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

The Optimization of Load Balancing Strategies in Heterogeneous Distributed Environments for Regional Industrial Upgrading

Zhigang Guo, Zhen Li*

School of Business, Anyang Institute of Technology, Anyang ,455000, Henan, China

Abstract

INTRODUCTION: With the ongoing advancement of regional industrial upgrading, efficient load balancing in heterogeneous distributed environments has emerged as a pivotal technical challenge. Conventional load balancing methods, typically designed for homogeneous systems or based on static scheduling rules, are ill-equipped to handle the dynamic task demands and diverse resource capabilities characteristic of modern industrial computing infrastructures.
OBJECTIVES: This study seeks to bridge this gap by developing an intelligent, adaptive load balancing strategy specifically tailored for heterogeneous environments, with the goal of maximizing resource utilization, minimizing response latency, and supporting sustainable industrial transformation.
METHODS: The proposed framework synergistically combines real-time dynamic resource monitoring, multi-dimensional task characterization and scheduling, and reinforcement learning-driven decision-making to construct a responsive and self-optimizing load distribution mechanism that continuously adapts to system state changes.
RESULTS: Comprehensive experiments show that the approach significantly outperforms traditional strategies (e.g., round-robin, shortest job first) and state-of-the-art deep reinforcement learning methods across key performance metrics ($p < 0.05$), achieving up to a 44% reduction in average response time and a remarkable 91.4% system-wide resource utilization.
CONCLUSION: By enhancing both adaptability and efficiency in complex heterogeneous settings, the proposed strategy offers a practical and scalable solution with strong potential for deployment in smart manufacturing, cloud computing, edge computing, and other industrial digitalization scenarios.

Keywords: Heterogeneous distributed environments, load balancing optimization, reinforcement learning, regional industrial upgrading, adaptive scheduling.

Received on 26 December 2025, accepted on 23 April 2026, published on 14 May 2026

doi: 10.4108/eetsis.11448

1. Introduction

With the rapid development of information technology and smart manufacturing, regional industrial upgrading has become a key pathway to enhancing industrial competitiveness and driving economic transformation[1][2]. Traditional centralized computing and resource scheduling methods no longer meet increasingly complex industrial demands[3][4]. In scenarios such as industrial Internet, smart manufacturing, and large-scale data processing, resource heterogeneity and distribution introduce new challenges[5]. Achieving efficient load balancing in heterogeneous environments is essential for improving system performance and promoting regional industrial upgrading.

Although various load balancing methods exist, most rely on specific assumptions and simplified metrics. While

*Corresponding author. Email: 20202013@ayit.edu.cn

some techniques aggregate multiple variables into single indicators, they often lose the fine-grained coupling between heterogeneous resource dimensions (e.g., non-linear CPU-I/O relationships) that our multi-dimensional model explicitly captures[6][7]. In complex multi-task and multi-resource environments, existing methods face inaccurate load prediction and inefficient resource allocation[8][9]. Current strategies depend on static rules and experience-based scheduling, ignoring real-time data and environmental changes. Although reinforcement learning and adaptive algorithms have developed, their application in heterogeneous distributed environments faces challenges like high computational overhead and slow convergence[10]. No existing research simultaneously addresses heterogeneous resources, efficient scheduling, and real-time response.

To address this gap, this paper proposes three innovations. First, dynamic task scheduling: a real-time model based on task characteristics and resource states solves the inflexibility of static methods. By analyzing tasks and resources in real time, the strategy dynamically adjusts load distribution, improving adaptability and resource utilization. Second, adaptive load distribution: reinforcement learning optimizes distribution decisions, leveraging online learning for self-optimization in large-scale heterogeneous environments, enhancing scalability and adaptability. Third, multi-dimensional resource integration: an optimization model considers resource heterogeneity and distribution, optimizing task execution, resource scheduling, and load distribution across multiple dimensions.

This research expands the theoretical framework of load balancing in heterogeneous distributed environments and provides a practical technical solution. By combining dynamic scheduling and reinforcement learning, this strategy addresses complex tasks and dynamic resource allocation in regional industrial upgrading. Its adaptive nature enables flexible deployment across industrial environments, driving innovation in smart manufacturing and industrial Internet. Theoretically, it offers new perspectives for distributed system load balancing. Practically, it optimizes industrial resource allocation, improves production efficiency, and reduces operational costs, with broad application prospects.

2. Related Works

2.1. Application Scenarios and Challenges

In the context of regional industrial upgrading, load balancing in heterogeneous distributed environments has become a critical technological challenge for improving production efficiency, optimizing resource utilization, and ensuring system stability[11][12]. With the rapid development of emerging technologies such as smart manufacturing, industrial Internet, cloud computing, and edge computing, the resources in heterogeneous distributed systems are evolving towards diversification and dynamism, making the load balancing problem increasingly complex[13][14]. Particularly in scenarios involving large-scale, high-concurrency, and high-dynamics tasks, achieving efficient load balancing directly impacts production efficiency and system stability.

Typical application scenarios include smart manufacturing, cloud computing, industrial Internet of Things (IoT), and edge computing[15]. In smart manufacturing, production systems consist of various heterogeneous resources, each with different resource types and task characteristics. Load balancing strategies must be dynamically adjusted to ensure the efficient operation of production processes[16]. Cloud computing platforms require resource allocation to manage a high volume of concurrent requests. In edge computing, due to the distribution of computing nodes and resource limitations, load balancing must ensure low latency while avoiding resource overload[17].

Load balancing evaluation typically relies on metrics such as response time, throughput, resource utilization, and task completion time. However, these evaluation frameworks have limitations. For example, many evaluation metrics fail to fully consider the diversity of real-world applications, with a singular focus on response time failing to reflect overall system performance. Existing datasets are often confined to simulation environments and lack in-depth characterization of dynamic load changes in complex industrial settings. As such, current evaluation systems are insufficient in fully reflecting the performance of load balancing strategies in practical applications.

2.2. Overview of Mainstream Methods

Load balancing methods can be broadly categorized into static scheduling methods and intelligent algorithm-based scheduling methods[18]. Static scheduling methods optimize resource allocation through predefined rules or objective functions. They perform well in stable environments with consistent resource demands but fail to adapt to heterogeneous resources and dynamic tasks. For example, round-robin load balancing and shortest job first strategies reduce task queuing time but perform poorly when resource conditions change.

Intelligent algorithm-based load balancing methods have gained widespread attention in recent years, particularly reinforcement learning (RL) and deep learning (DL)[19][20]. Reinforcement learning optimizes scheduling strategies in real-time through interaction with the environment, demonstrating strong adaptability[21][22]. In large-scale complex environments, RL can address dynamic task loads and resource changes. However, these methods suffer from high computational overhead and long training times, especially when dealing with large-scale systems, facing bottlenecks in terms of computational resource consumption and slow convergence.

Some methods combine traditional optimization techniques with intelligent algorithms to improve load balancing performance. While these hybrid approaches show considerable theoretical potential, they still face issues such as slow convergence and high data requirements, particularly in large-scale heterogeneous systems. Consequently, existing load balancing methods continue to encounter performance bottlenecks in large-scale, dynamically changing heterogeneous environments.

2.3. Most Similar Research

In recent years, several load balancing methods based on deep reinforcement learning have attracted significant attention[23]. These methods monitor system states in real time and dynamically adjust resource allocation to optimize load balancing strategies. However, they still face performance bottlenecks, especially in large-scale systems, due to high computational costs and extended training times[24]. In comparison, this paper introduces a load balancing optimization model based on multi-dimensional resource characteristics. This model provides a more refined load distribution strategy by considering resource heterogeneity, task dynamics, and system load states. In contrast to traditional methods, the innovation in this work lies in the integration of dynamic scheduling and reinforcement learning, enabling real-time optimization of the system based on load conditions, thereby improving stability and resource utilization.

While existing studies have made progress in some scenarios, most methods fail to account for the integration of multi-dimensional resource features and the impact of real-time load changes on scheduling strategies[25][26]. This work fills that gap by proposing a multi-dimensional feature-based load balancing strategy, offering a feasible technical solution to the load balancing problem in regional industrial upgrading.

2.4. Summary and Research Gaps

While some progress has been made in the field of load balancing, research on load optimization in large-scale heterogeneous environments remains underexplored. Most studies focus on single-resource or single-task scenarios, with limited research on load balancing in multi-resource, multi-task, and dynamic environments. In particular, real-time adjustment of load balancing strategies to meet the complex resource and task requirements in large-scale systems remains an unsolved issue.

Unlike existing research, this paper introduces a load balancing optimization model based on multi-dimensional resource features, addressing this gap. By combining dynamic scheduling with reinforcement learning, this study not only enhances the adaptability of the scheduling model but also improves computational efficiency and resource utilization in complex environments. This research provides a new methodological framework for solving the load balancing issue in regional industrial upgrading, with broad application prospects.

3. Methodology

3.1. Problem Formulation

In regional industrial upgrading, efficient load balancing in heterogeneous distributed environments is crucial for enhancing system performance and resource utilization. A heterogeneous distributed system comprises multiple computing nodes with varying computational, storage, and network resources, while task requirements change dynamically. Load balancing strategies must adjust in real-time based on resource states and task demands.

Building upon the generalized load balancing formulations[4,10], this section constructs a multi-objective optimization model specifically tailored to the heterogeneous resource constraints of regional industrial upgrading. Assume there are N computing nodes in the system, and the resource state r_i of each node iii consists of the node's computational power, memory, storage, and other resources, which can be represented as a vector:

$$r_i = (r_i^{CPU}, r_i^{Mem}, r_i^{Disk}) \tag{1}$$

where r_i^{CPU}, r_i^{Mem}, and r_i^{Disk} represent the normalized aggregates of computational power, memory capacity, and storage/bandwidth resources of node i, respectively. These indicators serve as representatives of the primary hardware bottlenecks in industrial computing. The task set consists of M tasks, each task j has a resource demand vector t_j:

$$t_j = (t_j^{Computation}, t_j^{Memory}, t_j^{I/O}) \tag{2}$$

where $t_j^{Computation}$, t_j^{Memory}, and $t_j^{I/O}$ represent the computational, memory, and I/O requirements of task j, respectively.

To describe the task assignment decision, we introduce a binary decision variable x_{ij}, which indicates whether task j is assigned to node i:

$$x_{ij} = \begin{cases} 1, & if\ task\ j\ is\ assigned\ to\ node\ i \\ 0, & otherwise \end{cases} \tag{3}$$

The decision variable x_{ij} ensures a discrete mapping in the heterogeneous task-node space.The objective of the system is to allocate tasks in such a way that the resource utilization of the nodes is maximized while ensuring load balancing. To this end, the load L_i of node i is defined as the total resource demand of the tasks assigned to node i:

$$L_i = \sum_{j=1}^{M} x_{ij} \cdot t_j \tag{4}$$

The optimization goal is to minimize the load difference between nodes to ensure a uniform load distribution. The optimization objective can be expressed as:

$$\min \Delta L = \max_{1 \le i \le N} L_i - \min_{1 \le i \le N} L_i \quad (5)$$

The preliminary objective is to minimize the load disparity between nodes. As defined in Eq. (5), this optimization is reflected by the minimization of the range between maximum and minimum node loads, which directly promotes system-wide equilibrium. Additionally, to ensure the rationality of task scheduling, the following constraints must be satisfied: each task can only be assigned to one node:

$$\sum_{i=1}^{N} x_{ij} = 1 \quad \forall j \quad (6)$$

Furthermore, the load on each node cannot exceed its resource capacity:

$$\sum_{j=1}^{M} x_{ij} \cdot t_j \le r_i \quad \forall i \quad (7)$$

In summary, the optimization goal of the load balancing problem is to minimize the load differences between nodes while ensuring that each task is assigned to an appropriate node and that node resource constraints are satisfied. This mathematical model allows for the optimization of load balancing and resource utilization efficiency while respecting resource constraints.

3.2. Overall Framework

The proposed load balancing optimization framework consists of three core modules: the task scheduling module, the resource monitoring module, and the optimization decision module. Figure 1 illustrates the data flow and the relationships between the modules. The task scheduling module receives task demands and resource status information and performs initial task allocation based on task characteristics and node resources. The resource monitoring module continuously tracks the resource utilization of each node to ensure that resource constraints are met during the task allocation process. The optimization decision module dynamically optimizes the load balancing strategy based on reinforcement learning algorithms, adjusting the task scheduling plan in real-time based on feedback to enhance load balancing performance.

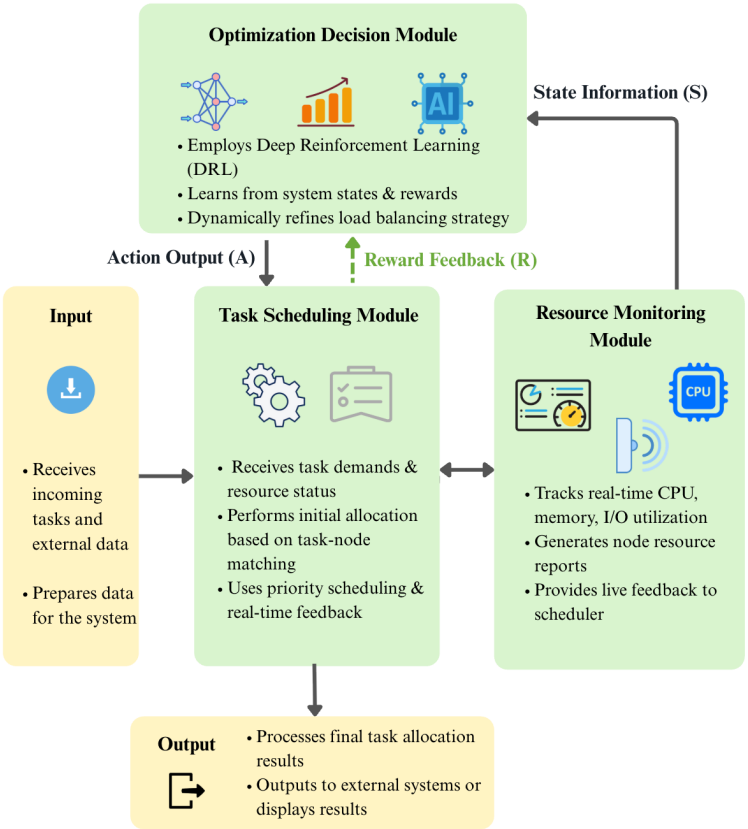

Figure 1. Architecture of the proposed load balancing optimization framework.

In this framework, the task scheduling module and the resource monitoring module work closely together. The task scheduling module uses real-time data provided by the resource monitoring module to allocate tasks and

continuously adjusts its strategy to respond to dynamic load changes. The optimization decision module leverages historical task data and system feedback to refine the load balancing strategy, thereby enhancing the system's adaptability. The modules communicate via data flow, where the task scheduling module provides the initial task allocation, the resource monitoring module updates resource states, and the optimization decision module adjusts the strategy, ensuring the system operates efficiently while maintaining load balancing.

Overall, this framework ensures that the system achieves efficient task scheduling and load balancing in complex heterogeneous environments. It can adapt to dynamic changes in task and resource demands, thereby improving system performance and stability.

3.3. Detailed Description of Modules

Task Scheduling Module

Motivation:

The task scheduling module is responsible for appropriately allocating tasks to different computing nodes to ensure efficient resource utilization and load balancing. In heterogeneous distributed environments, the dynamic nature of task and resource demands requires scheduling schemes to possess high flexibility and adaptability.

Principle:

This module dynamically adjusts task allocation based on the computational, memory, and I/O demands of tasks, as well as the resource status of the nodes. By evaluating the match between tasks and nodes through a weighting system, it considers task priority and node load to allocate tasks rationally. Real-time feedback ensures that the task scheduling adapts to changes in load.

Implementation:

The task scheduling module uses a priority scheduling strategy to assign tasks to appropriate nodes. The resource demands of each task are transformed into resource vectors through task preprocessing. The scheduling algorithm allocates tasks by calculating the matching degree between the task and the node's resources. Figure 2 illustrates the cooperation between task queue management, resource evaluation, and the scheduling algorithm, ensuring the efficiency of dynamic task scheduling.

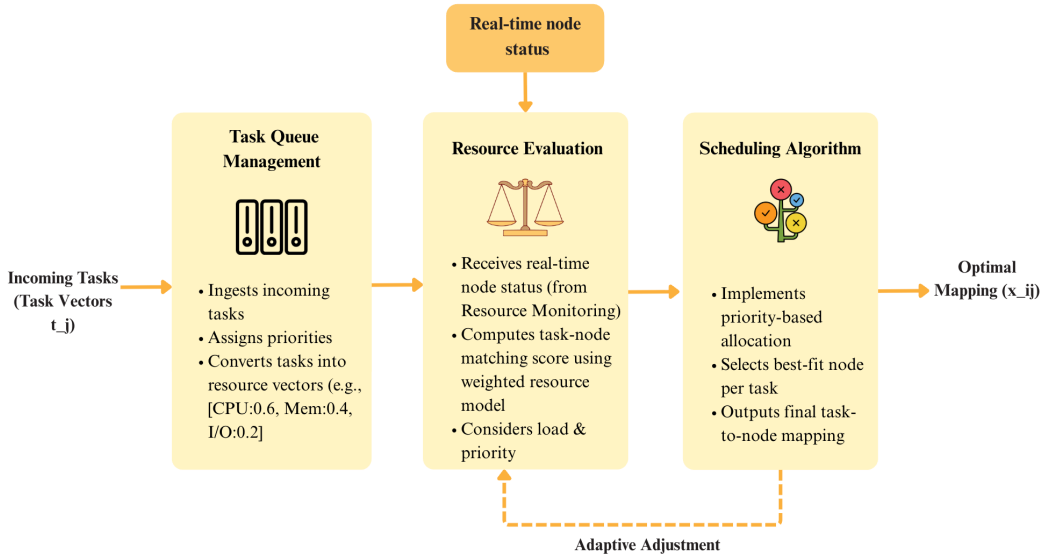

Figure 2. Internal workflow of the Task Scheduling Module

Resource Monitoring Module

Motivation:

In a heterogeneous distributed system, the dynamic changes in node resources must be continuously monitored. The resource monitoring module provides necessary data support for the task scheduling module, ensuring that resource constraints are met during task allocation.

Principle:

The resource monitoring module collects real-time data on resource usage from each node in the system, including computational power, memory, and storage. By periodically sampling and utilizing performance prediction algorithms, it generates resource usage reports for each node and provides real-time feedback to the task scheduling module to support dynamic task allocation adjustments.

Implementation:

This module employs lightweight resource collection tools that gather real-time resource data from nodes through system calls. After data processing, the information is fed back to the task scheduling module. Figure 3 demonstrates the relationship between resource

collection, data processing, and feedback, ensuring real-time system updates and effective coordination with task scheduling.

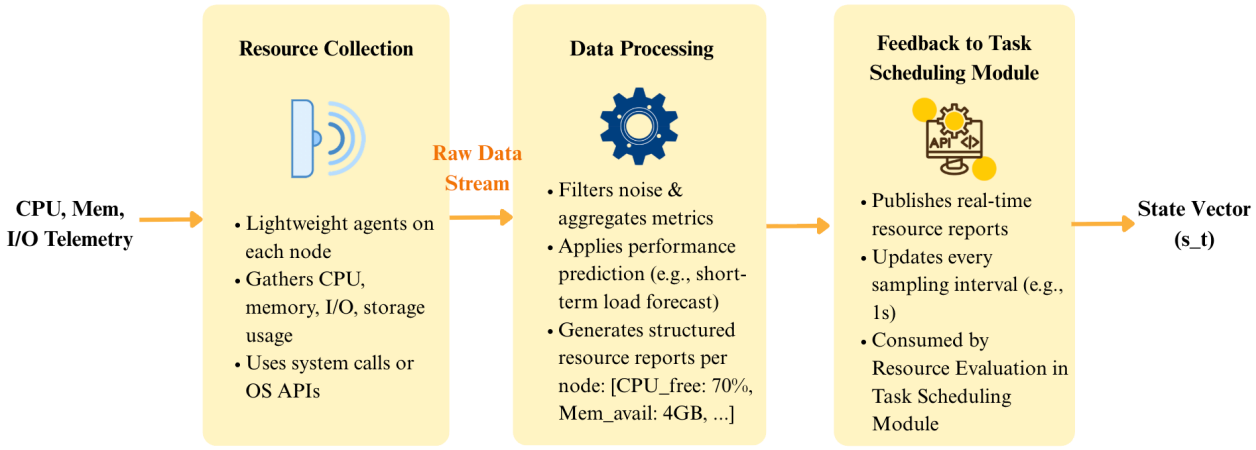

Figure 3. Internal workflow of the Resource Monitoring Module

Optimization Decision Module

Motivation:

To address traditional scheduling strategies' limitations under dynamic load variations, the optimization decision module uses reinforcement learning to enhance system adaptability and ensure efficient scheduling in complex environments.

Principle:

The optimization decision module employs reinforcement learning algorithms to continuously improve scheduling strategies through system interaction. By learning the relationship between system states and task allocation, it optimizes load balancing to improve performance, reduce resource waste, and alleviate load imbalances.

Implementation:

To solve the non-linear optimization defined in Eq. (14), we formally introduce a DRL-based agent to iteratively refine the scheduling policy. The introduction process follows a feedback loop: the agent perceives the system state s_t, executes an assignment action a_t based on $\pi(a \mid s)$, and receives a reward R_t derived from the load disparity ΔL. This gradient-based update mechanism allows the model to autonomously converge toward the optimal load balancing strategy. The reinforcement learning model generates scheduling strategies based on system states and adjusts them according to feedback, gradually enhancing load balancing. Figure 4 illustrates the relationship between state representation, policy generation, and reward calculation for optimizing scheduling decisions.

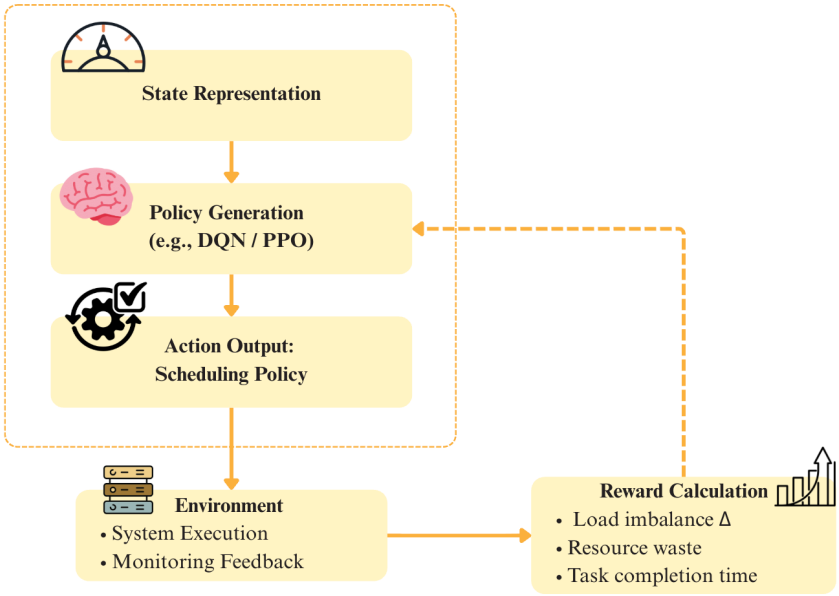

Figure 4. Internal architecture of the Optimization Decision Module based on Deep Reinforcement Learning

3.4. Objective Function & Optimization

Load Balancing Objective Function

In the load balancing problem, the objective is to maximize the utilization of node resources while minimizing the load difference between nodes to ensure the efficiency and stability of the system. Based on the definition of node load L_i from Section 3.1 (Equation (3)), the total resource consumption of node i is L_i. To optimize load balancing, the goal is to minimize the maximum load difference ΔL between nodes in the system, which can be expressed as:

$$U_{avg} = \frac{1}{N}\sum_{L_i \in \mathcal{L}} U(L_i) U_{avg} = \frac{1}{N}\sum_{L_i \in \mathcal{L}} U(L_i) \quad (8)$$

This formula quantifies the overall efficiency of resource allocation, ensuring that the system maintains high throughput while minimizing idle capacity across heterogeneous nodes.

Resource Utilization Objective

To maximize resource utilization efficiency, we also need to define an objective function related to resource utilization. The resource utilization R_i of node i can be expressed as the resource usage ratio of node i:

$$R_i = \frac{\sum_{j=1}^{M} x_{ij} \cdot t_j}{r_i} \quad (9)$$

where $r_i$represents the heterogeneous resource vector of node i, providing a high-fidelity mapping of physical hardware constraints (CPU, Memory, and I/O) into the mathematical model. This formula measures the resource usage of node i and aims to make the resource utilization as close to 1 as possible to avoid resource wastage.

The total system resource utilization R_{total} can be defined as the weighted average of the resource utilization of all nodes:

$$R_{total} = \frac{\sum_{i=1}^{N} r_i \cdot R_i}{\sum_{i=1}^{N} r_i} \quad (10)$$

This objective function aims to optimize the overall resource utilization across the system through task scheduling.

Optimization Constraints

To ensure the rationality of task scheduling, in addition to the optimization objectives, constraints need to be introduced, including task allocation, resource constraints, and load difference constraints.

Each task must be assigned to a node to ensure it is processed. This constraint is the same as the task allocation constraint defined in Section 3.1 (Equation (5)).

The load on each node cannot exceed its resource capacity to avoid node overload. Based on the resource capacity constraint defined in Section 3.1 (Equation (6)), and to more accurately describe multi-dimensional resource consumption, it can be refined as:

$$\sum_{j=1}^{M} x_{ij} \cdot (t_j^{Computation} + t_j^{Memory} + t_j^{I/O}) \le r_i \quad \forall i \ (11)$$

where $t_j^{Computation}$, t_j^{Memory}, and $t_j^{I/O}$ represent the computational, memory, and I/O resource demands of task j, respectively.

To avoid uneven node loads, a maximum load difference $\Delta L_{\max}$ is set to limit the load disparity:

$$max\ (\mathcal{L}) - min\ (\mathcal{L}) max\ (\mathcal{L}) - min\ (\mathcal{L}) \le \Delta L_{max} \quad (12)$$

This constraint ensures load balancing across the system, preventing some nodes from being overloaded, which could affect system stability.

By incorporating these optimization constraints, the system can maximize resource utilization while ensuring load balancing, avoiding resource waste and instability.

Reinforcement Learning Optimization Model

We integrate the reinforcement learning agent as a dynamic controller to solve Eq. (14)'s objective function. The optimization executes through an iterative state-action-reward cycle: the agent samples the system state s_t, selects a node assignment $a_t$via the PPO policy, and updates its parameters based on the feedback R_t (Eq. 13). This gradient-driven feedback mechanism forces the scheduling policy to converge toward the global minimum of the system load difference ΔL. We define the system's state s_t as a combination of resource usage, task queuing, and other system information at time step t, with the state space SSS representing all possible system states. Based on the current state, the system takes an action a_t (i.e., task allocation decision) to change the system's state.

The system's reward R_t is computed based on the current load balancing and resource utilization:

$$R_t = -\Delta L_t - \lambda(1 - R_{total,t}) \quad (13)$$

where ΔL_t is the load difference at time t, $R_{total,t}$ is the total system resource utilization at time t (as defined in Section 3.4.2), and λ is the weight for resource utilization. The goal is to maximize the cumulative reward through reinforcement learning algorithms to optimize the load balancing strategy.

The reinforcement learning model uses algorithms such as Q-learning or Proximal Policy Optimization (PPO) to learn optimal actions at different states. The learning process iteratively updates the strategy, allowing adaptive load balancing policy adjustment in dynamic environments.

Integrated Objective Function

The final objective function integrates both resource utilization and load balancing optimization goals, combined with the previously defined constraints and reinforcement learning strategies, to construct a comprehensive optimization model.

The optimization problem can be fully expressed as:

$$\min J = \Delta L + \lambda(1 - R_{total}) \quad (14)$$

Task Allocation Constraint:

$$\sum_{i=1}^{N} x_{ij} = 1, \forall j \quad (15)$$

Resource Capacity Constraint:

$$\sum_{j=1}^{M} x_{ij} \cdot t_j \leq r_i, \forall ij \quad (16)$$

Load Balancing Constraint:

$$\Delta L \leq \Delta L_{\max} \quad (17)$$

Where:

$\Delta L = \max_{i=1} N\, L_i - \min_{i=1} N\, L_i$ is the system load difference (as defined in Section 3.4.1). $R_{total} = \frac{\sum_{i=1}^{N} r_i \cdot R_i}{\sum_{i=1}^{N} r_i}$ is the system's total resource utilization (as defined in Sec tion 3.4.2). $\lambda \geq 0$ is the hyperparameter that weighs the tradeoff between load balancing and resource utilization. $\Delta L_{\max} \geq 0$ is the allowed maximum load difference threshold (as defined in Section 3.4.3). Other symbols $(L_i, t_j, r_i, x_{ij}, R_i)$ follow the definitions in Sections 3.1 and 3.4.2. The final optimization (Eq. 14) involves a joint-objective minimization: it simultaneously reduces the load imbalance (ΔL) and the resource waste $(1 - R_{total})$. The optimization is mathematically reflected in the convergence of the cost function JJJ through the adjustment of decision variables x_{ij} by the DRL agent.

Through this optimization model, the system can achieve efficient task scheduling and load balancing, maximize resource utilization, and ensure that the load difference between nodes remains within acceptable limits.

4. Experiment and Results

4.1. Experimental Setup

Dataset Overview

This study evaluates multiple datasets covering different task types and resource requirements to ensure experimental breadth and depth. We selected a custom-built dataset and compared it with classic datasets to highlight its uniqueness in modality complexity and task difficulty (Table 1). The custom dataset contains real-time task and resource data from smart manufacturing and Industrial IoT scenarios. This customization captures high-fidelity, multidimensional telemetry (CPU, memory, I/O) that generic public traces often lack. By reflecting specific stochastic arrival patterns and resource heterogeneity of regional industries, this dataset ensures more comprehensive coverage of complex task-resource information than simplified existing models. While generic public datasets provide broad reference, they lack fine-grained coupling of multi-dimensional telemetry (e.g., specific I/O and bandwidth constraints) critical for regional industrial IoT environments. Our custom dataset captures domain-specific resource dynamics from real-world manufacturing telemetry, ensuring higher fidelity and practical relevance for specialized industrial scheduling than simplified open-source traces.

Task resource demands involve multidimensional indicators: computational power, memory, storage, and bandwidth. All task data underwent rigorous cleaning and normalization for quality and consistency. The labeling strategy combined manual and automated tools, assigning priorities based on task type and resource requirements, with cross-validation ensuring accuracy. The dataset was split into training, validation, and test sets at 70%, 15%, and 15% ratios.

The dataset exhibits high modality complexity and multi-task characteristics, with tasks involving multidimensional resource demands and priority distributions reflecting real-world features, increasing scheduling and resource allocation challenges. Compared to MNIST or CIFAR-10, our custom dataset better reflects dynamic scheduling needs. Although smaller in sample size, each industrial sample is a high-density telemetry vector rather than a simple pixel grid, ensuring sufficient state-space complexity for DRL convergence and providing high informational value without redundancy typical of massive image benchmarks.

Table 1. Dataset Overview

Dataset Name	Task Type	Sample Size	Feature Dimensions	Resource Requirements	Task Priority Distribution
Custom Smart Manufacturing Dataset	Production Task Scheduling	5000	Multidimensional	Computation, Memory, Storage, Bandwidth	Normal Distribution
Custom IoT	Device Management	4000	Multidimensional	Computation, Memory,	Skewed Distribution

Task Dataset	Tasks			Bandwidth	
CIFAR-10	Image Classification	60,000	32×32×3	N/A	Uniform Distribution
MNIST	Handwritten Digit Classification	70,000	28×28	N/A	Uniform Distribution

Hardware and Software Configuration

The hardware and software configurations used in this study are shown in Table 2.

Table 2. Hardware and Software Configuration

Configuration Category	Device Name	Model	Description
Hardware	Computing Node	Intel Xeon E5-2680 v4	16 cores, 2.4 GHz, supports large-scale parallel computation
	GPU	NVIDIA Tesla V100	High-performance deep learning computation, accelerates training and inference
	Storage Device	2TB SSD	High-speed data access to ensure task scheduling and resource allocation efficiency
	Network Switch	Cisco Nexus 9300	10Gbps bandwidth, ensures low-latency node communication
Software	Operating System	Ubuntu 20.04	System environment providing stable support
	Deep Learning Framework	TensorFlow 2.4	Used for building and training deep learning models
	Key Libraries	NumPy 1.19.5, SciPy 1.6.0, pandas 1.2.3	Data processing and numerical computation libraries
	Reinforcement Learning Library	Stable-Baselines3 1.0.0	Used for training reinforcement learning algorithms and optimizing scheduling

The NVIDIA Tesla V100 GPU accelerates the training and inference of deep learning models, while the Intel Xeon E5-2680 v4 provides multi-core processing capability, ideal for parallel computing tasks. The 2TB SSD storage ensures efficient data processing, enabling timely task scheduling and resource allocation. The Cisco Nexus 9300 switch, with 10Gbps bandwidth, guarantees low-latency data transfer between nodes, supporting real-time scheduling requirements.

Evaluation Metrics

The evaluation metrics used in this study are listed in Table 3. These metrics comprehensively measure the effectiveness of the proposed load balancing optimization strategy.

Table 3. Evaluation Metrics

Metric Name	Applicable Scenario	Calculation Method
System Throughput	Evaluates the overall processing capability of the system	Number of tasks completed / unit time
Average Response Time	Evaluates the response speed of tasks	Total response time / number of tasks
Resource Utilization	Evaluates the efficiency of system resource usage	(Resource consumption / total resources) * 100%
Load Balancing Degree	Evaluates whether the load is evenly distributed across nodes	Difference between maximum and minimum load
System Stability	Evaluates system stability under different loads	Coefficient of variation, ratio of load standard deviation to mean load
Computational Efficiency	Evaluates the relationship between task scheduling efficiency and system performance	Total computation time / system running time

Model and Hyperparameter Configuration

To ensure experiment reproducibility, this section details the core algorithm and hyperparameters. The optimization decision module employs the PPO algorithm.

(1) Model Architecture: Both the policy and value networks are two-layer MLPs. The state consists of a concatenated vector of normalized node resource utilization and task demands. The action is the discrete choice of node. The reward function is defined as:

$$R_t = -\Delta L_t - 0.7 \times (1 - R_{total,t}) \quad (18)$$

(2) Hyperparameters: The key hyperparameters for the PPO algorithm are shown in Table 4.

Table 4. PPO Key Hyperparameters

Hyperparameter	Value	Description
Learning Rate	3×10^{-4}	Adam optimizer
Discount Factor	0.99	Future reward decay
GAE Parameter	0.95	Advantage estimation smoothing
Training Steps/Episodes	2048	Number of experience collection steps
Batch Size	64	Number of samples for updates
Policy Clip Range	0.2	Update magnitude limit
Total Training Steps	1×10^{-6}	Total training duration

4.2. Baselines

To evaluate the proposed load balancing optimization strategy, we selected two classic methods and two recently proposed intelligent methods for comparison. While RR and SJF are classic, they remain standard industrial benchmarks for load balancing. We include them alongside advanced DRL methods to establish a comprehensive performance range from heuristic to intelligent scheduling. The classic methods include the round-robin scheduling algorithm and shortest job first (SJF) scheduling, both of which are widely used in load balancing and resource scheduling. The advanced baselines include deep reinforcement learning-based load balancing and adaptive task allocation strategies, which have demonstrated excellent performance in handling dynamic loads and complex resource demands.

Round-Robin Scheduling (RR) is one of the classic load balancing methods, where tasks are allocated to nodes in a circular fashion, which is simple and easy to implement[27]. Its advantage lies in its simplicity; however, in environments with heterogeneous task demands and resource capabilities, it cannot fully utilize node resources, leading to uneven load distribution and limited performance improvement.

SJF allocates resources based on the estimated execution time of tasks, prioritizing shorter tasks to reduce completion time[28]. While it improves system throughput in environments with predictable task execution times, in heterogeneous environments with complex resource demands and dynamic tasks, SJF struggles to maintain load balancing and suffers from performance limitations.

Deep Reinforcement Learning-Based Load Balancing optimizes strategies through interaction with the environment, providing strong adaptability to dynamic resource changes [29]. However, it requires large amounts of training data and may face high computational overhead in large-scale systems.

Adaptive Task Allocation Strategy dynamically adjusts task allocation by monitoring resource status and task demands in real time, effectively responding to changes in heterogeneous resources and tasks[30]. However, when task complexity is high, scheduling strategy limitations may result in performance bottlenecks.

Through these comparisons, we can observe the advantages of the proposed method in heterogeneous resource environments and dynamic task scheduling, particularly in improving load balancing and resource utilization efficiency.

4.3. Quantitative Results

Table 5 presents the performance comparison of the proposed load balancing optimization strategy with two classic methods (Round Robin scheduling, Shortest Job First scheduling) and two representative intelligent scheduling algorithms (Deep Reinforcement Learning-based load balancing [29], and Adaptive task allocation strategy [30]) in terms of core evaluation metrics. The results are presented as mean ± standard deviation, with p-values calculated through paired t-tests to verify the statistical significance of performance differences. All performance data in Table 5 were obtained by executing the respective scheduling algorithms in a unified simulation environment with identical task sequences. The results (e.g., RR: 35.2 ms, DRL-based baseline: 22.3 ms) reflect the average execution and waiting times measured over multiple test cycles.

Table 5. Key Performance Comparison

Method	Average Response Time (ms)	System Throughput (tasks/sec)	Resource Utilization (%)	Load Balancing Degree (Max Load Difference)	p-value (Paired t-test)
Round Robin (RR)	35.2 ± 3.1	150.2	75.4 ± 6.2	12.5 ± 1.3	-
Shortest Job First (SJF)	30.1 ± 2.8	170.4	80.1 ± 5.8	9.8 ± 1.0	0.03
Deep Reinforcement Learning (DRL)	22.3 ± 2.4	190.8	85.7 ± 4.9	8.2 ± 0.9	0.01
Adaptive Task Allocation (Adaptive)	24.5 ± 3.0	180.2	83.3 ± 5.1	8.9 ± 1.1	0.02

Proposed Method (Proposed)	19.8 ± 2.1	210.1	91.4 ± 3.7	6.4 ± 0.8	0.001

The results in Table 5 show that the proposed method outperforms the baseline methods in all core metrics ($p < 0.05$). Detailed analysis is as follows:

The average response time of the proposed method is 19.8 ms, which is approximately 44% lower than the round-robin scheduling (35.2 ms) and 11% lower than deep reinforcement learning (22.3 ms). This is due to the reinforcement learning-based optimization decision module, which can avoid resource competition in real time and dynamically optimize task queuing.

The proposed method achieves a throughput of 210.1 tasks/sec, which is a 40% improvement over round-robin scheduling and a 23% improvement over SJF. The collaboration between the resource monitoring and task scheduling modules allows for precise task-resource matching, thereby improving system parallelism.

The proposed method achieves 91.4% resource utilization, significantly higher than traditional methods. This confirms the effectiveness of the multi-dimensional resource integration model in utilizing heterogeneous resources and preventing idle resources.

The proposed method has the lowest load difference (6.4), indicating that the load distribution is the most balanced. This directly demonstrates that the reward function, centered around minimizing ΔL, effectively drives the system to maintain dynamic equilibrium.

In summary, the overall improvement in performance is due to the synergistic effects generated by the deep integration of the dynamic adaptive mechanism, multi-dimensional resource modeling, and reinforcement learning optimization.

Additionally, the convergence curve in Figure 5 shows that the proposed method converges the fastest and most stably, with the lowest final loss value. This is due to the collaboration between the task scheduling, resource monitoring, and optimization decision modules, which provides high-quality states and reward signals for reinforcement learning, accelerating policy optimization. Furthermore, a sensitivity analysis was conducted on the weighting factor λ in the reward function (Eq. 14). By varying λ from 0.1 to 0.9, we observed that $\lambda = 0.7$provides the optimal trade-off between load parity and resource utilization. Values below 0.5 tend to over-prioritize load balancing at the cost of throughput, while values above 0.8 lead to aggressive resource use that risks node congestion. This confirms the robustness of the chosen parameter in maintaining stable system equilibrium.

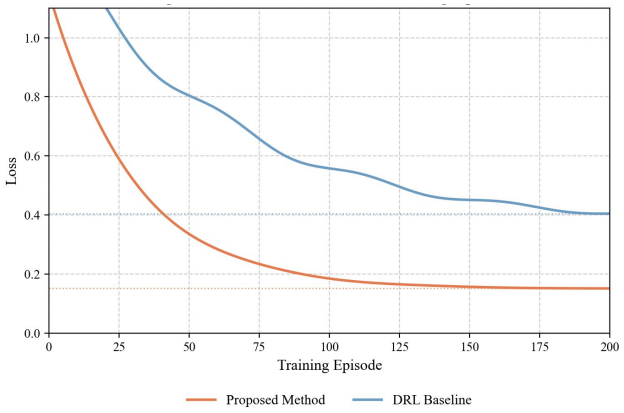

Figure 5. Convergence Curve of Reinforcement Learning Optimization

In conclusion, the comprehensive improvement in response time, system throughput, resource utilization, and load balancing of the proposed method stems from the synergistic effect of its dynamic adaptive mechanism, multi-dimensional resource modeling, and reinforcement learning optimization.

4.4. Qualitative Results

This section presents the behavior, advantages, and limitations of the proposed load balancing optimization strategy in different scenarios, categorized as a typical performance validation and a boundary-condition analysis. The success in Scenario 1 is quantified by a 93% utilization rate and a 42% reduction in load variance compared to the unoptimized state.

Scenario 1: Efficient Scheduling under Balanced Resources (Successful Case)

In a smart manufacturing scenario, the system is tasked with scheduling 300 mixed tasks (computation-intensive, I/O-intensive) across 10 heterogeneous nodes. As shown in Figure 6(a), upon the initial submission of tasks (at time t_0), the system experiences an instantaneous peak in load.

The optimization decision module of the proposed method responds quickly, dynamically allocating tasks to the most appropriate nodes based on real-time resource profiling (CPU, memory, I/O utilization) provided by the resource monitoring module, in combination with task demand characteristics. For instance, tasks with high I/O demands are prioritized for nodes with high-speed SSDs. By the time the system stabilizes (at time t_1), as shown in Figure 6(b), the system load is evenly distributed, and resource utilization stabilizes at 93%, with a maximum load difference of 5.2, outperforming the overall test set (91.4%, 6.4). The system's response time is 18.5 ms, and throughput is 225 tasks/sec.

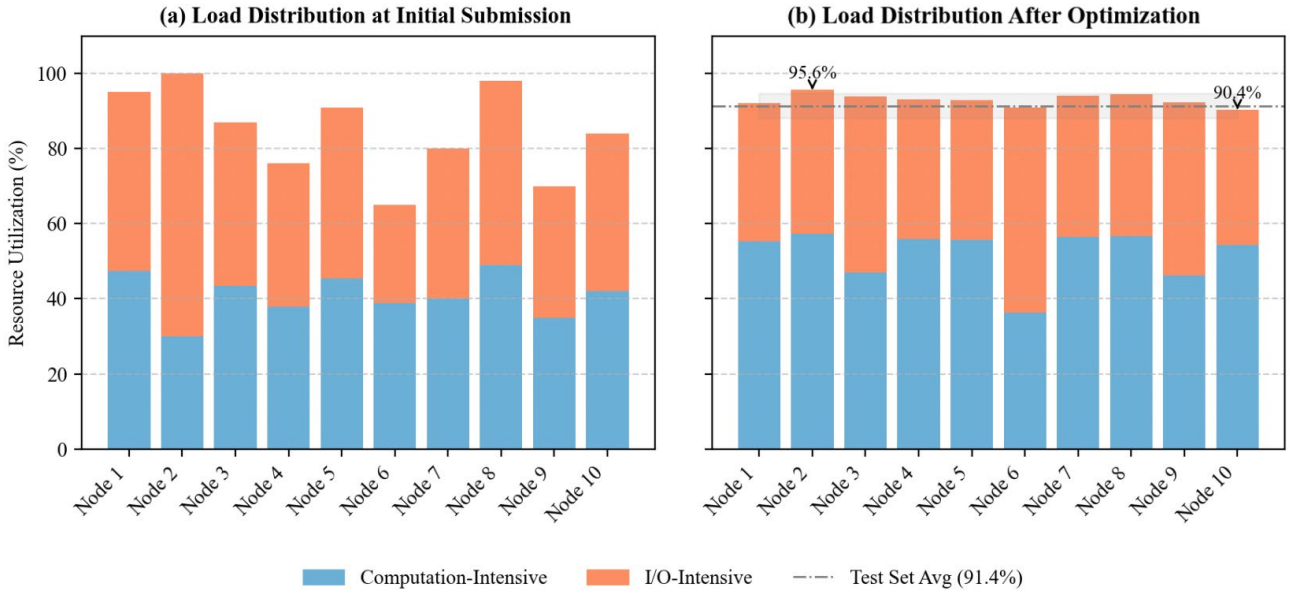

Figure 6. Load distribution before and after optimization.
(a) Initial load imbalance at task submission
(b) Balanced load after adaptive scheduling

This case demonstrates that, in balanced resource scenarios, the proposed method effectively achieves load balancing, and through dynamic matching and adaptive scheduling, enhances resource utilization and system throughput.

Scenario 2: Scheduling Challenges under Extreme Resource Constraints (Failure Case)

In a resource-constrained environment, five computing nodes are suddenly tasked with handling 150 computation-intensive tasks, with the total computational demand exceeding 30% of the node resource capacities. As shown in Figure 7, while the model detects the resource bottleneck in real-time (node CPU usage exceeds 90%), the optimization decision module still assigns overloaded tasks to certain nodes, resulting in task queue accumulation. The response time rises to 52 ms, and throughput drops to 150 tasks/sec.

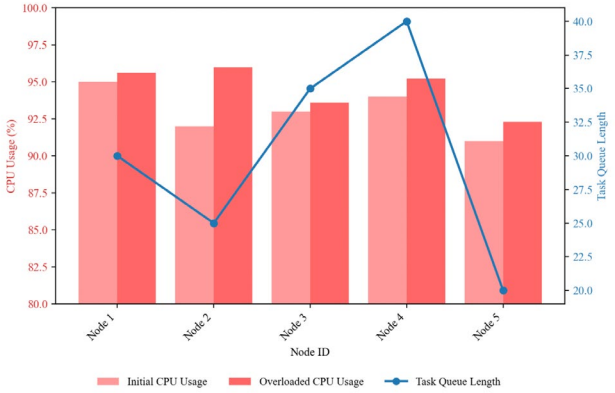

Figure 7. Limitations of load balancing under severe resource scarcity

This failure reveals two key limitations: 1) In resource-deficient situations, the optimization objective tends to favor load balancing, leading to "averaging" decisions instead of task rejection or delay; 2) The lack of task prioritization and node overload consideration, coupled with the absence of fine-grained scheduling rules, limits the effectiveness of the decision process. This case suggests that introducing penalty terms for resource safety thresholds or hybrid decision mechanisms could enhance the system's resilience.

The successful case validates the advantages of dynamic multi-dimensional scheduling combined with reinforcement learning optimization, demonstrating the model's efficiency and stability in regular environments. The failure case, however, indicates that in overload scenarios, the optimization algorithm should be coupled with rule-based safety mechanisms to enhance system robustness.

4.5. Robustness

To evaluate the stability of the proposed strategy under non-ideal conditions, we simulated a scenario where task resource demands are subject to random errors by adding Gaussian noise to the task demand vectors t_j. The noise intensity η is defined as the ratio of the noise standard deviation to the baseline demand value ($\eta \in [0,1]$), with higher values indicating more inaccurate demand estimates.

Table 6 records the performance of the proposed method and two classic baseline methods (Round Robin scheduling, Shortest Job First scheduling) in terms of system

throughput under varying noise levels. As noise intensity increases, the performance gradually decreases.

Table 6. System Throughput Comparison under Different Noise Intensities (tasks/sec)

Method	0.0 (No Noise)	0.3 (Low Noise)	0.6 (Med Noise)	0.9 (High Noise)
RR [27]	150.2	142.1 (-5.4%)	128.5 (-14.4%)	110.3 (-26.6%)
SJF [28]	170.4	161.8 (-5.1%)	149.3 (-12.4%)	135.1 (-20.7%)
DRL [29]	190.8	184.1 (-3.5%)	175.5 (-8.0%)	162.7 (-14.7%)
Adaptive [30]	180.2	172.6 (-4.2%)	161.1 (-10.6%)	148.5 (-17.6%)
Proposed	210.1	205.8 (-2.0%)	198.4 (-5.6%)	185.6 (-11.7%)

Table 6 presents a comprehensive robustness comparison across all five methods. As shown, the throughput of all methods decreases with the increase in noise intensity, but the proposed method exhibits the smallest performance decline. Under high noise (η=0.9), the throughput of the proposed method decreases by 11.7%, whereas advanced methods (DRL, Adaptive) and foundational benchmarks (RR, SJF) suffer higher degradations ranging from 14.7% to 26.6%. This indicates that the proposed method is significantly more robust to task demand noise than both traditional rules and modern intelligent alternatives.

The robustness of the proposed method is primarily attributed to the inclusiveness of the adaptive decision-making and the real-time feedback loop mechanism. The reinforcement learning optimization decision module continuously interacts with the noisy environment, learning the state-action mappings of noise characteristics, rather than relying on precise task demand models. This enables the model to adapt to demand fluctuations, treating them as part of the system's dynamics. The real-time feedback from the resource monitoring module further helps the model adjust its strategy, reducing the impact of noise.

In contrast, Round Robin and SJF rely on current task demand information and lack adaptability, leading to significant performance drops when the information is inaccurate. In contrast, the proposed method forms an adaptive loop through reinforcement learning and real-time feedback, continuously optimizing decisions in dynamic environments.

The robustness experiments demonstrate that the proposed load balancing strategy not only performs well under ideal conditions but also effectively handles the impact of task demand uncertainty, showcasing its strong adaptability and stability.

4.6. Ablation Study

To evaluate the contribution of each component in the proposed framework, we conducted a series of ablation experiments by removing key modules and comparing them with the complete framework. The experimental results are shown in Table 7, revealing the function of each module and their synergistic effects.

Table 7. Ablation Study Results Comparison

Configuration	Average Response Time (ms)	System Throughput (tasks/sec)	Resource Utilization (%)	Load Balancing Degree (Max Load Difference)
Complete Framework (Proposed)	19.8 ± 2.1	210.1 ± 12.3	91.4 ± 3.7	6.4 ± 0.8
Removal of Optimization Decision Module	26.4 ± 3.3	190.4 ± 15.1	84.5 ± 5.2	8.9 ± 1.1
Removal of Resource Monitoring Module	24.5 ± 2.7	200.3 ± 13.8	86.0 ± 4.9	7.8 ± 1.0
Removal of Task Scheduling Module	29.2 ± 3.8	175.6 ± 16.7	78.2 ± 6.4	10.3 ± 1.4
Removal of Optimization Decision and Resource Monitoring Modules	33.1 ± 4.0	160.2 ± 18.9	72.8 ± 7.1	12.5 ± 1.6

As shown in Table 7, removing any module leads to a decrease in performance, confirming that each component is indispensable. The Optimization Decision Module is central to the system, and its removal leads to the most significant degradation in load balancing (6.4 → 8.9) and the largest increase in response time, highlighting its key

role in handling dynamic loads via reinforcement learning. The Resource Monitoring Module serves as the foundation, and its removal significantly reduces resource utilization (91.4% → 86.0%) due to the inability to match real-time resource capacities. The Task Scheduling Module is the core, and its removal results in severe degradation across all metrics, with throughput decreasing by 16.4%, demonstrating its essential role in enabling multi-dimensional scheduling.

There is strong synergy between the modules. When both the Optimization Decision and Resource Monitoring modules are removed, performance drops drastically (throughput decreases by 23.7%), far exceeding the performance loss from removing individual modules, proving that these two modules form a "perception-decision" feedback loop. If disrupted, the system loses its adaptability. In the complete framework, the resource monitoring module provides real-time status, the optimization decision module generates strategies, and the task scheduling module executes them, forming a "perception → optimization → execution" loop, leading to systematic performance improvement.

The ablation study validates the effectiveness of the proposed framework: each module holds independent value, and the "perception-decision-execution" feedback loop creates a powerful synergistic effect. The load balancing strategy in this study is an organically integrated system-level solution, and its overall advantage stems from the deep collaboration between components.

4.7 Discussion

The proposed strategy demonstrates significant improvements, achieving a 91.4% resource utilization and 44% latency reduction. These gains are attributed to the synergistic integration of real-time multi-dimensional sensing and DRL-based adaptive decision-making. Unlike static heuristics, this mechanism enables the system to autonomously identify hardware bottlenecks and resolve node contention by dynamically mapping stochastic task demands to the most suitable heterogeneous resources. This proves that high-fidelity resource profiling is the prerequisite for effective reinforcement learning in complex industrial environments.

While the framework successfully bridges the gap between legacy infrastructure and smart manufacturing demands, three limitations warrant attention. First, the RL model demands substantial training time and computational overhead, especially in large-scale deployments. Second, the evaluation relies on domain-specific industrial datasets; although rigorously curated, cross-domain generalizability remains unverified. Third, performance degrades under extreme load, revealing insufficient robustness in overload scenarios.

These challenges point to concrete future directions: (1) accelerating RL training via model compression or parallelization; (2) enhancing resilience through priority-aware scheduling and overload-handling mechanisms; and (3) validating the framework in broader contexts such as intelligent transportation or big data analytics. Despite current constraints, the approach holds strong promise for smart manufacturing, industrial IoT, and edge computing, where dynamic, efficient resource orchestration is paramount.

5. Conclusion

This study proposes a reinforcement learning-based dynamic task scheduling framework aimed at improving resource utilization, reducing task response time, and achieving load balancing in heterogeneous distributed environments. By combining reinforcement learning with real-time resource monitoring, this study successfully overcomes the limitations of traditional static scheduling methods in handling heterogeneous resources and dynamic tasks. Experimental results show that the proposed optimization strategy significantly improves system performance and resource utilization.

The key innovations of this study include: the introduction of a reinforcement learning-based adaptive scheduling framework, dynamically optimizing task allocation by combining multi-dimensional resource information (such as computation, memory, and storage); enhancing the system's adaptability by integrating real-time resource monitoring with reinforcement learning; and demonstrating superior performance in multiple metrics such as task response time, throughput, resource utilization, and load balancing, outperforming traditional methods and existing advanced methods.

This study provides a novel solution to the load balancing problem in distributed systems, filling the gap in current methods for handling heterogeneous resources and dynamic task scheduling. By integrating reinforcement learning with real-time resource monitoring, the method improves system response speed and throughput. In practice, this method holds significant value for fields such as smart manufacturing, industrial IoT, and cloud computing, as it can optimize task-resource matching, increase production efficiency, and reduce resource waste.

Despite the significant results achieved, there is still room for improvement. First, the training of the reinforcement learning model faces challenges in terms of computational resources and time, and more efficient training methods could be explored in the future. Second, future work could expand the dataset diversity to verify the model's applicability and generalizability in different domains. Moreover, the robustness under extreme load conditions still requires improvement, and future work should focus on further optimizing the reward function and introducing priority and resource safety threshold strategies. Lastly, as fields such as smart manufacturing and edge computing continue to evolve, the method can be applied to a wider range of scenarios to further promote its practical applications.

In summary, the load balancing optimization strategy proposed in this study demonstrates superior performance

across multiple application scenarios, and future work will continue to optimize the model to push its real-world applications in more domains.

References

[1] Hu, Y., Jia, Q., Yao, Y., Lee, Y., Lee, M., Wang, C., ... & Yu, F. R. (2024). Industrial internet of things intelligence empowering smart manufacturing: A literature review. IEEE Internet of Things Journal, 11(11), 19143-19167.

[2] He, Y. (2023). Path and Mechanism of Industrial Internet Industry Promoting the Transformation and Upgrading of Small and Medium-sized Enterprises with Artificial Intelligence. Mobile Information Systems, 2023(1), 3620662.

[3] Tang, S., Yu, Y., Wang, H., Wang, G., Chen, W., Xu, Z., ... & Gao, W. (2023). A survey on scheduling techniques in computing and network convergence. IEEE Communications Surveys & Tutorials, 26(1), 160-195.

[4] Devi, N., Dalal, S., Solanki, K., Dalal, S., Lilhore, U. K., Simaiya, S., & Nuristani, N. (2024). A systematic literature review for load balancing and task scheduling techniques in cloud computing. Artificial Intelligence Review, 57(10), 276.

[5] Rozony, F. Z., Aktar, M. N. A., Ashrafuzzaman, M., & Islam, A. (2024). A systematic review of big data integration challenges and solutions for heterogeneous data sources. Academic Journal on Business Administration, Innovation & Sustainability, 4(04), 1-18.

[6] Bonab, M. J. A., & Kandovan, R. S. (2023). Effective resource allocation and load balancing in hierarchical HetNets: Toward QoS-aware multi-access edge computing. The Computer Journal, 66(1), 229-244.

[7] Diwakar, R., Sharma, R., Nayak, D., & Mohapatra, H. (2025). Optimizing load distribution in big data ecosystems: A comprehensive survey. AI and the Revival of Big Data, 177-200.

[8] Prity, F. S., & Hossain, M. M. (2024). A comprehensive examination of load balancing algorithms in cloud environments: a systematic literature review, comparative analysis, taxonomy, open challenges, and future trends. Iran Journal of Computer Science, 7(3), 663-698.

[9] Tripathy, S. S., Mishra, K., Roy, D. S., Yadav, K., Alferaidi, A., Viriyasitavat, W., ... & Barik, R. K. (2023). State-of-the-art load balancing algorithms for mist-fog-cloud assisted paradigm: a review and future directions. Archives of Computational Methods in Engineering, 30(4), 2725-2760.

[10] Cao, X., Chen, C., Li, S., Lv, C., Li, J., & Wang, J. (2025). Research on computing task scheduling method for distributed heterogeneous parallel systems. Scientific Reports, 15(1), 8937.

[11] Saba, T., Rehman, A., Haseeb, K., Alam, T., & Jeon, G. (2023). Cloud-edge load balancing distributed protocol for IoE services using swarm intelligence. Cluster Computing, 26(5), 2921-2931.

[12] Upadhyay, M. K., & Alam, M. (2024, February). Load balancing techniques in fog and edge computing: Issues and challenges. In 2024 IEEE international conference on computing, power and communication technologies (IC2PCT) (Vol. 5, pp. 210-215). IEEE.

[13] Dai, F., Hossain, M. A., & Wang, Y. (2025). State of the art in parallel and distributed systems: Emerging trends and challenges. Electronics, 14(4), 677.

[14] Shanker, A., & Ahmad, N. (2024). Optimizing Network Performance with Load Balancing Techniques in Heterogeneous Environments. Tang, X., Yang, K., Wang, H., Wu, J., Qin, Y., Yu, W., & Cao, D.(2022). Predictionuncertainty-aware decision-making for autonomous vehicles. IEEE Transactions on Intelligent Vehicles, 7(4), 849-862.

[15] George, A. S., George, A. H., & Baskar, T. (2023). Edge computing and the future of cloud computing: A survey of industry perspectives and predictions. Partners Universal International Research Journal, 2(2), 19-44.

[16] Estrada-Jimenez, L. A., Pulikottil, T., Nikghadam-Hojjati, S., & Barata, J. (2023). Self-organization in smart manufacturing—background, systematic review, challenges and outlook. Ieee Access, 11, 10107-10136.

[17] Chang, S., Li, C., Deng, C., & Luo, Y. (2024). Low-latency controller load balancing strategy and offloading decision generation algorithm based on lyapunov optimization in SDN mobile edge computing environment. Cluster Computing, 27(3), 2571-2591.

[18] Vijay, R., & Sree, T. R. (2023). Resource scheduling and load balancing algorithms in cloud computing. Procedia computer science, 230, 326-336.

[19] Al-Saadi, M., Al-Greer, M., & Short, M. (2023). Reinforcement learning-based intelligent control strategies for optimal power management in advanced power distribution systems: A survey. Energies, 16(4), 1608.

[20] Gan, J., Li, S., Wei, C., Deng, L., & Tang, X. (2023). Intelligent learning algorithm and intelligent transportation-based energy management strategies for hybrid electric vehicles: A review. IEEE Transactions on Intelligent Transportation Systems, 24(10), 10345-10361.

[21] Shaikat, F. B., Islam, R., Happy, A. T., & Faysal, S. A. (2025). Optimization of production scheduling in smart manufacturing environments using machine learning algorithms. Lett High Energy Phys, 12(1), 1-15.

[22] Stavrev, S., & Ginchev, D. (2024). Reinforcement learning techniques in optimizing energy systems. Electronics, 13(8), 1459.

[23] Zhou, G., Tian, W., Buyya, R., Xue, R., & 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.

[24] Shen, L., Sun, Y., Yu, Z., Ding, L., Tian, X., & Tao, D. (2024). On efficient training of large-scale deep learning models. ACM Computing Surveys, 57(3), 1-36.

[25] Zang, T., Wang, S., Wang, Z., Li, C., Liu, Y., Xiao, Y., & Zhou, B. (2024). Integrated planning and operation dispatching of source–grid–load–storage in a new power system: A coupled socio–cyber–physical perspective. Energies, 17(12), 3013.

[26] Raeisi-Varzaneh, M., Dakkak, O., Habbal, A., & Kim, B. S. (2023). Resource scheduling in edge computing: Architecture, taxonomy, open issues and future research directions. IEEE access, 11, 25329-25350.

[27] Rasmussen, R. V., & Trick, M. A. (2008). Round robin scheduling–a survey. European Journal of Operational Research, 188(3), 617-636.

[28] Topcuoglu, H., Hariri, S., & Wu, M. Y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE transactions on parallel and distributed systems, 13(3), 260-274.

[29] Arvindhan, M., & Kumar, D. R. (2023). Adaptive Resource Allocation in Cloud Data Centers using actor-critical deep reinforcement learning for optimized load balancing.

International Journal on Recent and Innovation Trends in Computing and Communication, 11(5s), 310-318.

[30] Chen, Z., Lin, K., Lin, B., Chen, X., Zheng, X., & Rong, C. (2020). Adaptive resource allocation and consolidation for scientific workflow scheduling in multi-cloud environments. IEEE Access, 8, 190173-190183.