Adaptive Mirage Search Optimization for Educational Workload Scheduling in Multi-Cloud Environments

Mohammed A.S. Mosleh[1], Elham Alzain[2]*, Hesham M.A. Abdullah[3], Ali Alshebami[4]
Al-Fayha Private College, Jubail, Saudi Arabia[1]
Applied College, King Faisal University, Al-Ahsa 31982, Saudi Arabia[2]
Al-Fayha Private College, Jubail, Saudi Arabia[3]
Applied College, King Faisal University, Al-Ahsa 31982, Saudi Arabia[4]

***Abstract*—AI-driven educational platforms increasingly rely on heterogeneous cloud workloads, including real-time tutoring, adaptive assessments, and large-scale learning analytics, which impose diverse and conflicting quality-of-service (QoS) requirements. Conventional cloud scheduling approaches fail to address such heterogeneity, often leading to service-level agreement (SLA) violations, increased execution cost, and poor resource utilization. This study proposes QoS-aware Adaptive Mirage Search Optimization Framework (QoS-AMSO), a QoS-aware multi-objective task scheduling framework based on an enhanced Mirage Search Optimization algorithm for intelligent workload management in multi-cloud environments. The proposed approach integrates four key innovations: adaptive refraction-based step sizing for improved convergence, QoS-feedback-driven dynamic weight adjustment, Lévy flight-based diversity preservation, and Pareto-based elite solution selection. Additionally, a task classification model categorizes workloads into latency-critical, throughput-intensive, deadline-soft, and batch-deferred classes, enabling context-aware scheduling decisions. The framework is evaluated using CloudSim 3.0 simulations with three cloud providers, 60 virtual machines, and workload sizes ranging from 100 to 1000 educational tasks under multiple workload scenarios. The QoS-AMSO achieves the best overall performance, reducing makespan to 798 s, execution cost to USD 292.8, SLA violation rate to 3.9%, and improving resource utilization to 97.2%. These results demonstrate the effectiveness and scalability of QoS-AMSO for QoS-aware scheduling in AI-powered educational multi-cloud environments.**

***Keywords*—Multi-cloud computing; task scheduling; Quality of Service (QoS); multi-objective optimization; mirage search optimization; AI-powered education; SLA-aware scheduling; metaheuristic algorithms**

I. Introduction

The advancements of artificial intelligence (AI) technologies have transformed modern educational systems, enabling the development of intelligent, scalable, and personalized learning analytics that require substantial computational resources to process diverse and dynamic workloads. Yet, the integration of AI-driven educational workloads with multi-cloud infrastructures introduces challenges in resource management and task scheduling [1]. The heterogeneous educational workloads exhibit diverse QoS requirements. In batch-oriented tasks, such as performance analytics and report generation, high throughput and resource-intensive execution are required. This heterogeneity results in conflicting optimization objectives, including decreasing execution time, operation cost, ensuring SLA compliance, and maintaining balanced resource utilization. Conventional cloud task scheduling approaches, including heuristic and metaheuristic algorithms such as Round Robin, Min-Min, Particle Swarm Optimization (PSO), and Genetic Algorithms (GA), are applied for resource allocation [2]. Many existing schedulers are developed for single-objective optimization, without considering the multi-dimensional nature of QoS requirements [3]. Most existing approaches rely on static objective weighting, fixed scheduling priorities, or heuristic decision strategies that are unable to adapt to continuously changing workload characteristics in multi-cloud environments. Although recent metaheuristic schedulers improve global search capability, they lack workload-aware decision-making and dynamic QoS adaptation, often resulting in suboptimal trade-offs among execution time, execution cost, SLA compliance, and resource utilization. The standard Mirage Search Optimization (MSO) algorithm provides an effective exploration–exploitation mechanism; however, it is a single-objective optimizer and does not incorporate task classification, runtime QoS feedback, adaptive search control, or Pareto-based solution selection required for heterogeneous educational workloads. Existing QoS-aware scheduling techniques and conventional MSO-based optimization remain inadequate for intelligent multi-cloud scheduling under dynamic educational environments. To address these challenges, this study proposes a QoS-aware Adaptive Mirage Search Optimization framework (QoS-AMSO) for intelligent task scheduling in AI-driven educational multi-cloud environments. The contribution of this research is a QoS-aware task classification model that categorizes educational workloads into latency-critical, throughput-intensive, deadline-soft, and batch-deferred classes, enabling context-aware scheduling decisions.

A QoS-AMSO algorithm enhances the standard MSO through adaptive refraction-based step sizing to improve convergence behavior in complex search spaces. A dynamic QoS feedback mechanism that adjusts objective weights in real time based on SLA violations, enabling responsive and adaptive scheduling. The model integrates a Lévy flight-based perturbation strategy to maintain population diversity and prevent premature convergence. A Pareto-based elite archive mechanism for effective multi-objective optimization, ensuring balanced trade-offs among competing QoS metrics.

The remainder of this study is organized as follows. Section II reviews existing works; Section III presents the system

*Corresponding author.

model and formulates the multi-objective optimization problem; Section IV describes the proposed QoS-AMSO framework; Section V discusses the experimental setup and evaluation, and Section VI concludes the study with future research.

II. Related Works

A. Cloud Task Scheduling Approaches

Rajput et al. [4] proposed a QoS-aware heuristic-based multi-cloud workflow scheduler for Spark tasks, achieving a 50% performance improvement, but it degrades under volatile conditions. Mangalampalli et al. [5] introduced ATS-IA3C, an A3C-based adaptive scheduler that improves makespan by 70.49% and cost by 77.42%, though it has high computational complexity. Mangalampalli et al. [6] proposed MOPTSA3C, which incorporates task and VM priorities into A3C, improving makespan and utilization but adding overhead due to priority computation. Kumar et al. [7] developed EAEFA based on electric fish swarm behavior, achieving over 30% makespan improvement in large-scale scenarios. Sandhu et al. [8] introduced TBW optimization combining Tabu Search, Bayesian classification, and WOA, achieving 95% efficiency but increasing computational complexity.

Acharya et al. [9] developed a Golden Search WOA (GSWOA) and attained 132s Makespan, 2.7s response time, and 74.34% RU. However, the GSWOA model has higher memory usage. Bansal et al. [10] presented a Hybrid PSO-Grasshopper Optimization Algorithm (HPSO-GOA) and attained 1636.44 total execution cost and 167.54J EC. Yet, the model is sensitive to initial population characteristics. Nagalakshmi et al. [11] presented an Energy-Aware TS using Orthogonal Learning PSO (MOEATS-OLPSO) and attained 180$ cost, 0.6J for energy consumption, and 37s time. Yet, the model lacked failure-handling mechanisms. Sanaj et al. [12] presented an EcoTaskOpt model that integrated Ant Colony Optimization (ACO) and the PSO algorithm. The model reduced makespan by 18%. Yet, the model's performance is affected by varying task lengths and heterogeneous systems.

B. QoS-Aware Scheduling Techniques

Tabary et al. [13] proposed SMPIA-GA-PSO, integrating GA and PSO with message passing to improve resource utilization, though it requires complex tuning. Long et al. [14] developed IVPTS, combining PSO with fuzzy logic to reduce makespan by 13%, but with higher computational cost. Tamilarasu et al. [15] introduced ICOATS using Coati Optimization to improve makespan and utilization. Sha et al. [16] proposed SPP-TLBO for deadline-aware service placement, improving efficiency by 8–19%. Xu et al. [17] introduced EAT for AIGC task scheduling across edge servers, reducing latency by up to 56%, but with high complexity and overhead.

C. AI Workloads in Cloud Environment

Sharma et al. [18] proposed ICO for AI-driven orchestration, improving scalability but with high implementation complexity. Adeyinka [19] developed LSTM-RL-DT for workload forecasting and dynamic allocation, reducing cost by 25% but struggling with rapid changes. Bauskar [20] introduced an AI-driven multi-cloud database optimization framework, facing consistency challenges across heterogeneous systems. Jorepalli et al. [21] proposed a cloud-native resilient AI architecture for smart education, offering scalability and fault tolerance but with complex deployment.

D. Research Gap Summary

Existing extensions of metaheuristic algorithms often fail to incorporate real-time feedback mechanisms and workload-specific prioritization, limiting their effectiveness in practical deployment scenarios. Table I illustrates the research gap summary.

TABLE I. Research Gap Summary

Method	Models	Strengths	Research Gaps
Classical Scheduling	RR, FCFS	Simple and fast execution	Poor performance in heterogeneous environments
Heuristic-Based	Min-Min, Max-Min	Efficient makespan reduction	Ignores multi-objective, poor adaptability
Swarm Intelligence	PSO	Strong global search capability	Premature convergence, lacks QoS integration
Nature-Inspired Metaheuristics	GWO, WOA	Balanced exploration and exploitation	Static parameters, no dynamic QoS adaptation
Physics-Based Optimization	MSO	Effective exploration–exploitation balance	Single-objective design, lacks Pareto optimization

Existing studies focus on improving individual QoS metrics such as makespan, execution cost, energy efficiency, or load balancing in isolation. Many optimization algorithms employ fixed search strategies and static objective weighting, limiting their adaptability to heterogeneous educational workloads with dynamically changing QoS requirements. Recent AI-based scheduling methods do not incorporate workload-aware task classification with runtime QoS adaptation and Pareto-based multi-objective decision-making. A research gap remains in developing an adaptive scheduling framework capable of simultaneously balancing multiple conflicting QoS objectives while responding to dynamic workload characteristics.

III. System Model and Problem Formulation

The QoS-AMSO framework models task scheduling in AI-driven educational multi-cloud environments as a QoS-constrained, multi-objective optimization problem, and heterogeneous workloads are dynamically mapped to distribute VMs across multiple cloud providers. Each task is represented by a feature vector including computational length, deadline, priority, data size, and execution requirements, and is classified into latency-critical, throughput-intensive, deadline-soft, or batch-deferred categories to enable context-aware scheduling. A centralized scheduler acts as a cloud broker, maintains global knowledge of tasks and resources to determine an optimal task-to-VM mapping that satisfies QoS constraints. The QoS-AMSO algorithm iteratively explores the solution space using adaptive mechanisms such as dynamic step sizing, QoS-driven weight adjustment, and diversity preservation [22]. Solutions are evaluated using a Pareto-based multi-objective framework, and a QoS feedback loop continuously monitors and adjusts performance during execution. The Pareto-optimal scheduling

plan deployed in the multi-cloud environment ensures adaptive and scalable scheduling under dynamic conditions [23].

A. Task and Resource Model

The system model represents the overall structure of the AI-driven educational multi-cloud environments and defines the interaction between tasks and resources during the scheduling process. Let the system consist of a set of tasks. $sT = \{sT_1, sT_2, sT_3, \dots sT_n\}$, a set of VMs distributed across multiple cloud providers $VM = \{vm_1, vm_2, vm_3, \dots, vm_m\}$, a set of cloud providers is given as $CP = \{cp_1, cp_2, \dots cp_1\}$ and each VM belongs to a cloud is defined as $vm_j \in c_k$. The task scheduling (mapping) function assigns each task (sT_i) to a vm_j and it is defined as $\sigma: sT \rightarrow VM$. The goal is to find an optimal mapping (σ) that decreases execution time, cost, SLA violations, and balances system load. Each task (sT_i) is defined as a multi-parameter entity, which includes $sT_i = (tL_i, d_i, p_i, m_i, c_i)$. Here, tL_i defines the task length, d_i defines the deadline, p_i defines the priority level, m_i defines the memory requirement, and c_i defines the task class. In the task classification role, based on parameters, a low deadline indicates latency-critical, a high data size indicates throughput-intensive, and a flexible deadline indicates the batch that influences the initial VM assignment and QoS weight importance. Each VM is defined as $vm_j = (MIPS_j, BW_j, RAM_j, cost_j)$. Here, $MIPS_j$ indicates processing speed, BW_j indicates bandwidth, RAM_j memory capacity and $cost_j$ indicates cost per unit execution time. The system model formulates the process as a mapping between heterogeneous educational workloads and distributed multi-cloud resources [24].

B. QoS Metrics

The proposed framework optimizes the scheduling process using these QoS metrics in a balanced manner. The Makespan represents the total completion time of all scheduled tasks, indicating the overall execution efficiency of the system. It is defined as the maximum finishing time among all VMs $\left(FT_{vm_j}\right)$ and it is defined as:

$$Mks = \max_{vm_j \in VM} \left(FT_{vm_j}\right) \quad (1)$$

Execution cost measures the total monetary expense incurred for executing all tasks across multiple cloud resources, and it is calculated as:

$$cost = \sum_{i=1}^{n} cost_j \times ET\left(t_i, vm_j\right) \quad (2)$$

Here, $cost_j$ indicates the cost per unit time of the VM, and $ET\left(t_i, vm_j\right)$ indicates the execution time of the task on the VM. The SLA violation rate quantifies the degree to which tasks fail to meet their deadlines, reflecting system reliability and service quality, and it is defined as:

$$SLA = \frac{|\{sT_i \mid CT_i > d_i\}|}{n} \quad (3)$$

Here, CT_i indicates the completion time of tasks, and n indicates the total number of tasks. Load imbalance measures the degree of uneven workload distribution across VMs, and it is given as:

$$LI = \frac{\max(Util_j) - \min(Util_j)}{\text{mean}(Util_j)} \quad (4)$$

Here, $Util_j$ indicates the utilization of the VM. Resource utilization reflects the efficiency of resource usage in the multi-cloud environment, and it is defined as:

$$RU = \frac{\sum_{j=1}^{m} used\ MIPS_j}{\sum_{j=1}^{m} Total\ MIPS_j} \quad (5)$$

Here, $used\ MIPS_j$ indicates computational capacity actively utilized, and $total\ MIPS_j$ indicates total available computational capacity.

C. Multi-Objective Optimization Problem

Given the heterogeneous educational workloads and the conflicting QoS requirements, the task scheduling problem is formulated as a multi-objective optimization problem, in which multiple performance metrics are optimized under system constraints [25], [26]. Let σ denote a candidate scheduling solution (mapping of tasks to VMs). In the task assignment constraint, each task is assigned to exactly one VM, and it is given as $\sum_{j=1}^{m} x_{ij} = 1, \forall sT_i \in T$. Here, $x_{ij} = 1$ if task is assigned to VM. The total resource demand must not exceed VM capacity, and it is represented as $\sum_{sT_i \in VM_j} \leq RAM_j$. In the deadline constraint, tasks should ideally complete within their deadline, and it is given as $CT_i < d_i, \forall sT_i \in T$. In the execution feasibility constraint, only feasible mappings are allowed, and it is represented as $ET\left(sT_i, vm_j\right) > 0$, and this ensures valid execution based on VM capability. The multiple objectives are optimized simultaneously, and this solution is defined in terms of Pareto optimality. A solution(σ_1) is said to dominate another solution(σ_2) and it is given as $\forall k: f_k(\sigma_1) \leq f_k(\sigma_2)$ and $\exists k: f_k(\sigma_1) < f_k(\sigma_2)$. The set of all non-dominated solutions is called the Pareto optimal set, and it is given as $p^* = \{\sigma \mid \nexists \sigma'\ such\ that\ \sigma' \prec \sigma\}$. From the Pareto set, a final solution is selected using a decision-making method, and it is given as $\sigma^* = \arg\min_{\sigma \in p^*} F_w(\sigma)$. This process ensures balanced QoS performance and context-aware scheduling.

IV. Proposed QoS-AMSO Framework

The proposed QoS-AMSO framework is a layered, end-to-end scheduling architecture that transforms AI-driven educational workloads into optimized task–resource mappings in a multi-cloud environment. It consists of five layers: workload ingestion and pre-processing, QoS-aware task classification, multi-cloud resource abstraction, QoS-AMSO optimization engine, and decision execution and monitoring. The workload ingestion layer collects heterogeneous tasks and extracts attributes such as computational size, deadline sensitivity, data volume, and priority, and converts them into a structured format. The QoS-aware classification layer categorizes tasks into latency-critical, throughput-intensive, deadline-soft, and batch-deferred classes that derive indicators. The resource abstraction layer models distributed cloud infrastructure by representing heterogeneous VMs with varying processing capabilities, costs, and dynamic states such as load and availability [27]. The QoS-AMSO optimization engine performs scheduling by initializing solutions using class-aware heuristics and iteratively refines through adaptive step sizing,

QoS-driven weight adjustment, and Lévy flight-based exploration. The execution and monitoring layer selects the optimal solution from the Pareto set, assigns tasks to VMs across cloud providers, and continuously monitors execution performance.

A. Overview of Mirage Search Optimization

Mirage Search Optimization (MSO) is a physics-inspired metaheuristic algorithm based on the optical phenomenon of mirage formation. The MSO mimics two phenomena, including superior mirage (global exploration) and inferior mirage (local exploitation). Each candidate solution is treated as an observation point (individual) in a multi-dimensional search space. Each solution is randomly initialized within bounds, and it is formulated as

$$x_u = lb_u + r.(ub_u - lb_u) \quad (6)$$

Here, ub_u, lb_u indicates lower and upper bounds, and $r \in [0,1]$ indicates a random number. This process ensures a diverse initial population. Superior mirage allows observing distant objects, modeled as a long-range search capability. The position update (x_{uv}^{t+1}) is formulated as:

$$x_{uv}^{t+1} = x_{uv}^{t} + \Delta x_{lower} \quad (7)$$

The inferior mirage phenomenon occurs due to refraction of light, and it is modelled to perform localized search around promising solutions, focusing on refining (fine-tuning) solutions near the global best. The distance and direction are modeled when the current solution is not the best, and it is formulated as

$$h_{uv} = \left|gbest_j - x_{uv}\right|.rand, D = \frac{(gbest_j - x_{uv}).rand}{h_{uv}} \quad (8)$$

The inferior mirage uses optical refraction principles to control movement direction. The angle (γ) gamma, angle (φ) and refraction angle ω (omega), and this process is formulated as

$$\gamma = \frac{\pi(t_{max} - t)}{2t_{max}}.rand \quad (9)$$

$$\emptyset = \left(\arctan\left(\frac{1}{2\tan\gamma}\right) - \arctan\left(\frac{\sin\gamma\cos\gamma}{1+\sin^2\gamma}\right)\right).rand + \arctan\left(\frac{\sin\gamma\cos\gamma}{1+\sin^2\gamma}\right) \quad (10)$$

$$\omega = \arcsin\left(\frac{n_2}{n_1}\sin(\emptyset + \gamma)\right) \quad (11)$$

The displacement calculation uses all geometric relations; the movement step is derived as:

$$\Delta x_{upper} = \frac{h}{\tan\gamma}\left(\frac{h}{\sin\gamma} - \frac{h\sin\emptyset}{\cos(\emptyset+\gamma)}\right).\frac{\cos\omega}{\cos(\omega-\gamma)} \quad (12)$$

This equation combines distance (h), angle $(\gamma, \varphi, \omega)$ and refraction behaviour. The final update rule is formulated as:

$$x_{uv}^{t+1} = x_{uv}^{t} + D.\Delta x_{upper} \quad (13)$$

The MSO employs a physics-driven search paradigm that combines superior mirage-based long-range exploration with inferior mirage-based fine-grained exploitation, allowing the algorithm to dynamically adapt its search behavior and improve solution quality.

B. Limitations of Standard MSO

MSO demonstrates effective exploration and exploitation through its reflection and refraction mechanisms; it exhibits several limitations when applied to QoS-driven task scheduling in multi-cloud environments. MSO is designed as a single-objective optimization algorithm, which restricts its ability to simultaneously handle multiple conflicting QoS metrics such as makespan, cost, SLA violation, and load balancing. The algorithm lacks QoS awareness, as it does not incorporate task-specific requirements such as deadlines, priorities, or workload characteristics, leading to suboptimal scheduling decisions in heterogeneous environments. MSO does not include a structured mechanism for maintaining multiple optimal solutions, limiting its capability to achieve Pareto-optimal trade-offs. To address these limitations, there is a need for an enhanced optimization framework that integrates multi-objective optimization, QoS awareness, adaptive parameter control, and task-specific intelligence.

C. Proposed QoS-AMSO Algorithm

1) Adaptive step sizing: In the standard MSO, the movement of individuals during both superior and inferior mirages is governed by fixed geometric step formulations. These formulations lack dynamic control over step magnitude. To overcome this limitation, the QoS-AMSO introduces an adaptive step sizing mechanism, which dynamically adjusts the movement step of individuals across iterations. Educational cloud workloads exhibit varying search characteristics during different optimization stages. Large search steps are beneficial during the early iterations to explore diverse scheduling solutions, and smaller steps are required in later iterations to refine promising task-to-resource mappings. A fixed movement strategy in the standard MSO cannot effectively accommodate this transition, leading to slow convergence and premature stagnation. The adaptive step size is introduced as a scaling factor $\alpha(t)$ and it is defined as:

$$\alpha(t) = \alpha_{max} - \left(\frac{t}{t_{max}}\right)(\alpha_{max} - \alpha_{min}) \quad (14)$$

Here, t indicates the current iteration, t_{max} indicates the maximum iterations, α_{max} indicates the initial large step size, and α_{min} indicates a small step size. The adaptive step size process is embedded into the MSO movement equations. The original MSO position update from Eq. (11) is modified with an adaptive step and formulated as:

$$x_{uv}^{t+1} = x_{uv}^{t} + \alpha(t) + \Delta x_{lower}, x_{uv}^{t+1}, \; = x_{uv}^{t} + \alpha(t).D.\Delta x_{upper} \quad (15)$$

This process scales step size dynamically with large jumps and controlled movement. The adaptive step sizing controls search intensity dynamically, improves convergence speed, enhances solution stability, and supports multi-objective balancing.

2) QoS feedback mechanism: The QoS feedback mechanism introduces a feedback-driven weighting mechanism that monitors QoS metrics, adjusts their importance dynamically, and guides the search toward better

solutions. The weighted fitness function is formulated as $F(\sigma) = \sum_{k=1}^{4} w_k(t).f_k(\sigma)$. Here, $w_k(t) = 1$ indicates the dynamic weight of k^{th} QoS metric. Static objective weights cannot effectively respond to runtime variations, causing optimization to focus on outdated scheduling priorities. Therefore, a QoS feedback mechanism is introduced in which weights are updated based on QoS deviation. For SLA violations, the SLA-specific feedback and it is formulated as:

$$w_{SLA}(t+1) = w_{SLA}(t) + \eta.SLA(t) \quad (16)$$

Here, η indicates the learning rate. This algorithm focuses on prioritizing deadlines. During fitness evaluation, this process can be integrated as:

$$F(\sigma, t) = \sum w_k(t)\,.\,f_k(\sigma) \quad (17)$$

The QoS feedback mechanism enhances the MSO framework by dynamically adjusting the importance of multiple QoS objectives based on real-time system performance.

3) Lévy flight strategy: Metaheuristic optimization algorithms suffer from premature convergence when candidate solutions become concentrated around local optima, with numerous feasible task-resource mappings. The proposed QoS-AMSO incorporates a Lévy Flight Strategy, which introduces random long-distance perturbations into the search process. The step size of the Lévy flight is defined as $L\sim levy\ (\beta)$, here $\beta \in (1,3)$ indicates the Lévy distribution parameter. Here, λ indicates the scaling factor, and $gbest$ indicates the global best solution. The Lévy flight is integrated with superior mirage and inferior mirage, and it is formulated as:

$$x^{t+1} = x^t + \alpha(t).\Delta x_{lower} + \lambda.L \quad (18)$$

$$x^{t+1} = x^t + \alpha(t).D.\Delta x_{lower} + \lambda.L.(x^t - gbest) \quad (19)$$

The QoS-AMSO Optimization Engine, Lévy flight enhances search diversity, prevents premature convergence, improves global optimality, and supports multi-objective exploration.

4) Pareto archive mechanism: Cloud task scheduling simultaneously optimizes multiple conflicting QoS objectives, including makespan, execution cost, SLA satisfaction, and load balancing. The proposed QoS-AMSO introduces a Pareto Archive Mechanism, which maintains a set of non-dominated solutions throughout the optimization process. A solution $(\sigma_a) \leq f_k(\sigma_b)$ and $\exists k: f_k(\sigma_a) < f_k(\sigma_b)$. At iteration $t = 0$, the archive is initialized as $\mathcal{A}(0) = \{\sigma_i\}$ is non-dominated in the population. The archive is updated as $\mathcal{A}' = \mathcal{A}(t) \cup \{\sigma_1^{t+1}, \sigma_2^{t+1}, \dots, \sigma_n^{t+1}\}$. The non-dominated sorting and it is formulated as $\mathcal{A}(t+1) = \{\sigma \in \mathcal{A}' | \nexists \sigma' \in \mathcal{A}' : \sigma' \prec \sigma\}$. To avoid excessive archive growth, a maximum size $(\mathcal{A}_{max})$ is defined as $|\mathcal{A}(t+1)| > \mathcal{A}_{max}$. The MSO update equations for superior and inferior mirage are defined as:

$$x^{t+1} = x^t + \alpha(t).D.\Delta x(\sigma_{archive}) \quad (20)$$

Here, $\sigma_{archive}$ indicates solutions selected from the Pareto set. The Pareto Archive Mechanism enhances the QoS-AMSO framework by maintaining a set of non-dominated solutions that capture optimal trade-offs among multiple QoS objectives.

D. Task Classification Model

1) Workload categories for AI-educational systems: The proposed QoS-AMSO framework classifies tasks into four distinct workload categories: latency-critical (LC), throughput-Intensive (TI), deadline-Soft (DS), and batch-deferred (BD). Each task sT_i is represented as a multi-dimensional feature vector that includes task length (tL_i), deadline (d_i), priority level (pL_i), memory requirement (m_i) and data size (ds_i). For latency-critical workloads, tasks require immediate response and strict deadline adherence. For example, LLM-based real-time Q&A systems live virtual laboratories and interactive tutoring sessions. The mathematical condition is formulated as $C_i = LC$, if $DTR_i < \theta_1$ and $p_i = high$. For throughput-intensive tasks, the mathematical condition is formulated as $C_i = TI$, if $ds_i < \theta_2$ and $tL_i =\gg 0$. In deadline-soft workloads, tasks with moderate deadlines have slight delays; the mathematical condition is formulated as $C_i = DS$, if $\theta_1 < DTR_i \leq \theta_3$. In batch-deferred workloads, the mathematical condition is formulated as $C_i = BD$, if $DTR_i > \theta_3$. This classification enables task-aware scheduling decisions, priority-based VM allocation, and QoS-driven optimization guidance. By guiding population initialization and dynamic QoS weight adaptation, task classification improves the quality of initial solutions and enhances the final scheduling performance.

2) Feature vector and classification: The proposed QoS-AMSO framework introduces a feature vector-based representation, followed by a classification mechanism that assigns each task to a QoS-aware category. This transformation enables the scheduler to incorporate task-level intelligence into the optimization process $F_i = [tL_i, d_i, pL_i, m_i, ds_i, DTR_i]$. In the task classification model using the feature vector, each task is assigned to a class, and it is formulated as $C_i = f(F_i)$. The classification output (C_i) is integrated into the optimization process at multiple levels. The fitness function enhancement is formulated as:

$$F(\sigma) = \sum_{i=1}^{n} W_{class}(C_i).f_i(\sigma) \quad (21)$$

The feature vector and classification enable context-aware decision-making, priority-based scheduling, efficient resource mapping, and improved SLA satisfaction.

3) Impact on AMSO initialization and weight setting: The proposed framework leverages task class information to guide the initial population generation and dynamically adjust QoS priorities during optimization. In the proposed class-Guided initialization, each task sT_i is assigned to a VM based on its class C_i and it is formulated as $\sigma_i^{(0)} = \arg\max_{vm_j \in VM} S(C_i, vm_j)$. Here, $\sigma_i^{(0)}$ indicates the initial mapping of tasks, $S(C_i, vm_j)$ indicates the suitability score. In the dynamic QoS weight

adjustment, the weights evolve using optimization, and it is given as:

$$w_k(t+1) = w_k(t) + \eta.\Delta_k(t) \quad (22)$$

$$\Delta_k(t) = \frac{f_k(t) - f_k^{target}}{f_k^{target}} \quad (23)$$

The task classification into AMSO initialization and weight setting enhances the optimization process by guiding initial solution generation and dynamically adapting QoS priorities based on workload characteristics. Algorithm 1 presents the complete algorithmic workflow of the proposed QoS-AMSO framework, which integrates task classification, adaptive optimization, and multi-objective decision-making into a unified scheduling process.

Algorithm 1: QoS-AMSO for Multi-Cloud Scheduling

Step 1: Workload Pre-processing and Feature Extraction
- Collect incoming tasks from AI-based educational applications.
- Extract task parameters and normalize features to ensure balanced representation.

Step 2: Task Classification
- Compute derived indicators and classify each task.

Step 3: Initialize Population (Class-Aware Initialization)
- Generate initial candidate scheduling solutions.
- Assign tasks to VMs based on task class.
  - High-performance resources for latency-critical tasks
  - Parallel resources for throughput-intensive tasks
  - Balanced resources for deadline-soft tasks
  - Low-cost resources for batch tasks

Step 4: Update Candidate Solutions (MSO Search)
- For each solution in the population:
  - Apply superior and inferior mirage strategies for exploration and exploitation.
  - Adjust movement using adaptive step sizing.
  - Modify search behavior based on iteration progress.
  - Introduce Lévy Flight Perturbation to selected solutions.
  - Combine current solutions with archive and perform non-dominated sorting.
  - Select final scheduling solution based on QoS preference.

Step 5: Task Execution and Monitoring
- Deploy selected scheduling plan.
- Assign tasks to respective VMs and monitor execution performance.

End

4) Computational complexity: The computational complexity of the proposed QoS-AMSO framework is analyzed to evaluate its scalability and efficiency in handling large-scale, heterogeneous multi-cloud scheduling problems. Let n indicate the number of tasks, M indicate the number, vm indicate the number of VMs, P indicate the population size, t indicate the number of iterations, and K indicate the number of QoS objectives. In task classification complexity, the complexity is defined as $O(n)$. For each iteration, all candidate solutions are updated, and it is formulated as $O(t.P.n)$. Each solution is evaluated across all QoS objectives, and it is given as $\left(O(t.P.n.K)\right)$. In the Pareto archive update, Non-dominated sorting is performed over the population and archive, and it is defined $\left(O(T.P^2)\right)$. The overall complexity is defined as:

$$TC = O(n) + O(P.n) + O(T.P.n) + O(T.P^2) \quad (24)$$

The proposed QoS-AMSO framework is computationally efficient due to linear scalability with task size, controlled population size limiting quadratic overhead.

V. Experimental Results and Discussion

A. Simulation Environment

To evaluate the effectiveness of the proposed QoS-AMSO framework, a simulation environment is replicated to replicate realistic multi-cloud conditions for educational workloads. Synthetic MOOC-based workload patterns are artificially generated workloads that mimic real MOOC platforms to simulate diverse educational task behaviors [25]. The synthetic workload is generated by modelling common educational activities observed in AI-enabled learning platforms, including video streaming sessions, online quizzes, assignment submissions, real-time tutoring requests, and learning analytics jobs. Task arrival times are generated using variable arrival rates to simulate normal and peak learning periods; computational demand, data size, and deadline constraints are assigned according to the characteristics of each educational activity.

A real-world workload dataset derived from Google cluster logs, modified to reflect educational task characteristics, was filtered based on computational intensity and execution duration. The original trace attributes, including CPU utilization, memory usage, task execution time, and arrival patterns, were retained, and application labels were mapped to representative educational workload categories. Interactive services are modelled as latency-critical tasks, large-scale assessment and analytics jobs as throughput-intensive tasks, routine learning activities as deadline-soft tasks, and background reporting processes as batch-deferred tasks. Table II illustrates the environmental settings.

TABLE II. Environmental Settings

Category	Parameter	Value
Simulation Tool	Platform	CloudSim 3.0
Cloud Infrastructure	Number of Cloud Providers	3
	Total Virtual Machines	60
	VM Distribution	20 per provider
VM Configuration	MIPS Levels	1000, 2000, 3000
	RAM	8 GB
	Bandwidth	100–1000 Mbps
	Storage	10–100 GB
Network Settings	Latency Range	5 ms – 80 ms
Workload	Task Count	100 – 1000 tasks
	Data Size	10 MB – 5 GB
	Priority Levels	Low, Medium, High
Optimization Parameters	Population Size (P)	50
	Max Iterations (T)	100
	Lévy Flight Parameter	1.5
	Archive Size	50 – 100 solutions

B. Workload Description

The workloads are structured to reflect realistic operational settings, ranging from small-scale deployments to large-scale distributed systems with dynamic load variations. The three workload scenarios are defined as small-scale (S1), medium-scale (S2), and large-scale dynamic environment (S3). Table III illustrates the workload scenario configuration.

TABLE III. WORKLOAD SCENARIOS CONFIGURATION

Scenario	Number of Tasks	Number of VMs	Scale	Workload Behavior	Special Condition
S1	100	20	Small-scale	Stable	No burst
S2	500	40	Medium-scale	Moderately dynamic	No burst
S3	1000	60	Large-scale	Highly dynamic	Burst injection at 50% time

C. Baseline Methods

To validate the effectiveness of the proposed QoS-AMSO framework, its performance is compared against a diverse set of baseline algorithms representing classical scheduling strategies and state-of-the-art metaheuristic optimization methods. The methods include Round Robin (RR), Min-Min, PSO, Grey Wolf Optimizer (GWO), Whale Optimization Algorithm (WOA), and Vanilla MSO. The RR algorithm is selected as a classical baseline due to its simple cyclic task assignment, highlighting the limitations of non-QoS-aware scheduling. The Min-Min algorithm is included as a heuristic that prioritizes tasks with minimum completion time. The PSO is chosen for its strong global search capability and popularity in scheduling problems. The MSO is used to evaluate the improvements achieved through adaptive step sizing, QoS feedback mechanisms, Lévy flight integration, and Pareto-based optimization.

D. Performance Metrics

To comprehensively evaluate the effectiveness of the QoS-AMSO framework, a set of performance metrics is selected. The metrics include Makespan (s), Total Cost (USD), SLA Violation Rate (%), Load Imbalance Index (LII), Resource Utilization (RU) (%), and Convergence speed.

E. Parameter Settings

The performance of the QoS-AMSO algorithm has hyperparameters that control the search behavior and convergence speed. Table IV illustrates the parameter settings.

TABLE IV. PARAMETER SETTINGS

Parameter	Standard value	Optimized value
Population Size (P)	20–30	50
Max Iterations (T)	100–150	100
Lévy Exponent (λ)	1.0–1.2	1.5
Archive Size	10–20	30
Initial Step Size (α_0)	0.5–0.7	0.9
Feedback Gain (γ)	0.01–0.03	0.05
Perturbation Interval	5–8	10

The optimization parameters are selected through preliminary sensitivity analysis by evaluating different parameter combinations under the medium-scale workload scenario (S2). Population size and maximum iteration count are chosen to provide a balance between convergence quality and computational overhead, and the Lévy exponent was selected to maintain sufficient search diversity without introducing excessive random exploration. The archive size and feedback gain are determined to ensure stable Pareto solution maintenance and responsive QoS adaptation throughout the optimization process.

F. Performance Comparison

This section presents the comparative performance analysis of the proposed QoS-AMSO framework against baseline algorithms is illustrated in Table V.

TABLE V. PERFORMANCE COMPARISON WITH BASELINE MODELS

Models	Makespan (s)	Total cost (USD)	SLA (%)	LII	RU (%)
RR	1285	512.4	12.8	0.42	68.3
Min-Min	1042	468.7	9.6	0.35	74.1
PSO	921	552.3	7.8	0.29	79.6
GWO	887	331.5	6.9	0.26	82.4
WOA	865	425.2	6.3	0.24	91.1
MSO	842	418.6	5.7	0.22	86.5
QoS-AMSO	798	292.8	3.9	0.18	97.2

The performance comparison results demonstrate that the proposed QoS-AMSO framework outperforms classical, heuristic, and metaheuristic baseline algorithms. The model attained a 798s makespan and 97.2% RU with reduced cost. The consistent performance improvement of QoS-AMSO over the baseline methods is attributed to the integration of its components. Task classification enables workload-aware resource allocation, adaptive step sizing balances global exploration and local exploitation for faster convergence, QoS feedback dynamically adjusts scheduling decisions according to runtime conditions, Lévy flight improves global search capability by avoiding local optima, and the Pareto archive preserves high-quality multi-objective solutions. The conventional scheduling algorithms optimize limited objectives; the coordinated operation of these mechanisms enables QoS-AMSO to achieve lower makespan and execution cost, fewer SLA violations, improved load balancing, and higher resource utilization across diverse educational workload scenarios.

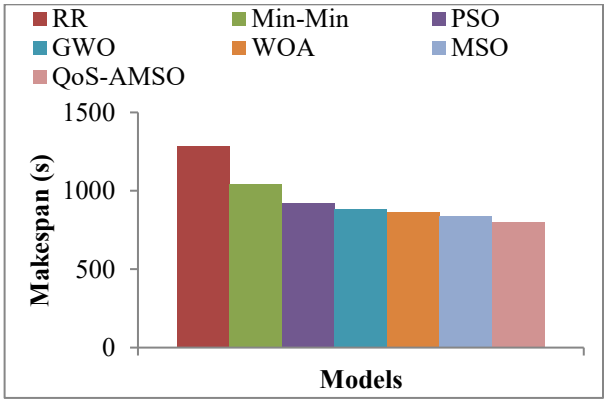


Fig. 1. Makespan graph comparison.

Fig. 1 illustrates a makespan graph comparison. The model attained a makespan of 798s, which is decreased by 5.22% to 37.89% compared to the RR, Min-Min, PSO, GWO, WOA, and MSO models. The QoS-AMSO dynamically adjusts its search process through adaptive step sizing, which enables control over exploration and exploitation during stages of optimization.

Fig. 2 illustrates the total cost graph comparison. The model attained a total cost of 292.8USD that is decreased by 10.04% to 41.08% compared to the RR, Min-Min, PSO, GWO, WOA, and MSO models. The QoS feedback mechanism allows the algorithm to adapt to runtime performance deviations.

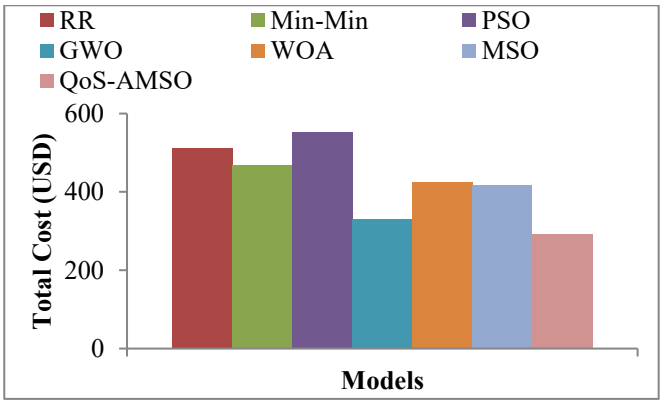


Fig. 2. Total cost graph comparison.

Fig. 3 illustrates the SLA violation graph. The model attained an SLA violation of 3.9%, which is decreased by 1.8% and 8.9% compared to the RR, Min-Min, PSO, GWO, WOA, and MSO models. The Lévy flight strategy introduces controlled randomness, enabling the algorithm to escape local optima and explore a solution space.

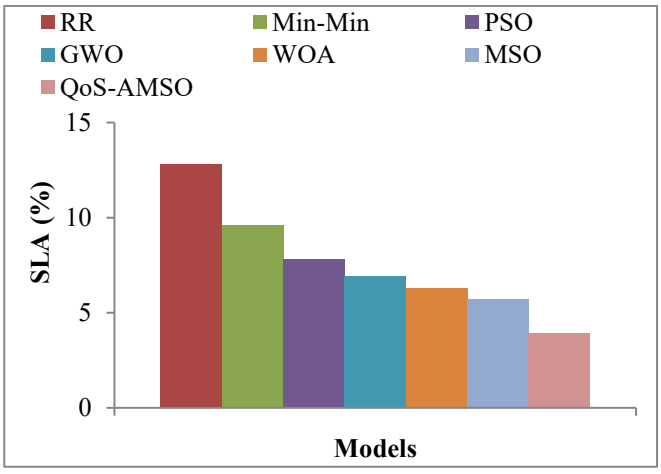


Fig. 3. SLA violation graph comparison.

Fig. 4 illustrates the load imbalance index graph. The model attained an LII of 0.18, which is decreased by 18.18% to 57.14% compared to the RR, Min-Min, PSO, GWO, WOA, and MSO models. The QoS-AMSO converges faster to high-quality solutions while maintaining consistency across different workload scenarios.

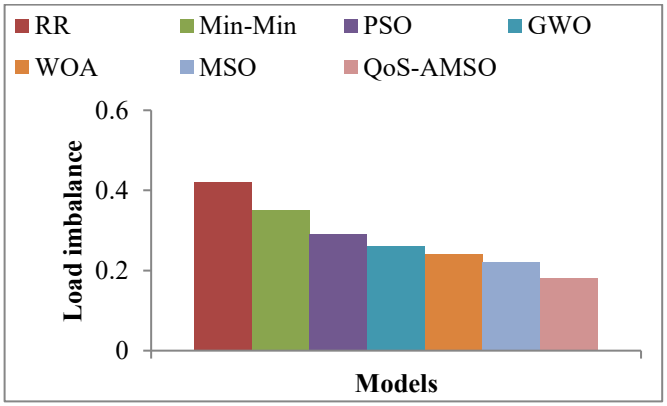


Fig. 4. Load imbalance index graph comparison.

Fig. 5 illustrates the resource utilization graph. The model attained resource utilization of 97.2%, which is increased by 6.1% to 28.9% than the RR, Min-Min, PSO, GWO, WOA, and MSO models. The QoS-AMSO incorporates task classification, enabling the scheduler to differentiate between latency-critical, throughput-intensive, and flexible workloads.

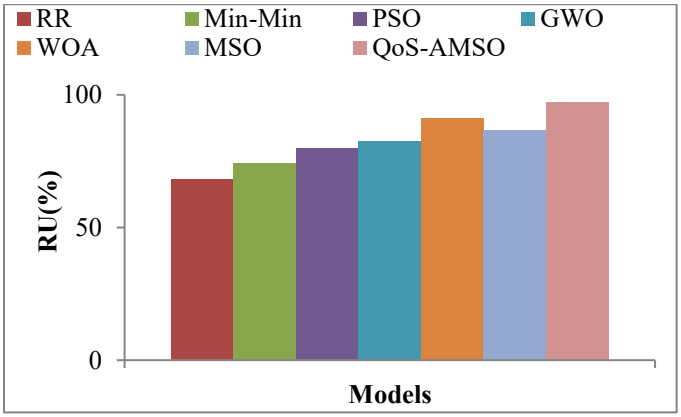
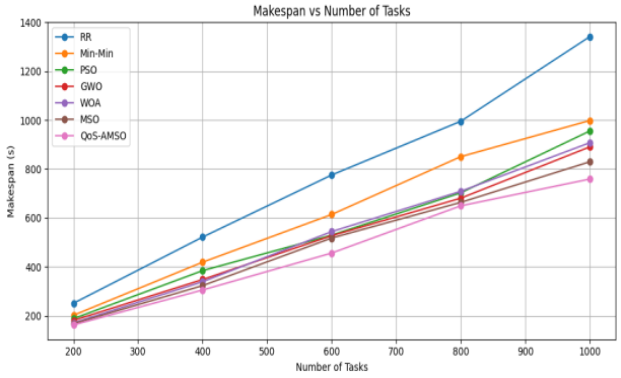


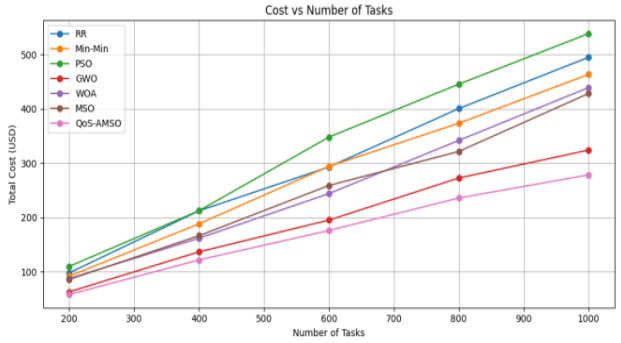
Fig. 5. Resource utilization graph comparison.

G. Scalability Analysis

The graphs in Fig. 6(a) – 6(b) show that QoS-AMSO maintains superior performance with increasing workload size, demonstrating better scalability compared to baseline methods.



(a)



(b)

Fig. 6. (a) Scalability analysis (makespan), (b) Scalability analysis (cost).

H. Convergence Analysis

Fig. 7 shows the plots that illustrate that QoS-AMSO converges faster and more smoothly to optimal solutions than other algorithms across iterations.

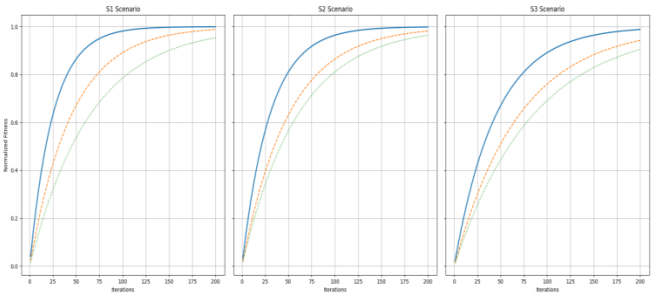


Fig. 7. Convergence analysis.

I. QoS Feedback Mechanism Effectiveness Analysis

This section evaluates how effectively the proposed QoS-AMSO framework adapts to dynamic workload changes using its QoS feedback mechanism under the large-scale scenario (S3) with burst injection. During the experiment, the evolution of the QoS weight vector is tracked across iterations. When a sudden workload spike occurs at 50% of the simulation time, an increase in SLA violations is observed due to resource contention. In response, the feedback mechanism dynamically adjusts the weights by increasing the importance of the SLA component. This reactive adjustment shifts the optimization focus toward meeting deadline constraints, resulting in improved task prioritization for latency-sensitive workloads, a reduction in SLA violations over subsequent iterations, and stabilization of system performance after the burst.

J. Ablation Study

An ablation study is conducted by systematically removing one mechanism, including without Lévy perturbation (LP), adaptive step sizing (AS), QoS feedback weights (QF), and class-aware initialization (CI) modules. Table VI illustrates the ablation study.

When Lévy perturbation is excluded, the makespan increases to 846s, and SLA violations increase to 5.2%, due to the loss of its ability to perform long-distance exploration, leading to localized search behavior. Removing adaptive step sizing degrades performance, with makespan increasing to 872s and cost to 327.9 USD, and fails to balance exploration and exploitation. When the QoS feedback mechanism is removed, makespan increases to 905s, with reduced utilization of 87.2% as the algorithm becomes static and unable to respond to runtime variations. Without class-aware initialization, makespan reaches 931s, and resource utilization drops to 85.6%, due to inefficient initial task-to-resource mappings, leading to poor overall scheduling performance. The complete QoS-AMSO model achieves the best results with a makespan of 798s and resource utilization of 97.2%. The Lévy perturbation, adaptive step sizing, QoS feedback, and class-aware initialization enable balanced exploration and exploitation, and dynamic QoS prioritization.

TABLE VI. ABLATION STUDY

Models	Makespan (s)	Total cost (USD)	SLA (%)	LII	RU (%)
w/o LP	846	318.5	5.2	0.23	91.4
w/o AS	872	327.9	5.8	0.25	89.7
w/o QF	905	341.6	7.1	0.28	87.2
w/o CI	931	356.4	7.9	0.31	85.6
QoS-AMSO	798	292.8	3.9	0.18	97.2

K. Statistical Validation

To ensure that the observed performance improvements of the proposed QoS-AMSO framework are not due to random variations, statistical validation is conducted using the Wilcoxon signed-rank test. Table VII illustrates the Statistical Significance results.

The statistical validation using the Wilcoxon signed-rank test confirms that the performance improvements of the QoS-AMSO framework are statistically significant across all baseline comparisons. The consistently low p-values and large effect sizes demonstrate that the proposed model achieves reliable enhancements in scheduling performance, establishing its robustness and effectiveness for multi-cloud task scheduling.

TABLE VII. STATISTICAL SIGNIFICANCE RESULTS

Comparison	p-value	Effect Size (r)
QoS-AMSO vs RR	0.00012	0.89
QoS-AMSO vs Min-Min	0.00031	0.82
QoS-AMSO vs PSO	0.00105	0.76
QoS-AMSO vs GWO	0.00218	0.71
QoS-AMSO vs WOA	0.00342	0.68
QoS-AMSO vs MSO	0.00487	0.64

L. Discussion

The proposed QoS-AMSO outperforms standard MSO by integrating decision intelligence into the optimization process. While MSO relies on physics-inspired search dynamics, it lacks awareness of workload characteristics and QoS requirements. The QoS-AMSO embeds scheduling context into the optimization loop, enabling the algorithm to focus on meaningful regions of the search space, resulting in stable

convergence and improved solution quality in heterogeneous environments. The Lévy flight mechanism helps maintain population diversity by introducing occasional long jumps, allowing exploration of distant regions and ensuring broader search coverage in complex multi-cloud scenarios. Adaptive step sizing enhances performance by dynamically reducing step size over iterations, enabling fine-grained adjustments near optimal solutions and improving convergence accuracy. In small-scale workloads, the Min-Min algorithm achieves slightly lower cost due to its focus on execution time minimization, but this leads to higher SLA violations and poor load distribution. The QoS-AMSO maintains a balanced multi-objective approach, ensuring system reliability, fairness, and QoS compliance, even if it incurs marginally higher cost in specific cases.

VI. Conclusion and Future Work

The QoS-AMSO framework is developed as a QoS-aware scheduling model for AI-driven educational workloads in multi-cloud environments. The contributions include the integration of task classification, adaptive optimization strategies, QoS feedback mechanisms, Lévy flight-based exploration, and Pareto-based decision-making within a unified architecture. Experimental results demonstrate that the proposed model achieves improved makespan, reduced cost, lower SLA violations, balanced load distribution, and higher resource utilization. Despite its performance, the current framework has certain limitations. The evaluation is conducted in a CloudSim-based simulation environment, which may not fully capture the dynamic behavior, network variability, and resource heterogeneity encountered in real multi-cloud deployments. The framework assumes accurate workload profiling and does not explicitly consider energy efficiency and security policies. Future work will focus on validating the framework in real multi-cloud testbeds, incorporating energy-aware and security-aware scheduling strategies, and integrating edge-cloud environments to improve adaptability for large-scale educational applications.

Funding Statement

This work was supported by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia [Funding No.KFU263786].

Acknowledgment

The authors are thankful to the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, and Al-Fayha Private College, Jubail, Saudi Arabia, for providing financial assistance.

References

[1] N. H. Anh, N. H. "Hybrid cloud migration strategies: balancing flexibility, security, and cost in a multi-cloud environment", *Transactions on Machine Learning, Artificial Intelligence, and Advanced Intelligent Systems*, Vol. 14, No. 10, pp. 14-26, 2024.

[2] W. K. Awad, K. A. Z. Ariffin, M. Z. A. Nazri, and E. T. Yassen, "Resource allocation strategies and task scheduling algorithms for cloud computing". *A systematic literature review. Journal of Intelligent Systems*, Vol. 34, No. 1, 2025.

[3] O. L. Abraham, M. A. B. Ngadi, J. B. M. Sharif, and M. K. M. Sidik, M. K. M. "Multi-objective optimization techniques in cloud task scheduling", *A systematic literature review. IEEE Access,* Vol. 13, pp. 12255-12291, 2025.

[4] K. Y. Rajput, X. Li, J. Zhang, and A. Lakhan, "A novel scheduling approach for Spark workflow tasks with deadline and uncertain performance in multi-cloud networks", *IEEE Transactions on Cloud Computing,* Vol. 12, No. 4, pp. 1145-1157, 2024.

[5] S. Mangalampalli, G. R. Karri, M. V. Ratnamani, S. N. Mohanty, B. A. Jabr, Y. A. Ali, and B. S. Abdullaeva, "Efficient deep reinforcement learning based task scheduler in multi-cloud environment*", Scientific Reports,* Vol. 14, No. 1, pp. 21850, 2024.

[6] S. S. Mangalampalli, G. R. Karri, S. N. Mohanty, S. Ali, M. I. Khan, S. Abdullaev, and S. A. AlQahtani, "Multi-objective Prioritized Task Scheduler using improved Asynchronous Advantage Actor-Critic (a3c) algorithm in multi-cloud environment", *IEEE Access,* vol. 12, pp. 11354-11377, 2024.

[7] M. S. Kumar, and G. R. Kumar, G. R. "EAEFA: an efficient energy-aware task scheduling in cloud environment", *EAI Endorsed Transactions on Scalable Information Systems,* Vol. 11, No. 3, 2024.

[8] R. Sandhu, M. Faiz, H. Kaur, A. Srivastava, and V. Narayan, V. "Enhancement in performance of cloud computing task scheduling using optimization strategies", *Cluster Computing,* Vol. 27, No. 5, pp. 6265-6288, 2024.

[9] B. Acharya, S. Panda, S. Das, S. K. Majhi, V. C. Gerogiannis, and A. Kanavos, "Optimizing task scheduling in cloud environments: a hybrid golden search whale optimization algorithm approach", *Neural Computing and Applications*, pp. 1-23, 2025.

[10] S. Bansal, B. S. Singla, and H. Aggarwal, "Workflow task scheduling in a cloud-fog environment: a hybrid PSO-GOA approach", *International Journal of System Assurance Engineering and Management,* pp. 1-23, 2025.

[11] B. Nagalakshmi, S. Subramanian, "Multi-objective energy aware task scheduling using orthogonal learning particle swarm optimization on cloud environment", *International Journal of Information Technology*, Vol. 17, No. 1, pp. 447-454, 2025.

[12] M. S. Sanaj, M. F. Horng, S. S. Shankar, C. C. Lo, R. Sunder, U. K. Lilhore, and E. S. Ghith, "Energy-Efficient Task Scheduling in Cloud Computing with EcoTaskOpt: A Hybrid Ant Colony and Particle Swarm Optimization Approach", *International Journal of Computational Intelligence Systems*, 2026.

[13] K. A. Tabary, H. Motameni, B. Barzegar, E. Akbari, H. Shirgahi, and M. Mokhtari, "QoS-aware task scheduling using hybrid genetic algorithm in cloud computing", *IEEE Access*, Vol. 13, pp. 51603-51616, 2024.

[14] G. Long, S. Wang, and C. Lv, "QoS-aware resource management in cloud computing based on fuzzy meta-heuristic method", *Cluster Computing*, Vol. 28, No. 4, pp. 276, 2025.

[15] P. Tamilarasu, and G. Singaravel, "Quality of service aware improved coati optimization algorithm for efficient task scheduling in cloud computing environment", *Journal of Engineering Research*, Vol. 12, No. 4, pp. 768-780, 2024.

[16] Y. Sha, H. Wang, D. Wang, and M. Ghobaei-Arani, "A multi-objective QoS-aware IoT service placement mechanism using Teaching Learning-Based Optimization in the fog computing environment", *Neural Computing and Applications*, Vol. 36, No. 7, pp. 3415-3432, 2024.

[17] Z. Xu, Z. Tang, J. Lou, Z. Yao, X. Xie, T. Wang, and W. Jia, "EAT: QoS-Aware Edge-Collaborative AIGC Task Scheduling via Attention-Guided Diffusion Reinforcement Learning", *IEEE Transactions on Mobile Computing*, 2026.

[18] S. Sharma, N. Kumar, Y. Dash, A. Dubey, and K. Devi, "Intelligent multi-cloud orchestration for AI workloads: enhancing performance and reliability", *In 2024 7th International Conference on Contemporary Computing and Informatics (IC3I)*, Vol. 7, pp. 1421-1426. IEEE, 2024.

[19] A. Adeyinka, "Dynamic resource allocation using AI-driven workload forecasting in multi-cloud environments", *World Journal of Advanced Research and Reviews,* Vol. 23, pp. 3188-3198, 2024.

[20] S. R. Bauskar, "Optimizing Multi-Cloud Environments Advanced Database Technologies for Scalable and Resilient Education and Training Systems", *In Integrating AI and Sustainability in Technical and Vocational Education and Training (TVET)*, pp. 189-206. IGI Global Scientific Publishing, 2025.

[21] S. K. R. Jorepalli, "Cloud-native AI applications designing resilient network architectures for scalable AI workloads in smart education", *In Smart Education and Sustainable Learning Environments in Smart Cities,* pp. 155-172. IGI Global Scientific Publishing, 2025.

[22] A. N. Malti, M. Hakem, and B. Benmammar, "A new hybrid multi-objective optimization algorithm for task scheduling in cloud systems", *Cluster Computing,* Vol. 27, No. 3, pp. 2525-2548, 2024.

[23] X. Tan, and X. Zhang, "Data-driven evolutionary algorithm with Pareto set topology learning for multimodal multi-objective optimization", *Knowledge-Based Systems*, pp. 115582, 2026.

[24] H. Abdullah, A. S. Kumar, A. Qasem Ahmed, and M. Saeed Mosleh, "Hybrid Optimization Based on Spectrum Aware Opportunistic Routing for Cognitive Radio Ad Hoc Networks" *Informatics and Automation,* Vol. 22, No. 4, pp. 880-905, 2023.

[25] A. Machado, K. Tenório, M. M. Santos, A, P. Barros, L. Rodrigues, R. F. Mello, and D. Dermeval, "Workload perception in educational resource recommendation supported by artificial intelligence: A controlled experiment with teachers", *Smart Learning Environments*, Vol. 12, No. 1, pp. 20, 2025.

[26] G. Prasad Arun, et al. "Artificial Intelligence in Computer Science: An Overview of Current Trends and Future Directions." Advances in Artificial and Human Intelligence in the Modern Era, edited by S. Suman Rajest et al., IGI Global Scientific Publishing, 2023, pp. 43-60.

[27] P. J, A. V. Senthil Kumar, and H. Mohammed Ali Abdullah, "A New Approach to Improve Energy Consumption Time and Life Time using Energy Based Routing in WSN," 2021 Emerging Trends in Industry 4.0 (ETI 4.0), Raigarh, India, 2021, pp. 1-6.