Journal of Engineering, Project, and Production Management
2026, 16(4), 2025-266

Dynamic Multi-Objective Resource Allocation in Cloud Computing

Yalan Yang[1] and Xiaoming Wu[2]
[1] Lecturer, School of Economics and Management, Southwest Petroleum University, Chengdu, 610500, China
[2] Supervisor, School of Economics and Management, Southwest Petroleum University, Chengdu, 610500, China, E-mail: wu.xming@outlook.com (corresponding author).

Project Management
Received November 12, 2025; revised April 8, 2026; accepted May 2, 2026
Available online May 29, 2026

Abstract: In response to issues of low efficiency in distributed project resource allocation and poor service quality in cloud computing environments, this paper constructs a multi-objective optimization model that explicitly accounts for deadline constraints, task priorities, and virtual machine load conditions. This model proposes a hybrid optimization algorithm that combines dynamic load assessment with active migration mechanisms. The algorithm achieves predictive control of Service Level Agreement (SLA) violations using an irregular cost function, manages dynamic workloads with a task-arrival time aware strategy, and performs load rebalancing through active migration. Experimental results from the CloudSim simulation platform demonstrate that the proposed algorithm significantly outperforms traditional benchmark algorithms across key metrics, including completion time, resource utilization, and SLA violation rate, thereby enhancing system performance and guaranteeing service quality. This research offers a practical technical solution for cloud service providers to optimize data center resource management, ensuring service quality while improving resource utilization efficiency, and holds valuable theoretical and practical significance.

Keywords: CloudSim simulation, resource allocation, load balancing, multi-objective optimization, task scheduling.

Copyright © Journal of Engineering, Project, and Production Management (EPPM-Journal).
DOI 10.32738/JEPPM-2025-266

1. Introduction

Cloud computing has emerged rapidly, becoming the core of modern information technology infrastructure (Cassel et al., 2022; Malyshkin, 2022). The growth of containerization and cloud-native technologies has led to new application deployment applications (Naayini 2025). One such approach is serverless computing to dynamically allocate resources in order to support complex application scenarios (Toosi et al., 2025). The potent integration of big data technology with cloud computing platforms has also introduced advanced performance improvement prospects for IT-based project management (Mahmud et al., 2025), yet resource allocation optimization is constantly a major bottleneck that inhibits system performance in cloud environments. In the Infrastructure as a Service (IaaS) cloud, improper scheduling strategies usually cause heavy imbalance among Virtual Machine (VM) loads, resulting in an increase in execution latency and even system crashes.

In resource allocation optimization, traditional load balancing algorithms are often inadequate to oversee dynamic workloads because they do not adequately consider task heterogeneity (RM et al., 2020). Developments in this area include meta-heuristic hybrid techniques to reduce completion time (Jena et al., 2022), load balancing algorithms specifically designed for data centers (Shafiq et al., 2021), a service discovery mechanism based on microservice architecture (Singh et al., 2023), and a new biologically-inspired scheduling approach that offers advantages in Makespan optimization and resource utilization (Alruwais et al., 2024; Singhal et al., 2024). Research on resource allocation optimization points to a trend toward a multi-objective collaborative optimization mode l (Chen et al., 2021; Ko et al., 2021). Workflow-aware analysis models based on performance and cost prediction provide quantitative support for scheduling decisions (Kumari et al., 2023). These models have been integrated into enterprise IT project management information systems (Pratama et al., 2023). Cost-aware allocation algorithms balance execution efficiency with economic considerations (Raeisi-Varzaneh et al., 2024). Reviews of parallel and distributed computing establish the theoretical foundations of optimization (Czarnul et al., 2025; Doostmohammadian et al., 2025), while applications of machine learning mark the evolution toward intelligent optimization methods (Wang and Yang, 2025).

The integration of federated learning and edge computing brings forth new avenues for distributed resource management. A systematic review identifies varied challenges for federated learning in edge computing, such as resource constraints (Abreha et al., 2022). Research on the multi-skilled resource-constrained project scheduling problems highlights the importance of considering skill matching (Peng et al., 2023). An extensive exploration of the collective resource allocation strategies of federated edge learning has identified future research directions (Zhang et al., 2024). The cloud resource scheduling technique of the deep reinforcement learning-based method proves the strong potential of artificial intelligence (Zhou et al., 2024). While the current research has accomplished fruitful achievements, deficiencies still exist. First, the consideration of the parameters of service quality, particularly the constraints of deadlines and the priority of tasks, lacks systematization. Second, the workload migration mechanism lacks sufficient research, and the active migration strategies should be effectively implemented when the virtual machine breaks the Service Level Agreement (SLA). Third, the adaptability of the dynamics of the resource reconfiguration algorithms to the random arrivals of tasks and the heterogeneous workload of dynamic environments should be further improved.

Distinguishing itself from existing approaches that predominantly address resource allocation through either static priority schemes or reactive load balancing mechanisms, this research develops a distributed project resource allocation optimization algorithm that integrates proactive violation prevention with adaptive workload management for cloud computing systems. Most current approaches tend to respond to Service Level Agreement (SLA) violations rather than predict and prevent them, which forms a critical gap in the quality assurance of dynamic environments. This paper overcomes this limitation by constructing a multi-objective optimization model that incorporates deadline constraints, task priorities, and virtual machine load states, while developing dynamic load evaluation and SLA violation prediction mechanisms to implement active resource migration and reconfiguration. The hybrid scheduling policy enables the effective treatment of random task arrivals through strategies that consider the time of task arrivals and adapt the allocation decisions to the real-time state of the system to optimize the resource allocation mapping relationship under time-varying workloads. This paper validates the proposed algorithm using the CloudSim simulation platform across key performance indicators: reducing the makespan, improving resource utilization, and lowering the SLA violation rate. It provides cloud service providers with a practical technical solution for data center resource management that promotes both theoretical understanding and operational efficiency of quality-assured cloud computing systems.

2. Methodology

2.1. System Model and Problem Formalization

The distributed project resource allocation problem in cloud computing environments involves complex multi-level interactions. In the IaaS cloud service model, data centers abstract physical server resources into multiple virtual machine instances through virtualization technology, and user-submitted task requests enter the system scheduling queue in the form of cloudlets.

This paper defines the task set $T = \{t_1, t_2, ..., t_n\}$ and virtual machine set $V = \{v_1, v_2, ..., v_m\}$, where each task t_i has attributes including length L_i (measured in Million Instructions), deadline D_i, arrival time A_i, and priority P_i. Each virtual machine v_j is characterized by core parameters, including processing speed $MIPS_j$ (Million Instructions Per Second), current load $Load_j$, and available capacity Cap_j.

The resource allocation optimization problem seeks the optimal mapping relationship from tasks to virtual machines subject to quality of service constraints. The decision variable $x_{ij} \in 0,1$ is defined to indicate whether task t_i is allocated to a virtual machine v_j. The optimization objective comprehensively considers three dimensions: minimizing makespan, maximizing resource utilization, and minimizing the SLA violation rate. Makespan is formally defined in Eq. (1).

$$Makespan = \max_{j=1}^{m} \{ \sum_{i=1}^{n} x_{ij} \cdot \frac{L_i}{MIPS_j} \} \tag{1}$$

Resource utilization is calculated according to Eq. (2).

$$ResourceUtilization = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} \cdot \frac{L_i}{MIPS_j}}{\sum_{j=1}^{m} MIPS_j \cdot Makespan} \tag{2}$$

The feasibility constraints for the allocation scheme comprise three components, as expressed in Eqs. (3), (4), and (5), respectively.

$$\sum_{j=1}^{m} x_{ij} = 1, \quad \forall i \tag{3}$$

$$\sum_{i=1}^{n} x_{ij} \cdot L_i \leq Cap_j, \quad \forall j \tag{4}$$

$$CompletionTime_{ij} \leq D_i \tag{5}$$

Eq. (3) ensures each task is assigned to exactly one Virtual Machine. Eq. (4) prevents resource overcommitment on any Virtual Machine. Eq. (5) enforces the deadline requirements embedded in service agreements.

To facilitate algorithm description and implementation, Table 1 systematically summarizes the mathematical symbols used in this paper, which run through the entire process of constructing and solving the optimization model.

Table 1. Symbol definitions and parameter specifications

Symbol	Definition	Symbol	Definition
T, V	Task set, Virtual machine set	L_i, D_i	Task length (MI), Deadline(s)
A_i, P_i	Arrival time(s), Priority	$MIPS_j$, $Load_j$	VM processing speed, Current load (%)
x_{ij}	Allocation decision variable (0/1)	Cap_j	VM available capacity (MI)

The symbolic system in Table 1 forms the mathematical basis of the resource allocation optimization model. Specifically, the task-related parameters describe the characteristics and constraints of user requests, the virtual machine parameters represent the computing power and state of the system, and the decision variables establish the mapping relationship between tasks and resources.

2.2. Optimization Algorithm Design and Implementation

To address the aforementioned resource allocation optimization problem, this paper proposes a hybrid optimization algorithm combining dynamic load evaluation and an active migration mechanism. The algorithm is inspired by the two-phase strategy based on twelve priority rules in randomized distributed multi-project scheduling (Yu et al., 2023), and its dynamic priority adjustment mechanism provides an important reference for task priority evaluation. At the same time, it draws upon the successful strategies of hybrid metaheuristic algorithms for resource allocation in the edge-cloud continuum within Internet of Things application scenarios (Dankolo et al., 2025), especially its adaptive parameter adjustment and multi-objective collaborative optimization idea. Based on the hierarchical design concept, this paper constructs a four-layer architecture including a physical resource layer, a virtualization layer, a resource management layer, and a scheduling decision layer. Information transmission and control instructions are delivered through standardized interfaces between layers. Fig. 1 shows the complete structure of this layered architecture and its internal components.

The four-layer architecture shown in Fig. 1 reflects the modular design concept of the system. The physical resource layer provides computing, storage, and network infrastructure. The virtualization layer performs resource abstraction and isolation through the Hypervisor. The resource management layer dynamically maintains the virtual machine pool and monitors the load status in real time, and the scheduling decision layer receives task requests and executes resource allocation decisions. The interlayer interaction ensures the timely feedback of load information and the rapid response to the allocation policy.

Traditional resource scheduling algorithms employ static allocation strategies that are inadequate for dynamic task arrival and heterogeneous workloads in cloud environments. The proposed algorithm addresses these limitations by using three main components. These components are dynamic load assessment, proactive task migration, and task arrival time awareness. Algorithm one describes the complete pseudo-code.

Algorithm one combines three essential components into a unified framework. The dynamic load evaluation mechanism combines deadline constraints with task priorities through a violation cost function, which predicts risks of SLA violations and prioritizes critical tasks. The proactive migration mechanism identifies high and low load virtual machine pairs to balance loads through dynamic Million Instructions Per Second (MIPS) allocation, focusing on low-priority tasks for preventive instead of remedial migrations. The value 0.75 optimizes resource usage and minimizes the risk of violation, as the variability in completion time increases with loads above this value. At the same time, the coefficient $\eta = 0.3$ limits excessive migrations and optimizes their distribution. The task arrives at the time-awareness mechanism, which processes tasks sequentially based on their arrival times, thereby helping adapt allocation decisions to the system's state in response to time-varying workload characteristics.

The algorithm has a computational complexity of $O(n \times m)$, with dynamic load evaluation requiring $O(m)$ comparisons per task and migration decisions evaluating $O(m)$ VM pairs upon activation. The polynomial tractability is maintained for the cloud environment with the application of threshold-based migration strategies.

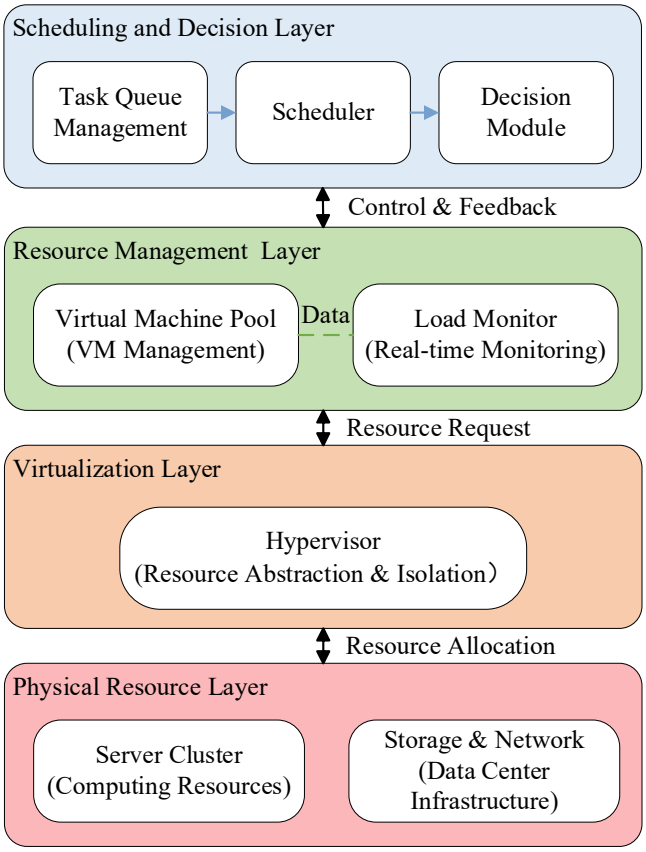


Fig. 1. The overall architecture diagram of the system

2.3. QoS Guarantee and SLA Management Mechanism

Quality of Service (QoS) guarantees are the key objective of resource provisioning in cloud computing. The QoS management mechanism proposed in the study enables efficient SLA management with the help of integrated multi-level constraint handling and predictive control. Under traditional task management models, state transitions are treated as linear processes, causing them to overlook potential risks of SLA violations and their corresponding remediation actions. In this study, conventional models are extended to account for transition and timeout states in order to better describe the complexity of task execution in a changing environment. A state transition diagram for the entire task lifecycle is shown in Fig. 2.

The task state transition logic and decision points for Quality of Service (QoS) management control are presented in Fig. 2. In this case, tasks move from a waiting state to an executing state based on resource availability and satisfaction of constraints. When SLA violation risks are detected, the system transitions the task to a migration state, thereby allowing a transfer of load before resuming and completing a task. Deadline violations result in a timeout state with recorded violation information.

Based on the quality transmission theory of multi-skills project scheduling optimization (Peng et al., 2025), this study proposes a task execution quality monitoring mechanism with delay risk prediction based on real-time task progress and resource consumption. Migration is triggered when the predicted task completion exceeds a predetermined deadline threshold. SLA violation prediction employs resource allocation models based on game theory (Somayeh, 2025), where violation probability estimates are generated based on current load state and task queue information. When predicted probabilities exceed thresholds, the algorithm initiates proactive migration strategies that select low-load virtual machines as targets, achieving load rebalancing through dynamic Million Instructions Per Second (MIPS) resource reallocation.

3. Results

3.1. Experimental Environment and Dataset

To evaluate the efficiency of the proposed algorithm, this paper develops a cloud computing simulation system through the CloudSim 3.0.3 simulation package. The experimental system is configured with a heterogeneous pool of virtual machine resources, and the number of virtual machine resources varies from 2 to 6 to evaluate the adaptability of the algorithm at different scales. Processing capabilities of each virtual machine are assigned between 1000 and 3000 MIPS to mirror the heterogeneous characteristics seen in real cloud environments. Memory capacity is assigned within the 2048-8192 MB range, and the storage capacity varies between 10000 and 50000 MB. Task characteristics are derived from Google Cluster Trace production workload patterns recorded over a 29-day monitoring period. In this dataset, task length distribution (10000-100000 MI) maps CPU usage percentiles to instruction counts, priority levels (1-5) correspond to dataset resource allocation hierarchies, while a Poisson arrivals model temporal distribution patterns.

Algorithm 1: Distributed project resource allocation optimization algorithm

Input: Task set T, VM set V, Parameters $(MIPS_j, D_i, P_i, A_i)$

Output: Allocation scheme X, Migration records M

Initialize: $MIPS_j \leftarrow TotalMIPS / m$, $Load_j \leftarrow 0$, $Threshold \leftarrow 0.75$

while $T \neq \varnothing$ do

$t_i \leftarrow$ Get next task by A_i // Arrival Time Awareness

for each $v_j \in V$ do // Dynamic Load Assessment

Compute: $C_{ij} \leftarrow A_i + \frac{\sum L_k + L_i}{MIPS_j}$, $V_{ij} \leftarrow \max(0, C_{ij} - D_i) \cdot P_i$

end for

$v_{target} \leftarrow \arg\min_j \{V_{ij}\}$ subject to capacity constraint

if v_{target} exists then Allocate t_i to v_{target}

Else // Active Migration Mechanism

Identify: $v_{high} \leftarrow \max\{Load_j\}$, $v_{low} \leftarrow \min\{Load_j\}$

if migration feasible then

Select $T_{migrate} \leftarrow$ low-priority tasks from v_{high}

Reallocate: $MIPS_{high/low} \leftarrow MIPS_{high/low} \mp \Delta$ where $\Delta = \eta(Load_{high} - Load_{low})TotalMIPS$

Migrate $T_{migrate}$: $v_{high} \rightarrow v_{low}$, allocate t_i to v_{low}, record in M

else t_i to waiting queue, increase P_i

end if

end if

end while

return X, M

To illustrate the heterogeneous nature of the experimental workload concretely, a representative sample of ten task instances drawn from the Google Cluster Trace dataset is presented in Table 2, covering the full range of task lengths, priority levels, arrival patterns, and deadline constraints encountered during simulation.

Table 2 demonstrates the heterogeneous nature of the experimental task workload. High-priority tasks such as T1 and T6 carry tight deadline constraints under a slack factor of 1.5, requiring rapid allocation responses, whereas low-priority tasks such as T3 and T9 operate under more relaxed deadline windows with slack factors of 3.0, providing greater scheduling flexibility. Task lengths span from 11,400 MI to 91,200 MI, reflecting the computational diversity of real production workloads. The arrival times follow a Poisson distribution to simulate dynamic workload scenarios, and task priorities are set to integer values from 1 to 5. The deadline is calculated according to Eq. (6).

$$D_i = A_i + \alpha \cdot \frac{L_i}{AvgMIPS} \tag{6}$$

Where $\text{AvgMIPS} = \sum_{j=1}^{m} \text{MIPS}_j / m$ represents the average processing capacity across all VMs, and α is the slack factor ranging from 1.5 to 3.0 to simulate service quality requirements of different urgency levels. The load threshold parameter is set to 0.75, and the migration sensitivity coefficient η is set to 0.3. Table 3 provides detailed configuration information for all key parameters.

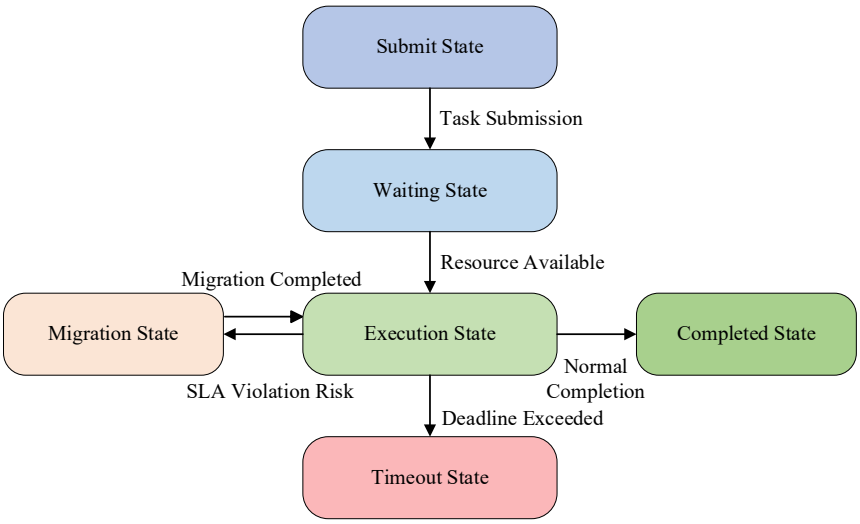


Fig. 2. Task lifecycle state transition diagram

Table 2. Representative sample of task instances from the experimental dataset

Task ID	Length L_i (MI)	Priority P_i	Arrival time A_i (s)	Slack factor α	Deadline D_i (s)
T1	15,200	5	0.0	1.5	11.4
T2	43,800	3	4.7	2.0	48.5
T3	72,500	1	9.3	3.0	118.6
T4	28,600	4	15.1	1.5	36.5
T5	91,200	2	21.8	2.5	135.8
T6	11,400	5	26.4	1.5	34.9
T7	56,300	3	33.2	2.0	89.4
T8	38,700	4	39.8	1.5	68.8
T9	83,100	1	45.6	3.0	170.1
T10	24,900	2	51.2	2.0	76.1

Table 3 shows the experimental configurations with virtual machines at the low, medium, and high-performance levels. The task parameters are based on the workload characteristics of Google Cluster Trace, and the algorithm parameters are set for a balance between performance and overhead. Each of these scenarios is simulated 30 times using a random seed between 1001 and 1030 for the Poisson distribution used in the arrival process. A warm-up period of 50 seconds is used before gathering performance metrics. The performance metrics provided are averages of all runs, and standard deviation measures are used to report dispersion.

3.2. Baseline Algorithms and Experimental Scenarios

This paper selects four representative algorithms as the comparison benchmarks. The First-Come-First-Served algorithm (FCFS) allocates resources based on the arrival time of tasks, which is simple but cannot consider the quality of service requirements. The Round Robin algorithm uses a time slice rotation mechanism to achieve load distribution. The Dynamic Load Balancing algorithm (Dynamic LBA) sorts and allocates tasks based on task length and virtual machine load, which can optimize the completion time but lacks deadline constraint handling. The Min-Min heuristic algorithm selects tasks with the shortest completion time for priority scheduling and performs well for small-scale tasks. Three sets of comparison scenarios are designed in the experiment. Algorithm performance is evaluated by varying or halving the number of virtual machines, 2, 4, or 6, in various scales of resources. To comprehend the transition of the level of performance, the workload of the tasks gradually increased from 10 to 40 tasks. The time interval of the incoming tasks is kept exponentially distributed, controlled by an average time interval of 5 seconds.

3.3. Performance Comparison Experiment Results

The completion time is the primary metric for judging the scheduling algorithm's efficiency, and the completion time directly indicates the system's total time to finish all tasks. To confirm the proposed dynamic load estimation mechanism's ability to reduce the completion time, an experimental setting is established where the resource environment consists of 4 virtual machines. Tasks are increased step by step from 10 tasks to 40 tasks, emulating the full scenario of light load and heavy load. The experiment displays the completion time trend of each algorithm by using a line graph, where the completion time trend of each algorithm varies with the increasing task load, as presented in Fig. 3.

Table 3. Experimental parameter configuration and dataset specification

Parameter category	Parameter	Value/range	Description
Simulation Platform	Simulator	CloudSim 3.0.3	Cloud computing simulation framework
	Operating System	Ubuntu 20.04 LTS 64-bit	Simulation environment
Virtual Machine Configuration	Number of VMs	2, 4, 6	Variable VM scale
	Processing Capacity	1000-3000 MIPS	Heterogeneous performance levels
	Memory	2048-8192 MB	Low/Medium/High configurations
	Storage	10000-50000 MB	Variable storage capacity
	Bandwidth	1000 Mbps	Network bandwidth
Task Parameters	Dataset Source	Google Cluster Trace	Real workload characteristics
	Number of Tasks	10, 20, 30, 40	Variable task load
	Task Length	10000-100000 MI	Heterogeneous task complexity
	Arrival Pattern	Poisson Distribution	Dynamic workload simulation
	Arrival Interval (Mean)	5 seconds	Average inter-arrival time
	Priority Level	1-5	Integer priority values
	Deadline Formula	$D_i = A_i + \alpha \cdot (L_i/AvgMIPS)$	QoS constraint
	Slack Factor (α)	1.5-3.0	Deadline tightness level
Algorithm Parameters	Load Threshold (T_{load})	0.75	VM overload threshold
	Migration Sensitivity (η)	0.3	Migration trigger coefficient
	Weight w1 (Priority)	0.4	Priority weight
	Weight w2 (Deadline)	0.3	Deadline urgency weight
	Weight w3 (Length)	0.3	Task length weight

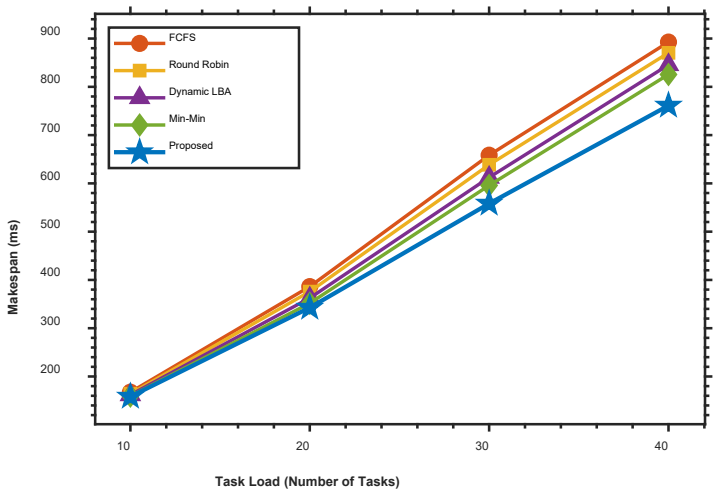


Fig. 3. Comparison of Makespan under different VM configurations and task loads

Fig. 3 shows that the makespan of the proposed algorithm is lower than that of the four benchmark algorithms in all task load scenarios. As the number of tasks increases from 10 to 40, the completion time of each algorithm shows a linear growth trend, but the growth rate of the proposed algorithm is the slowest. The performance gap between the proposed algorithm and the benchmark algorithms is more significant in heavy load scenarios, verifying the effectiveness of the

dynamic load assessment mechanism in reducing completion time. The 7.8% makespan reduction in comparison to Min-Min allows data centers to serve more workloads on existing infrastructure without capacity expansion, while reducing response times for time-critical applications, thus improving operational efficiency.

The primary goal of resource allocation optimization is to improve the utilization of resources and help cloud service providers serve more user requests with a limited hardware investment. To verify the performance of the proposed algorithm in terms of the efficiency of resource utilization, multiple virtual machine configuration scenarios are designed. The differences in resource utilization of each algorithm in 2-, 4-, and 6-VM environments are evaluated. It verifies that the active migration mechanism can avoid resource idleness by dynamically adjusting the load distribution, especially when the number of virtual machines increases, and the configuration complexity rises, and whether the proposed algorithm can still maintain efficient resource utilization. Fixing the task number at 40, the following experiment compares different VM configurations using grouped bar charts, as shown in Fig. 4.

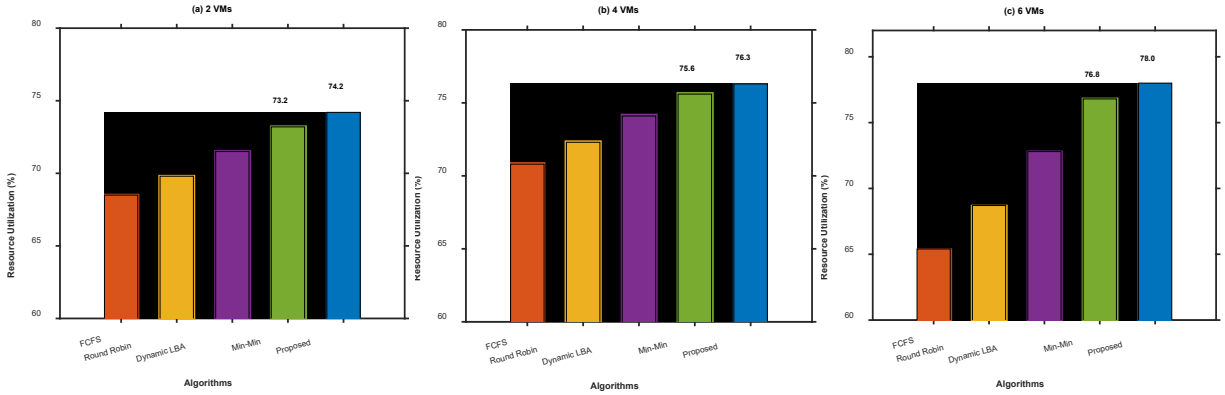


Fig. 4. Comparison of resource utilization in various scenarios

Specifically, the proposed algorithm achieves a remarkable 74.2% under the 2-VM setting, as evident from Fig. 4(a). Fig. 4(b) shows an even better result, where the proposed algorithm achieves 76.3% under the 4-VM setting. As shown in Fig. 4(c), the proposed algorithm achieves 78.0% under the 6-VM setting. From these results, the proposed algorithm is optimal for all three settings, which confirms its excellent ability to optimize.

To analyze whether the proposed algorithm introduces excessive computational overhead while ensuring service quality, an average execution time comparison test is designed. This experiment quantifies the additional time overhead brought by dynamic load assessment and violation cost calculation. A comparison with simple scheduling algorithms and complex heuristic algorithms identifies the performance optimization-computational overhead trade-off, as shown in Fig. 5.

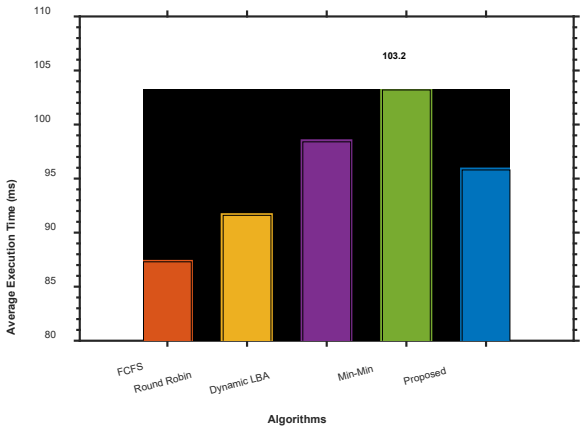


Fig. 5. Comparison of average execution times for different algorithms

Fig. 5 shows that the execution time of FCFS is the lowest at 87.3ms, while the proposed algorithm takes 95.8ms. Compared to the latter, it increases by 9.7% but is lower than the 103.2ms of Min-Min, achieving a better balance between performance and cost.

The managerial implications of execution time are particularly significant in operational decision-making contexts. In cloud data center environments, scheduling decisions are executed continuously and at high frequency, meaning that per-scheduling execution time directly determines system responsiveness under time-sensitive workload conditions. The 9.7% execution time overhead introduced by the proposed algorithm relative to FCFS, amounting to approximately 8.5ms per scheduling cycle, represents an acceptable operational cost, given that the algorithm simultaneously achieves up to 78.0%

resource utilization across VM configurations and reduces Makespan by 7.8% compared to Min-Min. Cloud service operations managers can interpret this trade-off as a favorable exchange: a marginal increase in per-decision computational latency yields substantially higher infrastructure utilization and faster task completion, improving the return on hardware investment without requiring additional capacity expansion.

3.4. In-Depth Analysis of Algorithm Performance

Service quality guarantee is the core competitiveness of cloud computing systems. SLA violation leads to user attrition and economic losses. To verify the effectiveness of the proactive migration mechanism in preventing SLA violations, a detailed statistical analysis of violations is designed in the experiment. It records the number of violations, violation rates, and the execution of migration operations for each algorithm in three load scenarios of light, medium, and heavy. Through comparing the performance differences with and without the migration mechanism, this experiment quantifies the improvement effect of the proactive migration strategy on the SLA satisfaction rate and analyzes key parameters such as migration success rate and migration frequency to evaluate the stability and practicality of the migration mechanism. The multi-dimensional statistical data are presented in tabular form, as shown in Table 4.

Table 4. SLA violations and migration statistics (mean ± sd, n=30)

Algorithm	Task load	Violation rate (%)	Migration count	Migration success rate (%)
FCFS	40	22.3±2.3	N/A	N/A
Round Robin	40	19.8±2.2	N/A	N/A
Dynamic LBA	40	18.5±1.9	N/A	N/A
Min-Min	40	17.6±1.8	N/A	N/A
Proposed	10	5.0±0.7	1.2±0.3	96.7±1.5
Proposed	20	8.5±1.2	3.8±0.7	95.2±1.8
Proposed	40	11.8±1.6	8.6±1.2	93.5±2.1

Table 4 shows that the proposed algorithm achieves a violation rate of 11.8 ± 1.6%, whereas FCFS achieves 22.3 ± 2.3%. The proposed algorithm achieves 8.6 ± 1 migration events across locations on average, with a success rate of 93.5 ± 2.1%, thereby ensuring successful migrations even in high-load conditions. The efficiency class balance rate for light, medium, and heavy loads justifies the proposed algorithm's effectiveness in handling migrations. The reduction in SLA violation rate by 47.1%, from 22.3% to 11.8%, has significant economic consequences for Cloud Service Providers (CSPs), such as lower monetary fines, higher customer loyalty, and a stronger service reputation. Additionally, the 93.5% migration success rate under heavy load indicates service reliability for production scenarios.

The load balancing concept represents a key goal in optimizing resources. The load imbalance impacts stability as well as efficiency in resources. A comparative analysis of virtual machine load distribution reveals that an active migration approach can achieve load balancing through real-time monitoring of status to dynamically adjust task distribution. An experiment is conducted using six virtual machines to execute forty tasks, with load balancing characteristics represented through box plots, as shown in Fig. 6.

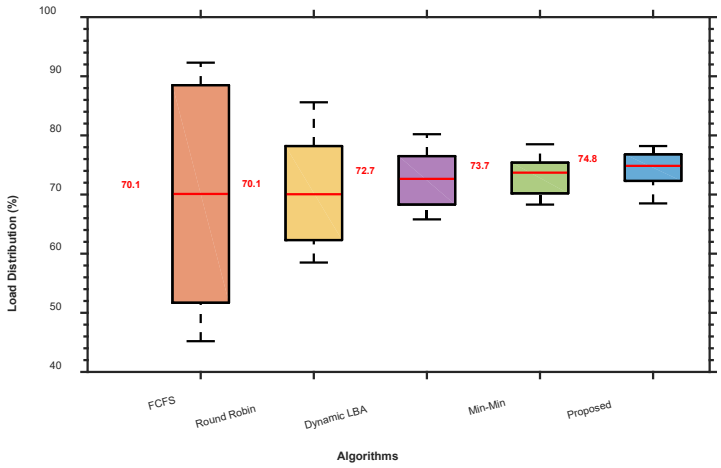


Fig. 6. Comparison of virtual machine load distribution

As depicted in Fig. 6, the box plots represent the characteristics of the load distribution for the algorithms on 6 virtual machines. The box for the FCFS algorithm is the tallest, with a median value of 70.1. However, the box for the proposed algorithm is the narrowest, with a median value of 74.8. This indicates that the active migration mechanism successfully balances the load distribution for the virtual machines, reducing severe imbalance.

A comprehensive performance evaluation requires assessing the algorithm from various aspects at the same time. A single indicator is not sufficient to reflect the overall advantages of the algorithm. To make the performance comparison

of the algorithm intuitive, the experiment selects five key dimensions, including completion time, resource utilization, satisfaction rate of SLA, execution time, and balance degree, to build the evaluation framework. All indicators are normalized such that higher values indicate higher performance. A grouped bar chart is used to plot multiple performance indicators for all comparison algorithms at once, allowing for a visual comparison of normalized values for all performance indicators, as shown in Fig. 7.

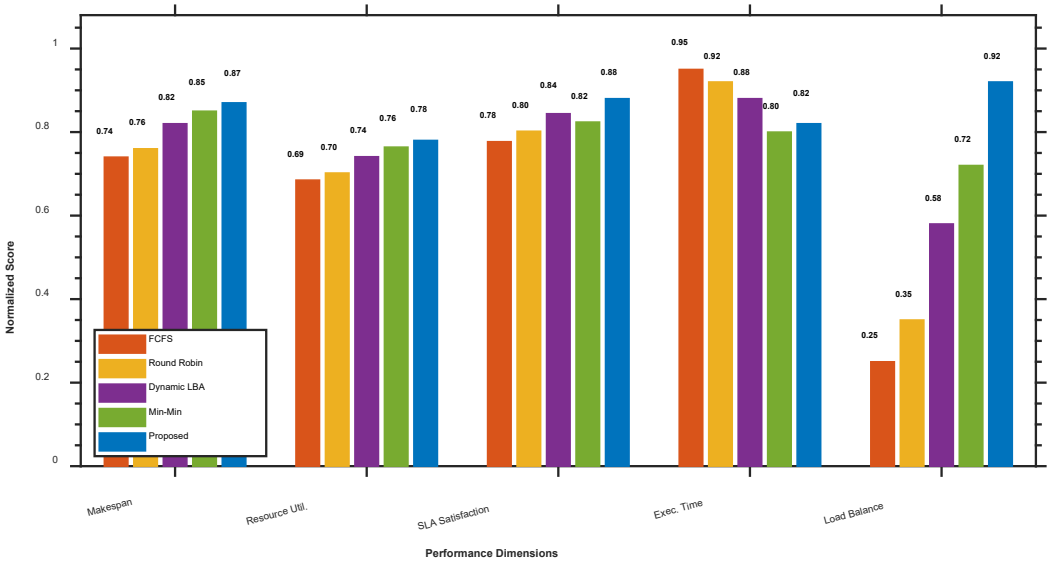


Fig.7. Normalized multi-dimensional performance comparison across algorithms

Fig. 7 shows the normalized score of each algorithm on five performance dimensions by means of a grouped bar chart, where a higher score indicates better performance. Notably, the proposed algorithm achieves the highest normalized score on four performance dimensions: completion time (0.87), resource utilization (0.88), SLA satisfaction rate (0.95), and load balancing degree (0.92). For execution time, the proposed algorithm achieves a score of 0.82, implying a small computational overhead of dynamic load evaluation and active migration. The algorithm does not achieve the minimum execution time, as FCFS achieves 87.3ms compared to 95.8ms. This indicates a trade-off between execution time and improved service quality. Hence, cloud operation managers can select an appropriate algorithm based on their operational requirements, and the proposed algorithm can be used in scenarios where contractual SLA requirements are considered in heterogeneous workload scenarios.

3.5. Comprehensive Performance Evaluation

To comprehensively evaluate the entire performance of the proposed algorithm and specifically delineate the advantages of the benchmark algorithm, all significant indices are required to be integrated and contrasted accordingly. This extensive survey integrates the principal data of the earlier experiments explained, providing an entire view of the comparison of the performance. Apart from providing the average data of the execution time and CPU time derived from 40 tasks and 4 virtual machine scenarios, the table even adds an algorithm features column, through which the design concepts and technical features of each algorithm can be contrasted, as shown in Table 5.

Table 5 demonstrates better performance of the present algorithm on three major indices: completion time, resource utilization, and rate of violating SLA. Compared to the Min-Min algorithm, the completion time decreases by 7.8%, the rate of resource utilization improves by 1.6 percentage points, and the rate of violating SLA decreases by 33.1%. The simultaneous achievement of a 1.6 percentage point increase in resource utilization and a 33.1% reduction in SLA violations demonstrates synergistic optimization effects, in which enhanced service quality and operational efficiency reinforce rather than compromise each other, validating the multi-objective optimization framework's capacity to balance competing performance dimensions. The quantitative information provided in Table 5 is helpful for the selection of evidence-based algorithms, allowing decision-makers to compare the makespan, resource usage, SLA violation rate, and execution time for all algorithms. In this regard, the best algorithm can be chosen based on the needs rather than relying on a single parameter.

4. Discussion

The experimental results confirm that the proposed algorithm is effective for multi-objective optimization. An overall makespan reduction by 7.8% is obtained from the violation cost function incorporating deadline constraints and task priorities, while the resource utilization reaches 78.0%, and load variance is reduced by 72.8% due to the active migration mechanism. The SLA violation rate is reduced from 22.3% to 11.8%, and migration success rates of 96.7%, 95.2%, and 93.5% are obtained for light, medium, and heavy load scenarios, respectively. The active migration mechanism introduces computational overheads regarding state transfer, context switching, and resource reallocation processes, reflected in the modest 9.7% increase in execution time. This is justified by the significant reduction in SLA violations and improved load

distribution because the mechanism performs selective migrations of only low-priority tasks when the risk of violations exceeds the threshold, allowing unnecessary migrations to be minimized while amplifying improvements in QoS. The baseline algorithms represent different scheduling paradigms-FCFS represents an arrival-order policy, Round Robin represents time-sharing mechanisms, Min-Min represents greedy heuristics, while Dynamic LBA represents load-aware mechanisms, which enables a systematic evaluation of priority-driven migration against fundamental approaches. In contrast to machine learning methods, the proposed algorithm offers deterministic behavior, which is important for SLA guarantees without training data or convergence of models. Deep learning research shows benefits under intricate tasks with hierarchical federated learning for device orchestration (Zhao et al., 2022), reinforcement learning for energy optimization (Li et al., 2023; Chen et al., 2024; Frank and Rutzen, 2024), and self-healing frames for Kubernetes clusters (Kaul, 2024). However, these methods are based on neural networks overseeing high-quality state spaces, which produce a large-scale computational load and data dependency. The proposed algorithm has the advantage of easily accommodating interpretable explicit priority functions and violation cost functions while retaining a straightforward way of obtaining solutions amenable to implementation in practice. The proposed method exhibits merits in settings with deterministic latency and algorithmic transparency for its two-parameter tuning instead of the multi-dimensional hyperparameter optimization task needed by the machine learning context. It performs robustly across diverse workload patterns without the need for training data demand, while existing neural network methods may achieve better performance on well-behaved traces with abundant history.

Table 5. Comparative analysis of comprehensive performance of different algorithms

Algorithm	Type	Key features	Makespan (ms)	Resource util. (%)	SLA violation (%)	Execution time (ms)
FCFS	Classic	First-come-first-serve	892.7	68.5	22.3	87.3
Round Robin	Classic	Time-slice rotation	870.4	70.2	19.8	91.6
Dynamic LBA	Recent	Length-based sorting	845.6	74.1	18.5	98.4
Min-Min	Heuristic	Min completion time	825.6	76.4	17.6	103.2
Proposed	QoS-aware	Priority + Migration	761.2	78.0	11.8	95.8

Research into hybrid optimization algorithms provides innovative solutions by integrating heuristic strategies (Mittal et al., 2024), while the proposed algorithm addresses the challenges brought about by random task arrivals using mechanisms that are time-aware for task arrivals. Blockchain-federated learning integration provides secure aggregation of distributed resources (Liu et al., 2024), and adaptive federated learning strategies optimize resource-limited IoT devices (Mughal et al., 2024; Liu et al., 2025). These targets are designed for edge computing scenarios that emphasize privacy protection and distributed cooperation, while the proposed algorithm focuses on QoS guarantees for cloud data centers. The active migration mechanism reduces SLA violation rates by 47.1% under heavy loads while maintaining 93.5% migration success, validating the effectiveness of dynamic assessment and predictive migration for QoS management. The experiment conducted on 6 virtual machines and 40 tasks only demonstrates the algorithmic correctness of the proposed method, although in real-life production environments much larger scales are used. Furthermore, the polynomial complexity of $O(n \times m)$ and a threshold-based selective migration mechanism suggest favorable scalability when scaled up, while maintaining analytical tractability as system scale increases.

5. Conclusion

In this study, distributed resource allocation optimization for project tasks in cloud computing is explored using a hybrid algorithm that uses dynamic load assessment with active migration mechanisms. The multi-objective optimization mechanism includes deadline constraints, priority-based tasks, and virtual machine loads to optimize tasks with adaptive scheduling mechanisms that are sensitive to arrivals. The evaluation using CloudSim shows significant benefits: a 7.8% reduction in completion time compared to the Min-Min algorithm, a resource utilization level of 78.0%, a reduction in SLA violation rate from 22.3% to 11.8%, and a 72.8% reduction in load variance. The success rate for migrations is between 93.5% and 96.7% for different loads.

This algorithm offers practical solutions to cloud service providers in terms of resource management in a data center, which includes predictive SLA control and dynamic balancing. Cloud management operation personnel using static allocation methods should consider priority scheduling methods, especially since reactive scheduling methods always yield SLA violation rates of more than 19% compared to 11.8% in the suggested algorithm. In addition, capacity planning management can utilize the active migration method for better resource utilization and to prevent premature hardware upgrades. The algorithm develops a direct link between scheduling decisions, service quality outcomes, and infrastructure cost efficiency, which transforms the problem of resource allocation from a purely technical issue to a managerially oriented framework. This provides a theory as well as a technology for intelligent resource scheduling and assurance for service quality in cloud computing. There are a few restrictions to be considered. Validation is based on simulations without

empirical evaluation on real infrastructure; experimental size is limited in comparison to a realistic scenario, with a maximum of 6 VMs active and forty concurrent tasks, and energy efficiency is not considered. The focus on Makespan, resource utilization, and SLA compliance considers service level guarantees more than energy optimization objectives. Energy-aware extensions would also have to encompass models of power consumption and hardware-specific parameters that are beyond the ability of current CloudSim simulations. Probable future directions for research include validation of our results on a real-world testbed, scalability evaluation over large clusters, multi-objective amalgamation in an energy-augmented context, and handling of heterogeneous traffic patterns beyond the Poisson process assumptions.

Author Contributions

Yalan Yang contributes to conceptualization, software, validation, draft preparation, manuscript editing and supervision. Xiaoming Wu contributes to conceptualization, methodology, investigation, data collection and manuscript editing. All authors have read and agreed with the manuscript before its submission and publication.

Funding

This research received no specific financial support from any funding agency.

Institutional Review Board Statement

Not applicable.

Declaration of Artificial Intelligence (AI) Tools

The authors used DeepSeek solely for language editing and readability improvement. The authors reviewed and verified all content and take full responsibility for the accuracy and integrity of the manuscript.

References

Abreha, H. G., Hayajneh, M., and Serhani, M. A. (2022). Federated learning in edge computing: a systematic survey. *Sensors*, 22(2), 450.

Alruwais, N., Alabdulkreem, E., Kouki, F., Aljehane, N. O., Allafi, R., Marzouk, R., Assiri, M., and Alneil, A. A. (2024). Farmland fertility algorithm based resource scheduling for makespan optimization in cloud computing environment. *Ain Shams Engineering Journal*, 15(6), 102738.

Cassel, G. A. S., Rodrigues, V. F., da Rosa Righi, R., Bez, M. R., Nepomuceno, A. C., and da Costa, C. A. (2022). Serverless computing for Internet of Things: A systematic literature review. *Future Generation Computer Systems*, 128, 299-316.

Chen, J., Du, T., and Xiao, G. (2021). A multi-objective optimization for resource allocation of emergent demands in cloud computing. *Journal of Cloud Computing*, 10(1), 20.

Chen, Z., Xiong, B., Chen, X., Min, G., and Li, J. (2024). Joint computation offloading and resource allocation in multi-edge smart communities with personalized federated deep reinforcement learning. *IEEE Transactions on Mobile Computing*, 23(12), 11604-11619.

Czarnul, P., Antal, M., Baniata, H., Griebler, D., Kertesz, A., Kessler, C. W., Kouloumpris, A., Kovačić, S., Markus, A., Michael, M. K., and Nikolaou, P. (2025). Optimization of resource-aware parallel and distributed computing: a review: P. Czarnul et al. *The Journal of Supercomputing*, 81(7), 848.

Dankolo, N. M., Radzi, N. H. M., Mustaffa, N. H., Arshad, N. I., Nasser, M., Gabi, D., and Yusuf, M. N. (2025). Optimizing resource allocation for IoT applications in the edge cloud continuum using hybrid metaheuristic algorithms. *Scientific Reports*, 15(1), 14409.

Doostmohammadian, M., Aghasi, A., Pirani, M., Nekouei, E., Zarrabi, H., Keypour, R., Rikos, A. I., and Johansson, K. H. (2025). Survey of distributed algorithms for resource allocation over multi-agent systems. *Annual Reviews in Control*, *59*, 100983.

Frank, R. and Rutzen, W. (2024). AI-Based Intelligent Offloading for Energy Optimization in IIoT and Cloud Systems.

Jena, U. K., Das, P. K., and Kabat, M. R. (2022). Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment. *Journal of King Saud University-Computer and Information Sciences*, 34(6), 2332-2342.

Kaul, D. (2024). AI-Driven Self-Healing Container Orchestration Framework for Energy-Efficient Kubernetes Clusters. *Emerg. Sci. Res,* 2024, 1-13.

Ko, H., Pack, S., and Leung, V. C. (2021). Performance optimization of serverless computing for latency-guaranteed and energy-efficient task offloading in energy-harvesting industrial IoT. *IEEE Internet of Things Journal,* 10(3), 1897-1907.

Kumari, A., Sahoo, B., and Behera, R. K. (2023). Workflow aware analytical model to predict performance and cost of serverless execution. *Concurrency and Computation: Practice and Experience,* 35(22), e7743.

Li, X., Zhang, J., and Pan, C. (2023). Federated deep reinforcement learning for energy-efficient edge computing offloading and resource allocation in industrial internet. *Applied Sciences,* 13(11), 6708.

Liu, Y., Liu, D., Khoukhi, L., Wang, B., Zhang, L., and Xu, Y. (2025). Energy-efficient device selection and resource allocation for edge-driven hierarchical federated learning. *Results in Engineering*, 27, 105992.

Liu, Y., Jia, Z., Jiang, Z., Lin, X., Liu, J., Wu, Q., and Susilo, W. (2024). BFL-SA: Blockchain-based federated learning via enhanced secure aggregation. *Journal of Systems Architecture*, 152, 103163.

Mahmud, F., Orthi, S. M., Saimon, A. S. M., Moniruzzaman, M., Miah, M. A., Ahmed, M. K., Khair, F. B., Islam, M. S., and Manik, M. M. T. G. (2025). *Big data and cloud computing in IT project management: A framework for enhancing performance and decision-making* [online]

Malyshkin, V. (2022). Parallel computing technologies 2020. *The Journal of Supercomputing,* 78(4), 6056-6059.

Mittal, P., Kumar, S., and Sharma, S. (2024). Revolutionizing Cloud-Based Task Scheduling: A Novel Hybrid Algorithm for Optimal Resource Allocation and Efficiency in Contemporary Networked Systems. *International Journal of Computing and Digital Systems,* 15, 1551-1563. http://dx.doi.org/10.12785/ijcds/1501110

Mughal, F. R., He, J., Das, B., Dharejo, F. A., Zhu, N., Khan, S. B., and Alzahrani, S. (2024). Adaptive federated learning for resource-constrained IoT devices through edge intelligence and multi-edge clustering. *Scientific Reports*, 14(1), 28746.

Naayini, P. (2025). Building ai-driven cloud-native applications with kubernetes and containerization. *International Journal of Scientific Advances*, (IJSCIA), 6(2), 328-340.

Peng, J., Su, Z., and Liu, X. (2025). Multi skill project scheduling optimization based on quality transmission and rework network reconstruction. *Scientific Reports*, 15(1), 13545.

Peng, J., Su, Z., and Liu, X. (2025). Multi skill project scheduling optimization based on quality transmission and rework network reconstruction. *Scientific Reports*, 15(1), 13545.

Pratama, I. N., Dachyar, M., and Pratama, N. R. (2023). Optimization of resource allocation and task allocation with project management information systems in information technology companies. *TEM Journal,* 12(3), 1814.

Raeisi-Varzaneh, M., Dakkak, O., Fazea, Y., and Kaosar, M. G. (2024). Advanced cost-aware Max–Min workflow tasks allocation and scheduling in cloud computing systems. *Cluster Computing*, 27(9), 13407-13419.

RM, S. P., Bhattacharya, S., Maddikunta, P. K. R., Somayaji, S. R. K., Lakshmanna, K., Kaluri, R., Hussien, A., and Gadekallu, T. R. (2020). Load balancing of energy cloud using wind driven and firefly algorithms in internet of everything. *Journal of Parallel and Distributed Computing*, 142, 16-26.

Shafiq, D. A., Jhanjhi, N. Z., Abdullah, A., and Alzain, M. A. (2021). A load balancing algorithm for the data centres to optimize cloud computing applications. *IEEE Access*, 9, 41731-41744.

Singh, N., Hamid, Y., Juneja, S., Srivastava, G., Dhiman, G., Gadekallu, T. R., and Shah, M. A. (2023). Load balancing and service discovery using Docker Swarm for microservice based big data applications. *Journal of Cloud Computing*, *12*(1), 4.

Singhal, S., Ali, S., Awasthy, M., Shukla, D. K., and Tiwari, R. (2024). Rock-hyrax: An energy efficient job scheduling using cluster of resources in cloud computing environment. *Sustainable Computing: Informatics and Systems*, *42*, 100985.

Somayeh, D. (2025). Resource Allocation in Cloud Computing Systems Using Game Theory. *Pegem Journal of Education and Instruction,* 15(2), 257-267. https://www.pegegog.net/index.php/pegegog/article/view/4283

Toosi, A. N., Javadi, B., Iosup, A., Smirni, E., and Dustdar, S. (2025). Serverless computing for next-generation application development. *Future Generation Computer Systems*, 164, 107573.

Wang, Y. and Yang, X. (2025). Intelligent resource allocation optimization for cloud computing via machine learning. *arXiv preprint arXiv,* 2504.03682.

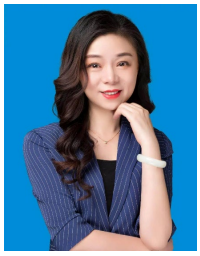
Yu, Y., Xu, Z., and Zhao, S. (2023). A two-stage algorithm based on 12 priority rules for the stochastic distributed resource-constrained multi-project scheduling problem with multi-skilled staff. *IEEE Access*, 11, 29554-29565. doi: 10.1109/ACCESS.2023.3260847

Zhang, J., Wu, Q., Fan, P., and Fan, Q. (2024). A comprehensive survey on joint resource allocation strategies in federated edge learning. *arXiv preprint arXiv:2410.07881*.

Zhao, T., Li, F., and He, L. (2022). DRL-based joint resource allocation and device orchestration for hierarchical federated learning in NOMA-enabled industrial IoT. *IEEE Transactions on Industrial Informatics,* 19(6), 7468-7479.

Zhou, G., Tian, W., Buyya, R., Xue, R., and Song, L. (2024). Deep reinforcement learning-based methods for resource scheduling in cloud computing: A review and future directions. *Artificial Intelligence Review*, 57(5), 124.

Yalan Yang is a doctoral student and lecturer in the School of Economics and Management at Southwest Petroleum University. Her research interests include economics and management, as well as project engineering management.



Xiaoming Wu is a Professor and doctoral supervisor at the School of Economics and Management, Southwest Petroleum University. Prof. Wu specializes in regional and energy economics. She has published 15 papers in high-impact journals, such as SCI and CSSCI, and has independently authored two monographs.