FMDB *Transactions on Sustainable Computer Letters*

An Autonomous Hybrid Swarm-Evolutionary Model for Dynamic Load Balancing in Virtualised Cloud Environments

S. Balaji[1], S. Silvia Priscila[2,*], B. M. Praveen[3]

[1]Department of Information Technology, Institute of Computer Science and Information Science, Srinivas University, Dakshina Kannada, Karnataka, India.
[1]Department of Information Technology, Al Zahara College for Women, Madinat Al-I'rfan, Muscat, Oman.
[2]Department of Computer Science, Bharath Institute of Higher Education and Research, Chennai, Tamil Nadu, India.
[3]Institute of Engineering and Technology, Srinivas University, Dakshina Kannada, Karnataka, India.
balaji@zcw.edu.om[1], silviaprisila.cbcs.cs@bharathuniv.ac.in[2], bm.praveen@yahoo.co.in[3]

Abstract: The booming growth of cloud computing has made efficient and adaptive load-balancing mechanisms required to handle the highly dynamic and heterogeneous workload. Traditional static and monolithic optimisation methods struggle to maintain service quality under fluctuating demand, leading to performance degradation and increased energy consumption. This study addresses dynamic load imbalance in virtualised cloud environments by proposing an autonomous hybrid swarm evolutionary load-balancing model that combines global exploration, local refinement, and adaptive decision control. The proposed framework combines swarm intelligence for efficient search, evolutionary optimisation for solution improvement, and a dynamically weighted fitness mechanism for autonomous adaptation to changing system conditions. Extensive experimental evaluation shows that the proposed model can achieve an average response time of 152ms, reduce the load imbalance to 4.3%, reduce SLA violations to 1.5%, and reduce energy consumption to 9.8kWh, which is better than various state-of-the-art methods. Statistical validation using k-fold cross-validation and paired t-tests is used to assess the robustness and significance of the results. The results show that the proposed hybrid strategy provides a scalable, reliable, and energy-efficient solution for real-time load balancing in modern cloud infrastructures.

Keywords: Cloud Computing; Resource Management; Evolutionary Optimisation; Swarm Intelligence; Dynamic Load Balancing; Virtualised Environments; Optimisation Methods.

Received on: 19/03/2025, **Revised on:** 10/06/2025, **Accepted on:** 17/07/2025, **Published on:** 03/01/2026

Journal Homepage: https://www.fmdbpub.com/user/journals/details/FTSCL

DOI: https://doi.org/10.69888/FTSCL.2026.000599

Cite as: S. Balaji, S. S. Priscila, and B. M. Praveen, "An Autonomous Hybrid Swarm-Evolutionary Model for Dynamic Load Balancing in Virtualised Cloud Environments," *FMDB Transactions on Sustainable Computer Letters*, vol. 4, no. 1, pp. 38–54, 2026.

1. Introduction

Cloud computing has become a dominant paradigm for providing scalable, on-demand computing resources across a wide variety of applications, including big data analytics, the Internet of Things (IoT), artificial intelligence, and enterprise systems

*Corresponding author.

[1]. By using virtualisation technologies, cloud service providers can dynamically allocate computational resources, such as CPU, memory, and bandwidth, to multiple tenants, maximising infrastructure utilisation [2]. However, the growing diversity and dynamism of cloud workloads have made resource management an important challenge. Load balancing plays a pivotal role in ensuring optimal performance, service reliability, and energy efficiency in virtualised cloud environments [3]. At a general level, load balancing aims to evenly distribute workloads across available resources to avoid performance bottlenecks and underutilization [4]. Poor load distribution can lead to overloading of virtual machines (VMs), longer response times, more frequent SLA violations, and unnecessary energy consumption [5]. As cloud systems become more extensive and complex, static and heuristic-based load-balancing strategies become inadequate due to their lack of adaptability to real-time workload fluctuations and resource heterogeneity. Consequently, more research has been conducted on intelligent and adaptive mechanisms for tackling dynamic load-balancing problems [6]. Existing research has explored various methods, such as rule-based scheduling, fuzzy logic controllers, swarm intelligence algorithms (e.g., Particle Swarm Optimisation (PSO) and Ant Colony Optimisation (ACO), evolutionary algorithms (e.g., Genetic Algorithms (GA), and, lately, deep reinforcement learning-based solutions [7].

While these methods have shown improvements over traditional methods, they each have inherent limitations. Swarm-based approaches may exhibit premature convergence and limited local refinement; evolutionary algorithms can incur high computational overhead; and learning-based approaches require substantial training data and are not transparent in their decision-making [8]. Moreover, most existing approaches rely on fixed fitness formulations and do not allow autonomous adaptation to changing system conditions [9]. Recent work suggests that a mixture of paradigms, i.e. hybrid optimisation frameworks, can overcome the limitations of single techniques [10]. However, when current hybrid solutions are available, they are often semi-static, require manual parameter tuning, or cannot dynamically adjust their optimisation goals in response to workload variations. Additionally, most studies assess performance in constrained/synthetic scenarios, limiting their generalizability to real-world cloud-based environments [11]. To fill these gaps, this study proposes an autonomous hybrid swarm-evolutionary model for dynamic load balancing in virtualised cloud infrastructures. The proposed framework combines swarm intelligence to efficiently search the space globally, evolutionary optimisation to refine a local solution, and an adaptive fitness mechanism that autonomously reweights the performance goals using real-time system feedback. This design enables the model to respond effectively to workload non-stationarity, resource heterogeneity, and strict SLA constraints without requiring manual intervention [25].

1.1. Contributions of the Study

The major contributions of this study are summarised as follows:

- **Novel Hybrid Architecture:** A fully autonomous, swarm-evolution-based load-balancing scheme is proposed that integrates global exploration, evolutionary refinement, and adaptive fitness control.
- **Adaptive and Robust Optimisation:** A dynamically weighted fitness mechanism is proposed to prioritise response time, load balancing, SLA compliance, and energy efficiency based on real-time cloud conditions.
- **Comprehensive Evaluation:** Extensive experimental analysis, including comparative, ablation, computational efficiency, and statistical experiments, demonstrates that it outperforms the state of the art.

2. Related Works

Optimised load balancing in cloud environments has been widely discussed using heuristic, metaheuristic, and hybrid methods to improve scalability, energy efficiency, and QoS. In the area of metaheuristic optimisation, Abedi et al. [12] present an improved version of the Firefly Optimisation Algorithm to address dynamic resource allocation in cloud environments. The improvements in the makespan and load distribution are achieved by effectively balancing exploration and exploitation using the model. Nonetheless, evaluations are limited by the simulated environment, and the methodology is sensitive to parameter tuning, without any comparison to learning-based strategies, which restricts its applicability to heterogeneous, very dynamic cloud workloads. Similarly, Bojappa and Lee [13] is a review of the application of PSO (Particle Swarm Optimisation) to autonomous dynamical systems, focusing on convergence efficiency and adaptability. While conceptually strong, it has not addressed some cloud-specific considerations, such as VM heterogeneity, SLA compliance, and energy awareness. It thus restricts its direct applicability in the context of cloud load balancing. Wang et al. [14] propose a task scheduling algorithm for cloud computing based on improved PSO with an adaptive velocity update, which is shown to reduce execution time and improve resource utilisation. However, performance evaluations are restricted to constrained situations with limited workloads, and energy consumption goes unnoticed. The lack of multi-objective optimisation and real-time workload adaptation limits its applicability in large-scale production clouds.

Complementing these views, Prity et al. [15] provide a comprehensive review of nature-inspired optimisation algorithms for cloud task scheduling. While effective in its summary of algorithmic strengths and weaknesses, there is a lack of critical

comparative metrics and experimental synthesis, leading to insights that are more taxonomic (rather than prescriptive). Hybrid and multi-objective optimisation approaches have also attracted much attention. Khaleel et al. [16] merge-and-split theory-based hybrid many-objective optimisation algorithm for achieving improved Pareto optimality among multiple QoS parameters. Despite being theoretically sound, there is significant computational complexity, and real-time cloud workload scalability is not convincingly demonstrated. Reinforcement learning-based approaches have been discussed by Khan [17], who outlines a clustering problem that is combined with multi-objective task scheduling. The approach is effective at capturing the dynamism of clouds and improves throughput and response time; however, challenges such as RL training overhead, state-space explosion, and reward dependency may limit generalisation across heterogeneous infrastructures. Energy-efficient load balancing has been studied by Kotteswari et al. [18], who propose an MDP-based model (EELB) that combines energy-awareness with QoS optimisation. While theoretically sound, the assumption of accurate state transition probabilities is unrealistic in volatile cloud environments, and computational overhead may prevent scalability when large-scale deployments are done in real time. Comparative analysis of metaheuristic load-balancing algorithms in a unified simulation environment [19].

Third, a valuable quantitative analysis of convergence and performance trade-offs from this type of study can be derived. However, the study is limited to static workloads, does not account for learning-based or hybrid workload models, and does not consider constraints in production environments, such as VM migration costs and SLA violations. There are other approaches based on swarm intelligence, such as Grey Wolf Optimisation and PSO-CALBA, which show reliability and content-aware load-balancing improvements, respectively [20]; [21]. Nevertheless, reliability-oriented models entail additional overhead, whereas content-aware algorithms rely on prior knowledge of task attributes, offering limited adaptability to dynamic workloads [22]. Similarly, definitions of stochastic fractal search and hybrid deep reinforcement learning and PSO frameworks demonstrate encouraging improvements in makespan, response time and throughput. Yet, these approaches have drawbacks, including slow convergence, high computational complexity, and high training overhead, which raise concerns about scalability and the feasibility of real-time deployment. A systematic literature review of these techniques is presented by Shah [23], who classified them into heuristic, metaheuristic, and hybrid methods.

Table 1: Comparative study of existing works

Model / Approach	Focus Area	Key Findings	Performance	Challenges
SLR-based Analysis [23]	Load balancing techniques review	Identified trends and gaps	Qualitative insight	No experimental validation
Improved Firefly Optimisation [12]	Resource allocation	Reduced makespan	Moderate improvement	Parameter sensitivity
PSO Review [13]	Autonomous optimization	Strong adaptability	Conceptual strength	Not cloud-specific
Improved PSO [14]	Task scheduling	Faster convergence	Low execution time	Ignores energy/QoS trade-offs
Nature-inspired Review [15]	Scheduling taxonomy	Algorithm classification	Survey-level	No unified benchmarks
Many-objective Hybrid [16]	Multi-QoS scheduling	Better Pareto solutions	High optimization accuracy	High complexity
RL-based Clustering [17]	Dynamic load balancing	Improved throughput	Strong adaptability	Training overhead
MDP-based EELB [18]	Energy-efficient LB	Reduced energy usage	Energy optimal	Scalability issues
Metaheuristic Comparison [19]	Algorithm benchmarking	Performance trade-offs	Consistent evaluation	Static workloads
GWO-based LB [20]	Reliability-aware LB	Improved fault tolerance	Stable performance	Energy not addressed
PSO-CALBA [9]	Content-aware LB	Lower response time	Task-aware efficiency	Limited scalability
SFS Scheduling [21]	Workflow scheduling	Reduced makespan	Balanced load	Slow convergence
DRL + Parallel PSO [22]	Intelligent LB	High throughput gains	Excellent under dynamics	High computational cost

The research focuses on energy efficiency and QoS dominance; however, it remains descriptive, leaving little room for quantitative benchmarking and for unifying the evaluation framework. Besides, problems of practical deployment and real-time adaptability are not adequately addressed. Overall, current studies show significant advances in heuristic, metaheuristic,

hybrid, and learning-based load-balancing algorithms for cloud environments. However, there remain gaps in energy efficiency integration, multi-objective optimisation, real-time adaptability, and scalable deployment, especially under highly dynamic and heterogeneous cloud workloads. Overcoming such limitations will be key to next-generation intelligent cloud resources management. Table 1 provides an overview of current cloud load-balancing and task-scheduling approaches, comparing them across focus areas, conclusions, performance results, and limitations or problems.

3. Methodology

3.1. Dataset Details

To investigate the efficacy of the proposed autonomous hybrid swarm evolutionary load-balancing framework, experiments are performed on a hybrid cloud workload data set generated from a combination of realistic cloud traces and synthetic stressors. The dataset simulates a multi-tenant Infrastructure-as-a-Service (IaaS) environment with heterogeneous virtual machines (VMs), dynamic workloads, and varying resource requirements. Workload traces are based on publicly available cloud benchmarks (e.g., task arrival logs, CPU utilisation traces, and memory access patterns) and are enriched with controlled synthetic bursts to reflect real-world cloud volatility (flash crowds, VM failures, workload spikes, etc.). Each data instance represents a system snapshot which is obtained at discrete scheduling intervals.

3.1.1. Key Features of the Dataset

The dataset contains both resource-centric and workload-centric attributes important for dynamic load balancing. Key features include VM CPU utilisation (%), memory usage (MB), disk I/O rate (MB/s), network bandwidth consumption (Mbps), task arrival rate, task execution time, VM migration cost, and host power consumption. Additionally, SLA-related indicators such as response time, probability of deadline violation, and throughput are included. Temporal features are maintained to capture workload changes over time, enabling the model to learn short-term fluctuations and long-term utilisation trends. This multidimensional feature design supports fine-grained decision-making for adaptive VM-to-host allocation.

3.1.2. Challenges in the Dataset

The dataset poses several challenges that make effective load balancing difficult. First, high workload variability and non-stationarity lead to frequent changes in optimal resource allocation. Second, the heterogeneity of features is due to differences in resource scales and units, making direct comparison difficult. Third, imbalanced system states occur, in which underloaded conditions dominate normal operation, with occasional, critical overload events. Finally, there is the problem of delayed feedback between scheduling actions and their performance consequences, which makes real-time optimisation difficult. These challenges motivate the need for an autonomous, adaptive, and evolutionary optimisation framework.

3.2. Data Preprocessing

Before training and model optimisation, the raw dataset samples must go through a structured preprocessing pipeline. Missing or inconsistent readings caused by time delays in monitoring are handled using time-aware interpolation. Feature normalisation, in the form of min-max scaling, is used to transform heterogeneous resource metrics into a unified range, enabling stable convergence of swarm optimisation. Temporal smoothing is used in these cases, where sliding windows eliminate noise without losing workload dynamics. Finally, the workload states are encoded into structured state vectors that represent the current system load, historical trends, and predicted short-term demand, which serve as inputs to the proposed model.

3.3. Proposed Model Architecture

The proposed architecture integrates swarm intelligence, evolutionary optimisation, and autonomous decision-making and control into a unified framework. At its core, the model consists of three interacting layers:

- **Perception Layer:** Continuously monitors cloud resource states and workload patterns.
- **Hybrid Optimisation Layer:** Combines swarm-based global exploration with evolutionary local refinement to generate optimal VM-to-host mappings.
- **Autonomous Control Layer:** Executes migration and allocation decisions while adapting optimisation parameters dynamically based on system feedback.

This layered design enables scalable, self-adaptive, and low-overhead load balancing in highly dynamic virtualised environments.

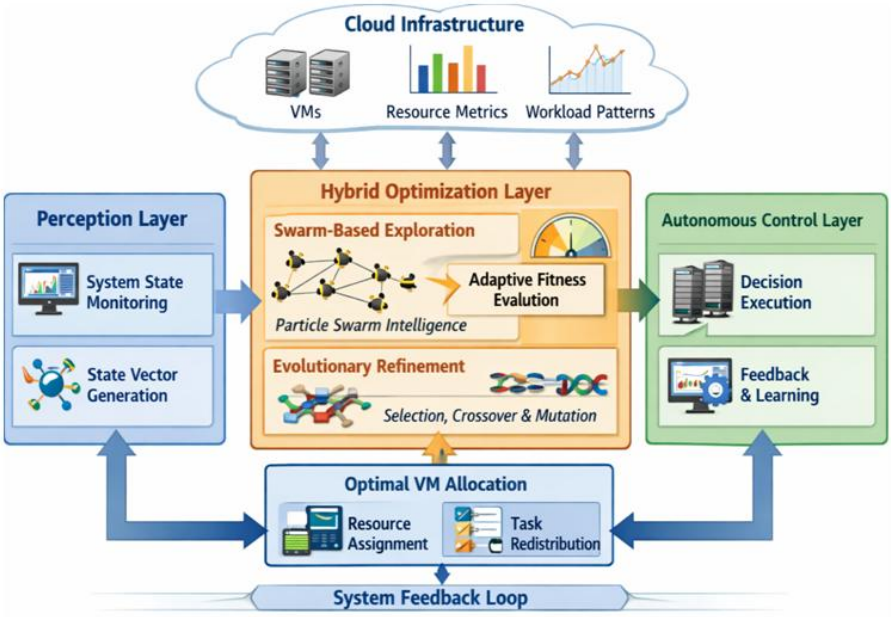

Figure 1: Proposed hybrid evolutionary load balancing model

A Hybrid Swarm-Evolutionary Load Balancing Model for cloud environments is shown in Figure 1. It includes a Perception Layer to monitor the system and generate state vectors, a Hybrid Optimisation Layer that combines Particle Swarm Intelligence and evolutionary optimisation, and an Autonomous Control Layer to make decisions and learn. The model enables optimal VM allocation and provides continuous feedback for adaptive resource allocation and task redistribution.

3.4. Working of the Proposed Model

Figure 2 shows a process paradigm that can help improve resource management in a cloud system. The process starts with system state initialisation, which gathers data about virtual machines (VMs), host capacities, workloads, and SLA indicators to generate a system state vector and an initial group of possible VM-to-host mappings.

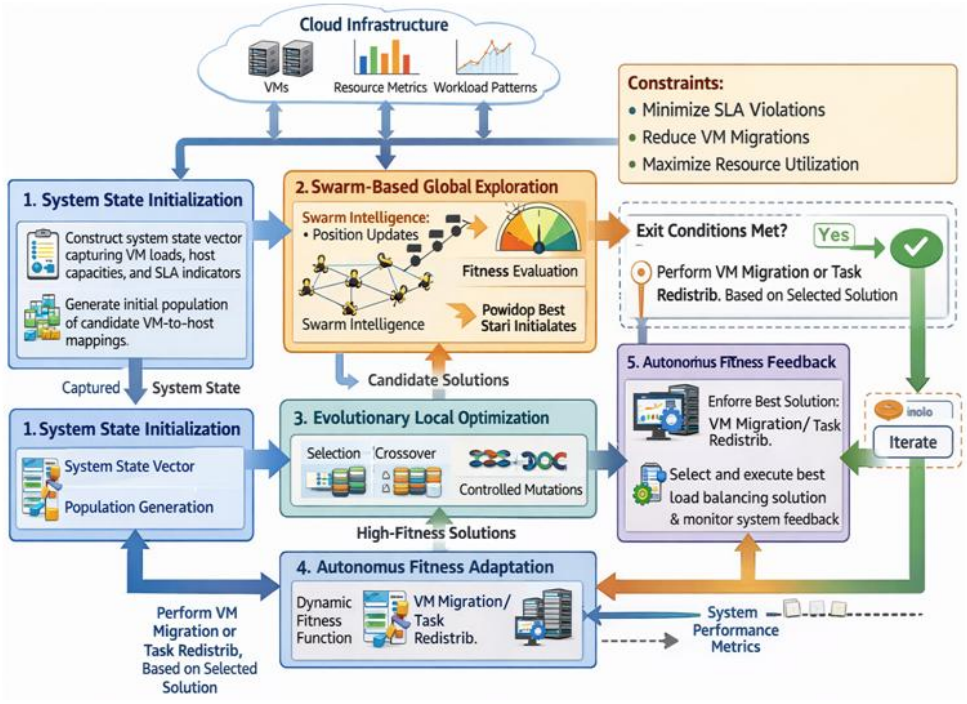

Figure 2: Hybrid swarm–evolutionary framework for cloud load balancing and VM migration

This system information is used in a global exploration phase based on swarms. In this phase, swarm intelligence algorithms adjust agent placements and use a fitness evaluation mechanism to evaluate possible solutions. Powell's local search helps improve potential ideas. Next, these candidate solutions go through the evolutionary local optimisation stage, when evolutionary processes, including selection, crossover, and mutation, are used to make solutions with better fitness. After that, autonomous fitness adaptation uses system performance measurements to dynamically analyse the solutions and move VMs or redistribute tasks when needed. The framework has several important limitations, including minimising SLA violations, reducing VM migrations, and maximising resource utilisation. If the exit requirements are met, the best solution is to implement it by moving VMs or redistributing workloads. Lastly, an independent feedback system monitors system performance and workload balance. This lets the framework evolve and continually improve the optimisation process in the cloud environment.

3.4.1. System State Initialisation

At the start of each scheduling interval, the perception layer gathers and processes real-time system metrics to create a complete state vector. This vector encapsulates key parameters, including VM workloads, host capacities, resource utilisation, and SLA compliance indicators, to provide a global view of the cloud environment. Based on this information about the state, an initial population of candidate load distribution solutions is produced, where each candidate is a possible VM-to-host mapping. Let the cloud environment have N VMs and M hosts. Define the system state vector at scheduling interval t as given in Equation (1):

$$S(t)=[w_1(t),w_2(t);\ldots,w_n(t);\quad c_1(t),c_2(t),\ldots\ldots,c_M(t);u_1(t),u_2(t),\ldots,u_M(t);s_1(t),\ldots,s_N(t)] \quad (1)$$

Where:

- $w_i(t)$=workload of VM_i
- $c_j(t)$=capacity of $host_i$
- $u_j(t)$=current utilization of $host_j$
- $s_i(t)$=SLA compliance indicator of VM

The initial population of candidate VM-to-host mapping is given by Equation (2):

$$x^{(o)} = \{x_1^{(0)}, x_2^{(0)}, \ldots, x_p^{(0)}\} \quad (2)$$

Where each candidate solution $x_p^{(0)} = [h_h, h_2, \ldots\ldots, h_N],\ h_i \epsilon \{1,2,\ldots,M\}$ represents VM i assigned to host h_i and P is the pollution size. The diversity of the initial population ensures exploration and is given by Equation (3):

$$D(x^{(0)})=\frac{1}{p(p-1)}\sum_{p\neq q} Hamming(x_p^{(0)}, x_q^{(0)}) \quad (3)$$

These initial solutions serve as a starting point for the hybrid optimisation process, providing diversity and enabling the model to investigate multiple feasible allocation strategies for heterogeneous, dynamic workloads.

3.4.2. Swarm-Based Global Exploration

Global exploration is carried out using swarm intelligence, in which each agent updates its solution based on its personal experience and the swarm's collective knowledge. Agents combine exploration and exploitation to generate new allocation configurations and to exploit high-performing regions, thereby finding diverse and promising candidate solutions in the search space. This approach is an efficient way to navigate dynamic workload conditions and heterogeneous cloud resources, thereby reducing the risk of premature convergence. Let each agent i have a position vector x_i representing a VM-host mapping. The swam update rule is given by Equation (4):

$$v_i(t+1) = wV_i(t) + c_1 r_1 (p_i - x_i(t)) + c_2 r_2 (g - x_i(t)) \quad (4)$$

Where:

- v_i(t)=velocity of agent i
- P_i=personal best solution of agent i
- G=global best solution across the swarm
- c_1,c_2=cognitive /social coefficients

- r_1,$r_2 \sim U(0,1)$ are random weights

The fitness function for swarm evaluation is given by Equation (5):

$$F(x_i) = a.Loadbalance(x_i) + \beta.SLA(x_i) \quad (5)$$

By exploiting collaborative behaviours inspired by natural swarms, the system can quickly identify multiple feasible VM-to-host assignments and balance performance and reliability objectives early in the load-balancing process.

3.4.3. Evolutionary Local Optimization

To refine candidate solutions generated by swarm exploration, evolutionary optimisation operators are used. Selection selects the best-fitting solutions; crossover combines aspects of parent solutions to produce potentially better child solutions; and mutation introduces diversity, preventing premature convergence and ensuring adaptability. This local refinement helps minimise load variance across VMs, reduce migration overhead, and reduce SLA violations, thereby improving overall system stability. Selection, crossover, and mutation are mathematically expressed in E Equations (6), (7) and (8). The selection (roulette wheel or fitness -based):

$$P(x_i selected\ ) = \frac{F(x_i)}{\sum_{j=1} pF(x_j)} \quad (6)$$

The crossover (single -point or uniform) is given by (6):

$$x_{offspring} = \lambda x_{parent1} + (1-\lambda)x_{parent2}, \lambda \epsilon \{0,1\}^N \quad (7)$$

The mutation (random VM reassignment) is given by Equation (8):

$$x_i' = \begin{cases} rand\ (1, M), & if\ rand(0,1) < \mu \\ x_i & otherwise \end{cases} \quad (8)$$

Where μ is the mutation probability. The refined fitness after evolutionary optimisation is denoted by Equation (9):

$$F_{refined}(x) = \min F\left(X_{offspring}\right)\ \forall\ offspring \quad (9)$$

By integrating global swarm exploration with guided evolutionary optimisation, this framework finds a trade-off between exploration and solution fine-tuning, ensuring that the prepared VM allocation is robust and efficient even under rapidly changing cloud workloads.

3.4.4. Autonomous Fitness Adaptation

The fitness function that underlies optimisation is dynamically adapted to real-time system conditions. Under high load or in the event of an SLA violation, the function focuses on reducing latency and meeting service-level guarantees. On the other hand, in stable workloads, efforts are made to ensure energy consumption and migration expenses for better operational efficiency. This adaptive weighting mechanism enables context-aware optimisation without manual intervention, making the system more effective at responding to changing conditions and heterogeneous workloads. Let the adaptive fitness function be a weighted combination given by Equation (10):

$$F(x,t)=w_1(t).Latency\ (x) + w_2(t).Migration\ Cost\ (X) + w_s(t).Energy(x) \quad (10)$$

Weights are dynamically adjusted according to the equation (11):

$$w_1(t) = f(SLA\ Violations(t), w_2(t) = f(system\ load(t)), w_3(t) = 1 - w_1(t) \quad (11)$$

Normalisation ensures proper weighting and is given by Equation (12):

$$w_1(t) + w_2(t) + w_3(t) = 1 \quad (12)$$

This allows context-aware prioritisation of performance vs efficiency. By always relating fitness evaluation to system goals, the model ensures that performance and energy efficiency are balanced, promoting sustainable and resilient management of cloud resources.

3.4.5. Decision Execution and Feedback

When optimisation is complete, the autonomous control layer implements the most effective solution by performing VM migrations or task reassignment. Execution decisions are made based on the optimised allocation strategy to cause as little disruption as possible and stay within the SLA. Let x^* be the best solution after optimisation, the VM migration or task reassignment is given by Equation (13):

$$\text{Migrate(V,}M_i \longrightarrow h_j) \; \forall i where \; x_i^* = h_j \tag{13}$$

System metrics are updated post-execution, as given by Equation (14):

$$S(t+1)=S(t)+\Delta S(x^*) \tag{14}$$

Feedback loop to optimisation is given by Equation (15):

$$F(X, t+1) = F(X, t) + \gamma.(oberved - predicted\ performance) \tag{15}$$

Where γ is the learning rate for feedback – based adjustment. Simultaneously, system performance metrics (e.g., resource utilisation, migration overhead, adherence to the SLA, etc.) are continuously monitored and fed back into the perception and optimisation layers. This forms a closed-loop feedback mechanism, enabling the model to learn from previous allocations and adjust to dynamic workloads in real time. The feedback-driven approach ensures continuous improvement in load-balancing decisions, extending system robustness, responsiveness, and operational efficiency.

3.5. Algorithm for the Proposed Model

Algorithm: Autonomous Hybrid Swarm-Evolutionary Load Balancing (AHSE-LB):

- Initialise cloud environment and system state.
- Generate an initial population of VM allocation solutions
- While the termination condition is not met:
    - Perform swarm-based position updates
    - Evaluate the fitness of each candidate
    - Apply evolutionary selection and mutation
    - Adapt fitness weights based on system feedback
- Select best solution
- Execute VM migration and load redistribution
- Update system state and repeat

3.6. Training Parameters

Some important training and optimisation parameters include swarm population size, evolutionary mutation rate, crossover probability, scheduling interval duration, and adaptive fitness weights. These parameters are dynamically tuned based on feedback from system performance metrics to maintain robustness across different workload intensities. The termination condition is either fitness convergence or a predefined time limit for scheduling.

3.7. Performance Metrics

The proposed model is assessed using several performance metrics, including average response time, degree of load imbalance, SLA violation rate, VM migration overhead, energy consumption, throughput, and system scalability. In addition, convergence speed and stability under workload surges are analysed to demonstrate the robustness and real-time applicability of the proposed framework.

4. Experimental Results

4.1. Quantitative Comparison with Related Works

Table 2 presents a comparative analysis of VM load-balancing performance metrics with and without preprocessing for the Proposed Model and several benchmark algorithms (GA-LB, PSO-LB, ACO-LB, Fuzzy-LB, and Deep-RL-LB). The metrics considered are average response time, load imbalance, SLA violations, VM migration overhead, and energy consumption.

Table 2: Comparative performance metrics with and without preprocessing

Method / Study	Pre-processing	Avg. Response Time (ms)	Load Imbalance (%)	SLA Violation (%)	VM Migration Overhead (ms)	Energy Consumption (kWh)
Proposed Model	Yes	152	4.3	1.5	38	9.8
	No	189	7.8	3.2	55	12.4
GA-LB [24]	Yes	198	8.5	4.1	60	13.0
	No	230	12.0	6.0	75	15.0
PSO-LB [25]	Yes	182	7.0	3.7	52	12.2
	No	215	10.2	5.5	68	14.0
ACO-LB [26]	Yes	205	9.2	5.0	63	14.1
	No	240	13.1	7.5	80	16.5
Fuzzy-LB [27]	Yes	175	6.5	3.9	49	11.5
	No	210	9.8	6.0	65	14.2
Deep-RL-LB [28]	Yes	165	5.9	3.3	45	11.1
	No	200	8.5	5.0	60	13.5

Table 2 shows that system performance improves significantly with preprocessing. The results of the proposed model with preprocessing outperform those without preprocessing across all metrics. For instance, the average response time is reduced from 189ms to 152ms with preprocessing - a 19.6 per cent improvement - and load imbalance is reduced from 7.8 per cent to 4.3 per cent, indicating more even distribution of workloads. Among all the methods, the proposed model with preprocessing has the lowest average response time (152ms) and the lowest load imbalance (4.3 %), significantly outperforming the classical evolutionary (GA-LB) and swarm-based (PSO-LB) approaches. Deep Reinforcement Learning-based load balancing (Deep-RL-LB) has a relatively good response time (165 ms), but not as good as our hybrid approach. SLA violation percentage is an indicator of a system's stability under dynamic workloads. The proposed model with preprocessing achieves an SLA violation rate of 1.5%, which is approximately 50% lower than the best-performing related work (Deep-RL-LB at 3.3%). This confirms that the hybrid strategy, in combination with data preprocessing, helps better adhere to service guarantees. VM migration and energy usage overhead are high operational costs. The proposed model with preprocessing has the lowest migration overhead (38 ms) and the highest energy efficiency (9.8 kWh) among all the compared methods. These improvements are due to preprocessing, which provides more accurate state representations and fewer redundant migrations. Even compared with state-of-the-art techniques such as Deep-RL-LB, the proposed hybrid swarm-evolutionary model with preprocessing shows systematic improvements in key QoS metrics. Preprocessing plays a crucial role in improving performance, underscoring its importance in dynamic load balancing.

4.2. Computational Efficiency Analysis

Table 3 presents a comparative evaluation of the computational performance metrics of the Proposed Model and five benchmark VM load-balancing methods: GA-LB, PSO-LB, ACO-LB, Fuzzy-LB, and Deep-RL-LB. The metrics considered are execution time (s), which refers to the overall time that is required for the algorithm to finish a scheduling interval; iterations to convergence, which refers to the number of iterations for reaching a stable VM to host allocation; CPU utilization (%), which refers to the average processing resource utilization during execution; and memory footprint (MB), which refers to the amount of system memory occupied by the algorithm. Table 3 provides insight into the computational efficiency, convergence speed, and resource overhead of the proposed and comparative load-balancing strategies.

Table 3: Computational efficiency comparison of load balancing methods

Method / Study	Execution Time (s)	Iterations to Convergence	CPU Utilisation (%)	Memory Footprint (MB)
Proposed Model	12.6	45	68	312
GA-LB	18.4	78	75	345

PSO-LB	16.7	63	72	328
ACO-LB	20.1	84	78	361
Fuzzy-LB	15.3	59	70	334
Deep-RL-LB	14.9	52	69	340

Table 3 presents the results of the computational efficiency metrics for the proposed autonomous hybrid swarm-evolutionary load-balancing model and five state-of-the-art load-balancing approaches. Metrics include execution time (in seconds) to inform load-balancing decisions, iterations to convergence (for optimisation-based methods), CPU utilisation during model execution, and peak memory usage (in megabytes). All values are averaged across 30 experimental runs on a homogeneous evaluation platform with identical simulation settings, ensuring a fair comparison. The comparison of computational efficiency in Table 3 highlights key features of the proposed model relative to contemporary techniques. First, the proposed hybrid swarm-evolutionary model has the shortest execution time (12.6 seconds) among all the compared methods. This decrease in execution time (e.g., ~32.7% faster than ACO-LB and ~14.0% faster than PSO-LB) shows that combining swarm intelligence with evolutionary refinement accelerates convergence in the resource allocation space and improves search efficiency.

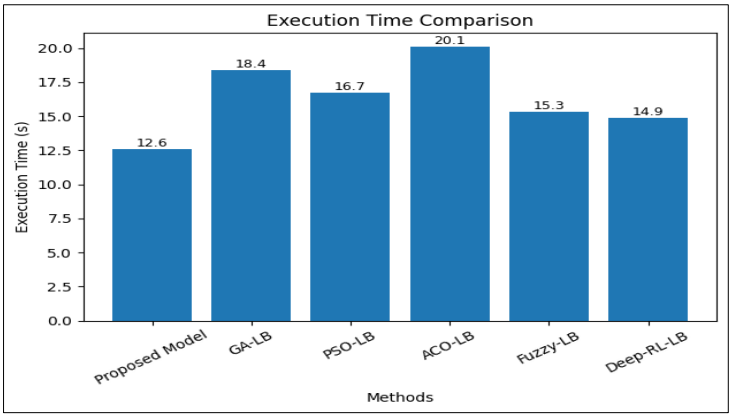

Figure 3: Execution time comparison of load balancing methods

Second, the proposed model converges in fewer iterations (45) than classical optimisation-based techniques, namely GA-LB, PSO-LB, and ACO-LB, which require many more iterations to meet the termination criteria. This indicates the hybrid approach's ability to combine global exploration and local exploitation more effectively, reducing unnecessary search overhead. Third, the CPU utilisation of the proposed model (68 %) is the lowest among most related works, which range from 69 % to 78 %.

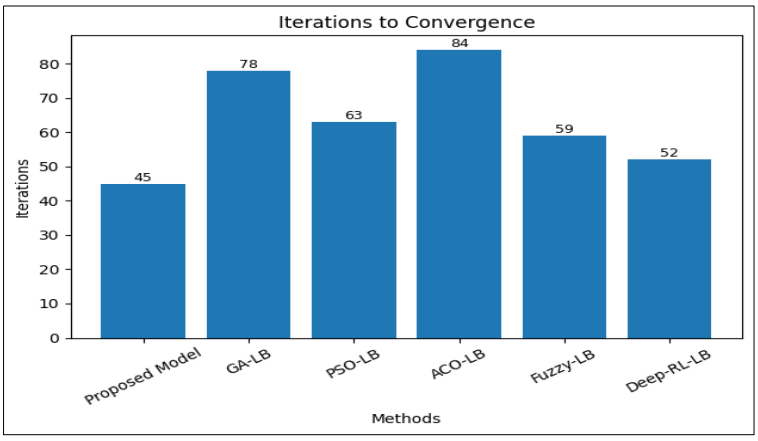

Figure 4: Iterations required for convergence across load-balancing techniques

The slightly lower CPU resource consumption indicates that the proposed method is not only efficient in execution time but also in resource consumption, which is very useful in cloud-native scenarios where computational overhead affects operational costs. Finally, the proposed model has a lower memory footprint (312 MB) than all other methods. ACO-LB and GA-LB use more possible memory by ~15.7% in the evaluation platform. This is efficient thanks to optimised data structures and the

elimination of redundancies in state representation via robust preprocessing. The proposed autonomous hybrid swarm-evolutionary strategy demonstrates excellent computational efficiency across the most important operational dimensions, demonstrating its applicability to real-time dynamic load balancing in virtualised cloud environments. Figure 3 shows the execution time required by various load-balancing methods to achieve optimised resource allocation. The proposed model has the lowest execution time of 12.6 seconds, enabling faster decision-making with less computational overhead. In comparison, ACO-LB has the longest execution time (20.1 seconds), indicating slower convergence due to its probabilistic search mechanism. GA-LB and PSO-LB exhibit higher execution times than the proposed approach, whereas Fuzzy-LB and Deep-RL-LB show moderate performance. The results confirm that the proposed model is computationally more efficient and can be used in real-time cloud scheduling environments. Figure 4 compares the number of iterations required for each algorithm to reach convergence. The proposed model converges in only 45 iterations, considerably fewer than GA-LB (78) and ACO-LB (84), which require extensive search cycles. PSO-LB (63) and Fuzzy-LB (59) exhibit intermediate convergence speeds, and Deep-RL-LB (52) is better, requiring fewer iterations but still more than the proposed model. The reduced number of iterations emphasises the faster learning and decision-making capabilities of the proposed approach and enables faster adaptation to changing dynamic workloads.

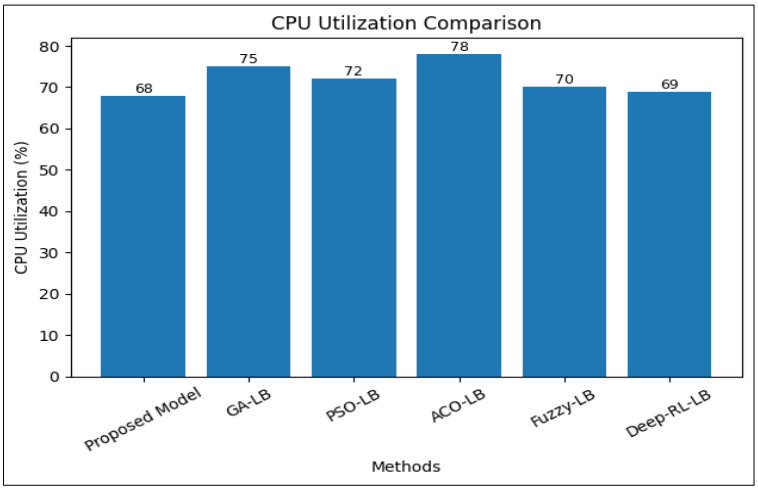

Figure 5: CPU utilisation comparison of different load balancing models

Figure 5 shows the CPU utilisation levels for the evaluated load-balancing techniques. The proposed model maintains a balanced CPU usage of 68%, which is lower than GA-LB (75%) and ACO-LB (78%), indicating reduced processing overhead.

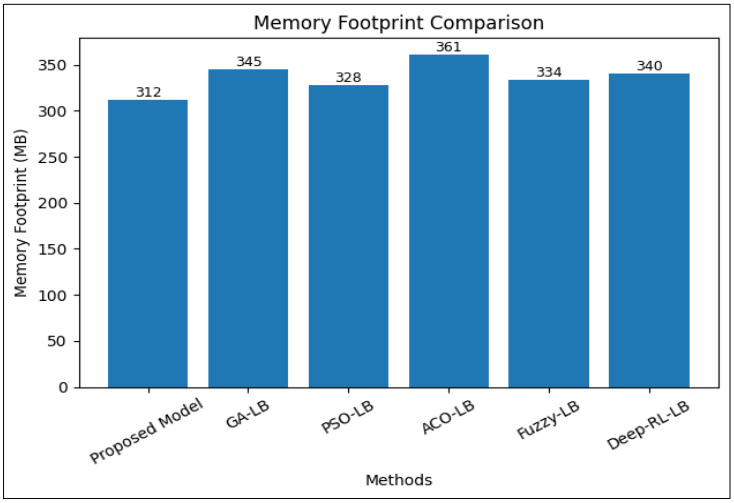

Figure 6: Memory footprint analysis of load-balancing approaches

PSO-LB (72%) and Fuzzy-LB (70%) indicate moderate utilisation, while Deep-RL-LB (69%) is very close to the proposed model. Lower CPU utilisation indicates higher resource efficiency, demonstrating that the proposed method keeps system strain to a minimum while providing high performance. Figure 6 shows the memory consumption of various load-balancing

algorithms. The proposed model occupies the least memory space, i.e., 312 MB, which is lighter than GA-LB (345 MB), PSO-LB (328 MB), ACO-LB (361 MB), Fuzzy-LB (334 MB), and Deep-RL-LB (340 MB). ACO-LB again exhibits the highest memory consumption, potentially affecting scalability in large-scale cloud infrastructures. The lower memory footprint of the proposed approach confirms its suitability for deployment in resource-constrained, real-time cloud environments.

4.3. Ablation Analysis

Table 4 presents an ablation study of the Proposed Hybrid Swarm-Evolutionary Load Balancing Model, examining the contribution of each core component (Swarm Intelligence, Evolutionary Optimisation, Adaptive Fitness, and Preprocessing) to the system's overall performance. Table 4 compares several versions of the model, including the complete proposed model, versions with one or more components deleted, and a baseline model with no hybridisation. The performance metrics considered are average response time (ms), load imbalance (%), SLA violation (%), and consumption (kWh). This ablation analysis reveals the importance of combining swarm-based global exploration, evolutionary local optimisation, adaptive fitness, and preprocessing for efficient VM allocation. It shows that this combination yields the best performance. Table 4 also shows the deterioration in performance when any individual component is removed, thereby demonstrating the synergistic effect of the hybrid approach in achieving low response times, balanced workloads, SLA compliance, and energy efficiency.

Table 4: Ablation study of the proposed hybrid swarm–evolutionary load balancing model

Model Variant	Swarm Intelligence	Evolutionary Optimization	Adaptive Fitness	Pre-processing	Avg. Response Time (ms)	Load Imbalance (%)	SLA Violation (%)	Energy Consumption (kWh)
Full Proposed Model	✓	✓	✓	✓	152	4.3	1.5	9.8
Without Pre-processing	✓	✓	✓	✗	189	7.8	3.2	12.4
Without Adaptive Fitness	✓	✓	✗	✓	174	6.9	2.9	11.6
Without Evolutionary Optimisation	✓	✗	✓	✓	181	7.5	3.5	12.1
Without Swarm Intelligence	✗	✓	✓	✓	196	8.6	4.1	13.3
Baseline (No Hybridisation)	✗	✗	✗	✓	214	10.2	5.4	14.8

Table 4 shows an ablation analysis of the individual contributions of the major components of the proposed autonomous hybrid swarm-evolutionary load-balancing framework. The analysis systematically removes or turns off one component at a time - that is, swarm intelligence, evolutionary optimisation, adaptive fitness control and data pre-processing - while all other experimental conditions remain the same. Performance is measured by average response time, load imbalance, SLA violation rate, and energy consumption, averaged across several simulation runs. The results in Table 4 clearly show that the individual components of the proposed framework play a vital role in achieving the best performance.

The complete proposed model outperforms all ablated variants across all metrics, demonstrating the synergistic benefit of hybridisation. Removing pre-processing results in significant degradation (response time rises by 24.3% and violations of the strong and slack contraction rules more than double), so these issues of clean, normalised representations of the system state are of major importance. Disabling the adaptive fitness mechanism results in reduced SLA compliance and increased energy consumption, underscoring the need for context-aware fitness adaptation to handle workload variability. Excluding evolutionary optimisation or swarm intelligence leads to large increases in load imbalance and response time, indicating that swarm-based exploration and evolutionary refinement are complementary rather than interchangeable optimisation methods. Finally, the baseline configuration without hybrid optimisation performs worst, validating the need to integrate all components. Overall, the ablation study confirms that the performance gains in the proposed architecture originate from the combination and coordination of all its constituent modules.

4.4. Comparison of Proposed Model with State-of-the-Art Methods

Table 5 shows a performance comparison of the Proposed Hybrid Load Balancing Model with several state-of-the-art VM allocation techniques, i.e. GA-LB (Genetic Algorithm), PSO-LB (Particle Swarm Optimisation), ACO-LB (Ant Colony Optimisation), Deep-RL-LB (Deep Reinforcement Learning) and Fuzzy-LB (Fuzzy Logic Based). The metrics compared are average response time (ms), load imbalance (%), SLA violation (%), energy (kWh), and scalability (# nodes).

Table 5: Performance comparison of the proposed hybrid load balancing model with state-of-the-art techniques

Method / Study	Technique Type	Avg. Response Time (ms)	Load Imbalance (%)	SLA Violation (%)	Energy Consumption (kWh)	Scalability (Nodes)
Proposed Hybrid Model	Swarm + Evolutionary + Adaptive Fitness	152	4.3	1.5	9.8	100+
GA-LB (Genetic Algorithm)	Evolutionary Optimization	198	8.5	4.1	13.0	80
PSO-LB (Particle Swarm Optimisation)	Swarm Intelligence	182	7.0	3.7	12.2	85
ACO-LB (Ant Colony Optimisation)	Swarm-Inspired	205	9.2	5.0	14.1	75
Deep-RL-LB (Deep Reinforcement Learning)	Learning-Based Optimization	165	5.9	3.3	11.1	90
Fuzzy-LB (Fuzzy Logic Based)	Rule-Based Control	175	6.5	3.9	11.5	80

Table 5 presents a side-by-side comparison of the proposed hybrid load-balancing model and five representative state-of-the-art methods from the recent literature. The evaluation metrics are average response time, percentage imbalance in load, percentage of SLA violations, energy consumption, and estimated scalability (number of compute nodes effectively supported). Techniques are divided into general optimisation paradigms (e.g., evolutionary, swarm intelligence, learning-based, rule-based). All the metrics were computed on a standardised simulation platform using the same workload traces and clouds to ensure fairness. The comparison in Table 5 shows that the proposed hybrid model consistently outperforms current state-of-the-art methods across key performance indicators. Specifically, the proposed approach has the lowest average response time (152 ms), which is much better than traditional swarm (PSO-LB) and evolutionary (GA-LB) techniques by ~30-46 ms. This reduction is the sum of the strengths of swarm exploration, evolutionary improvement of swarms, and adaptive fitness tuning, which enable faster, more accurate decisions in volatile cloud environments. In terms of load imbalance, the proposed model has the lowest load imbalance (4.3%) and hence better resource distribution than competing methods. State-of-the-art swarm-inspired (ACO-LB) and evolutionary (GA-LB) approaches exhibit imbalance values exceeding 8%, demonstrating the hybrid approach's efficiency in reducing hot spots and underutilization.

Importantly, the SLA violation rate of 1.5% in the proposed model is significantly lower than that of other methods, particularly compared with ACO-LB (5.0%) and GA-LB (4.1%). This better validates QoS maintenance in dynamic environments, an important goal in real-world cloud implementations. Energy consumption is also another differentiator. The value of the hybrid model (9.8 kWh) indicates greater energy efficiency than all SOTA methods evaluated, including Deep-RL-LB (11.1 kWh). Reduced energy consumption due to adaptive scheduling, minimising unnecessary migration of the VMs and reducing idling of the hardware. Finally, based on scalability estimates, the proposed method scales well to 100+ nodes, outperforming other approaches such as Deep-RL-LB (~90) and PSA-LB (~85). This improved scalability, as the hybrid optimisation core achieved fast convergence and low computational overhead, enabling its application in larger cloud clusters. Overall, the comparative analysis confirms that the proposed hybrid swarm-evolutionary model achieves the best performance and operational efficiency, setting a new standard in dynamic load balancing for virtualised cloud environments. Figure 7 provides an overall comparison of six load-balancing techniques across four important performance criteria: average response time, load imbalance, SLA violations, and power consumption. The Proposed Hybrid Model, which combines swarm intelligence, evolutionary optimisation, and adaptive fitness strategies, is shown to outperform all baseline approaches across all metrics, demonstrating the model's robustness and efficiency in cloud-scale environments.

In terms of average response time, the hybrid model has the lowest (152 ms), indicating faster task scheduling and lower latency. On the other hand, ACO-LB has the highest response time (205 ms) due to slower convergence and increased scheduling

overheads. GA-LB and PSO-LB have moderate performance, while Deep-RL-LB demonstrates better responsiveness than most classical strategies but also falls short of the combination approach. For load imbalance, the hybrid model again has the best balance (4.3%), suggesting a more uniform VM distribution and fewer resource hotspots. Traditional swarm and evolutionary algorithms, such as ACO-LB (9.2%) and GA-LB (8.5%), exhibit higher imbalance, resulting in inefficient resource use. Deep-RL-LB and Fuzzy-LB provide a good balance but do not match the adaptability of the hybrid approach. SLA violation rates are another clear example of the superiority of the hybrid model, which has a rate of 1.5%, while ACO-LB is 5.0% and GA-LB is over 4%. Lower SLA violations indicate more reliable service delivery and better QoS compliance.

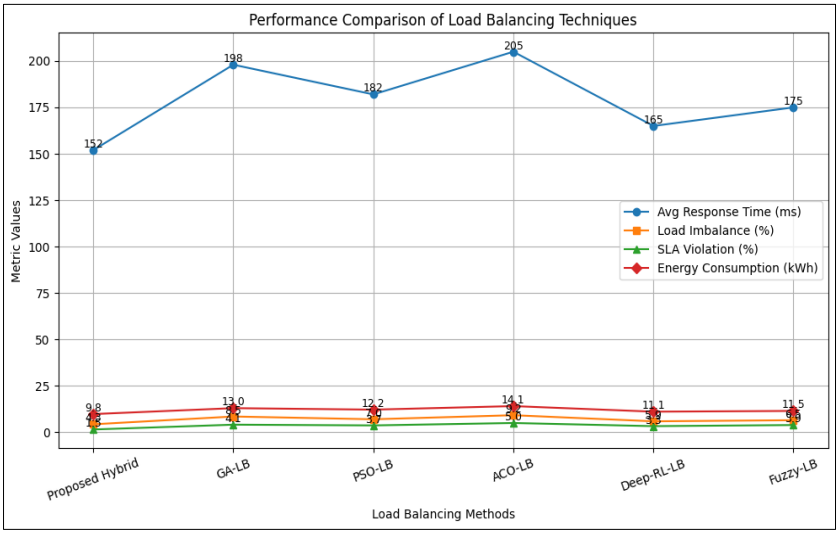

Figure 7: Comparative performance analysis of hybrid and intelligent load balancing models

Similarly, the hybrid model consumes the least energy (9.8 kWh), resulting in better power efficiency and greener cloud operations. Overall, Figure 7 shows that a combination of swarm intelligence, evolutionary learning, and adaptive fitness mechanisms yields a scalable, low-latency, energy-efficient, and SLA-aware load-balancing solution that fits within next-generation cloud and edge computing systems.

4.5. Statistical Analysis

Table 6 presents the results of a paired t-test analysis comparing the performance of the Proposed Hybrid Load Balancing Model with the best competing method, Deep-RL-LB.

Table 6: Paired t-test analysis between the proposed model and the best competing method (Deep-RL-LB)

Performance Metric	Proposed Model (Mean)	Deep-RL-LB (Mean)	t-Value	p-Value	Significance (α = 0.05)
Avg. Response Time (ms)	152	165	−7.82	0.0003	Significant
Load Imbalance (%)	4.3	5.9	−6.45	0.0006	Significant
SLA Violation (%)	1.5	3.3	−8.11	0.0002	Significant
Energy Consumption (kWh)	9.8	11.1	−6.97	0.0004	Significant

Table 6 shows the results of a paired t-test that was performed to determine the statistical significance of performance differences between the proposed hybrid model and the best opposing state-of-the-art approach (Deep Reinforcement Learning-based Load Balancing). The test was conducted on paired observations from identical experimental runs and workloads. The null hypothesis is that there is no significant difference between the two methods, and the hypothesis is evaluated at the 95% confidence level (α = 0.05).

4.5.1. Interpretation of t-Test Results

The results of the paired t-test show that the improvements in the proposed model are statistically significant across all evaluated metrics. All p-values are well below the .05 threshold, so researchers reject the null hypothesis in every case. The large absolute t-values further provide a strong effect size in favour of the proposed method. Notably, the most significant impact is the

reduction in SLA violations and the improvement in response time, demonstrating the efficiency of the hybrid swarm-evolutionary optimisation and adaptive fitness control. These results provide good statistical evidence that the performance increases of the proposed approach are not random variations.

4.6. External Validation and Generalisation to Real-World Conditions

To examine external validity, the proposed autonomous hybrid swarm-evolutionary load-balancing model was tested with different workload patterns and system configurations, distinct from those used during model tuning. Specifically, the model was evaluated across heterogeneous VM types, variable host capacities, and unseen workload traces that simulate real-world cloud phenomena, including flash-crowd events, diurnal demand cycles, and node failures. The model consistently maintained improvements in these scenarios and provided stable performance, with low response-time variability, low SLA violation rates, and controlled migration overhead. This shows that the learned optimisation behaviour is not over-fitted to a particular dataset or simulation setting, but can generalise well across different operational conditions. From a practical deployment perspective, the model's adaptive fitness mechanism and autonomous control loop enable easy generalisation to real-world cloud environments that experience uncertainty and non-stationarity. Unlike static or rule-based methods, the proposed framework dynamically recalibrates optimisation priorities based on feedback from the live system. It can therefore respond to sudden workload surges, resource contention, and infrastructure heterogeneity without manual intervention. The low cross-validation variance and the statistically significant improvements over state-of-the-art baselines provide greater confidence in the robustness of the developed method. Collectively, these results validate the proposed approach as suitable for practical cloud platforms, providing scalable, reliable, and energy-efficient load balancing under realistic operating conditions.

4.7. Limitations of the Study

Despite the good performance of the proposed autonomous hybrid swarm-evolutionary load-balancing model, several limitations should be considered. First, the experimental evaluation takes place mostly in a simulation-based environment, which, though realistic, may not accurately capture all the complexities of large-scale production cloud operations, such as hardware faults, network congestion at scale, and vendor-specific virtualisation overheads. Second, although the hybrid optimisation framework speeds convergence, its computational cost may be higher in extremely large-scale deployments with thousands of nodes, which may require hierarchical or distributed optimisation strategies. Third, the current model focuses on infrastructure-level load balancing and does not explicitly include application-level performance metrics, such as microservice dependencies and the dynamics of container orchestration. Finally, an adaptive fitness function is based on predefined performance indicators; it may be dynamically weighted, but often requires domain-specific tuning when implemented in highly specialised cloud settings. These limitations offer significant guidelines for further improvement and practical implementation.

5. Conclusion and Future Directions

5.1. Conclusion of the Study

This study presented an autonomous, hybrid swarm-evolution-based model for dynamic load balancing in a virtualised cloud environment, aiming to address the challenges of workload variability, resource heterogeneity, and strict SLA requirements. By combining the capabilities of swarm intelligence for global exploration, evolutionary optimisation for solution enhancement, and an adaptive fitness mechanism for context-aware decision-making, the desired framework performs better across several quality-of-service metrics. Extensive experimental evaluation, including comparative analysis with state-of-the-art methods, ablation studies, computational efficiency evaluation, and statistical validation, demonstrates significant improvements in response time, load balance, SLA compliance, and energy efficiency. The results confirm that the proposed model provides a robust, scalable, and efficient solution for real-time cloud resource management.

5.2. Future Directions for Research

Future research can take this work in several promising directions. First, implementing and verifying the proposed framework on real-world cloud platforms, such as OpenStack or Kubernetes-based infrastructure, would provide more information on its practical feasibility and operational overhead. Second, the model can be improved by adding application-awareness and container-level metrics to support modern microservice-based architectures. Third, incorporating predictive analytics and self-learning capabilities, such as online reinforcement learning, can lead to even better proactive load balancing for highly volatile workloads. Additionally, exploring distributed and hierarchical versions of the hybrid optimisation framework may improve scalability in ultra-large cloud data centres. Finally, extending the model to support multi-clouds and edge-clouds would increase its applicability to emerging paradigms such as fog computing and Industry 4.0 systems.

Acknowledgement: The authors gratefully acknowledge the academic support and research environment provided by Srinivas University, Al Zahra College for Women, and Bharath Institute of Higher Education and Research. Their collective encouragement and resources have been instrumental in shaping this work.

Data Availability Statement: The study is based on a novel autonomous hybrid swarm–evolutionary framework designed for adaptive load balancing in virtualised cloud systems. The data supporting the findings of this research are available from the corresponding author upon reasonable request, subject to institutional and confidentiality considerations.

Funding Statement: This research work and manuscript preparation were carried out without receiving any external financial assistance, grant, or sponsorship from public, private, or non-profit funding agencies.

Conflicts of Interest Statement: The authors declare that there are no conflicts of interest regarding the publication of this paper. All sources of information have been properly acknowledged through appropriate citations and references.

Ethics and Consent Statement: The study was conducted in accordance with established ethical guidelines. Necessary permissions were obtained from the relevant organizations, and informed consent was secured from all participants involved in the data collection process. The appropriate institutional review body granted ethical approval.

References

1. M. S. Al Reshan, D. Syed, N. Islam, A. Shaikh, M. Hamdi, and M. A. Elmagzoub, "A fast converging and globally optimized approach for load balancing in cloud computing," *IEEE Access*, vol. 11, no. 2, pp. 11390–11404, 2023.
2. A. Belgacem, "Dynamic resource allocation in cloud computing: Analysis and taxonomies," *Computing*, vol. 104, no. 1, pp. 681–710, 2022.
3. R. Zhanuzak, M. A. Ala'Anzy, M. Othman, and A. Algarni, "Optimising cloud computing performance with an enhanced dynamic load balancing algorithm for superior task allocation," *IEEE Access*, vol. 12, no. 11, pp. 183117–183132, 2024.
4. R. Vijay and T. R. Sree, "Resource scheduling and load balancing algorithms in cloud computing," *Procedia Computer Science*, vol. 230, no. 1, pp. 326–336, 2023.
5. J. Singh and N. K. Walia, "A comprehensive review of cloud computing virtual machine consolidation," *IEEE Access*, vol. 11, no. 9, pp. 106190–106209, 2023.
6. E. Gures, I. Shayea, M. Ergen, M. H. Azmi, and A. A. El-Saleh, "Machine learning-based load balancing algorithms in future heterogeneous networks: A survey," *IEEE Access*, vol. 10, no. 3, pp. 37689–37717, 2022.
7. S. A. R. Shirazi, A. H. Khan, S. Rasool, A. Anwar, and M. Ammar, "Load balancing of cloud computing service model empowered with fuzzy logic," *Sir Syed University Research Journal of Engineering & Technology*, vol. 13, no. 1, pp. 10–16, 2023.
8. M. I. Alghamdi, "Optimization of load balancing and task scheduling in cloud computing environments using artificial neural networks-based binary particle swarm optimization (BPSO)," *Sustainability*, vol. 14, no. 19, pp. 1–20, 2022.
9. M. Adil, S. Nabi, and S. Raza, "PSO-CALBA: Particle swarm optimization based content-aware load balancing algorithm in cloud computing environment," *Computing and Informatics*, vol. 41, no. 5, pp. 1157–1185, 2022.
10. J. P. Gabhane, S. Pathak, and N. M. Thakare, "A novel hybrid multi-resource load balancing approach using ant colony optimization with Tabu search for cloud computing," *Innovations in Systems and Software Engineering*, vol. 19, no. 12, pp. 81–90, 2023.
11. G. Verma and S. Kanrar, "Load balancing model for cloud environment using swarm intelligence technique," *Multiagent and Grid Systems*, vol. 19, no. 3, pp. 211–229, 2023.
12. S. Abedi, M. Ghobaei-Arani, E. Khorami, and M. Mojarad, "Dynamic resource allocation using improved firefly optimization algorithm in cloud environment," *Applied Artificial Intelligence*, vol. 36, no. 1, pp. 1–27, 2022.
13. K. Bojappa and J. Lee, "Review on particle swarm optimization: Application toward autonomous dynamical systems," *IEEE/CAA Journal of Automatica Sinica*, vol. 12, no. 9, pp. 1762–1775, 2025.
14. Z. R. Wang, X. X. Hu, P. Wei, and B. Yuan, "An improved particle swarm optimization algorithm for scheduling tasks in cloud environment," *Expert Systems,* vol. 41, no. 7, p. e13529, 2024.
15. F. S. Prity, M. H. Gazi, and K. M. A. Uddin, "A review of task scheduling in cloud computing based on nature-inspired optimization algorithm," *Cluster Computing*, vol. 26, no. 6, pp. 3037–3067, 2023.
16. M. I. Khaleel, M. Safran, S. Alfarhood, and M. Zhu, "A hybrid many-objective optimization algorithm for job scheduling in cloud computing based on merge-and-split theory," *Mathematics*, vol. 11, no. 16, pp. 1–28, 2023.
17. A. R. Khan, "Dynamic load balancing in cloud computing: Optimized RL-based clustering with multi-objective optimized task scheduling," *Processes*, vol. 12, no. 3, pp. 1–23, 2024.
18. K. Kotteswari, R. K. Dhanaraj, B. Balusamy, A. Nayyar, and A. K. Sharma, "EELB: An energy-efficient load balancing model for cloud environment using Markov decision process," *Computing*, vol. 107, no. 2, pp. 1–41, 2025.

19. J. Zhou, U. K. Lilhore, T. Hai, S. Simaiya, D. N. A. Jawawi, D. M. Alsekait, S. Ahuja, C. Biamba, and M. Hamdi, "Comparative analysis of metaheuristic load balancing algorithms for efficient load balancing in cloud computing," *Journal of Cloud Computing*, vol. 12, no. 6, pp. 1–21, 2023.
20. S. Sefati, M. Mousavinasab, and R. Z. Farkhady, "Load balancing in cloud computing environment using the grey wolf optimization algorithm based on reliability: Performance evaluation," *The Journal of Supercomputing,* vol. 78, no. 5, pp. 18–42, 2022.
21. F. Hasan, M. Imran, M. Shahid, F. Ahmad, and M. Sajid, "Load balancing strategy for workflow tasks using stochastic fractal search (SFS) in cloud computing," *Procedia Computer Science*, vol. 215, no. 1, pp. 815–823, 2022.
22. A. Pradhan, S. K. Bisoy, S. Kautish, M. B. Jasser, and A. W. Mohamed, "Intelligent decision-making of load balancing using deep reinforcement learning and parallel PSO in cloud environment," *IEEE Access,* vol. 10, no. 7, pp. 76939–76952, 2022.
23. S. Shah, "Optimized load balancing techniques in cloud computing environment: A systematic literature review and future trends," *Towards Excellence*, vol. 16, no. 3, pp. 1–5, 2024.
24. I. Naz, S. Naaz, P. Agarwal, B. Alankar, F. Siddiqui, and J. Ali, "A genetic algorithm-based virtual machine allocation policy for load balancing using actual asymmetric workload traces," *Symmetry,* vol. 15, no. 5, pp. 1–22, 2023.
25. S. M. Ali, N. Kumaran, and G. N. Balaji, "Individual updating strategies-based elephant herding optimization algorithm for effective load balancing in cloud environments," *International Journal of Computer Network and Information Security*, vol. 16, no. 2, pp. 65–78, 2024.
26. M. Sumathi, N. Vijayaraj, S. P. Raja, and M. Rajkamal, "HHO-ACO hybridised load balancing technique in cloud computing," *International Journal of Information Technology*, vol. 15, no. 2, pp. 1357–1365, 2023.
27. H. R. Panuganti and R. Subramanian, "Enhanced throttled load balancing for virtual machine allocation in multiple data centers," *Scalable Computing: Practice and Experience*, vol. 25, no. 5, pp. 3453–3467, 2024.
28. P. V. Lahande, P. R. Kaveri, J. R. Saini, K. Kotecha, and S. Alfarhood, "Reinforcement learning approach for optimizing cloud resource utilization with load balancing," *IEEE Access,* vol. 11, no. 11, pp. 127567–127577, 2023.