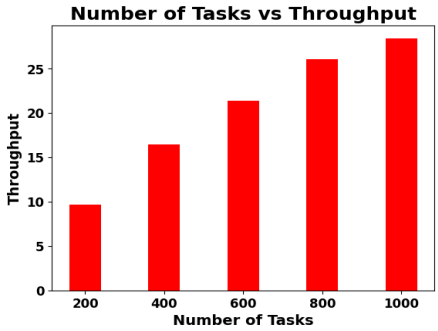

Fig. 5: Number of tasks vs Throughput

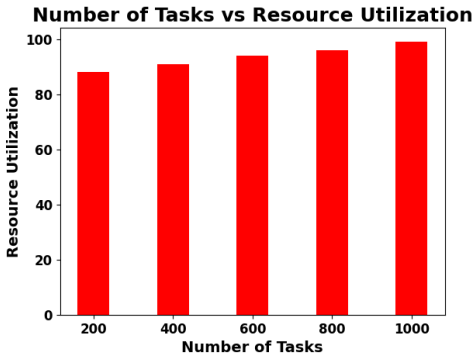

Fig. 6: Number of tasks vs Resource utilization (%)

Performance Analysis Based on Number of VMs

Figures 7 to 11 present the performance of proposed DQN and SFO-ISHR model with varying number of VMs across primary metrics.

As the number of VM count increases, energy consumption, makespan, and execution time consistently minimizes because of optimized task-to-VM allocation through predictive scheduling of DQN and exploration and exploitation search of SFO. Throughput consistently maximizes, thereby representing the model's scalability and ability to process more tasks in parallel. Resource utilization shows an efficient load distribution and less capacity at high VM counts. These results indicate the model's ability to obtain high scalability, minimized operation costs, and energy efficacy scheduling in cloud environments.

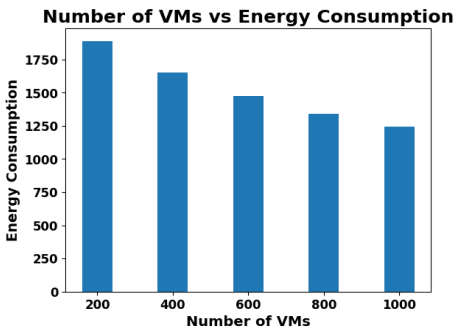

Fig. 7: Number of VMs vs Energy Consumption (J)

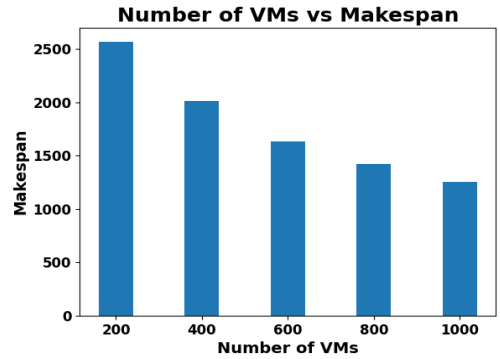

Fig. 8: Number of VMs vs Makespan

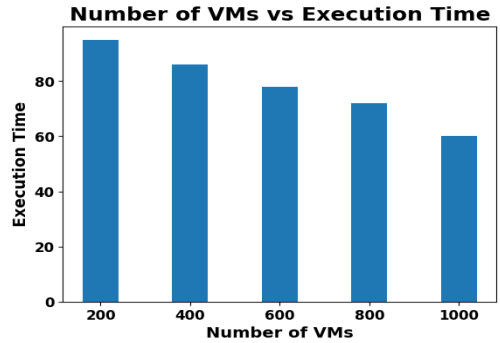

Fig. 9: Number of VMs vs Execution time (s)

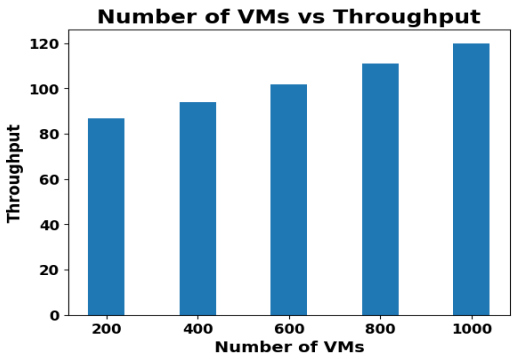

Fig. 10: Number of VMs vs Throughput

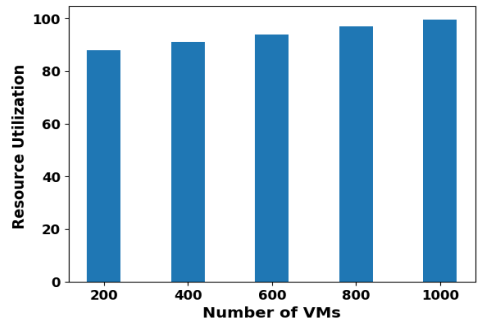

Fig. 11: Number of tasks vs Resource utilization (%)

Table 3 presents the performance of proposed DQN-SFO, determines superior performance across primary measures such as energy consumption, makespan, execution time, throughput, and resource utilization. DQN uses Q-learning to dynamically predict optimum task to VM mappings by increasing discounted cumulative rewards and efficiently reducing the scheduling latency as well as execution cost.

SFO employs the biologically inspired multiple-phase behaviors for adaptively refining search space solutions. The incorporation of inverted S-shaped escape energy preserves exploration in early iterations while improving the exploitation and hierarchical reverse

learning while also avoiding local optimum by redistributing population diversity based on fitness functions. The proposed model ensures robust convergence, balanced resource allocation, resulting in less energy overhead and enhanced scalability across traditional metaheuristic algorithms such as Cuckoo Search Optimization (CSO), Bird Mating Optimization (BMO), Lyrebird Optimization Algorithm (LOA) and Secretary Bird Optimization (SBO).

The Table 4 presents the performance of proposed model with various DQN variants such as Rainbow DQN, Distributed DQN and Vanilla DQN. The proposed method obtains less energy consumption, less makespan and minimal execution time, thereby showing high throughput and optimum resource utilization. The proposed model improves decision-making by including SFO that dynamically explores task scheduling space through multiple phase process. The inverted S-shaped escape energy and hierarchical reverse learning mechanisms prevent premature convergence and local optima, thereby enabling better task allocation and resource balancing. The proposed model effectively outperformed the traditional DQN models in energy efficacy and scalability in cloud task scheduling process. The Table 5 represents the statistical performance of the proposed DQN with ISHR-SFO method against existing approaches with standard deviation over multiple independent runs and p-value. The proposed model consistently obtains less energy consumption, makespan and execution time, while obtaining high throughput and resource utilization, thereby representing superior efficacy and scalability. The p-value determines that the performance improvements of the proposed model are statistically significant, where higher p-values for existing models represent comparatively weak improvements. These results show that improvements obtained by integrating inverted S-shaped escape energy and hierarchical reverse learning with DQN not because of random variations and reliable improvement over existing cloud load balancing algorithms.

Table 3: Performance of proposed model with different optimization algorithms

Methods	Energy consumption(J)	Makespan	Execution time (s)	Throughput	Resource Utilization
CSO	1731	8865	42.7	105.6	94.62
BMO	1688	9865	41.3	108	95.16
LOA	1616	9599	39.9	110.4	96.14
SBO	1573	9344	38.5	114	97.22
Proposed DQN with ISHR-SFO	1443	8495	35.15	120	99.12

Table 4: Performance of proposed model with different DQN variants

Methods	Energy Consumption (J)	Makespan	Execution time (s)	Throughput	Resource Utilization (%)
Rainbow DQN	1587	9344	38.5	114	97.15
Distributional DQN	1659	9769	40.25	108	95.18
Vanilla DQN	1803	10619	43.75	102	93.22
Proposed DQN with ISHR-SFO	1443	8495	35.15	120	99.12

Table 5: Statistical analysis of proposed model against existing models with standard deviation and p-value

Method	Energy Consumption (J)	Makespan	Execution Time (s)	Throughput (kbps)	Resource Utilization (%)	p-value
CSO	1731 ± 41.2	8865 ± 152	42.7 ± 1.1	105.6 ± 3.2	94.62 ± 1.8	0.04
BMO	1688 ± 38.5	9865 ± 176	41.3 ± 1.0	108.0 ± 2.9	95.16 ± 1.6	0.03
LOA	1616 ± 35.8	9599 ± 164	39.9 ± 0.9	110.4 ± 2.7	96.14 ± 1.5	0.05
SBO	1573 ± 33.4	9344 ± 148	38.5 ± 0.8	114.0 ± 2.4	97.22 ± 1.3	0.03
Rainbow DQN	1587 ± 36.2	9344 ± 151	38.5 ± 0.9	114.0 ± 2.6	97.15 ± 1.4	0.04
Distributional DQN	1659 ± 39.6	9769 ± 169	40.25 ± 1.0	108.0 ± 2.8	95.18 ± 1.6	0.03
Vanilla DQN	1803 ± 44.1	10619 ± 182	43.75 ± 1.2	102.0 ± 3.4	93.22 ± 1.9	0.03
Proposed DQN with ISHR-SFO	1443 ± 28.7	8495 ± 121	35.15 ± 0.7	120.0 ± 2.1	99.12 ± 0.9	0.02

To validate statistical significance of performance improvements, paired t-test is tested between the proposed DQN with ISHR-SFO model and every baseline model. The null Hypothesis (H0) assumes that there is no statistically significant different between proposed and comparative models, when alternative Hypothesis (H1) assumes that the proposed model obtains statistically significant performance improvements. All experiments are executed over N independent runs and results are reported as mean $\pm$ standard deviation. The p-value less than 0.05 represents that the observed performance improvements are statistically significant and not due to random variation.

Comparative Analysis

The Table 6 presents comparative analysis of the proposed approach to the existing model GNN-DMQOS (Kalaskar and Thangam, 2024) over response time, throughput, and resource utilization as the number of tasks increases from 100 to 500. The response time is significantly less in a proposed model, integrated to DQN's decision-making and SFO's adaptive task allocation that minimizes scheduling delay and network overhead. The throughput is higher in the proposed model because of better parallelism and VM utilization allows more tasks to get processed. SFO's dynamic exploration and exploitation mechanism for optimum workload balancing increases data handling rates. Resource utilization in the proposed model facilitates through hierarchical reverse learning and escape energy strategies in ISHR-SFO, which ensures that all available resources are effectively used with less states. The proposed model scales effectively under increasing load and ensures more reliable and energy efficacy, especially in cloud-based dynamic environments simulated in CloudSim 3.0 toolkit.

Table 7 presents comparative analysis of the proposed approach with existing models like Meta-RHDC (Krishna and Vali, 2025) under the simulation of CloudSim 5.0. The proposed model maintains less makespan consistently through effectively mapping tasks by deep Q-learning policies and refinements from SFO. This resulted in fast task completion and enhanced scheduling performance. Energy consumption is minimized because of optimized task allocation, which avoids redundant computation and VM overload. Moreover, resource utilization is optimal, which enables hierarchical reverse learning and stage-wise balancing in ISHR-SFO and ensures balanced VM utilization under high load. As presented in Table 8, increasing VM counts of the proposed model scales better with less makespan and energy utilization than Meta-RHDC (Krishna and Vali, 2025). Significantly, when Meta-RHDC represents the declined resource usage with high VM counts, the proposed model maintains superior usage, thereby indicating its ability to distributed tasks uniformly and prevent resource underutilization.

Table 6: Comparative analysis of proposed model with GNN-DMQOS (Kalaskar and Thangam, 2024)

Metrics	Methods	Number of tasks				
		100	200	300	400	500
Response time (ms)	GNN-DMQOS (Kalaskar and Thangam, 2024)	1.345	1.987	2.567	2.665	2.876
	Proposed DQN with ISHR-SFO	1.1211	1.682	2.189	2.275	2.418
Throughput (kbps)	GNN-DMQOS (Kalaskar and Thangam, 2024)	1513.17	1534.19	1576.19	1588.13	1598.14
	Proposed DQN with ISHR-SFO	1580.34	1602.76	1644.29	1665.32	1684.78
Resource Utilization (%)	GNN-DMQOS (Kalaskar and Thangam, 2024)	90.56	91.22	92.78	93.44	94.78
	Proposed DQN with ISHR-SFO	93.45	95.12	96.83	98.26	99.41

Table 7: Comparative analysis of proposed model with Meta-RHDC (Krishna and Vali, 2025) in terms of number of tasks

Metrics	Methods	Number of tasks				
		1000	2000	3000	4000	5000
Makespan (ms)	Meta-RHDC (Krishna and Vali, 2025)	125.55	260.11	390.16	520.21	650.27
	Proposed DQN with ISHR-SFO	115.32	240.27	367.84	495.73	624.91
Energy consumption (kw)	Meta-RHDC (Krishna and Vali, 2025)	3710.74	3502.95	3641.55	3641.66	3568.54
	Proposed DQN with ISHR-SFO	3521.48	3305.63	3402.17	3414.09	3328.36
Resource Utilization (%)	Meta-RHDC (Krishna and Vali, 2025)	86.91	90.20	93.54	96.54	99.01
	Proposed DQN with ISHR-SFO	91.86	94.38	96.73	98.92	99.76

Table 8: Comparative analysis of proposed model with Meta-RHDC (Krishna and Vali, 2025) in terms of number of VMs

Metrics	Methods	Number of VMs				
		40	80	120	160	200
Makespan (ms)	Meta-RHDC (Krishna and Vali, 2025)	105.81	210.29	315.74	420.72	525.06
	Proposed DQN with ISHR-SFO	98.62	196.41	298.36	397.12	496.54
Energy consumption (kw)	Meta-RHDC (Krishna and Vali 2025)	4.89	6.19	7.37	9.84	11.31
	Proposed DQN with ISHR-SFO	4.61	5.84	6.89	8.92	10.38
Resource Utilization (%)	Meta-RHDC (Krishna and Vali 2025)	90.12	88.98	87.45	85.12	83.98
	Proposed DQN with ISHR-SFO	92.46	90.38	89.72	98.92	99.76

This is because of an integrated decision intelligence of DQN and global-local optimization by ISHR-SFO. The proposed model obtains superior scalability, energy efficacy, and scheduling performing that validates for dynamic cloud environments with huge workloads and different resource configurations.

Research Implication

The integration of DQN with ISHR-SFO presents significant advancements in a domain of intelligent cloud resource management. The primary implications of this research are given below:

- Determines the RL-DQN and ISHR-SFO, which is integrated to solve dynamic scheduling issues in cloud environments more efficiently than conventional heuristic or learning models
- It highlights the impact of intelligent task scheduling on minimizing the overall energy consumption in data centers, which contributes to cloud computing
- Introduced novel advancements to ISHR-SFO (inverted S-escape energy and hierarchical reverse learning), which is generalized and employed to different optimization issues in scheduling and networking

Discussion

The proposed model integrated DQN with ISHR-SFO, which addresses essential drawbacks in the existing cloud load balancing algorithms by introducing dynamic, adaptive, and energy-efficacy task scheduling mechanism. Traditional models such as GNN-DMQOS (Kalaskar and Thangam, 2024) and Meta-RHDC (Krishna and Vali, 2025), lacks real-time adaptability and suffered from convergence problems also ineffective resource allocation under fluctuating workloads. The component of DQN of the proposed model ensures intelligence decision-making by continuous learning of an optimum task to VM mapping based on system states and rewards. The ISHR-SFO provides robust metaheuristic algorithm that improves global exploration and local exploitation by biologically inspired multiple phase behavior and advanced improvements such as inverted S-shaped escape energy and hierarchical reverse learning. The simulation results on CloudSim 3.0 and CloudSim 5.0 over different task and VM configuration represents that the proposed model effectively minimizes response time, makespan, and energy consumption while increasing throughput and resource utilization. Additionally, the proposed model introduced scheduling algorithm that uses intelligent policy learning with adaptive optimization to load balancing in cloud environments. Unlike traditional RL or metaheuristic algorithms, the proposed model establishes interaction where DQN generated adaptive scheduling policies and ISHR-SFO refines by population-based optimization. This cooperative interaction improves convergence speed and solution quality under dynamic workloads.

Superior performance of the proposed DQN with ISHR-SFO model from efficient energy where inverted S-shaped escape energy enhances exploration and hierarchical reverse learning prevents premature convergence under dynamic workloads. Despite these strengths, the manuscript has certain limitations that involves reliance on CloudSim-based simulations rather than real-world cloud deployments, fixed parameter setting across experiments and evaluation on limited scale of tasks and virtual machines. Moreover, training overhead of DQN increase for huge-scale cloud environments. The performance improvement of the proposed model is attributed to cooperative interaction between DQN and ISHR-SFO components. The inverted S-shaped escape energy maintains prolonged exploration in early iterations, prevents premature convergence and ensures broader search of task-VM mappings. As iteration progresses, escape energy shifts towards exploitation and refines promising scheduling solutions. Moreover, hierarchical reverse learning preserves population diversity by re-evaluating lower-ranked individuals and minimization stagnation in local optima. This process balance between exploration and exploitation as well as directly contributes to minimized resource usage and lower energy consumption under dynamic workloads.

Threats to Validity the experimental evaluation is conducted by CloudSim-based simulation that does not completely represents real-world cloud heterogeneity, network latency variations, and hardware-level uncertainties. The workload patterns are generated and do not extract highly dynamic traffic conditions. Moreover, hyperparameter settings and reward design choices influence performance results, which potentially limits generalization ability across various cloud infrastructures.

Limitation

Despite their high performance, the proposed model has certain drawbacks. The evaluation is conducted primarily using CloudSim simulations than real-world cloud deployments that do not completely capture practical system uncertainties. The model utilizes fixed hyperparameter settings across experiments and computational overhead of DQN training that increase for large-scale cloud environments.

Conclusion

In this manuscript, a DQN-SFO-based cloud load balancing algorithm is proposed by integrating the intelligent decision-making capability of DQN and ISHR-SFO. Through dynamic task-to-VM allocation, the proposed model obtains minimized makespan, execution time and energy consumption, thereby increasing throughput and resource utilization under varying

workloads and VM configurations. The incorporation of inverted shaped escape energy and hierarchical reverse learning enhance the balance between exploration and exploitation, which prevents premature convergence and ensures stable convergence towards optimum solutions. Experimental results demonstrate that the proposed model consistently outperformed existing methods in terms of scalability, energy efficacy, and load distribution. The proposed model obtained less execution time of 1443s, makespan of 8495, energy consumption of 35.25 J, high throughput of 120kbps and 99.12% of resource utilization. The integration of DQN and ISHR with SFO model provides a robust solution for intelligent cloud resource management, which makes it well-suited for large-scale, and dynamic cloud computing environments.

Future Work

In future work, the proposed model will extend to real-world cloud deployments beyond CloudSim simulation with adaptive hyperparameter tuning for DQN and dynamic population control in ISHR-SFO to enhance scalability. Moreover, multi-objective SLA-aware scheduling strategies will incorporate to improve sustainability and practical applicability in large-scale cloud environments.

Acknowledgment

The authors would like to express their sincere gratitude to Presidency University, Yelahanka, Bengaluru, Karnataka, India, for providing the necessary infrastructure and support to carry out this research. The authors also extend their appreciation to colleagues and peers for their valuable suggestions and encouragement during the course of this work.

Funding Information

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

Author's Contributions

B. R. Yogeetha: Conceptualization of the research work, methodology design, supervision, and manuscript preparation.

S. P. Anandaraj: Data collection, analysis, experimentation, and preparation of the initial draft. Both authors reviewed and approved the final version of the manuscript.

Ethics

This research does not involve human participants or animals. The authors declare that there are no ethical conflicts associated with this study.

References

Abdulghani, A. M. (2024). Hybrid Task Scheduling Algorithm for Makespan Optimisation in Cloud Computing: A Performance Evaluation. *Journal on Artificial Intelligence*, *6*(1), 241–259. https://doi.org/10.32604/jai.2024.056259

Akerele, J. I., Uzoka, A., Ojukwu, P. U., & Olamijuwon, O. J. (2024). Optimizing traffic management for public services during high-demand periods using cloud load balancers. *Computer Science & IT Research Journal*, *5*(11), 2594–2608. https://doi.org/10.51594/csitrj.v5i11.1710

Alruwais, N., Alabdulkreem, E., Kouki, F., Aljehane, N. O., Allafi, R., Marzouk, R., Assiri, M., & Alneil, A. A. (2024). Farmland fertility algorithm based resource scheduling for makespan optimization in cloud computing environment. *Ain Shams Engineering Journal*, *15*(6), 102738. https://doi.org/10.1016/j.asej.2024.102738

Brahmam, M. G., & Reddy, V. A. (2024). VMMISD: An Efficient Load Balancing Model for Virtual Machine Migrations via Fused Metaheuristics With Iterative Security Measures and Deep Learning Optimizations. *IEEE Access*, *12*, 39351–39374. https://doi.org/10.1109/access.2024.3373465

Choppara, P., & Lokesh, B. (2025). Efficient Task Scheduling and Load Balancing in Fog Computing for Crucial Healthcare Through Deep Reinforcement Learning. *IEEE Access*, *13*, 26542–26563. https://doi.org/10.1109/access.2025.3539336

Chraibi, A., Alla, S. B., & Ezzati, A. (2021). Makespan Optimisation in Cloudlet Scheduling with Improved DQN Algorithm in Cloud Computing. *Scientific Programming*, *2021*, 1–11. https://doi.org/10.1155/2021/7216795

Ghorbani, M., Khaledian, N., & Moazzami, S. (2025). ALBLA: an adaptive load balancing approach in edge-cloud networks utilizing learning automata. *Computing*, *107*(1), 34. https://doi.org/10.1007/s00607-024-01380-0

Kalaskar, C., & Thangam, S. (2024). A graph neural network-based approach with dynamic multiqueue optimization scheduling (DMQOS) for efficient fault tolerance and load balancing in cloud computing. *International Journal of Intelligent Systems*, *2024*(1), 6378720. https://doi.org/10.1155/int/6378720

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

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

Kumar, N. V., Mohanty, S., & Pattnaik, P. K. (2025). Hybrid algorithm for optimized clustering and load balancing using deep Q reccurent neural networks in cloud computing. *Bulletin of Electrical Engineering and Informatics*, *14*(2), 1570–1578. https://doi.org/10.11591/eei.v14i2.9123

Li, P., Wang, H., Tian, G., & Fan, Z. (2024). Towards Sustainable Cloud Computing: Load Balancing with Nature-Inspired Meta-Heuristic Algorithms. *Electronics*, *13*(13), 2578. https://doi.org/10.3390/electronics13132578

Menaka, M., & Kumar, K. S. S. (2024). Supportive particle swarm optimization with time-conscious scheduling (SPSO-TCS) algorithm in cloud computing for optimized load balancing. *International Journal of Cognitive Computing in Engineering*, *5*, 192–198. https://doi.org/10.1016/j.ijcce.2024.05.002

Nagarajan, S., Mahalingam, S., Velayutham, K., & Daniel, E. (2025). Deep Kronecker ResNeXt forward fractional network-driven clustering for enhanced load balancing in cloud computing. *The Journal of Supercomputing*, *81*(8), 973. https://doi.org/10.1007/s11227-025-07359-8

Rajawat, A. S., Goyal, S. B., Kumar, M., & Malik, V. (2024). Adaptive resource allocation and optimization in cloud environments: Leveraging machine learning for efficient computing. *Applied Data Science and Smart Systems*, 499–508. https://doi.org/10.1201/9781003471059-64

Sahu, S., & Verma, P. (2026). Energy-Aware Task Scheduling and Load Balancing in Cloud Computing Using AI. *Cybernetics and Systems*, *57*(1), 84–121. https://doi.org/10.1080/01969722.2025.2588787

Simaiya, S., Lilhore, U. K., Sharma, Y. K., Rao, K. B. V. B., Maheswara Rao, V. V. R., Baliyan, A., Bijalwan, A., & Alroobaea, R. (2024). A hybrid cloud load balancing and host utilization prediction method using deep learning and optimization techniques. *Scientific Reports*, *14*(1), 1337. https://doi.org/10.1038/s41598-024-51466-0

Singal, M., & Verma, G. (2024). Hybrid Load Balancing Technique for Cloud Environment Using Swarm Optimization. *The Review of Socionetwork Strategies*, *18*(2), 167–183. https://doi.org/10.1007/s12626-024-00160-8

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

Ullah, A., Alomari, Z., Alkhushayni, S., Al-Zaleq, D., Bany Taha, M., & Remmach, H. (2024). Improvement in task allocation for VM and reduction of Makespan in IaaS model for cloud computing. *Cluster Computing*, *27*(8), 11407–11426. https://doi.org/10.1007/s10586-024-04539-8

Verma, G. (2024). Load Balancing in Cloud Environment Using Opposition Based Spider Monkey Optimization. *Wireless Personal Communications*, *137*(2), 977–996. https://doi.org/10.1007/s11277-024-11445-0

Zhang, X. (2024). Optimizing scientific workflow scheduling in cloud computing: a multi-level approach using whale optimization algorithm. *Journal of Engineering and Applied Science*, *71*(1), 175. https://doi.org/10.1186/s44147-024-00512-9

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