Transactions on Computational and Scientific Methods | Vo. 5, No. 7, 2025
ISSN: 2998-8780
https://pspress.org/index.php/tcsm
Pinnacle Science Press

# Reinforcement Learning Based Adaptive Scheduling Method for Throughput Improvement in Distributed Computing Environments

**Yulin Wang**[1]
[1]New York University, New York, USA
*Corresponding author: Yulin Wang; wylbenji@gmail.com

**Abstract:** To address the declining scheduling efficiency caused by the continuous growth of task size, increasingly heterogeneous resource types, and dynamic changes in system state in distributed computing environments, this paper proposes a reinforcement learning-based adaptive scheduling method aimed at improving task throughput. This method models the distributed task scheduling process as a continuous decision-making process. By jointly representing task queue state, node resource occupancy, communication load information, and task attribute characteristics, an input representation reflecting the system's operating state is constructed. Based on this, a state refinement mechanism is introduced to enhance the policy learning's ability to perceive complex environmental changes. To improve the rationality of task allocation, a task node compatibility scoring module is further designed. This module comprehensively evaluates candidate allocation relationships from the perspectives of computational adaptability, communication cost, and load balancing requirements, enabling scheduling decisions to more accurately match task needs with node capabilities. Simultaneously, a reward function is constructed around the throughput improvement objective, integrating task completion efficiency, waiting time control, load balancing constraints, and service stability into the optimization process. This guides the scheduling strategy to achieve continuous improvement in overall system performance in the long run. This method achieves coordinated coupling of system state modeling, compatibility matching evaluation, throughput-oriented reward shaping, and policy optimization updates within a unified framework, transforming the scheduling process from a traditional rule-driven approach to an adaptive optimization process oriented towards dynamic environments. Comparative results show that the proposed method achieves superior overall performance in terms of throughput, resource utilization, task waiting control, and quality of service assurance, providing an effective technical solution for intelligent task scheduling in complex distributed computing environments.

**Keywords:** Reinforcement learning scheduling; task throughput optimization; resource collaborative allocation; dynamic decision modeling

## 1. Introduction

With the continuous development of cloud computing, edge computing, and high-performance computing, distributed computing environments have become the core infrastructure supporting large-scale data processing, complex model training, and the operation of various types of online services[1], [2]. Against this backdrop, computing tasks exhibit significant characteristics such as continuously expanding scale, increasingly complex structures, diversified resource requirements, and highly dynamic arrival processes. Traditional scheduling methods relying on static rules or fixed priorities are gradually becoming inadequate to adapt to the ever-changing resource competition and task execution needs in real-world scenarios[3]. Especially under conditions of multi-node collaboration, heterogeneous resource coexistence, and enhanced service quality constraints, how to achieve efficient task allocation and orderly execution in complex

operating environments has become one of the key problems that urgently need to be solved in the field of distributed computing.

Task throughput, as an important indicator for measuring the overall service efficiency and resource utilization level of a distributed system, directly affects the system's processing capacity, response efficiency, and operating costs. In the reality of continuously growing task scale and increasingly severe resource constraints, improving throughput is not only related to the computing platform's ability to handle high-concurrency requests but also to whether resources can be utilized more fully, evenly, and stably[4]. When scheduling strategies lack effective awareness of task status, node load, resource availability, and future trends, systems often experience problems such as local congestion, resource fragmentation, queue backlog, and decreased execution efficiency, thus limiting further improvements in overall throughput. Therefore, research on scheduling methods for dynamic environments aimed at improving task throughput has significant theoretical value and practical necessity.

Reinforcement learning offers a new research approach for solving complex dynamic decision-making problems. Its core advantage lies in its ability to learn the optimal decision-making strategy for long-term gains through continuous interaction with the environment. Compared to traditional heuristic or rule-driven methods, reinforcement learning is more suitable for handling scheduling scenarios with complex state spaces, delayed feedback relationships, and continuously evolving decision-making processes[5]. In distributed computing environments, task scheduling is essentially a typical sequential decision-making problem. Scheduling behavior not only affects the current resource allocation but also has a continuous impact on subsequent task queuing, node load evolution, and overall system throughput. Introducing reinforcement learning into distributed task scheduling helps overcome the limitations of fixed strategies in adapting to complex scenarios, enabling scheduling systems to possess stronger environmental awareness, autonomous decision-making capabilities, and dynamic adjustment capabilities, thereby providing a more intelligent technical path for improving system throughput performance.

Researching reinforcement learning-based adaptive scheduling methods to improve task throughput in distributed computing environments aligns with the technological trends in the evolution of intelligent computing infrastructure and addresses the practical need for efficient resource collaborative management. From a theoretical perspective, this research helps promote the deep integration of reinforcement learning methods and distributed system scheduling mechanisms, expands the application boundaries of intelligent optimization methods in complex computing environments, and provides a new analytical framework for dynamic resource allocation and long-term benefit modeling. From an application perspective, this research is expected to enhance the adaptive scheduling capabilities of distributed platforms when facing heterogeneous tasks, high dynamic loads, and complex resource constraints, improving system throughput, resource utilization efficiency, and service stability, providing crucial support for the efficient operation of cloud data centers, edge collaboration platforms, and large-scale intelligent computing systems. Therefore, conducting in-depth research in this direction has significant academic and engineering value.

## 2. Background

The core characteristic of distributed computing environments lies in the collaborative operation of multiple computing nodes, storage nodes, and network resources to jointly complete large-scale task processing and support complex business operations[6]. In actual operation, the system not only needs to address issues such as uncertain task arrival times, significant differences in task scale, heterogeneous resource types, and node performance fluctuations, but also needs to balance multiple objectives including latency, load balancing, resource utilization, and service continuity. With the increasing demands for artificial intelligence training, massive data analysis, online inference services, and industrial-grade real-time computing, the task organization methods and resource scheduling relationships in distributed platforms have become more

complex. Traditional scheduling logic relying on human experience is no longer sufficient to continuously meet the requirements of flexibility, real-time performance, and global coordination in complex scenarios. Against this backdrop, building intelligent scheduling mechanisms that can adapt to environmental changes and continuously optimize the decision-making process has gradually become an important direction in distributed computing research.

Reinforcement learning provides a modeling approach for distributed scheduling problems based on interactive feedback and long-term reward optimization. Its key value lies in its ability to unify state perception, action selection, and reward evaluation in complex environments into a dynamic decision-making framework. Compared to traditional optimization methods that rely solely on local information or short-term goals, reinforcement learning is better suited to characterizing the real-world characteristics of task scheduling, such as the interplay between successive decisions, the continuous evolution of resource states, and the coupling of system performance metrics. In distributed computing scenarios, different scheduling decisions continuously alter queue lengths, node occupancy, task waiting relationships, and the overall execution rhythm; therefore, the scheduling process itself exhibits significant temporal dependencies and global interconnectedness[7]. Combining reinforcement learning with distributed task scheduling helps to shift scheduling methods from passive, reactive configuration to proactive, learning-based optimization, laying the methodological foundation for autonomous adjustment and continuous performance improvement in complex computing environments.

Earlier reinforcement learning scheduling studies provide the direct algorithmic foundation for adaptive resource orchestration in distributed computing. Deep reinforcement learning has been used for resource management, multi-resource multi-machine job scheduling, data-processing cluster scheduling, data-center job scheduling, automated HPC scheduling, cluster scheduling, and distributed machine learning scheduling [8], [9], [10], [11], [12], [13], [14]. These studies show that scheduling policies must jointly consider queue dynamics, heterogeneous resource occupation, long-term completion efficiency, and feedback-driven policy refinement.

Recent distributed-system learning research further extends this foundation toward secure collaboration, robust state modeling, and semantic decision support. Heterogeneity-aware federated optimization with secure aggregation strengthens distributed data mining under non-uniform clients and privacy constraints [15]. Reconstruction-residual discrimination and temporal state modeling improve the detection of abnormal system behavior and latent distribution shifts in distributed environments [16], [17]. Graph-based knowledge representation, knowledge graph-augmented semantic fusion, and logistic similarity learning also provide transferable mechanisms for aligning structured system states, task attributes, and candidate execution decisions [18], [19], [20].

The growing use of large language model adaptation and graph-based representation also offers useful methodological guidance for scheduling models that must remain expressive yet efficient. Frequent subgraph mining and structural prompt injection highlight the value of extracting reusable structural patterns from complex interaction graphs [21]. Structured pruning-driven low-rank adapter learning and task-aware continual fine-tuning show how large models can retain adaptability while controlling parameter cost and catastrophic forgetting [22], [23]. Cross-modal graph neural recommendation, spatiotemporal transformer-based latency prediction, and variational autoencoder-based query-plan anomaly warning further connect temporal behavior fusion, latency forecasting, and anomaly-aware cost control to dynamic scheduling and throughput optimization [24], [25], [26].

## 3. Methodology

To improve task throughput in distributed computing environments with dynamic arrivals, heterogeneous resources, and time-varying contention, the scheduling problem is formulated as a sequential decision

process in which the scheduler continuously observes the global system condition and selects a task–node assignment action that maximizes long-horizon execution efficiency. Rather than treating each dispatching step as an isolated optimization event, the proposed policy gradient method models the coupling among queue evolution, resource occupancy, and execution feedback, so that the scheduling policy can explicitly account for how current decisions reshape future service capacity. This paper further presents the overall model architecture, as shown in Figure 1.

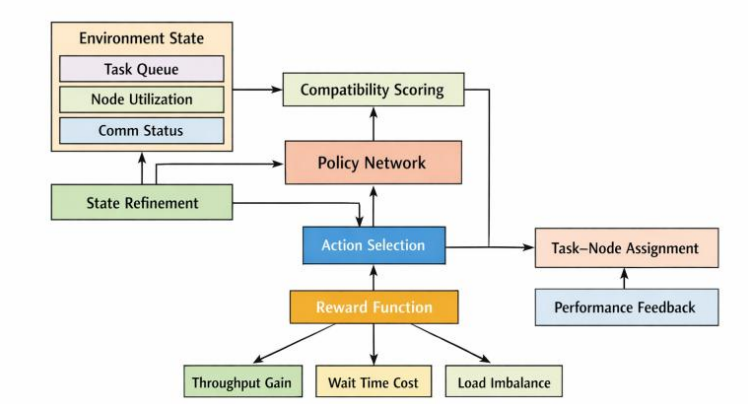


**Figure 1.** Overall model architecture

At decision step $t$, the environment state is represented by a compact throughput-oriented descriptor that jointly summarizes pending tasks, node utilization, communication load, and execution urgency:

$$s_t = [\boldsymbol{q}_t; \boldsymbol{h}_t; \boldsymbol{c}_t; \boldsymbol{d}_t]$$

in which $\boldsymbol{q}_t$ denotes the task queue profile, $\boldsymbol{h}_t$ characterizes node-side computing availability, $\boldsymbol{c}_t$ reflects communication congestion, and $\boldsymbol{d}_t$ encodes task urgency and resource demand. Since raw system measurements usually exhibit different scales and unstable fluctuations, a state refinement transformation is introduced to improve representation consistency before policy inference:

$$\tilde{s}_t = \phi\left(W_s s_t + b_s\right)$$

with learnable parameters $W_s$ and $b_s$ projecting the original observation into a decision-friendly latent space through nonlinear mapping $\phi(\cdot)$. This design is important because throughput is jointly influenced by both immediate load status and hidden structural interactions among tasks and nodes, and the refined state makes those coupled patterns easier to capture during policy learning. In this framework, the scheduler no longer relies on fixed heuristics such as earliest dispatch or static priority sorting, but instead develops an adaptive representation of execution pressure, service opportunity, and future congestion risk.

The state refinement module draws on this broader line of robust distributed representation learning. Secure federated optimization supports stable multi-source feature aggregation under heterogeneous resource conditions [15], while residual-based anomaly discrimination and temporal state modeling help the scheduler represent abnormal load transitions and latent drift before they produce queue congestion [16], [17].

Next, a compatibility-driven scoring mechanism is constructed to evaluate whether a candidate node can efficiently execute a specific task under the current system condition. By combining computational fit, communication overhead, and load balancing tendency, the method measures the quality of each task–node pair through:

$$m_{ij}^{(t)} = \alpha \frac{r_i^{\mathrm{req}} \cdot r_j^{\mathrm{cap}}}{\| r_i^{\mathrm{req}} \| \ \| r_j^{\mathrm{cap}} \|} - \beta l_{ij}^{(t)} - \gamma u_j^{(t)}$$

where $r_i^{\mathrm{req}}$ and $r_j^{\mathrm{cap}}$ denote the demand vector of task $i$ and the capacity vector of node $j$, while $l_{ij}^{(t)}$ and $u_j^{(t)}$ quantify communication latency and current utilization, respectively. A probabilistic scheduling policy is then used to transform these matching scores into executable actions, allowing the scheduler to preserve exploration ability when the environment is still changing:

$$\pi_\theta \left( a_t | \tilde{s}_t \right) = \mathrm{softmax} \left( M_t \right)$$

in which $M_t$ is composed of all valid matching scores $m_{ij}^{(t)}$ and $\theta$ denotes the policy parameters. Such a formulation is meaningful because the scheduler must not only identify high-quality assignments at the current moment, but also maintain enough adaptability to respond to future queue shifts, node recoveries, and bursty task arrivals.

The compatibility scoring module also benefits from semantic and structural alignment ideas. Graph-based knowledge representation and knowledge graph-augmented semantic fusion support the modeling of relationships among tasks, nodes, and resource constraints [18], [19], while logistic similarity learning offers a concise way to transform task-node fit into a comparable matching score under sparse or cold-start operating conditions [20].

Meanwhile, a throughput-aware reward mechanism is introduced to guide the policy toward decisions that increase completed workload while suppressing congestion accumulation and resource fragmentation. Instead of rewarding only short-term dispatch success, the reward explicitly emphasizes sustainable service efficiency across multiple scheduling steps:

$$r_t = \lambda_1 \Delta \mathrm{Thr}_t - \lambda_2 \Delta \mathrm{Wait}_t - \lambda_3 \Delta \mathrm{Imb}_t - \lambda_4 \Delta \mathrm{Drop}_t$$

where $\Delta \mathrm{Thr}_t$ measures the increment of completed tasks or processed workload, $\Delta \mathrm{Wait}_t$ captures waiting-time growth, $\Delta \mathrm{Imb}_t$ reflects the change in load imbalance, and $\Delta \mathrm{Drop}_t$ penalizes scheduling inefficiency that leads to deferral or resource waste. Because throughput improvement should not be achieved by sacrificing overall stability, this reward design encourages the scheduler to seek a better tradeoff among execution speed, fairness of resource occupation, and robustness under dynamic pressure. Under this objective, actions that appear locally attractive but induce downstream queue blockage receive lower long-term returns, thereby reducing myopic scheduling behavior in highly coupled distributed environments.

Finally, policy optimization is performed by maximizing discounted cumulative return so that the scheduler can internalize delayed performance feedback and progressively learn a stable adaptive dispatching strategy:

$$J(\theta) = \mathbb{E}_{\pi_\theta} \left[ \sum_{t=0}^{T} \gamma^t r_t \right]$$

where $\gamma \in (0,1)$ controls the contribution of future rewards to current policy improvement. To stabilize learning in the presence of abrupt load variation and nonstationary resource competition, the policy is updated with an advantage-guided objective:

$$\mathcal{L}(\theta) = \mathbb{E}_t \left[ \log \pi_\theta \left( a_t | \tilde{s}_t \right) \widehat{A}_t \right]$$

with $\widehat{A_t}$ estimating whether the selected action performs better than the expected baseline at step $t$. Through this mechanism, the proposed method converts distributed scheduling from a rule-based reactive process into a feedback-driven adaptive optimization procedure, which is valuable because the scheduler can continuously revise its decision preference according to evolving workload structures and node states. In consequence, the method provides a unified path for jointly modeling system observation, task–node matching, throughput-oriented reward shaping, and long-horizon policy adaptation within one coherent reinforcement learning framework.

During policy optimization, structural pattern extraction, compact adaptation, and continual learning provide additional guidance for maintaining scheduler stability. Frequent subgraph mining can expose recurring task-resource motifs for policy learning [21], low-rank adapter pruning and task-aware continual fine-tuning help reduce adaptation cost while preserving learned scheduling knowledge [22], [23], and cross-modal graph recommendation, latency prediction, and query-plan anomaly warning strengthen the policy's ability to combine behavioral, temporal, and cost-deviation signals during long-horizon throughput optimization [24], [25], [26].

## 4. Experimental Results and Analysis

### 4.1 Dataset

This study uses Google ClusterData 2019 as the experimental dataset. Released publicly by Google, this dataset is open-source cluster trajectory data for real-world production-grade distributed computing environments. It records the operational information of eight Borg cluster units during May 2019, covering multiple dimensions such as task submission, resource usage, machine status, and job relationships. Compared to data containing only static resource snapshots or single-node monitoring metrics, this dataset more comprehensively reflects the dynamic coupling relationships between task arrival, resource contention, node collaboration, and scheduling feedback in large-scale distributed systems. Therefore, it has high consistency with reinforcement learning adaptive scheduling research aimed at improving task throughput in distributed computing environments.

In terms of data content, Google ClusterData 2019 not only includes resource requirements and usage information at the job and task levels, but also provides structured fields such as CPU usage histograms statistically analyzed in 5-minute intervals, shared resource reservation information, and master-slave task relationships. This allows for a more detailed characterization of load changes, resource consumption trends, and task association patterns during the scheduling process. Since improving task throughput essentially relies on joint modeling of queue evolution, node availability, and task matching efficiency, this dataset can provide a reliable data foundation for state representation construction, task node compatibility evaluation, and reward function design, thus better supporting the dynamic decision-making needs of reinforcement learning scheduling models in complex distributed environments.

### 4.2 Experimental Results and Analysis

To verify the comprehensive performance of the proposed method in task scheduling scenarios in distributed computing environments, representative related methods in reinforcement learning-driven resource management, cluster scheduling, and job scheduling were selected as comparison objects. A unified comparison was conducted from multiple dimensions, including task throughput, resource utilization level, job completion efficiency, waiting overhead, slow job suppression, load balancing capability, service quality satisfaction, and fault tolerance performance. This allowed for a more systematic characterization of the overall scheduling characteristics of different scheduling strategies in complex dynamic environments. The experimental results are shown in Table 1.

**Table 1.** Experimental results compared with other models

| Method | THR | UTL | JCT | WAT | SLD | LBI | SLA | FTD |
|---|---|---|---|---|---|---|---|---|
| DeepRM | 82.14 | 74.26 | 49.87 | 21.43 | 2.41 | 0.17 | 90.24 | 4.91 |
| DRL-MRMS | 83.62 | 75.18 | 47.95 | 20.26 | 2.28 | 0.17 | 91.35 | 4.63 |
| Decima | 85.73 | 77.84 | 44.68 | 18.77 | 2.11 | 0.15 | 92.48 | 4.21 |
| DeepJS | 86.95 | 79.12 | 42.83 | 17.64 | 1.98 | 0.15 | 93.26 | 3.94 |
| RLScheduler | 88.34 | 80.57 | 40.29 | 16.52 | 1.86 | 0.14 | 94.11 | 3.68 |
| DRAS | 89.27 | 81.46 | 38.74 | 15.81 | 1.79 | 0.13 | 94.86 | 3.32 |
| RLPTO | 90.18 | 82.63 | 37.56 | 14.94 | 1.71 | 0.12 | 95.38 | 3.05 |
| Ours | 92.47 | 85.21 | 34.18 | 12.36 | 1.49 | 0.11 | 97.12 | 2.41 |

All the compared methods achieved certain results in the distributed task scheduling scenario, but significant differences still exist in overall performance. With scheduling strategies evolving from early resource management methods based on deep reinforcement learning to more optimized models oriented towards complex cluster environments, THR and UTL generally show an upward trend, while cost-related indicators such as JCT, WAT, SLD, LBI, and FTD continue to decline, indicating that existing research has been able to improve resource utilization efficiency and alleviate task congestion to some extent. In contrast, the proposed method achieves the best performance across all eight indicators, with THR reaching 92.47 and UTL reaching 85.21, demonstrating that the proposed adaptive scheduling mechanism can more fully exploit the system's service capabilities and improve overall resource utilization. Simultaneously, JCT, WAT, and SLD decrease to 34.18, 12.36, and 1.49, respectively, indicating that the method can effectively shorten task completion cycles, reduce waiting overhead, and improve scheduling response efficiency. Furthermore, the proposed method achieves LBI, SLA, and FTD scores of 0.11, 97.12, and 2.41 respectively, further demonstrating that the constructed state-aware and policy optimization process not only enhances the balance of system load distribution but also improves service quality assurance capabilities and task execution stability. Overall, the proposed method achieves a better balance between throughput improvement, resource coordination, latency control, and service reliability, demonstrating strong global scheduling capabilities and dynamic environment adaptability.

To further analyze the impact of each component module on the overall scheduling performance, this paper conducts an ablation study on the state refinement mechanism, task node compatibility scoring mechanism, and throughput-oriented reward mechanism in the method. The relevant results are shown in Table 2. Comparing the performance changes under different settings can more clearly reveal the specific roles of each key design in throughput improvement, resource utilization, latency control, and quality of service assurance, and further illustrate the overall collaborative advantages of the complete model in a distributed computing environment.

Different modules all make substantial contributions to the overall scheduling performance, and the complete model maintains the best performance across all evaluation metrics. This indicates that the components in the proposed method are not simply superimposed, but rather form an effective synergy between throughput improvement, resource utilization optimization, and latency control. After removing State Refinement, THR and UTL decrease to 90.91 and 83.12, respectively, while JCT and WAT increase to 36.72 and 13.84, indicating that the state refinement process plays an important role in enhancing the system's state representation capability and improving the policy's perception accuracy of queue changes

and node load fluctuations. After removing Compatibility Scoring, all metrics further decrease, especially THR, JCT, WAT, and FTD, demonstrating that compatibility assessment between tasks and nodes is crucial for improving scheduling matching quality and reducing execution overhead caused by unreasonable allocation. In contrast, removing the Throughput-Aware Reward resulted in the most significant performance degradation, with the THR dropping to 88.76, and JCT, WAT, and FTD deteriorating to 39.54, 15.27, and 3.18, respectively. This indicates that the throughput-oriented reward mechanism plays a crucial role in guiding the strategy to balance long-term service efficiency, wait control, and load coordination. Overall, the complete model achieves a THR of 92.47, a UTL of 85.21, and the lowest JCT, WAT, SLD, LBI, and FTD, while maintaining an SLA of 97.12. This demonstrates that the proposed method can effectively improve adaptive scheduling capabilities and overall task throughput in distributed computing environments through joint optimization of state modeling, compatibility matching, and reward shaping.

**Table 2.** Ablation study results under different method settings

| Setting | THR | UTL | JCT | WAT | SLD | LBI | SLA | FTD |
|---|---|---|---|---|---|---|---|---|
| w/o State Refinement | 90.91 | 83.12 | 36.72 | 13.84 | 1.63 | 0.12 | 96.28 | 2.77 |
| w/o Compatibility Scoring | 89.84 | 82.47 | 38.11 | 14.63 | 1.70 | 0.13 | 95.94 | 2.96 |
| w/o Throughput-Aware Reward | 88.76 | 81.35 | 39.54 | 15.27 | 1.82 | 0.14 | 95.41 | 3.18 |
| Full Model | 92.47 | 85.21 | 34.18 | 12.36 | 1.49 | 0.11 | 97.12 | 2.41 |

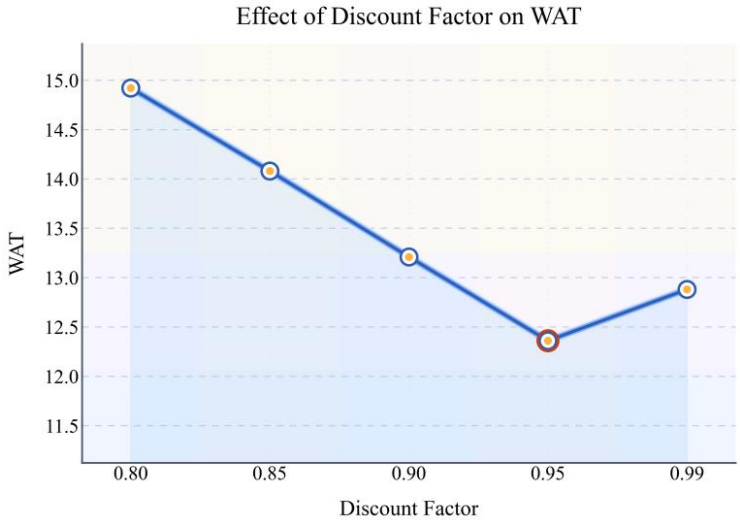


**Figure 2.** The effect of discount factor on WAT experimental results

As shown in Figure 2, as the discount factor gradually increases from 0.80 to 0.95, WAT shows a continuous downward trend. This indicates that within this range, a moderate focus on future returns in the scheduling strategy can more effectively alleviate task queuing and waiting accumulation problems, making the current allocation decision more considerate of subsequent system state evolution, thereby reducing overall waiting overhead. When the discount factor further increases to 0.99, WAT rebounds to some extent, indicating that an excessively strong emphasis on long-term returns can weaken the strategy's ability to respond quickly to current waiting pressure, making scheduling behavior more susceptible to the influence of long-term return estimation biases. Overall, the discount factor has a significant moderating effect on waiting time control, with 0.95 corresponding to the lowest WAT, indicating that this value can better balance the relationship between short-term scheduling efficiency and long-term global optimization, thus

enabling the proposed method to achieve better waiting time performance in continuous decision-making scenarios.

## 5. Conclusion

This paper focuses on the core issue of improving task throughput in distributed computing environments. Addressing the insufficient adaptability of traditional scheduling methods under conditions of dynamic task arrival, heterogeneous resource contention, and complex system state coupling, it proposes a reinforcement learning-based adaptive scheduling method oriented towards long-term benefit optimization. Starting from the practical needs of distributed task scheduling, this method organically integrates system state modeling, task node compatibility assessment, throughput-oriented reward shaping, and policy adaptive optimization into a unified framework. This enables the scheduling process to more fully perceive the correlation between queue changes, resource consumption, and execution feedback in continuous decision-making scenarios. In this way, the model not only improves the overall service capacity and resource utilization efficiency of the system but also achieves a more reasonable coordination between task completion latency, waiting control, load balancing, and service stability. The results show that the proposed method exhibits strong comprehensive advantages across multiple key evaluation indicators, demonstrating that this research provides a targeted and scalable intelligent optimization path for distributed scheduling problems. This method has high application potential for cloud computing platforms, high-performance computing clusters, edge collaboration systems, and resource management scenarios for large-scale artificial intelligence tasks. It helps improve the operational efficiency and service quality of complex computing infrastructure under high concurrency, high dynamics, and high load conditions, and provides methodological support for the autonomous scheduling of intelligent computing platforms.

Future research can further expand the modeling depth and deployment value of this method in more complex practical application contexts. On the one hand, more systematic joint modeling can be carried out around finer-grained task dependencies, cross-node communication overhead, heterogeneous hardware collaboration, and multi-objective resource constraints. This will enable scheduling strategies to not only focus on throughput improvement but also simultaneously consider richer system requirements such as energy consumption control, latency assurance, and service fairness. On the other hand, this method can be further integrated with the online scheduling mechanisms of real-world distributed platforms to enhance the robustness and generalization ability of the model in non-stationary load environments, sudden task impacts, and multi-tenant competition scenarios, thereby bridging the gap between algorithm design and engineering implementation. As intelligent computing infrastructure continues to evolve, reinforcement learning scheduling methods for complex operating environments are expected to play a more important role in data center resource orchestration, industrial internet computing scheduling, intelligent manufacturing task collaboration, and new edge-cloud fusion systems. Therefore, conducting in-depth research in this direction can not only enrich the theoretical system of distributed intelligent scheduling, but also provide important support for the construction of the next generation of efficient, autonomous, and sustainable computing service platforms.

## References

[1] K. Siddesha, G. V. Jayaramaiah, and C. Singh, "A novel deep reinforcement learning scheme for task scheduling in cloud computing," Cluster Computing, vol. 25, no. 6, pp. 4171-4188, 2022.

[2] Z. Lin, C. Li, L. Tian, et al., "A scheduling algorithm based on reinforcement learning for heterogeneous environments," Applied Soft Computing, vol. 130, p. 109707, 2022.

[3] G. Chen, J. Qi, Y. Sun, et al., "A collaborative scheduling method for cloud computing heterogeneous workflows based on deep reinforcement learning," Future Generation Computer Systems, vol. 141, pp. 284-297, 2023.

[4] J. Wang, S. Li, X. Zhang, et al., "Deep reinforcement learning task scheduling method based on server real-time performance," PeerJ Computer Science, vol. 10, p. e2120, 2024.

[5] S. Li, W. Dai, Y. Chen, et al., "Optimization of high-performance computing job scheduling based on offline reinforcement learning," Applied Sciences, vol. 14, no. 23, p. 11220, 2024.

[6] Z. Zhang, C. Xu, K. Liu, et al., "A resource optimization scheduling model and algorithm for heterogeneous computing clusters based on GNN and RL," The Journal of Supercomputing, vol. 80, no. 16, pp. 24138-24172, 2024.

[7] S. Lajili, Z. Brahmi, M. N. Omri, et al., "Federated Reinforcement Learning-based Adaptive Stream Applications Scheduling in Edge and Cloud Computing," Future Generation Computer Systems, p. 108235, 2025.

[8] H. Mao, M. Alizadeh, I. Menache, et al., "Resource management with deep reinforcement learning," in Proceedings of the 15th ACM Workshop on Hot Topics in Networks, 2016, pp. 50-56.

[9] W. Chen, Y. Xu, and X. Wu, "Deep reinforcement learning for multi-resource multi-machine job scheduling," arXiv preprint arXiv:1711.07440, 2017.

[10] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, et al., "Learning scheduling algorithms for data processing clusters," in Proceedings of the ACM Special Interest Group on Data Communication, 2019, pp. 270-288.

[11] S. Liang, Z. Yang, F. Jin, et al., "Data centers job scheduling with deep reinforcement learning," in Pacific-Asia Conference on Knowledge Discovery and Data Mining, Cham: Springer International Publishing, 2020, pp. 906-917.

[12] D. Zhang, D. Dai, Y. He, et al., "RLScheduler: an automated HPC batch job scheduler using reinforcement learning," in SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, 2020, pp. 1-15.

[13] Y. Fan, B. Li, D. Favorite, et al., "Dras: Deep reinforcement learning for cluster scheduling in high performance computing," IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 12, pp. 4903-4917, 2022.

[14] X. Lu, C. Liu, S. Zhu, et al., "RLPTO: A reinforcement learning-based performance-time optimized task and resource scheduling mechanism for distributed machine learning," IEEE Transactions on Parallel and Distributed Systems, vol. 34, no. 12, pp. 3266-3279, 2023.

[15] Y. Ke, "Heterogeneity-Aware Federated Optimization with Secure Aggregation for Distributed Data Mining," 2024.

[16] Z. Liu and D. Abdurehim, "Distributed System Anomaly Detection via Collaborative Discrimination of Reconstruction Residuals and Latent Distribution Shifts," 2025.

[17] Y. Zhan, "Adaptive Representation Learning and Temporal State Modeling for Robust Anomaly Detection in Distributed Systems," 2024.

[18] J. Zhou, "Graph-Based Knowledge Representation and Semantic Alignment for Large Language Model-Driven Entity Recognition," 2025.

[19] Z. Zhu, "Knowledge Graph-Augmented Semantic Fusion for Large Language Model-Based Long-Text Classification," 2024.

[20] R. Hu, "Unified Semantic Matching with Logistic Similarity Learning for Cold-Start Advertising Recommendation," Transactions on Computational and Scientific Methods, vol. 4, no. 2, 2024.

[21] Z. Xiao, C. Moretti, and Q. Li, "Frequent Subgraph Mining and Structural Prompt Injection for Large Language Model Fine-Tuning," 2025.

[22] O. Lin, G. Brennan, and X. Xu, "Structured Pruning-Driven High-Expressive Low-Rank Adapter Learning for Large Language Models," 2025.

[23] J. Hu, Y. Lyu, and J. Jiang, "Task-Aware Regularized Continual Fine-Tuning for Mitigating Catastrophic Forgetting in Large Language Models," 2025.

[24] Z. Ke, "MCA-TAN: Cross-Modal Graph Neural Recommendation with Temporal Behavior and Social Network Fusion," 2025.

[25] Z. Xiao, K. Ying, and B. Venkatesan, "Spatiotemporal Transformer-Based Latency Prediction for Large-Scale Cloud Computing Clusters," 2025.

[26] L. Ren, S. Li, and Z. Liu, "Variational Autoencoder-Based Query Plan Anomaly Detection and Cost Deviation Warning for Database Systems," 2024.