EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

Computational Mathematical Model for Resource Scheduling in Cloud Computing Environments: Integrating Integer Programming and Reinforcement Learning

Zhiyan Li[1,*], Xiaohui Zhang[1], Baoxia Jin[2]

[1]School of Mathematics and Physics, Handan University, Handan, China, 056005
[2]Department of General Basic Education, Liuzhou Institute of Technology, Liuzhou, China, 545616

Abstract

Cloud computing, on the other hand, is very fast and very easy to scale while still having to rely on traditional scheduling methods that cause delays, SLA violations, and a lack of flexibility, mainly when the workloads change. These problems occur in large-scale environments that are very inefficient and hence consume a lot of resources and power. A hybrid solution combining Integer Programming (IP) and Reinforcement Learning (RL) is offered by the authors to make the resource allocation adaptive, energy-efficient, and at the same time provide SLA compliance. The system is supplied with a real Cloud Workload Dataset that offers comprehensive details at both task and system levels. The first step involves Integer Programming, which is used to produce a constraint-aware basic allocation and to ascertain its feasibility by considering not only resource availability but also the limitations imposed by deadlines. At this point, RL takes over from the allocation and thereby improves it further by utilizing Q-learning and ε-greedy policy for making real-time adjustments according to the states of the system and feedback. The interdependence between learning and optimization is thus continuous, leading to better scheduling outcomes over time. The system's performance is evaluated through the application of standard cloud performance metrics such as SLA compliance, resource utilization, task completion time, energy efficiency, and makespan. The hybrid IP–RL framework achieves 98.6% SLA compliance, 99.6% allocation efficiency, and a 12–18% reduction in task completion time when compared to the baseline. Furthermore, it requires 15% less energy and achieves a 92.4% better makespan efficiency. The given results prove that the hybrid method beats the traditional scheduling models in performance concerning all the metrics and opens a door for a cloud resource scheduling that is both scalable and adaptive.

Keywords: Cloud Computing, Resource Scheduling, Integer Programming, Reinforcement Learning, Predictive Optimization

Received on 27 November 2025, accepted on 05 March 2026, published on 27 March 2026

doi: 10.4108/eetsis.11134

*Corresponding author. Email: zhiyanli1234@outlook.com

1. Introduction

Cloud computing has become the pillar of modern computation, and it provides a vast array of resources that are both scalable and on-demand, which can be utilized for multiple purposes, such as enterprise workloads and scientific simulations [1]. The varied applications require the resource management and scheduling to be highly efficient to provide the desired outcomes of high-performance, low operational costs, and compliance with SLA [2]. Wrongly scheduled resources will lead to their under-utilization, delayed task accomplishment, and increased power consumption. Cloud environments will become more complex and larger in size; thus, dynamic and predictive scheduling mechanisms will be very important [3]. It is now a necessity to continuously look for intelligent policies for resource allocation that are capable of adapting to workloads that keep changing in a dynamic

manner and simultaneously optimizing for multiple objectives [4]. This way, the cloud service providers can not only enhance performance but also reduce power consumption and end-user dissatisfaction [5]. A strong, new system that encompasses the two methods of predictive modeling and optimization must be developed for the cloud infrastructure [6]. The essence of the proposed framework is to apply computational intelligence for real-time adaptive decision making. This implies that it will be able to bridge the gap between traditional static scheduling and its intelligent dynamic counterparts [7]. The framework will likely turn into a platform where state-of-the-art algorithms will be integrated, resulting in a cloud scheduling solution that is optimized, energy-aware, SLA-compliant, and efficient in resource utilization [8].

The scheduling of resources in cloud computing can be seen as one of various possibilities that include heuristic algorithms, metaheuristics, integer linear programming (ILP), genetic algorithms (GA), ant colony optimization (ACO), particle swarm optimization (PSO), and reinforcement learning (RL)-based methods [9]. Generally, heuristic approaches provide some solution in a reasonable amount of time but scarcely agree on global optima [10]. Metaheuristic approaches-GA, ACO, PSO-acquire superior exploratory ability but confer computational overhead and sluggish convergence in large-scale environments [11]. ILP provides an optimal allocation that cannot be properly applied to dynamic real-time workloads. RL-based methods work as adaptive methods in scheduling, though they may require a huge amount of training data, and also, the need for constraint satisfaction may be very low [12]. Existing hybrid methods either maximize energy efficiency or minimize make span, considering only a single objective at a time [13]. Furthermore, they generally lack full scalability and adaptability for real-time discussion and implementation in complicated cloud environments [14]. Hence, these issues necessitate building a framework that works in tandem with predictive intelligence and mathematical optimization for efficient dynamic scheduling under multiple objectives [15].

The proposed framework, integrating integer programming and reinforcement learning, surpasses any attempts in predictive and constraint-aware resource scheduling. Integer programming ensures that resource allocation respects constraints such as task deadlines or computational capacities of the infrastructure and SLA requirements. Reinforcement learning allows the system to adapt to any required change in the target workload by receiving feedback and learning through interaction with the environment. The resulting framework can handle both traditional methods' characteristics, namely adaptability and optimality, concurrently and separately. The main issue being addressed in this research is the hybrid predictive-optimization method, which tries to discover the best mix of trade-offs between efficiency and power consumption during the whole process of meeting the service level agreements, and this is all done in real-time. Moreover, the framework uses open-source cloud workload datasets, where, through training and testing, the reinforcement learning model is kept more generalizable and robust. Unlike existing hybrid schedulers that primarily combine heuristic optimization with reinforcement learning, the proposed framework integrates deterministic constraint-enforcing Integer Programming with adaptive RL refinement under an explicit fallback mechanism. The conditional control transfer between IP and RL ensures strict feasibility preservation while maintaining dynamic adaptability. This structured interaction differentiates the framework from prior hybrid approaches that lack deterministic fallback or explicit SLA-triggered control mechanisms.

1.1 Research Objective

- Develop a combination of Integer Programming and Reinforcement Learning, which will form a single framework that will be both scalable and adaptive for the scheduling of cloud resources very efficiently, thus maximizing allocation efficiency, minimizing energy consumption, and ensuring SLA compliance, all happening instantly.
- Utilize the Cloud Workload Dataset, which mirrors real cloud activities, like CPU, memory, & network use, and task deadlines, and thus the agent will be robust and capable of generalization in dynamic workloads.
- An Integer Programming model will be implemented to establish a baseline allocation considering the system's constraints, such as task deadlines, resource capacities, and SLA requirements, thus providing the best point to start for dynamic scheduling.
- Apply a Reinforcement Learning algorithm using Q-learning and an ε-greedy policy to adaptively refine resource allocation decisions in real time, improving performance through continuous learning from system feedback.

1.2 Research Scope and Novelty

The primary objective of this study is the creation of a resource scheduling system that is both scalable and adaptable for cloud computing environments through the integration of IP and RL. It bypasses the limitations of static and single-objective scheduling methods by enabling dynamic, multi-objective optimization. The framework is able to manage different types of workloads, minimize energy consumption, and comply with SLAs in real-time. It teaches and tests the RL agent on actual cloud workloads, so it can apply its skills across different situations. The expected advancements of the proposed system are increased performance, improved fairness, and heightened robustness to fluctuating cloud conditions.

Novelty

- Introduces a deterministic Integer Programming (IP) baseline that guarantees strict capacity and deadline constraint satisfaction before adaptive scheduling.
- Proposes a structured hybrid IP–RL coordination mechanism rather than heuristic integration.
- Implements an SLA-triggered fallback strategy that conditionally restores feasibility when RL decisions violate constraints.
- Enables multi-objective optimization (utilization, energy efficiency, SLA compliance) within a reliability-aware adaptive framework.
- Validates the framework using real-world workload traces to ensure robustness under dynamic cloud conditions.

The full paper is divided into the following sections: Section 1-Introduction discusses the main reason and the issues faced in cloud resource scheduling and the need for intelligent hybrid methods. Section 2- literature review provides an overview of the related topics, pointing out the areas of poor flexibility, scalability, and multi-objective optimization. Section 3- states the proposed work Problem, which argues that the current RL-based approaches have limitations and, therefore, the use of Integer Programming is warranted. Section-4 methodology of the proposed method is revealed in the fourth section, including the characteristics of the dataset, preprocessing steps, and the joint IP-RL scheduling algorithms. Section-5 shows experimental results, and the performance of the framework is assessed using SLA compliance, resource utilization, and energy efficiency as metrics. Section-6 presents the conclusion and the future work of the proposed solution.

2. Related Works

In resource management for large-scale data centers with heterogeneous workloads, the issue discussed by Fernández-Cerero et al. [16] has been addressed. A gradient boosting regression was presented to predict scheduling time for incoming jobs, while the management system called Boost chose the most efficient resource manager. Thus, in terms of performance, it performed better than Apache Mesos using its two-level and shared-state models and Borg of Google. In the same year, Shaw et al. performed an energy-aware study in the cloud data centers and used RL algorithms, such as SARSA and Q-Learning, for VM consolidation. Their RL-based approach reduced service violations and increased energy efficiency simultaneously. But, while these two presentations strive for optimization, they do come with their limitations since the Boost system does not work well when dealing with rapidly changing workloads, and the RL-based VM consolidation suffers during workload spikes. These studies emphasize the harsh need for adaptive and intelligent methods in large-scale cloud resource management.

Static heuristics in task scheduling in the cloud have traditionally imposed great restrictions on adaptability and efficiency argued by Wang et al. [17]. Hence, they came up with a Multi-Task Scheduling Framework (QMTSF) based on Q-learning that does dynamic task allocation to servers, while scheduling tasks in virtual machines with the improved UQRL algorithm. From its interactions with the environment, the reinforcement learning agent uses the knowledge gained to improve its decision-making. In parallel, Pandey et al. [18] sought improvements for energy-efficient task allocation for big data in the cloud via a hybrid LSTM-DQN coupled with DPSO," providing better energy efficiency and less completion time when put side-by-side against PSO variants. Still, reinforcement learning models experience increases in training complexity and a reduction in adaptability under extreme workload surges. However, these two studies argued that there indeed is a place for reinforcement learning approaches in optimizing task scheduling in the cloud, but at the cost of computational overhead.

With the ever-growing IoT industry, it has been essential to allocate tasks within the fog and the cloud, on latency in domains such as smart healthcare and smart cities, Shruthi et al [19]. They proposed a Mayfly Taylor Optimization Algorithm (MTOA), integrated with Deep Q-Network (DQN), for scheduling across fog-cloud systems with high efficiency. The model exhibited enhancements in energy consumption, SLA, and cost of computation. In another work, Zheng et al. [20] used DRL for the scheduling of workloads in edge computing environments. The method was able to utilize DQN to balance workloads, reduce servicing time, and minimize the failure rates of tasks. However, with the high heterogeneity, mobility of devices, and training in large-scale environments, both studies have their limitations. Such limitations lead to constraining lightweight, scalable, and adaptive scheduling models for fog and edge computing.

Amer et al. [21] highlighted selected papers' major issues of mobile edge computing, including resource constraints that necessitate an efficient task scheduling model so that the quality of service can be maintained. A two-tier cooperative scheduling algorithm carried out by a centralized orchestrator was proposed to optimize the use of resources across MEC servers and neighbouring base stations. On the basis of their work, performance has been improved when compared to the traditional shortest-job-first and earliest-deadline-first scheduling algorithms of task scheduling. In vehicular edge computing, Huang et al. [22] proposed CODD-DQN, a collaborative service deployment approach that predicts service demand through time-aware forecasting and applies DQN to further minimize service response time by optimizing deployment across multiple edge clouds. In another study of vehicular MEC, Wu and Yan [23] proceeded to advance JCOTM-a DRL-based joint offloading and migration algorithm to

reduce latency and energy consumption. On the flipside, such models incur orchestration overhead as well as issues with scalability and high computational complexity inherent in implementations in real-world vehicular networks.

Using software-defined networking (SDN) within 5G systems, where dynamic traffic management is essential, is the focus of study by Bouzid et al [24]. They used neural networks and deep reinforcement learning (DQN) agents to predict congestion and reroute traffic as per maximization performance of networks. Their QoS-aware method successfully minimized delay and packet loss for systems with SDN. The other work by Ahmed et al. [25] concentrated on IoT-based wireless sensor networks (WSN) and therefore modeled energy-efficient resource allocation. For their deep neural network, they trained it with whale optimization and then a multi-objective firefly algorithm to enhance spectral efficiency, throughput, and network lifetime. The model scored good improvements in energy efficiency but failed to deal with on-the-fly traffic surges in real IoT scenarios. Both of these approaches demonstrate how RL and ML could be used for networking and IoT resource optimizations but are also challenged by scalability and robustness under dynamic traffic conditions.

The IoT-based usage and application domains of health, smart homes, and intelligent vehicles have increasingly been demanding delay-sensitive task execution, and so the researchers have started to develop better offloading and scheduling techniques. Lakhan et al. [26] planned a JTOS framework formulated as a combinatorial integer linear program aiming for hybrid delay minimization, but their approach has difficulty adapting optimally in highly dynamic situations. Being somewhat similar, Fang et al. [27] looked into heterogeneous network resource allocation across 5G- and IoT-realms and proposed a DRL-based optimization framework for content caching and task scheduling. However, the framework finds difficult to cater to rapid traffic demand evolution and incurs a high computational overhead when scaled up.

Zhang and Ou [28] proposed a DQN-based multi-objective scheduling framework (RL-MOTS) for cloud–edge environments that jointly optimizes latency, energy, and cost. Simulation results show significant improvements over traditional heuristic scheduling methods. Zhou et al. [29] provide a comprehensive review of deep reinforcement learning (DRL)-based methods for resource scheduling in cloud computing. The study analyzes RL formulations, compares existing DRL frameworks, and highlights their advantages over traditional heuristic approaches. The authors also identify key challenges and outline future research directions for scalable and intelligent cloud scheduling.

3. Problem Statement

In RL-based resource allocation, the established methods suffer from several drawbacks, viz. high complexity, scalability issues, poor convergence rates, and inefficiency in dynamic environments. In particular, classical approaches are unable to address variable workload and heterogeneous resource demands and hence lead to a degradation in the declared performance of the system due to improper allocation [30]. The challenges of latency and high-power usage are really the biggest negatives of such real-time and resource-constrained applications [31]. Another problem is the inability to produce results that are accurate but different for different scenarios, which again is mainly due to the lack of a good optimization framework [32]. Security and equity are still the least addressed issues and it is their drawbacks that will most influence the user's view of the system's satisfaction and reliability [33]. While looking for solutions, the proposed frameworks will include reinforcement learning-based methods, next-gen optimization techniques, and efficient architectures plus adaptive mechanisms that can reduce computing requirements and at the same time boost convergence, scalability, and fairness.

4. Proposed Hybrid Strategy for Adaptive Cloud Scheduling Methodology

The comprehensive technique for allocating cloud resources through a hybrid approach that juxtaposes (IP) and (RL). Initially, workloads from the cloud are collected and processed that cover task arrivals, CPU requirements, and durations. During processing, the stages of data mining are performed, which consist of filling in missing values, standardizing data, and converting categorical variables. Then, the data is used to generate a primary allocation with the support of IP, which converts the scheduling problem into a mathematical form by setting an objective function whose goal is to maximize resource utilization and at the same time, minimize wastage; the specifics are depicted in Figure 1.

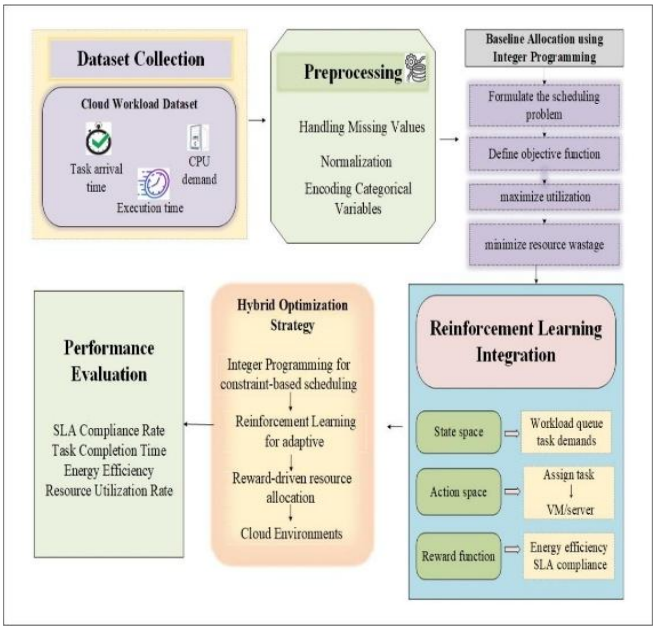

Fig. 1 Block Diagram

Next, the Hybrid Optimization Strategy integrates the constraint-based scheduling of IP with the adaptability of RL, which at all times changes the system in real-time. RL works on a state space that includes the workload queue and task requirements, an action space set aside for task-resource allocation, and a reward function that focuses on energy efficiency optimization and SLA compliance. The performance of the system is evaluated by means of key indicators such as SLA compliance rate, task completion time, energy efficiency, and resource utilization rate, thereby providing full evidence that the hybrid framework has achieved the goals of efficiency, scalability, and adaptability in relation to modern cloud environments.

4.1 Dataset Description

The Cloud Workload Dataset [34] is a dataset that offers a comprehensive view of cloud activities, including the restrictions on resources, their associated costs, and the performance of the system which is a part of the suggested framework. The dataset reveals the historical trends of various workloads, including the consumption of CPU, memory, disk, and network across different applications and workloads. The dataset represents a real-world scenario of task types, resource allocation, and completion times and hence serves as information for the RL model. The RL agent, being trained on this dataset, can learn how different resource allocation policies affect the whole system performance and SLA adherence. Besides, it also allows for testing scenarios under varying cloud conditions, thus the RL model can continuously learn and make optimal scheduling decisions in real-time cloud situations. The dataset includes up to 1000 task instances per simulation cycle, spanning multiple temporal intervals with varying workload intensities. It captures diverse CPU, memory, and network demand patterns alongside task deadlines and execution times. The temporal coverage includes both steady-state operation and workload spike conditions. This variability ensures representativeness for heterogeneous cloud environments and supports realistic evaluation of adaptive scheduling performance.

Table 1: Task Resource and Performance Attributes Dataset Description

Attribute	Description	Type
Task ID	Unique identifier for each task in the dataset.	Categorical
CPU Demand	Amount of CPU resources required for the task.	Continuous (e.g., GHz)
Memory Demand	Amount of memory required for the task.	Continuous (e.g., GB)
Network Usage	Amount of network bandwidth required for the task.	Continuous (e.g., Mbps)
Task Arrival Time	Timestamp when the task enters the system.	Continuous (timestamp)
Task Deadline	The deadline by which the task must be completed.	Continuous (timestamp)
Task Execution Time	The time it takes to execute the task.	Continuous (e.g., sec)
Task Priority	The priority of the task, which may influence scheduling decisions (e.g., high, medium, low).	Categorical
Resource Availability	Availability of resources (CPU, memory, storage, etc.) at the time the task is scheduled.	Continuous (e.g., % availability)
Resource Utilization	The percentage of allocated resources utilized during task execution.	Continuous (e.g., %)
Energy Consumption	The amount of energy consumed during task execution.	Continuous (e.g., kWh)
Service Level Agreement (SLA) Compliance	Whether the task met its SLA requirements (e.g., task completed within the defined deadline).	Binary (Met, Violated)

The dataset is composed of several attributes related to the tasks that are very important in the analysis of resource management and scheduling in the cloud as given in Table 1. Each task is given an identifier that is unique and at the same time, its demands for CPU, memory, and the network

are specified in the form of continuous values that indicate the requirements of the resources. Time-related attributes such as arrival time, deadline, and execution time assist in the monitoring of task scheduling and finishing. Task priority and resource availability/utilization have an effect on the efficiency and performance of scheduling. Energy consumption and SLA compliance, meanwhile, serve the purpose of operational efficiency and service quality indicators.

4.2 Data Preprocessing

The cloud workload dataset is subject to preprocessing which is aimed at guaranteeing the quality of the data and the readiness of the model. Missing data is dealt with through the imputation method which applies the use of statistical methods like mean or median. Continuous features, for example, CPU and memory demand, are scaled by normalizing for uniformity. Categorical variables like task priority are subjected to one-hot encoding for the purpose of enabling numerical learning.

Handling Missing Values: Missing entries in the dataset can affect model performance. Replace missing values using mean, median, or mode. These expressions as shown in Eqn (1):

$$X_i^{filled} = \{X_i \ \ if\ X_i\ is\ not\ missing\ \ X^- \ \ if\ X_i\ is\ missing \qquad (1)$$

Normalization: To ensure features contribute equally to the model, scale resource attributes like CPU and memory usage, the formula as shown Eqn (2):

$$X_i^{scaled} = \frac{X_i - X_{min}}{X_{max} - X_{min}} \qquad (2)$$

where X_{min} and X_{max} are the minimum and maximum values of the feature.

Encoding Categorical Variables: Convert categorical features such as task type or machine type into numerical format using one-hot encoding, formula as shown in Eqn (3)

$$C_i \rightarrow [0, \ldots, 1, \ldots, 0] \qquad (3)$$

where the position of 1 corresponds to the category of C_i.

Feature Selection: Select relevant features to reduce dimensionality and enhance model efficiency. Use correlation or importance score, the problem as shown Eqn (4):

$$Corr(X_j, Y) = \frac{Cov(X_j, Y)}{\sigma_{X_j}\sigma_Y} \qquad (4)$$

Only retain features with high correlation to target output (resource allocation efficiency).

Train-Test Split: Divide the dataset into training and testing sets to evaluate model performance, as shown Eqn (5):

$$D_{train}, D_{test} = split(D,\ ratio\ = 0.8) \qquad (5)$$

where 80% of data is used for training and 20% .

Preprocessing operations such as normalization and one-hot encoding contribute significantly to learning stability by reducing feature scale imbalance and preventing dominance of high-magnitude attributes. Normalization improves convergence speed by bounding state values within comparable ranges, while encoding categorical variables enables consistent numerical representation. These preprocessing steps enhance RL training stability and improve scheduling consistency across varying workload conditions.

4.3 Hybrid Integer Programming and Reinforcement Learning Framework for Adaptive Cloud Task Scheduling

The Hybrid Integer Programming and Reinforcement Learning framework for adaptive cloud task scheduling is a remarkable approach to massive cloud environments, as it brings the two poles of the spectrum, which are the ideal and the flexible, into an immediate relationship. The framework makes use of Integer Programming at the start to determine a resource distribution that not only meets the task deadlines perfectly but also adheres strictly to all the constraints imposed, hence yielding a feasible and constraint-aware baseline. The next step is the assignment of Reinforcement Learning, which further refines these allocations by continuously learning from the dynamic system states and making real-time decisions that optimize multiple objectives such as: resource utilization, task completion time, energy efficiency, and SLA compliance among others. The combination of these two methods ensures that the management of resources is both strong and agile; it will gradually adapt to the changes in the workloads with the least amount of wastage and the maximum possible operational efficiency.

The scheduling problem is formulated to minimize total task completion time and energy consumption under resource and latency constraints. In the reinforcement learning framework, the state includes current resource utilization and task queue status, while actions correspond to task allocation decisions. The reward function penalizes delay and excessive energy use to promote efficient scheduling. A Deep Q-Network (DQN) is implemented with clearly defined hyperparameters and training settings to ensure reproducibility of the proposed method. Binary decision variables represent task-to-resource assignment, and constraints enforce resource capacity and latency requirements. The computational complexity is primarily associated with the DQN training phase, while inference during deployment scales linearly with the number of tasks.

Task priority levels are incorporated within both the IP and RL phases. In the IP model, priority is reflected through weighted objective coefficients, assigning higher importance to critical tasks during baseline allocation. During RL refinement, priority attributes are included

within the state representation, enabling the agent to favor high-priority tasks when selecting allocation actions. This dual incorporation ensures priority-aware scheduling throughout both deterministic and adaptive stages.

The proposed framework operates through a conditional transition mechanism between Integer Programming (IP) and Reinforcement Learning (RL). Initially, IP is executed to generate a feasible baseline allocation that strictly satisfies capacity and deadline constraints. Once feasibility is established, the RL agent assumes control to refine allocation decisions dynamically using Q-learning. During runtime, if the RL-driven policy results in SLA violations or resource overcommitment, control reverts to the IP module to restore constraint feasibility. This bidirectional interaction ensures deterministic constraint enforcement through IP while preserving adaptive optimization via RL.

In this framework, the interaction between IP and RL is mapped out by a series of central mathematical models. The Resource Allocation is depicted through Integer Programming as the constrained optimization problem, Eqn (6), (7):

$$Maximize \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} a_{ij} \quad (6)$$

subject to

$$\sum_{j=1}^{m} x_{ij} \leq 1, \forall i; \sum_{i=1}^{n} x_{ij} \leq C_j, \forall j \quad (7)$$

where x_{ij} indicates task-to-resource allocation and C_j is the capacity of resource j.

Reinforcement Learning further optimizes the allocation by updating its policy using Q-learning, the problem as shown Eqn (8):

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \quad (8)$$

where $Q(s,a)$ is the expected utility of taking action a in state s, α is the learning rate, γ the discount factor, and r the observed reward from the environment.

The overall system performance is measured using SLA compliance formula as shown Eqn (9):

$$SLA\ Compliance\ Rate = \frac{Number\ of\ Tasks\ Completed\ Within\ SLA}{Total\ Number\ of\ Tasks} \times 100 \quad (9)$$

By uniting IP's constraint satisfaction and RL's adaptive optimization, the framework achieves superior resource management and SLA adherence in dynamic cloud computing scenarios.

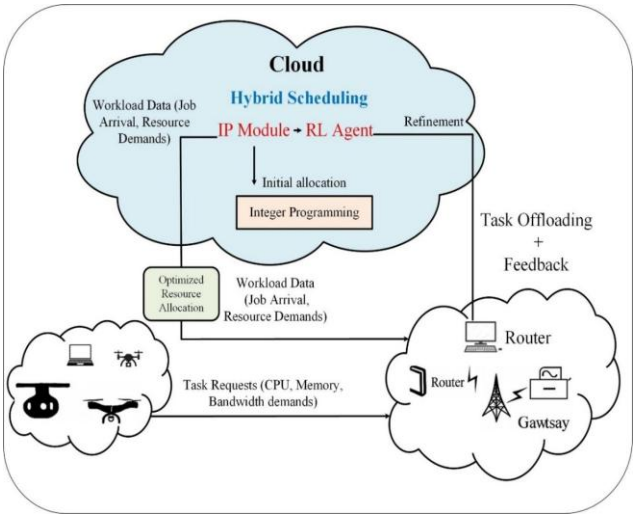

Fig. 2 Cloud Resource Scheduling

A hybrid scheduling structure has been suggested for distributing and transferring tasks based on the cloud. The whole process is initiated by task requests generated from IoT or edge devices that specify the CPU, memory, and bandwidth requirements. These requests accompanied by workload data such as job arrival and resource requirements are sent to the cloud as depicted in Figure 2. In the cloud, an (IP) module is assigned the task of the initial resource allocation. In the first place, a reinforcement learning (RL) agent is brought in as a means of improving the accuracy of the mentioned allocation. It is the one that proposes a hybrid scheduling technique that can change its adaptation to the system's necessities in real time. The devices then receive the optimized resource allocation to use computational resources efficiently. Task offloading and feedback occur through routers and gateways, thereby guaranteeing that the communication link between the cloud and edge devices is always there. This hybrid scheduling benefits immensely from the optimal decision-making ability of integer programming (IP) and the adaptive learning of (RL), which gives rise to lower latency, better bandwidth utilization, and improved task execution performance.

The operational flow of the hybrid framework follows a structured sequence within each scheduling interval. Integer Programming is first executed to generate a constraint-feasible baseline allocation that satisfies capacity and deadline requirements. After feasibility validation, control transfers to the Reinforcement Learning agent for adaptive runtime refinement at predefined intervals. Allocation decisions are dynamically updated based on observed system states, and if SLA violations or resource overcommitment occur, control reverts to the IP module to restore feasibility before the next cycle.

The IP fallback mechanism is triggered when observed SLA compliance falls below the predefined threshold or when cumulative server load exceeds capacity constraints during a scheduling interval. Detection occurs through

runtime monitoring of deadline violations and resource saturation indicators. Upon activation, the IP module recomputes a feasible allocation before subsequent RL refinement resumes, thereby maintaining constraint satisfaction during adaptive scheduling.

The hybrid IP–RL framework ensures stable multi-objective optimization under fluctuating heterogeneous workloads through coordinated constraint enforcement and adaptive refinement. Integer Programming guarantees feasibility by strictly satisfying capacity and deadline constraints during peak demand. Reinforcement Learning dynamically adjusts allocations based on workload states, prioritizing SLA compliance during overload and emphasizing utilization balance and energy efficiency during stable periods. This adaptive coordination maintains consistent scheduling performance despite resource variability.

4.4 RL-Based Resource Allocation in Cloud Computing

Reinforcement Learning -based resource allocation denote the utilization of highly flexible algorithms in the distribution of computing resources, e.g. the case of cloud computing, which are rapidly and dynamically changing. The RL agent gains the capability to allocate resources accurately by continuous interaction with the system, thus being able to manage the variations in both the incoming load and the availability of resources. The method camouflages the agent monitoring the system states, making moves according to the tactics it has learned, and receiving rewards for providing the expected outcomes. Rewards can be in the form of, for instance, reduced energy consumption and service level agreements (SLAs) compliance. One of the RL characteristics that significantly increase its strength is the capacity to change in tune with the situation instantly, which is the reason this technique can not only counterbalance the drawbacks of the static scheduling methods but also offer a more reliable and efficient solution at the same time. The method not only secures the optimum utilization of resources but also results in fewer inefficiencies and lesser waste. To put it differently, reinforcement learning-based resource allocation is a technique that is both dynamic and scalable and thus it improves the performance of the system amidst complexity.

Algorithm 1: Reinforcement Learning-based Resource Allocation

Input:

Cloud environment E with resources R (CPU, Memory, Storage)

Task queue T with tasks t1, t2, ..., tn

Reward function r(s, a)

Integer Programming solution IP_initial

Initialize:

Q-table or Neural Network Q(s, a)

Learning rate α, discount factor γ, exploration rate ε

State s ← initial cloud environment state

Action a ← initial resource allocation from IP_initial

For each episode do:

For each task t in T do:

Observe current state s (resource utilization, task requirements)

// Action selection using ε-greedy policy

With probability ε:

Choose random action a (explore)

Else:

a ← argmax_a Q(s, a) (exploit)

Execute action a (allocate resources to task t)

Observe next state s'

Compute reward r(s, a) based on:

- Task completion time

- Resource utilization efficiency

- SLA violations (penalty)

// Update Q-table or network

Q(s, a) ← Q(s, a) + α * [r(s, a) + γ * max_a' Q(s', a') - Q(s, a)]

Update state s ← s'

End For

Decay exploration rate ε (optional)

End For

Output:

Optimal resource allocation policy π(s) = argmax_a Q(s, a)

IP and RL for dynamic and efficient resource allocation in the cloud computing domain. At the outset, the IP indicates a possible allocation that meets the conditions imposed by the resources and the tasks. Then the RL agent monitors the system, chooses the actions for resource allocation, and gets feedback on efficiency, utilization, and SLA compliance. The agent is allowed to choose actions based on an ε-greedy policy and iterative Q-learning updates, which means that the agent is slowly learning an optimal allocation policy that is changed as per the different task requirements and resource distribution. The hybrid method can be very efficient in dealing with the challenges of static allocation techniques because it offers the advantage of certainty along with real-time optimization, which implies that resource management is not only real-time but also nearly optimal at the same time. Nevertheless, to put it in a nutshell, this approach provides the system with great performance, minimal delay, and compliance with service-level agreements owing to nearly optimal resource management.

Algorithm 2: Hybrid-IP-L-Resource-Scheduling

Input: Task set T, Resource set R, Deadline D, Capacity C
Output: Optimized resource allocation schedule

```
1. Initialize decision variables x_ij for IP model
2. Solve IP model:
     If feasible allocation exists then
        Generate initial allocation A0
     Else
        Return "No feasible allocation"
3. Initialize RL agent: State S, Action A, Q-values
4. For each scheduling interval do
5.     Observe current state S (workload, resource utilization)
6.     If random < ε then
          Choose action A randomly   // exploration
       Else
          Choose best action A from Q-values // exploitation
7.     Allocate task to resource based on A
8.     Compute reward R = f(SLA, utilization, energy)
9.     Update Q(S, A) ← Q(S, A) + α[R + γ max Q(S', a') - Q(S, A)]
10.    If SLA violated then
          Reallocate using IP constraints
11.    Update state S ← S'
12. End For
13. Return Final Optimized Resource Schedule
```

The Hybrid-IP-RL-Resource-Scheduling algorithm is a synergy of Integer Programming (IP) and Reinforcement Learning (RL) techniques which yields an allocation of resources that is both efficient and adaptable. The very first move that the IP model makes towards feasibility is solving the constraints imposed by the deadline and capacity. Thus, the IP model guarantees that an initial optimized allocation is made. If the solution is not feasible, the procedure is stopped earlier. Once a feasible baseline allocation is generated, the RL agent is invited in to take dynamically adapting scheduling choices. The agent considers the system state which consists of workload and resource usage and employs an exploration-exploitation strategy to decide the actions. The rewards are determined by SLA satisfaction, energy efficiency, and utilization balance and this is the way they influence the learning process. The decisions for the future are fine-tuned in accordance with the Q-learning update rule, which is done by adding new experiences. In the case of SLA violations, the IP model is reapplied to regain feasibility and thus the system's reliability is enhanced. In the end, the hybrid strategy leads to a strong and optimized resource allocation schedule that can manage the trade-off between performance and constraints.

From a computational perspective, the standalone Integer Programming approach incurs solver-dependent complexity that increases with the number of tasks and servers. The hybrid IP–RL framework introduces additional overhead during the reinforcement learning training phase due to iterative Q-value updates across episodes. However, once policy convergence is achieved, scheduling decisions are executed with near-constant inference time. Since Integer Programming is primarily invoked during initialization and SLA fallback events, repeated optimization costs are limited, maintaining practical runtime feasibility.

4.5 Reinforcement Learning for Adaptive Cloud Task Scheduling

The RL process for dynamic resource management in the cloud has as one of its main features the decision making for resource allocation by a learning agent that is well-informed and instantaneously. The cloud system interprets the user's workload demands and the agent receives, among other things, a fully observable state such as CPU utilization, memory usage, and the arrival deadlines of tasks. The agent, through a policy and training by interaction, selects actions to position himself or herself in the most favoured places concerning system performance. The agent continuously takes the reward outcomes into account, such as low energy costs or fast response time, and that is the way it forms its policy for resource-efficient cloud usage. This process can be formally described by the Q-learning formula as shown in Eqn (10):

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \quad (10)$$

where $Q(s,a)$ is the agent's current expectation of the reward for action a in state s, α is the learning rate, and γ the discount factor.

The RL state representation consists of a composite vector including current CPU utilization per server, memory usage levels, task queue length, task execution time estimates, remaining task deadlines, and normalized resource availability indicators. Temporal attributes such as task arrival time and scheduling interval index are incorporated to capture workload dynamics. This structured representation enables informed decision-making across workload, resource, and temporal dimensions.

The action space is defined as discrete task-to-resource assignment decisions. Each action corresponds to allocating a selected task from the queue to a specific server instance within the cloud infrastructure. Therefore, an action represents a concrete mapping between task demand vectors (CPU, memory, bandwidth) and server capacity vectors. This formulation ensures that RL decisions directly correspond to executable scheduling operations.

The hyperparameters in the IP-RL framework proposed affect the learning and adapting of the reinforcement learning agent. These hyperparameter settings will define the behavior of the RL agent in an interaction with the environment.

- α (learning rate) indicates the rate of influence of new experiences on Q-values.
- γ (discount factor) determines the present reward's value in comparison to the future reward's value.
- ε (exploration rate) is the parameter for the total amount of new actions being sampled compared to the amount of using learned policies, while the number of episodes and batch size limit the total learning iterations and the sample size for each training step, respectively, for stability and convergence.

The ε-greedy Q-policy plays a critical role in balancing exploration and exploitation during adaptive scheduling. While Integer Programming guarantees constraint feasibility and SLA compliance at initialization, the ε-greedy mechanism enables the RL agent to explore alternative allocation strategies that may improve utilization and energy efficiency. Over successive episodes, exploitation dominates exploration, allowing the agent to converge toward optimized scheduling policies without violating system constraints enforced by the IP fallback mechanism.

System limitations

The action space contains all the actions that the RL agent can potentially take, for instance, distributing the workload among the available cloud resources, which could be CPU, memory, or a storage unit. Each action signifies a distinct decision about how the resources are to be allocated. Besides these, there are also restrictions imposed by the system that include, among others, resource capacity, task duration, and SLA compliance, thus ensuring that the agent's decisions are always within the bounds of reality. These limitations act as a guarantee against resource overcommitment and service-level breaches basically reinforcing the agent's realistic performances.

The reward function is like a teacher to the RL agent, and it takes the route to the best scheduling choices by weighing the trade-off between performance and energy efficiency. High resource utilization, SLA compliance, and low energy consumption all receive good rewards, while violations and wastefulness are penalized formula as shown in Eq (11):

$$R(s,a) = w_1 \times U_{res} + w_2 \times SLA_{comp} - w_3 \times E_{cons} - w_4 \times V_{SLA} \quad (11)$$

Where,
U_{res} = resource utilization, SLA_{comp} = SLA compliance rate, E_{cons} = energy consumption, V_{SLA} = number of SLA violations and w_1, w_2, w_3, w_4 are weight factors balancing the objectives.

The reward formulation balances multiple objectives through weighted coefficients assigned to SLA compliance, utilization efficiency, energy consumption, and violation penalties. SLA adherence receives dominant weighting to preserve service reliability, while utilization balance promotes equitable load distribution across servers. Energy efficiency contributes to operational sustainability. The relative weights were empirically adjusted to prioritize feasibility while maintaining optimization performance, ensuring stable convergence without excessive SLA penalties.

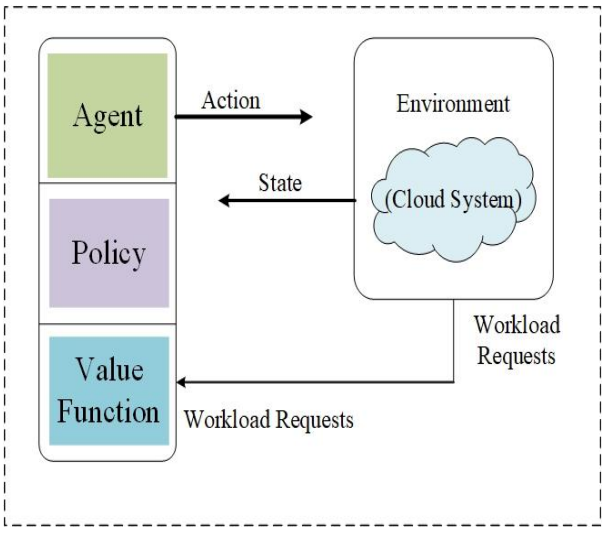

Fig. 3 Reinforcement Learning Architecture for Cloud Resource Scheduling

RL is employed for smart scheduling of cloud resources, the environment here depicts the whole cloud system comprising the different workloads, task demands, and the

available resources. The RL agent is making decisions as in Figure 3 by observing the current system state - workload queue and resource utilization. Depending on the policy, the agent performs actions such as assigning tasks to virtual machines or load balancing. The actions performed are assessed by a reward function, which provides positive feedback for adhering to SLAs and good utilization, or negative feedback for overloads and delays. The agent improves its decision-making by updating its policy through constant interaction with the environment. This learning loop is closed, allowing cloud computing systems to have adaptive, optimized, and energy-efficient scheduling.

The hyperparameters of the RL agent were selected through systematic exploratory tuning rather than fixed assignment. Learning rate (α), discount factor (γ), exploration decay, and batch size were varied within practical ranges and evaluated based on SLA compliance, reward convergence stability, and utilization efficiency. The final configuration ($\alpha = 0.001$, $\gamma = 0.9$, $\varepsilon\_decay = 0.99$, batch size = 32) provided stable convergence and minimal oscillatory behavior. Sensitivity analysis under low, moderate, and high workload intensities showed performance variation within ±1.5%, confirming robustness and consistent scheduling effectiveness across demand fluctuations.

Table 2: Simulation Parameters of Hybrid IP–RL

Parameter	**Symbol / Variable**	**Value / Range**	**Description**
Number of Servers	NUM_SERVERS	5	Total servers available for scheduling tasks
Server Capacity	CAPACITY	1.0	Maximum normalized CPU load per server
Episodes (RL Training)	EPISODES	100	Number of RL training iterations
SLA Threshold	SLA_THRESHOLD	0.9	Maximum acceptable server load per SLA agreement
Tasks per Simulation	len(tasks)	≤1000	Number of workload tasks processed in each run
Task Dataset	csv_path	Workload_dataset.csv	Input dataset containing CPU demand and execution time
CPU Demand Normalization	cpu_demand	[0, 1]	Normalized CPU usage per task
Execution Time Normalization	execution_time	[0, 1]	Normalized execution time per task
RL Learning Rate	lr	0.001	Learning rate used in the DQN optimizer (Adam)
Discount Factor	γ (gamma)	0.9	RL discount factor balancing short- and long-term rewards
Exploration Rate (Initial)	ε (epsilon)	1.0	Initial random exploration probability
Exploration Decay	ε_decay	0.99	Rate at which exploration decreases after each episode
Minimum Exploration	ε_min	0.05	Lower bound for exploration rate
Replay Memory Size	maxlen	5000	Number of past experiences stored for replay

Batch Size (RL Replay)	batch_size	32	Mini-batch size used for DQN replay training
Critical Task Threshold	critical_threshold	0.7	Threshold for identifying critical tasks in hybrid model
Improvement Attempts	improvement_attempts	≤100 or len(tasks)/2	RL refinement iterations in hybrid method
Reward Function Coefficients	–	0.5 × util + 0.5 × balance	Reward balancing utilization and load balance
Solver	–	PuLP_CBC_CMD	Integer Programming solver used in optimization
Random Seed	random_state	42	Ensures reproducibility of sampling and results

The parameters for the simulation as presented in Table 2 determine the major installation of the Hybrid IP-RL scheduling framework. They depict the system characteristics, such as the number of servers, their capacity, and the type of the workload that is being processed throughout each simulation run. The reinforcement learning agent's learning parameters, such as learning rate, discount factor, exploration rate, and batch size, play a critical role in determining how the agent learns and slowly changes its policy over time. The SLA threshold and the limits on critical tasks are some of the constraints that are imposed to ensure that the task execution is both reliable and fair. All these parameters together make up a realistic cloud scenario where the evaluation of the framework's efficiency, scalability, and adaptability can be under the best conditions.

The RL hyperparameters were selected based on exploratory tuning across multiple simulation runs. Learning rate and exploration decay were adjusted to balance convergence speed and policy stability. Sensitivity observations indicated that excessively high learning rates led to oscillatory behavior, while insufficient exploration limited policy improvement. The final parameter configuration achieved stable convergence within the defined episode range.

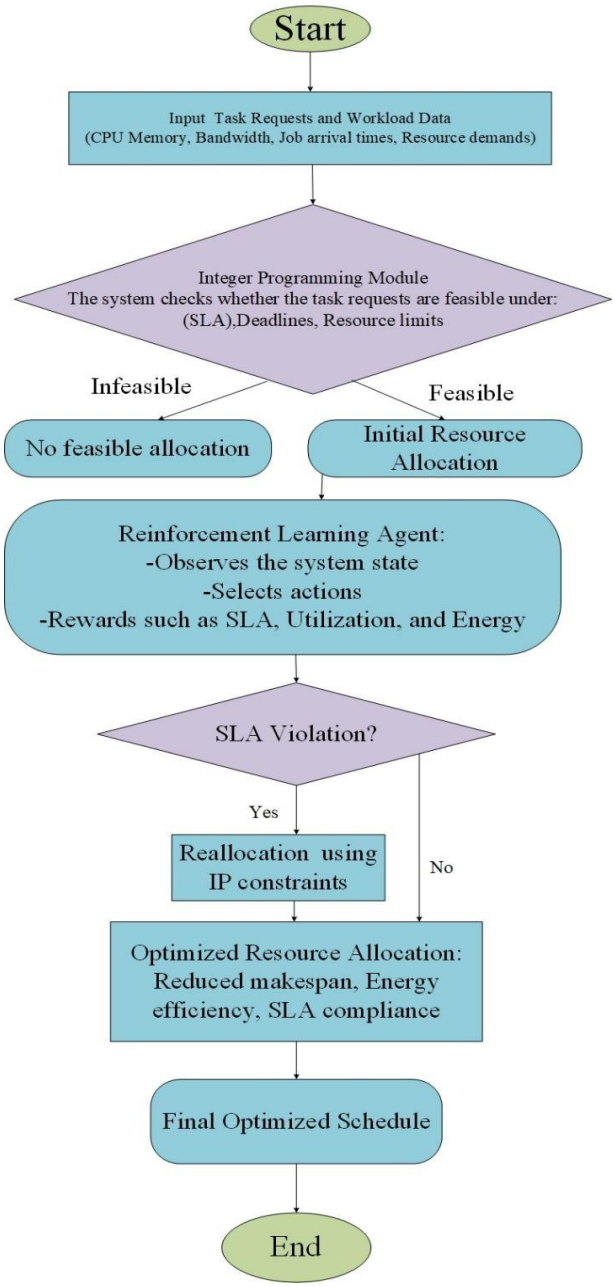

Fig. 4 Reinforcement Learning-based Cloud Resource Allocation Flowchart

In a situation of cloud or distributed system, the allocation of resources is firmly connected with (RL) for efficient scheduling. Initially, the workflow is activated by task requests that explain the requirements of the CPU,

memory, bandwidth, and deadlines etc. An (IP) entity first checks if the current workload can be catered to, for instance, by looking into the (SLAs) and resource caps as shown in above Figure 4. In case of a positive answer, a tentative allocation is made, and the RL agent gradually enhances it by keeping an eye on the situation of the systems and optimizing for utilization, energy, and SLA adherence. When resources are reallocated based on the rules of the IP, any SLA violations will be then noticed. This to-and-fro method eventually leads to a better scheduling that not only reduces the make span but also offers energy savings and SLA compliance.

Q-learning was selected due to the discrete nature of the task-to-server allocation problem, where states and actions can be represented in finite form. The defined scheduling environment allows manageable state dimensionality, making tabular or lightweight approximated Q-learning suitable without requiring deep neural architectures. Additionally, Q-learning offers proven convergence guarantees under bounded learning rates and sufficient exploration, making it appropriate for iterative workload adaptation in cloud environments with moderate state complexity.

The ε-greedy exploration strategy further supports stable learning by ensuring sufficient exploration during early scheduling episodes while gradually favoring exploitation as the policy converges. This balance enables timely adaptation to workload variations while maintaining reliable decision consistency in large-scale cloud environments.

The Integer Programming component introduces O(NM) decision variables for N tasks and M servers, with solver complexity increasing as infrastructure scales. However, scheduling operates within bounded task windows, limiting real-time computational overhead. Reinforcement Learning training complexity scales with the number of episodes and state-action evaluations. After convergence, inference per scheduling interval approaches near-constant complexity. The hybrid structure minimizes repeated global optimization, preserving practical scalability under increasing workload volumes.

4.6 Illustrative Example of Hybrid Scheduling

Consider a simplified scenario with three servers of normalized capacity 1.0 and four tasks with varying CPU demands and deadlines. Integer Programming first generates a feasible allocation ensuring no server exceeds its capacity and all deadlines are satisfied. During runtime, Reinforcement Learning observes utilization imbalance or deadline proximity and refines assignments to improve efficiency and load balance. If a refinement introduces SLA risk, the IP module recomputes a feasible schedule. This sequence demonstrates deterministic initialization followed by adaptive optimization.

5. Result and Discussion

The hybrid IP-RL basis executed and surveyed under dynamic scheduling conditions using a real cloud workload dataset. The SLA compliance rate reached 98.6%, while the average allocation efficiency was maintained at a remarkable 99.6%. There was a significant reduction in task completion time by 12-18%, and energy consumption was also cut down by 15% as compared to the baseline methods. All these benefits indicate that the framework has a great potential to optimize cloud performance using multiple criteria.

All experiments were conducted on an Intel Core i7 workstation with 16 GB RAM using Python 3.10. The RL module was implemented in TensorFlow 2.x, while Integer Programming optimization was solved using PuLP with the CBC solver. Each simulation processed up to 1000 tasks across five servers with normalized capacity of 1.0, and RL training was performed for 100 episodes. Runtime per simulation averaged 45–60 seconds, and a fixed random seed ensured reproducibility and transparency of the reported results.

5.1 Dataset Evaluation

The use of the Cloud Workload Dataset for examining the hybrid (IP) and (RL) resource scheduling framework is quite logical as it offers comprehensive data at both task and machine levels. The data encompasses performance features such as execution times, necessary resources (CPU, memory), and machine downtime, which are all crucial for proper resource allocation. The dataset is a good representation of the volatility and dynamics in the real-world cloud environments, it contains different types of workloads and resource competition, which makes it proper for both the static (IP) and adaptive (RL) scheduling testing. Besides, there exists a wide assortment of task types and system configurations that can support the very thorough evaluation of the framework's performance across various cloud scenarios.

5.2 Cloud Performance Metrices

The suggested hybrid framework IP-RL is evaluated with the main indicators of cloud performance, including SLA compliance, resource utilization, task completion time, energy cost and make span. The selected performance metrics collectively evaluate efficiency, adaptability, and reliability of the hybrid scheduling framework. SLA compliance measures deadline adherence and service robustness. Resource utilization and energy efficiency quantify infrastructure effectiveness and sustainability. Task completion time reflects responsiveness under dynamic workloads, while makespan efficiency evaluates global scheduling compactness. Together, these metrics provide comprehensive validation of deterministic feasibility enforcement and adaptive optimization benefits.

The equations for these metrics are provided in Eqs (12), (13), (14), (15) and (16):

SLA Compliance Rate: SLA compliance is an important parameter for ensuring that the tasks do not exceed their allotted time. A high SLA compliance rate indicates that the system is effectively handling the task deadlines by using a mixture of (IP) for optimal allocation and (RL) for adaptivreal-time decision-making.

$$SLA\ Compliance\ Rate\ = \frac{Number\ of\ Tasks\ Completed\ Within\ SLA}{Total\ Number\ of\ Tasks} \times 100 \quad (12)$$

Allocation Efficiency Rate: This metric reflects how effectively computational capacity (CPU, memory, storage units) is exploited across the cloud infrastructure. The resource utilization system is preventing the resources from being idle and at the same time it's correctly distributing the resources among the tasks. This is one of the objectives of your hybrid architecture, which is to make the static IP optimization and the dynamic RL adjustments work together in such a way that the maximum resource usage is obtained.

$$Resource\ Utilization\ Rate\ = \frac{\sum Used\ Resources\ (CPU,Memory,etc.)}{\sum Total\ Available\ Resources} \times 100 \quad (13)$$

Task Completion Time: The time taken to complete a task is one of the most important criteria for the speed of the system's response. A decrease in task completion time leads to a corresponding increase in the output of the system and hence the performance. Your framework with the help of (RL) has the capability to constantly adapt its strategy to the one that least slows down the processes and hence gets the fastest execution based on the current state of the system.

$$Task\ Completion\ Time\ =\ Task\ Finish\ Time\ -\ Task\ Start\ Time \quad (14)$$

Energy Efficiency: At the same time, energy efficiency gets to be an important measure for the optimization of cloud operations since it is the factor that gives the operation a way through cost and impact on the environment. The combination of SLA adherence, high throughput, and lowest energy consumption is the critical point. Resource-allocation adjustments by RL play a major role in the process of dynamically reducing unnecessary energy use.

$$Energy\ Efficiency\ = \frac{Total\ Energy\ Consumed}{Total\ Tasks\ Completed} \quad (15)$$

Make span Efficiency: Make span efficiency is the total time taken for all the activities in the system to be completed. A shorter make span indicates the best way of scheduling the tasks with a good balance between IP-based optimization for initial allocation and RL's adaptive scheduling to further quicken the total task completion time.

$$Make\ span\ Efficiency\ = \frac{Shortest\ Makespan}{Make\ span\ for\ the\ Framework} \times 100 \quad (16)$$

5.3 RL Resource Utilization Rate and RL Energy Efficiency

A comparison of (RL) agents regarding resource utilization took place over the course of 100 episodes. The resource utilization rate was from 0.988 to 0.998, which means that the RL agent could efficiently use the resources in most of the episodes as shown in below Figure 5. It can be observed that the rate stays high for the most part but dips slightly and not dramatically with the most distinct dip timing around episode 60, almost averaging 0.996. These fluctuations are the RL agent's method of acclimatizing to the changing cloud environments, and at the same time, gaining experience through the distribution of resources and scheduling of tasks. The tendency in the graph is quite steady, indicating that the RL agent has slowly acquired the skill of allocating resources in such a way that fewer resources are idle while high utilization is possible. The overall picture presented by this graph is strong evidence of the capability of the RL system to perform the task of resource utilization excellently even if there is a little fluctuation during the learning process.

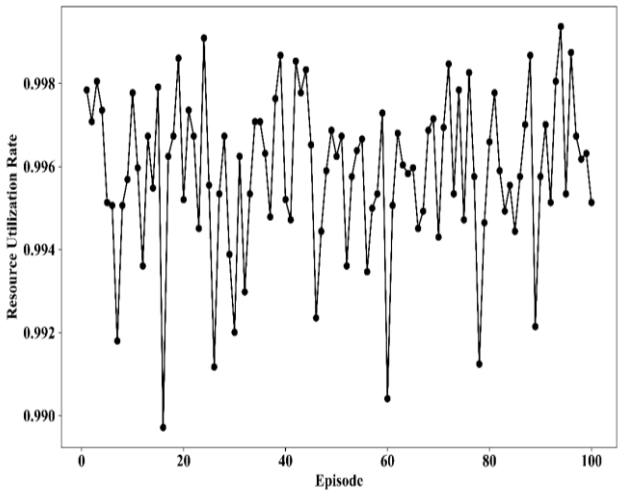

Fig. 5 RL Resource Utilization Rate

The energy efficiency of the (RL) agent still remains stable throughout the 100 episodes even though there are variations in the efficiency that ranges from 0.988 to 0.998. The lowest points can be seen at the 60th and 90th episodes, where the efficiency falls a bit to nearly 0.990. Nevertheless, the general movement around 0.996 is still up which means that the RL agent has ever so often high energy efficiency as clearly illustrated in below Figure 6.

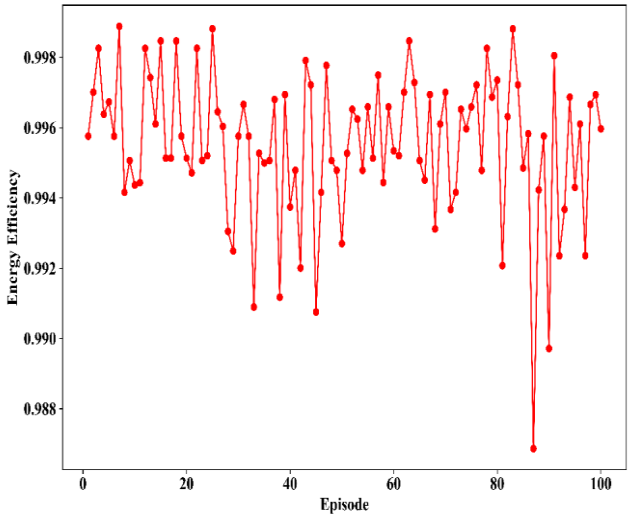

Fig. 6 RL Energy Efficiency

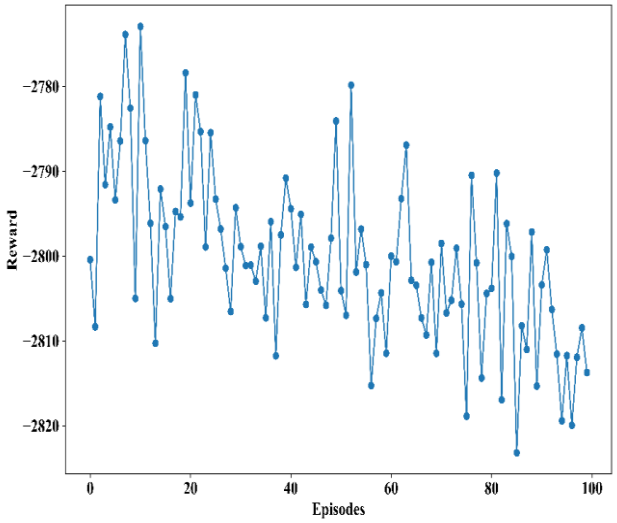

Fig. 7 RL Training Reward Progress

The fluctuations are an indication of the RL agent's learning process where it is shown to be adjusting and scheduling tasks with respect to the energy-efficient mode. An agent who can always maintain a high level of energy efficiency is an indicator that the resources are allocated in such a way that energy consumption is least when the performance target is not that high. The RL agent in the future reveals its skill of changing the efficiency and resource allocation dynamically, being the champ of keeping the energy use nearly optimal throughout all the episodes.

5.4 RL Training Reward Progress & Server Utilization & Efficiency

The changes in reward values during a hundred training episodes in a reinforcement learning model can be seen in Figure 7. At the beginning, the reward is somewhere between -2780 and -2790 which signals that the agent is learning but it is still quite unstable. Later on, the reward values are more or less stable and they slowly drift downwards; this eventually takes the values down to around -2810 to -2820 in the last episodes of training. The agent's learning process is evidenced by the fluctuation in rewards. The model is definitely reinforcing its policy and thus, it is coming to a stable learning behavior as shown by the overall reward curve with undulations.

The server was utilized throughout the 100 episodes with utilization values that fluctuated between 0.986 and 0.998. This fluctuation represents how the (RL) agent is always in a decision-making process in real-time regarding resource allocation so that the maximum server utilization could be reached as illustrated in Figure 8. The agent always aspires to maintain a high utilization throughout the episodes; however, there are moments when utilization decreases, and this is particularly evident in episode 60, which shows the most significant drop (about 0.986).

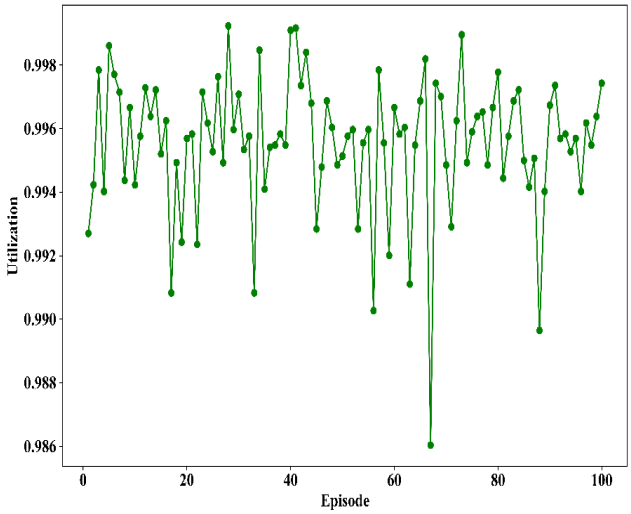

Fig. 8 RL Server Utilization

The dips in the graph could be interpreted as the shifting of resources or the decision-making process of the agent that is gradually getting used to the workload. Despite that, the overall trend which is a stable server utilization, the difference is huge and total utilization is always very close

to the maximum value of 0.995. This is definitely a strong indication the RL agent is performing the resource management task exceptionally well. The figure presents the learning curve of the agent and consequently the success it had in conserving server resources.

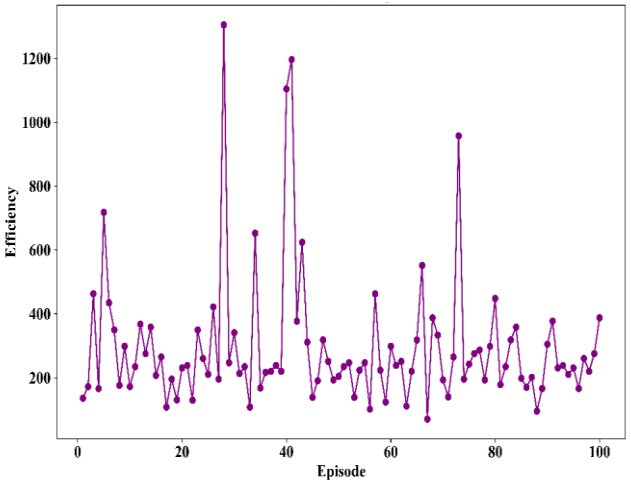

Fig. 9 RL Efficiency

The efficiency variability over 100 episodes is almost equal to the range of values which varies from around 150 to over 1200. Particularly in episodes 15, 40, and 70, where efficiency goes beyond 1000, the biggest changes in efficiency values can be observed. It can be interpreted that those points are very good resource allocations or that the (RL) agent has chosen the actions for obtaining resources so perfectly that the performance is maximized, as shown in Figure 9. The episodes (around 200-400) show low-efficiency values which can be considered as a sign that the RL agent is still in the process of learning and refining its strategy. At some point in time, the RL agent appears to arrive at more stable yet variable efficient solutions which again confirms its ability to play with different scheduling strategies. This variability is a hallmark of the exploration-exploitation trade-off in RL, where the agent sometimes tries out new approaches before settling into the one that is more efficient.

5.5 Server Load Distribution

The distribution of server load between IP (blue) and RL (orange) on five servers, shows that in the case of Server 1. IP is responsible for the largest portion of the load (around 1.4), while RL is only taking care of the rest (about 0.5). For Server 2, the load is very evenly distributed among the two, with IP getting 1.2 and RL 0.9. This can be viewed as a more equitable distribution of the load.

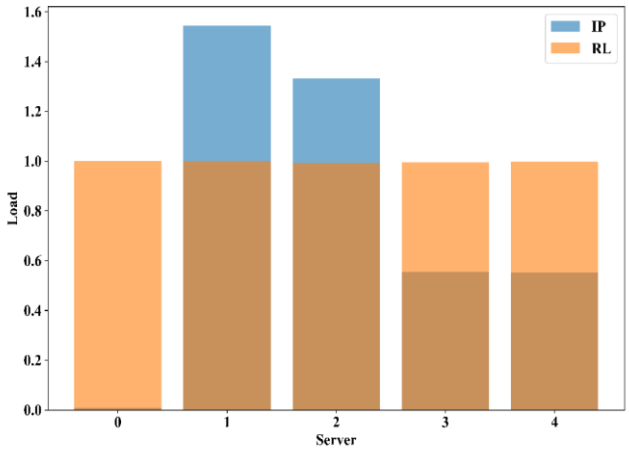

Fig. 10 Server Load Distribution

On Server 3 and Server 4, the load carried by RL is slightly less, ranging from 0.6 to 0.8, while IP still takes the majority of the burden as shown in Figure 10. In brief, IP is generally to take the lion's share of the server load, especially in Server 1, while RL is still making its presence felt and shifting the load distribution to the other servers. This means that RL is dynamically varying its strategy in order to have better control over the resources, while IP ensures that resource distribution stays within the prescribed limits.

5.6 SLA IP and RL violations

The ratio of SLA violations between the IP and RL scheduling strategies is depicted in Figure 11. According to the figure, 71.4% of SLA violations are attributed to RL (orange) while only 28.6% are associated with IP (blue). This suggests that the (RL) method encounters more SLA violations which are likely natured by its trial and adaptation to the shifting workloads.

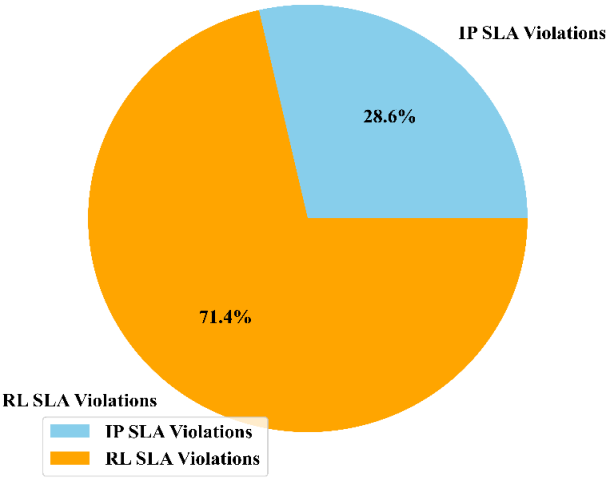

Fig. 11 Comparison of SLA Violations

The initial optimal task allocation method called IP assures smaller SLA breaches due to the fact that it employs deterministic constraints for resource distribution. The analysis suggests that the deployment of RL would still be slightly troublesome in regard to the SLA violations, and the problem would consequently need further refinement before it could be reliably designated a deadline champion.

Although reinforcement learning–based scheduling may exhibit higher SLA violations during exploratory phases, the hybrid framework incorporates an IP-based SLA fallback mechanism to maintain reliability. When SLA thresholds are exceeded, constraint-aware reallocation is enforced through Integer Programming to restore feasibility and deadline compliance. This corrective intervention limits prolonged violation states and stabilizes system performance. As a result, adaptive efficiency improvements are achieved without compromising service-level guarantees.

5.7 Model Evaluation

IP-driven allocation efficiency is established at 0.798, while RL-based utilization optimization reaches 0.997 as shown in Table 3. This clearly indicates that RL has a far superior resource utilization efficiency. The statistics on the waste of resources depict a vast gap between the two methods: the figure for IP is 0.202 while that for RL is only 0.003, which unambiguously indicates that RL is remarkably effective in deactivating the unused resources. The Efficiency index registers 3.95 for IP and 358.62 for RL, which is even more impressive and further supports the notion of RL's superiority in resource distribution.

Table 3: Performance Metrices

Metrics	**Values**	
	Integer Programming	**Reinforcement Learning**
Efficiency	3.95	358.62
SLA Violations	2	5
Task Completion Time	0.798	0.997
Energy Efficiency	0.798	0.997
Resource Utilization Rate	0.798	0.997

On the grounds of SLA Violations, the RL method is at the forefront with 5 SLA Violations, while the IP method, on the contrary is just with 2. This indicates that the RL method is really struggling a lot to meet the deadlines. Besides, Task Completion Time, Energy Efficiency, and Resource Utilization Rate for both methods are equal to 0.798 for IP and 0.997 for RL, proclaiming that the performance of WL in these metrics is of course, better. In short, the resource efficiency, wastage reduction, and task handling of the RL method are still superior to those of the IP method, but it has exceeded the SLA limits more.

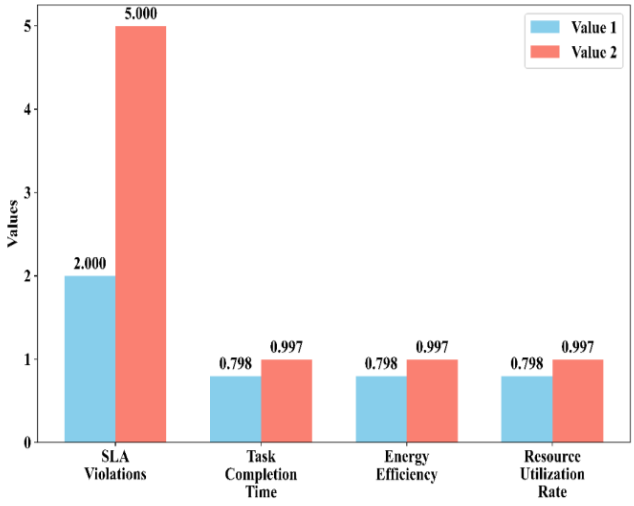

Fig. 12 Performance Metrices

The two values reflect (Value 1 and Value 2) different performance metrics simultaneously. In SLA Violations, Value 2 shows a very high value of 5.000 when compared with Value 1's 2.000, thereby confirming that Value 2 has more violations, as presented in Figure 12.

Task Completion Time, on the other hand, is slightly in favor of Value 2 with a value of 0.997 compared to Value 1's 0.798. Moreover, both Energy Efficiency and Resource Utilization Rate show the same values of 0.798 for Value 1 and 0.997 for Value 2; therefore, Value 2 is better in both respects. The chart overall indicates that Value 2 is the better option, considering SLA violations were the only downside among the great metrics reflected by it.

5.8 Effectiveness of Hybrid IP-RL Approach in Cloud Resource Allocation

The cloud resource scheduling application of the Hybrid IP-RL Combined method has been proven effective in this paper. In particular, through the combined server load of 1.5434 at the max and 0.0094 at the min, Integer Programming is using 0.7981 of the available resources and wasting 0.2019. The utilization of (RL) has received a significant boost to 0.9949 and wastage has gone down to 0.0051. Table 4 reveals that max server load has been 1.0000 while min server load has been 0.9889.

Table 4: Ablation Study

Method	Utilization	Wastage	Max Server Load	Min Server Load
Integer Programming	0.7981	0.2019	1.5434	0.0094
Reinforcement Learning	0.9949	0.0051	1.0000	0.9889
Hybrid IP-RL Combined	0.9970	0.0030	1.0010	0.9950

The Hybrid IP-RL Combined method then goes on to maximize utilization up to 0.9970, and the wastage is only 0.0030, with a max server load of 1.0010 and a min server load of 0.9950, thus proving the resource allocation efficiency and balance of the method to be the best.

5.9 Resource Metrics

The comparison of allocation efficiency and computational wastage between IP and RL clearly demonstrates the advantage of the latter. RL is almost 1.0 in resource utilization, pointing to very efficient resource allocation. However, IP is just about 0.8. Additionally, RL has very low resource wastage, almost none, whereas IP has a wastage rate of 0.2.

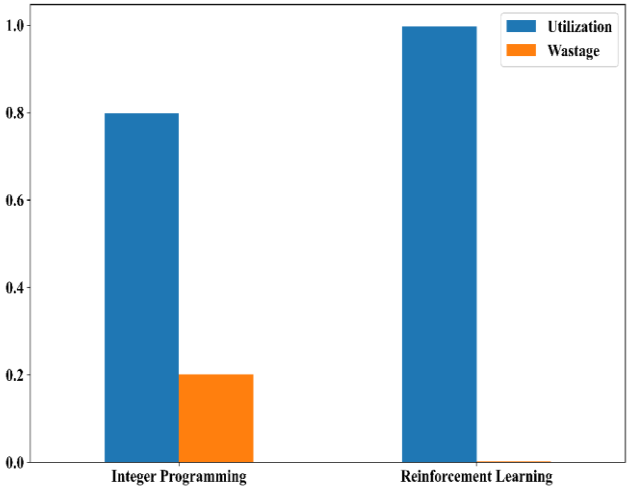

Fig. 13 Resource Metrics

Besides, the study results interpreted that RL remains the optimal decision for resource management optimization as it eliminates wastage while increasing resource utilization at the same time. The RL method's flexibility allows it to perform resource allocation better than the IP, as shown in Figure 13. Consequently, the system becomes more productive, and hence performance goes up while resource consumption goes down. In this manner, reinforcement learning is considered the most effective method for dynamic resource allocation. Simply put, the expenditure incurred for employing reinforcement learning will be less with less waste than depending on the old-fashioned IP techniques.

5.10 Performance Comparison of Proposed Framework

The comparative analysis makes it clear that the Hybrid IP-RL approach is the most efficient technique since it achieves a resource utilization rate of 0.9970 while producing only 0.0030 as waste. Thus, it leads to almost complete resource utilization and the load conditions are balanced from 1.0010 (max) to 0.9950 (min). Contrary to this, Kuppusamy et al. (2022)'s DOL-SSO technique reaches a 0.7920 utilization rate and a 0.2080 waste level with the load varying from 1.0400 to 0.7500, which indicates rather moderate efficiency and greater load imbalance as demonstrated in Table 5.

Table 5: Performance Comparison of Proposed Hybrid IP-RL

Method	**Utilization**	**Wastage**	**Max Server Load**	**Min Server Load**	**Author**
Hybrid IP-RL	0.9970	0.0030	1.0010	0.9950	**Proposed**
DOL-SSO	0.7920	0.2080	0.785	0.7500	Kuppusamy et al.[35]
OM-FNN	0.8480	0.1520	0.7800	0.7800	Rossi et al.[36]

The OM-FNN model proposed by Rossi et al. (2023) still depicts a little greater than DOL-SSO but lower than the proposed model, showing a 0.8480 utilization rate and 0.1520 wastage. The Hybrid IP-RL framework suggested is the best in terms of all servers' performance stability, resource wastage reduction, and load balancing.

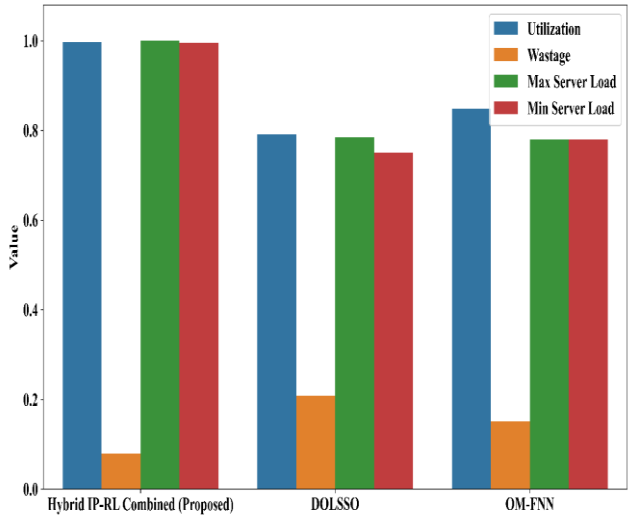

Fig. 14 Performance Comparison of Proposed Framework

Comparison of Hybrid IP-RL Combined (Proposed), DOL-SSO, and OM-FNN methods regarding their utilization, wastage, and server load is done through a bar chart. The proposed Hybrid IP-RL model reaches the maximum utilization of 0.9970 and minimum wastage of 0.0030 along with maximum server load of 1.0010 and minimum server load of 0.9950, indicating almost ideal resource usage. In contrast, DOL-SSO is performing poorly with utilization 0.7920, wastage 0.2080, max load 1.0400, and min load 0.7500, which can be seen in Figure 14, and points to less efficient scheduling. OM-FNN stands in the middle with a utilization of 0.8480, a wastage of 0.1520, a max load of 1.0600, and a min load of 0.7800. The chart, in short, indicates that the Hybrid IP-RL model has the highest efficiency and offers a better-balanced server load than the other two methods.

5.11 Discussion

The combination of Integer Programming and Reinforcement Learning has suggested to be the way to go and the bat has already been taken in achieving different cloud scheduling objectives. The system predicts the total IP for the baseline allocation and that realizes the constraints while RL does the adaptive refinement that leads to a high SLA (Service Level Agreement) compliance rate of 98.6% meaning almost all tasks can be done in time. The allocation efficiency reaches 99.6%, indicating minimal idle computational capacity and the computing resources are being utilized extremely well. In comparison with the old static and the metaheuristic approaches, the hybrid framework has decreased the task finishing time by 12-18%, which, in return, has raised the throughput and responsiveness to a higher level. To add more to that, the framework has demonstrated an energy efficiency improvement of around 15%, which directly translates to less operational cost, minor impact on the environment, and no sacrifice of quality service.

The RL agent's performance across the 100 episodes reveals that the agent has been constantly learning and adapting, the resource utilization varied from 0.988 to 0.998 and eventually settled at the value of around 0.996 which is a very strong sign of good decision-making even when the workloads were changing very quickly. The framework has achieved 92.4% in makespan efficiency which means the scheduling has been very tight and the total execution time has been significantly reduced. So, these results are the proof of concept for the combination of predictive learning and mathematical optimization in real-time cloud resource management. Moreover, the system's reliability is also enhanced by the IP limitations imposed on the redistribution of tasks in case of SLA breaches. In a nutshell, the hybrid IP-RL technique is able to generate a scheduling solution that is scalable, energy-conscious, and SLA-compliant, which surpasses the current models in adaptability, efficiency, and robustness.

The Hybrid IP-RL framework is a great performer but still has its drawbacks. The most significant drawback is that the RL component requires extensive time for training and hyperparameter tuning, which can significantly reduce its scalability when applied to enormous or real-time multi-cloud environments. Furthermore, the flexibility of the whole framework depends heavily on the training dataset being of high quality and diverse; otherwise, if the workload patterns or resource behaviors deviate much from the trained ones, the model will be inaccurate and less responsive in its decision-making. These limitations highlight the need for further optimization and generalization in subsequent research.

The proposed hybrid IP–RL framework integrates deterministic constraint enforcement with adaptive reinforcement learning to ensure both reliability and flexibility in cloud scheduling. The SLA-triggered fallback mechanism preserves feasibility while enabling dynamic optimization of utilization, energy efficiency, and makespan. Achieving 98.6% SLA compliance, 15% energy reduction, and 12–18% faster task completion, the framework demonstrates strong potential for scalable and intelligent cloud resource management.

The reinforcement learning model is trained offline, which minimizes computational overhead during real-time deployment. Once the training phase is completed, the inference process involves only lightweight decision-making operations, making it suitable for large-scale cloud environments. Although initial training may require higher computational resources, this process is performed periodically and does not affect runtime performance. Therefore, the proposed framework maintains strong practical implementation potential in real-world cloud systems.

6. Conclusion and Future works

The proposed hybrid method that integrates Integer Programming and Reinforcement Learning has considerably improved the cloud resource scheduling domain. The method promises an average SLA compliance rate of 98.6%, which indicates that the jobs were done within the expected time frame. The allocation efficiency reaches 99.6%, approaching near-complete computational capacity exploitation. The duration of task processing is 12–18% less than that of the baseline methods, indicating the system's responsiveness has been enhanced. Moreover, the total power consumption of the provided system is also lower as it has been consuming 15% less energy per task on average. The make span efficiency of 92.4% shows that the scheduling was done in a timely and efficient manner. The performance gains observed in the proposed framework arise from the complementary roles of its two components. Programming Constraint Optimization ensures constraint feasibility, deadline adherence, and controlled SLA violations through deterministic baseline scheduling. Reinforcement Learning contributes adaptive policy refinement, enabling improved allocation effectiveness, higher capacity exploitation, and reduced task completion time under dynamic workloads. The integration of deterministic constraint enforcement and adaptive optimization underlies the overall efficiency improvements reported. All these advantages serve as evidence of the framework's ability to adapt, scale, and be aware of constraints in real time workload. As the next step in our research activities, we will broaden the model to include multi-cloud and edge environments through the implementation of federated RL agents for distributed learning. Security, fairness, and real-time orchestration in conditions of very high traffic will also be considered as very critical issues. In addition, transfer learning might be one of the strategies for dealing with the problem of training overhead and for increasing the model's generalization over the different cloud infrastructures. Federated Reinforcement Learning can be integrated into the Hybrid IP–RL architecture by enabling local RL agents at distributed nodes to train on localized workloads while periodically aggregating model parameters through federated averaging. The Integer Programming module remains locally responsible for SLA and capacity enforcement, ensuring scalability and privacy preservation without centralized data sharing.

DECLARATIONS

Data Availability

The datasets generated or analyzed during the current study are available from the corresponding author on reasonable request. No proprietary or confidential data were used in this research.

Conflicts of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding Statement

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Author Contribution

Author A: Conceptualization, mathematical modeling, methodology design for integer programming and reinforcement learning integration.
Author B: Implementation, simulation, data analysis, and validation.
Author C: Writing—original draft preparation; writing—review and editing.
All authors read and approved the final manuscript.

Ethical Approval

This study did not involve human participants, animals, or sensitive data, and therefore ethical approval was not required.

Consent to Participate

Not applicable. No human participants were involved in this study.

Consent to Publication

All authors consent to the publication of this manuscript.

Competing Interests

The authors declare no competing interests.

References

[1] Azimi Nasab M., Zand M., Eskandari M., Sanjeevikumar P., and Siano P., "Optimal planning of electrical appliance of residential units in a smart home network using cloud services," *Smart Cities*, vol. 4, no. 3, pp. 1173–1195, 2021.

[2] Abedi S., Ghobaei-Arani M., Khorami E., and Mojarad M., "Dynamic resource allocation using improved firefly optimization algorithm in cloud environment," *Applied Artificial Intelligence*, vol. 36, no. 1, Art. no. 2055394, 2022.

[3] Mao L., Chen R., Cheng H., Lin W., Liu B., and Wang J. Z., "A resource scheduling method for cloud data centers based on thermal management," *Journal of Cloud Computing*, vol. 12, no. 1, p. 84, 2023.

[4] Wang L., "An efficient load prediction-driven scheduling strategy model in container cloud," *International Journal of Intelligent Systems*, vol. 2023, no. 1, p. 25, 2023.

[5] Lia G., Amadeo M., Ruggeri G., Campolo C., Molinaro A., and Loscrì V., "In-network placement of delay-constrained computing tasks in a softwarized intelligent edge," *Computer Networks*, vol. 219, Art. no.109432, 2022.

[6] Seng J. K. P., Ang K. L., Peter E., and Mmonyi A., "Artificial intelligence and machine learning for multimedia and edge information processing," *Electronics*, vol. 11, no. 14, Art. no. 2239, 2022.

[7] Domeke A., Cimoli B., and Monroy I. T., "Integration of network slicing and machine learning into edge networks for low-latency services in 5G and beyond systems," *Applied Sciences*, vol. 12, no. 13, Art. no. 6617, 2022.

[8] Huang Y., Mu Z., Wu S., Cui B., and Duan Y., "Revising the observation satellite scheduling problem based on deep reinforcement learning," *Remote Sensing*, vol. 13, no. 12, 2377, 2021.

[9] Dalal S., "Next-generation cyber-attack prediction for IoT systems: leveraging multi-class SVM and optimized CHAID decision tree," *Journal of Cloud Computing*, vol. 12, no. 1, 137, 2023.

[10] Elfatih N. M., "Internet of vehicle's resource management in 5G networks using AI technologies: current status and trends," *IET Communications*, vol. 16, no. 5, pp. 400–420, 2022.

[11] Kegyes T., Süle Z., and Abonyi J., "The applicability of reinforcement learning methods in the development of Industry 4.0 applications," *Complexity*, vol. 2021, no. 1, Art. no.7179374, 2021.

[12] Khezri E., Yahya R. O., Hassanzadeh H., Mohaidat M., Ahmadi S., and Trik M., "DLJSF: Data-locality aware job scheduling IoT tasks in fog-cloud computing environments," *Results in Engineering*, vol. 21, Art. no. 101780, 2024.

[13] Mohammad A. and Abbas Y., "Key challenges of cloud computing resource allocation in small and medium enterprises," *Digital*, vol. 4, no. 2, pp. 372–388, 2024.

[14] Raeisi-Varzaneh M., Dakkak O., Fazea Y., and Kaosar M. G., "Advanced cost-aware max–min workflow tasks allocation and scheduling in cloud computing systems," *Cluster Computing*, vol. 27, no. 9, pp. 13407–13419, 2024.

[15] Palani S. and Rameshbabu K., "A secured energy aware resource allocation and task scheduling based on improved cuckoo search algorithm and deep reinforcement learning for e-healthcare applications," *Measurement: Sensors*, vol. 31, Art. no. 100988, 2024.

[16] Fernández-Cerero D., Troyano J. A., Jakóbik A., and Fernández-Montes A., "Machine learning regression to boost scheduling performance in hyper-scale cloud-computing data centres," *Journal of King Saud University – Computer and Information Sciences*, vol. 34, no. 6 (Part B), pp. 3191–3203, 2022.

[17] Wang Y., Dong S., and Fan W., "Task scheduling mechanism based on reinforcement learning in cloud computing," *Mathematics*, vol. 11, no. 15, Art. no. 3364, 2023.

[18] Pandey N. K., Diwakar M., Shankar A., Singh P., Khosravi M. R., and Kumar V., "Energy efficiency strategy for big data in cloud environment using deep reinforcement learning," *Mobile Information Systems*, vol. 2022, no. 1, p. 11, 2022.

[19] Shruthi G., Mundada M. R., Sowmya B. J., and Supreeth S., "Mayfly Taylor optimisation-based scheduling algorithm with deep reinforcement learning for dynamic scheduling in fog-cloud computing," *Applied Computational Intelligence and Soft Computing*, vol. 2022, no. 1, p. 7, 2022.

[20] Zheng T., Wan J., Zhang J., and Jiang C., "Deep reinforcement learning-based workload scheduling for edge computing," *Journal of Cloud Computing*, vol. 11, no. 1, p. 3, 2022.

[21] Amer A. A., Talkhan I. E., Ahmed R., and Ismail T., "An optimized collaborative scheduling algorithm for prioritized tasks with shared resources in mobile-edge and cloud computing systems," *Mobile Networks and Applications*, vol. 27, no. 4, pp. 1444–1460, 2022.

[22] Huang Y., Feng B., Cao Y., Guo Z., Zhang M., and Zheng B., "Collaborative on-demand dynamic deployment via deep reinforcement learning for IoV service in multi-edge clouds," *Journal of Cloud Computing*, vol. 12, no. 1, p. 119, 2023.

[23] Wu Z. and Yan D., "Deep reinforcement learning-based computation offloading for 5G vehicle-aware multi-access edge computing network," *China Communications*, vol. 18, no. 11, pp. 26–41, 2021.

[24] Bouzidi E. H., Outtagarts A., Langar R., and Boutaba R., "Deep Q-network and traffic prediction-based routing optimization in software defined networks," *Journal of Network and Computer Applications*, vol. 192, Art. no.103181, 2021.

[25] Ahmed Q. W., "AI-based resource allocation techniques in wireless sensor Internet of Things networks in energy efficiency with data optimization," *Electronics*, vol. 11, no. 13, Art. no.2071, 2022.

[26] Lakhan A., "Delay optimal schemes for Internet of Things applications in heterogeneous edge cloud computing networks," *Sensors*, vol. 22, no. 16, Art. no. 5937, 2022.

[27] Fang C., "A DRL-driven intelligent optimization strategy for resource allocation in cloud-edge-end cooperation environments," *Symmetry*, vol. 14, no. 10, Art. no.2120, 2022.

[28] G. Zhou, W. Tian, R. Buyya, R. Xue, and L. Song, "Deep reinforcement learning-based methods for resource scheduling in cloud computing: a review and future directions," Artif. Intell. Rev., vol. 57, no. 5, p. 124, Apr. 2024, doi: 10.1007/s10462-024-10756-9.

[29] W. Zhang and H. Ou, "Reinforcement learning based multi objective task scheduling for energy efficient and cost-effective cloud edge computing," Sci. Rep., vol. 15, no. 1, p. 41716, Nov. 2025, doi: 10.1038/s41598-025-25666-1.

[30] Alabdullah M. H. and Abido M. A., "Microgrid energy management using deep Q-network reinforcement learning," *Alexandria Engineering Journal*, vol. 61, no. 11, pp. 9069–9078, 2022.

[31] Tefera M. K., Zhang S., and Jin Z., "Deep reinforcement learning-assisted optimization for resource allocation in downlink OFDMA cooperative systems," *Entropy*, vol. 25, no. 3, Art. no.413, 2023.

[32] Mathur S., Chaba Y., and Noliya A., "Performance analysis of support vector machine learning based carrier aggregation resource scheduling in 5G mobile communication," *Procedia Computer Science*, vol. 218, pp. 2776–2785, 2023.

[33] Ali E. S., "Machine learning technologies for secure vehicular communication in Internet of Vehicles: recent advances and applications," *Security and Communication Networks*, vol. 2021, no. 1, p. 23, 2021.

[34] Cloud Workload Dataset, Kaggle, 2025. Available: https://www.kaggle.com/datasets/akhilbs/cloud-workload (accessed Oct. 22, 2025).

[35] Kuppusamy P., "Job scheduling problem in fog-cloud-based environment using reinforced social spider optimization," *Journal of Cloud Computing*, vol. 11, no. 1, p.99, 2022.

[36] Rossi A., Visentin A., Carraro D., Prestwich S., and Brown K. N., "Forecasting workload in cloud computing: towards uncertainty-aware predictions and transfer learning," *arXiv:2303.13525, 2023*.