Transactions on Computational and Scientific Methods | Vol. 5, No. 4, 2025
ISSN: 2998-8780
https://pspress.org/index.php/tcsm
Pinnacle Science Press

Topology-Aware Decision Making in Distributed Scheduling via Multi-Agent Reinforcement Learning

Bingxing Wang
Illinois Institute of Technology, Chicago, USA
wanguo0528@gmail.com

Abstract: This study focuses on the problem of multi-node task scheduling and resource optimization in distributed systems. A collaborative optimization method based on multi-agent reinforcement learning is proposed. By modeling the system as a partially observable multi-agent Markov decision process, each agent is enabled to make autonomous decisions based on local observations, while a global reward mechanism ensures overall optimization. In terms of algorithm design, a graph attention mechanism is introduced to enhance the modeling of inter-agent dependencies. In addition, a value function decomposition framework is adopted to improve the stability of joint policy convergence. The experimental setup is built on the real-world Cluster Trace dataset. A simulated environment is constructed to evaluate the proposed method across multiple metrics, including average task completion time, resource utilization, and system throughput. The performance is compared with traditional scheduling strategies and representative reinforcement learning algorithms. Results show that the proposed method achieves significant improvements in scheduling efficiency and resource usage. It effectively enhances the operational performance and intelligent coordination level of distributed systems.

Keywords: Multi-agent reinforcement learning; distributed scheduling; graph attention mechanism; resource optimization.

1. Introduction

With the rapid development of information technology and communication networks, distributed systems have gradually become a key component of modern computing architectures [1]. They are widely applied in fields such as edge computing, cloud computing, industrial automation, intelligent transportation, and energy scheduling. The core features of distributed systems include collaborative operation among nodes, resource sharing, and parallel task processing, which help improve overall computational efficiency and system reliability. However, as system scale expands and task complexity increases, achieving efficient collaboration, dynamic task allocation, and optimal resource scheduling across heterogeneous nodes has become a critical challenge in distributed system research. Traditional centralized control methods often suffer from high computational complexity, communication bottlenecks, and poor system stability in high-dimensional, dynamic, and uncertain environments. These limitations make them inadequate for the performance demands of future intelligent systems [2].

To address these challenges, reinforcement learning (RL), as an intelligent decision-making approach with adaptability and self-optimization capabilities, has attracted significant attention in recent years. Particularly in complex systems with large and dynamic state spaces, RL can optimize policies through interaction with

the environment, aiming to maximize long-term objectives. Building on this, Multi-Agent Reinforcement Learning (MARL) introduces multiple agents to learn and make decisions collaboratively. This allows each agent to operate autonomously while enabling system-wide coordination and optimization. MARL naturally aligns with the structural characteristics of distributed systems and effectively enhances task responsiveness, resource efficiency, and overall robustness. Therefore, applying MARL to collaborative optimization tasks in distributed systems holds strong theoretical and practical significance [3].

One of the main challenges in multi-agent systems lies in the design of cooperation mechanisms, communication structures, and ensuring the stability of policy convergence. In distributed systems, issues such as information asymmetry, uneven resource distribution, and limited communication may arise between nodes. These factors make it difficult for traditional RL methods to be directly applied in multi-agent scenarios. The MARL framework offers a decentralized control paradigm where each agent can learn and make decisions based on local information while achieving global coordination through specific interaction mechanisms. With the advancement of deep learning, deep multi-agent reinforcement learning has further expanded the expressiveness and applicability of MARL algorithms. This makes it possible to handle high-dimensional state and policy spaces, offering new solutions for distributed system optimization.

Currently, MARL-based optimization methods have demonstrated superior performance compared to traditional approaches in various application areas, including energy scheduling, traffic signal control, intelligent manufacturing, and autonomous system collaboration. These methods provide advantages in decision accuracy and real-time responsiveness. They also offer strong transferability and scalability, enabling effective adaptation to complex and dynamic environments with multiple concurrent tasks. Therefore, advancing research on MARL-based collaborative optimization in distributed systems not only promotes the development of intelligent systems toward autonomy, efficiency, and reliability but also provides solid algorithmic and theoretical support for building future ubiquitous intelligent networks [4].

In conclusion, this study aims to conduct in-depth research on "Multi-Agent Reinforcement Learning-Based Collaborative Optimization for Distributed Systems." It explores effective mechanisms and algorithmic frameworks for achieving efficient cooperation among distributed agents in complex and dynamic environments. This research contributes to the theoretical foundations at the intersection of multi-agent systems and distributed optimization and holds broad engineering value.

2. Related work

In recent years, collaborative optimization in distributed systems has emerged as one of the core issues in intelligent systems research. Traditional optimization methods mostly rely on centralized control or heuristic algorithms. These include resource scheduling and task allocation strategies based on integer programming, graph theory models, or genetic algorithms. While effective in specific scenarios, such methods often struggle in distributed environments characterized by multiple objectives, strong coupling, and high dynamics. They fail to adapt quickly to environmental changes or update strategies in time. Moreover, centralized approaches face risks of single-point failure and scalability bottlenecks, which impose significant performance constraints in large-scale deployments.

To address the limitations of centralized control, researchers have gradually introduced distributed optimization methods based on reinforcement learning. Notable progress has been made especially in single-agent reinforcement learning. In scenarios where multiple edge nodes or subsystems are modeled separately, reinforcement learning replaces traditional rules by learning optimal control strategies through continuous interaction. However, single-agent approaches are severely limited in multi-agent systems. They struggle to handle instability between policies and the high-dimensional coordination required by dynamic environments. In response, multi-agent reinforcement learning has emerged as a natural extension. It has been widely

applied to model and optimize collaborative behaviors among agents. Representative methods include Independent Q-Learning [5], MADDPG [6], and QMIX. These explore both practical and theoretical aspects under different frameworks such as fully independent learning, centralized training with decentralized execution, and value function mixing.

Despite the promising results achieved by existing approaches in multi-agent settings, many challenges remain. These include policy non-stationarity, communication cost control among agents, difficulty in joint policy convergence, and limited generalization ability. Recent studies have explored solutions by introducing attention mechanisms, graph neural networks, and meta-learning frameworks to enhance dependency modeling and information integration among agents. Additionally, customized collaboration mechanisms tailored to specific tasks have become a research focus. Examples include task-driven group collaboration and distributed game-theoretic strategy integration. These methods have contributed to the practical viability of multi-agent collaborative optimization. As a result, current research trends are shifting toward more general, stable, and interpretable multi-agent reinforcement learning models to meet the increasingly complex and dynamic coordination demands in distributed systems.

3. Method

In order to cope with the challenges of high task complexity, strong node heterogeneity and limited information transmission in distributed systems, this paper proposes a collaborative optimization method based on Multi-Agent Reinforcement Learning (MARL), which aims to optimize the overall performance of the system through decentralized strategy learning. In this method, each agent represents a sub-node in the system, realizes autonomous learning and decision-making through local perception and interaction with the environment, and realizes collaborative behavior among agents with the help of global reward design and coordination mechanism. This method can effectively overcome the shortcomings of centralized methods in scalability and robustness, and show good adaptability and optimization effect in multi-task collaboration and dynamic resource scheduling scenarios. Its model architecture is shown in Figure 1.

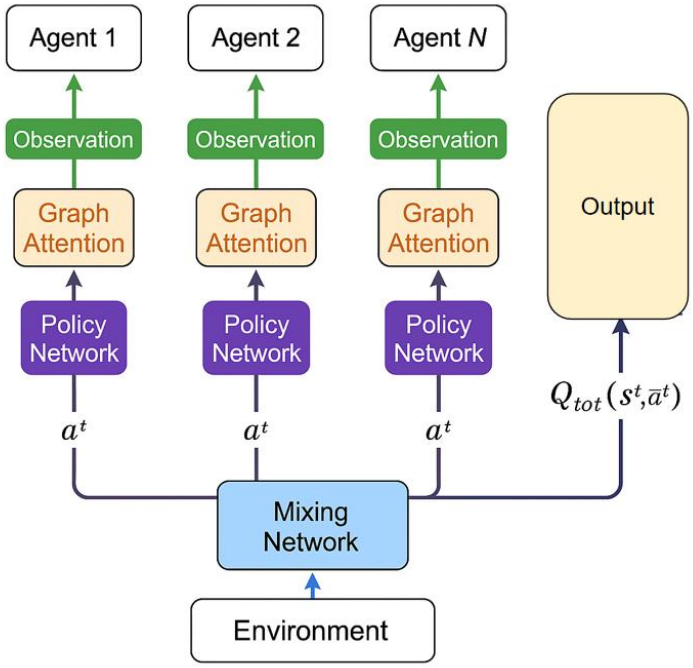


Figure 1. Overall model architecture

As shown in Figure 1, the multi-agent reinforcement learning network architecture proposed in this paper consists of multiple agents. Each agent extracts key features based on local observation information through the graph attention mechanism and outputs the current action based on the policy network. The action information of all agents is fused through the Mixing Network to form a joint action value function to maximize the global return. Under the framework of centralized training and decentralized execution, this structure effectively improves the stability of policy learning and the overall collaborative optimization capability of the system.

In the model construction, the entire distributed system is modeled as a multi-agent extension of a partially observable Markov decision process (POMDP). Suppose there is a set of agents $A = \{a_1, a_2, ..., a_N\}$, each agent a_i receives a local observation o_i^t at time t and selects an action a_i^t to maximize the expected cumulative return. The system is recorded as:

$$s_t \in S$$

In the system, the joint action is $a_t = (a_i^t, ..., a_N^t)$, the global reward is $r^t = R(s^t, a^t)$, and the environment transfer function is $P(s^{t+1} \mid s^t, a^t)$ .Each agent makes decisions through the policy function $\pi_i(a_i^t \mid o_i^t; \theta_i)$. The goal is to jointly optimize all policy parameters $\{\theta_1, ..., \theta_N\}$ under the centralized training decentralized execution (CTDE) framework to maximize the system's overall reward function $E \mid \sum_t \gamma^t r^t \mid$.

In order to enhance the stability and convergence of the strategy, this paper introduces a hybrid architecture based on value function decomposition and uses the QMIX algorithm to model the joint action value function. Specifically, let the joint state-action value function be $Q_{tot}(s^t, a^t)$, and assume that it can be decomposed into a nonlinear additive function of the individual value function, that is, there exists a hybrid network $f_{mix}(\cdot)$ such that:

$$Q_{tot}(s^t, a^t) = f_{mix}(Q_1(o_1^t, a_1^t), ..., Q_N(o_1^t, a_1^t); s^t)$$

Each Q_i is the local value function of the corresponding agent, and the hybrid network is designed to satisfy the monotonicity constraint $\frac{\partial Q_{tot}}{\partial Q_i} \geq 0$, to ensure that the system performance will not degrade when the strategy of each agent is improved. The TD error minimization criterion is used in the training phase, and the optimization objective function is:

$$L(\theta) = E(s^t, a^t, r^t, s^{t+1})[(Q_{tot}(s^t, a^t) - y^t)^2]$$

The target Q value is:

$$y^t = r^t + \gamma \max_{a^{t+1}} Q_{tot}(s^{t+1}, a^{t+1})$$

In addition, in order to address the problem of limited information transmission that may exist in actual systems, this paper integrates the attention mechanism into the policy network to enhance the implicit modeling capabilities between multiple agents. Specifically, the interaction between agents is realized through a graph attention module, and each node performs attention weighting based on neighbor information to learn a more context-aware policy representation. This mechanism enables agents to achieve near-optimal global behavior based on local collaboration in the absence of global information. Combining the above designs, the method proposed in this paper achieves efficient collaboration and optimal task allocation among multiple agents in a distributed system while ensuring decentralized decision-making and execution of the system.

4. Experiment

4.1 Datasets

The dataset used in this study is the Alibaba Cluster Trace (Alibaba Cluster Data V2018), released by Alibaba Group [7]. It is widely used in research on task scheduling, resource allocation, and system optimization in large-scale distributed computing clusters [8-10]. The dataset is derived from server logs in Alibaba's real production environment. It contains 24-hour operational records from over 8,000 machines. The data includes detailed information on CPU, memory, and disk usage, task lifecycles, and job scheduling. It is one of the most representative public datasets for cloud resource scheduling research.

The dataset is well-structured and consists of three core subsets: machine usage, batch task, and container usage. These subsets enable the analysis of system behavior under conditions such as dynamic task submission, resource contention, and service quality assurance. Each record is time-stamped, allowing for the tracking of workload trends across heterogeneous nodes. It also supports modeling and evaluation of system efficiency at various levels, including individual nodes, sub-clusters, and the entire system. These features make the dataset highly suitable for training and validating multi-agent collaborative optimization strategies.

In this study, a highly dynamic and non-stationary distributed system simulation environment was constructed based on scheduling behaviors and resource usage patterns extracted from the dataset. Each agent corresponds to an independent node in the cluster. Agents learn optimal task scheduling and resource allocation strategies through local observation. A global reward mechanism is also incorporated to guide the system toward overall load balancing, resource utilization maximization, and task completion time minimization. This setup validates the feasibility and effectiveness of the proposed method in real-world scenarios.

4.2 Experimental Results

1) Comparative experiment with traditional heuristic scheduling algorithm

This paper first presents a comparative experiment with the traditional heuristic scheduling algorithm, and the actual results are shown in Table 1.

Table 1: Comparison of Scheduling Performance between MARL and Traditional Heuristic Algorithms

Method	**Avg Task Completion Time (s)**	**Resource Utilization (%)**	**System Throughput (tasks/h)**
FIFO[11]	12.7	68.3	2870
Round Robin[12]	10.9	72.1	3120
DRF[13]	9.4	78.5	3345
DQN[14]	9.2	78.1	3403
MARL(Ours)	7.6	85.7	3780

Experimental results show that the proposed MARL method outperforms traditional heuristic algorithms and single-agent reinforcement learning methods across multiple core scheduling metrics. Specifically, MARL achieves the best performance in terms of average task completion time, reaching only 7.6 seconds. This is significantly better than FIFO (12.7 seconds), Round Robin (10.9 seconds), and more advanced methods such as DRF (9.4 seconds) and DQN (9.2 seconds). These results indicate that collaborative learning and policy sharing among agents lead to more efficient task scheduling and resource matching.

In terms of resource utilization, MARL also demonstrates a clear advantage, reaching 85.7%. This represents a notable improvement over DQN (78.1%) and DRF (78.5%). Traditional strategies like FIFO and Round Robin perform poorly, with utilization rates of only 68.3% and 72.1%, respectively, due to their lack of adaptive learning. This suggests that MARL can dynamically optimize resource scheduling strategies and improve overall node utilization in highly dynamic and heterogeneous system environments.

Regarding system throughput, MARL again performs best, completing 3,780 tasks per hour. This marks an improvement of more than 20% over traditional algorithms. The result stems from both the efficient collaboration mechanisms among agents and the model's stability and generalization in complex environments. Overall, MARL enhances the scheduling performance of distributed systems and confirms the feasibility and effectiveness of using multi-agent learning frameworks for collaborative optimization in large-scale heterogeneous settings.

2) Hyperparameter sensitivity experiments

Furthermore, this paper gives the results of hyperparameter sensitivity experiments, mainly focusing on Epoch and Lr. First, the experimental results of Epoch are given, as shown in Table 2.

Table 2: Hyperparameter sensitivity experiment results (Epoch)

Epoch	Avg Task Completion Time (s)	Resource Utilization (%)	System Throughput (tasks/h)
50	10.8	72.4	3090
125	8.9	79.8	3455
150	8.3	82.1	3610
175	7.8	84.3	3715
200	7.6	85.7	3780

Experimental results show that the model's performance in task scheduling improves steadily with the increase in training epochs. The average task completion time decreases from 10.8 seconds at epoch 50 to 7.6 seconds at epoch 200, exhibiting a stable and significant convergence trend. This indicates that, through continuous interaction and learning, the multi-agent reinforcement learning model can gradually form better strategies to improve task assignment and resource matching efficiency.

Resource utilization also shows a clear upward trend, increasing from an initial 72.4% to 85.7%. This result reflects the agents' enhanced perception of node resource states as training progresses. Their strategies become more reasonable, reducing idle resources and waste. Moreover, performance growth becomes stable after 150 epochs, suggesting that the policy is approaching an optimal region and that further training yields diminishing returns.

System throughput increases as well with the number of training epochs, ultimately reaching 3,780 tasks per hour. This trend demonstrates that a multi-agent system, after sufficient training, can coordinate node behaviors efficiently, achieving load balancing and parallel task execution. Therefore, an appropriate training duration is essential for unlocking full model performance. At the same time, it highlights the need to balance performance gains against training costs in practical deployment.

At the same time, the experimental results of different learning rates are given, as shown in Table 3.

Table 3: Hyperparameter sensitivity experiment results (Learning Rate)

Learning Rate	Avg Task	Resource Utilization	System Throughput

	Completion Time (s)	(%)	(tasks/h)
0.001	9.7	74.2	3260
0.0005	8.6	79.5	3475
0.0003	8.1	82.0	3605
0.0002	7.8	84.1	3720
0.0001	7.6	85.7	3780

Experimental results show that the learning rate has a significant impact on the performance of multi-agent reinforcement learning models. A higher learning rate (e.g., 0.001) accelerates early convergence but leads to poor final performance. The average task completion time reaches 9.7 seconds, with both resource utilization and system throughput remaining low. This may be due to large update steps causing oscillations in policy learning, which affect the model's stability and overall scheduling performance.

When the learning rate is reduced to 0.0005 and 0.0003, the model shows clear performance improvements. The average task completion time drops to 8.1 seconds, and system throughput increases to 3,605 tasks per hour. This indicates that a moderate learning rate helps the model converge smoothly to better policies. Agents can coordinate more effectively in task scheduling and resource allocation. Resource utilization also rises steadily, reflecting enhanced adaptability of the learned strategies to system states.

Further reducing the learning rate to 0.0002 and 0.0001 leads to the best overall performance. At 0.0001, the model achieves the most optimal scheduling outcome, with task completion time reduced to 7.6 seconds, resource utilization reaching 85.7%, and throughput rising to 3,780 tasks per hour. These results suggest that a smaller learning rate allows for more fine-grained exploration in the policy space, helping agents avoid local optima and enabling better collaborative optimization. Therefore, properly setting the learning rate is crucial for achieving stable scheduling and improved global performance in reinforcement learning applied to distributed systems.

3) Experiment on the influence of local observation and global reward design on strategy convergence

In this section, this paper further analyzes the impact of local observation and global reward design on strategy convergence. The experimental results are shown in Figure 2.

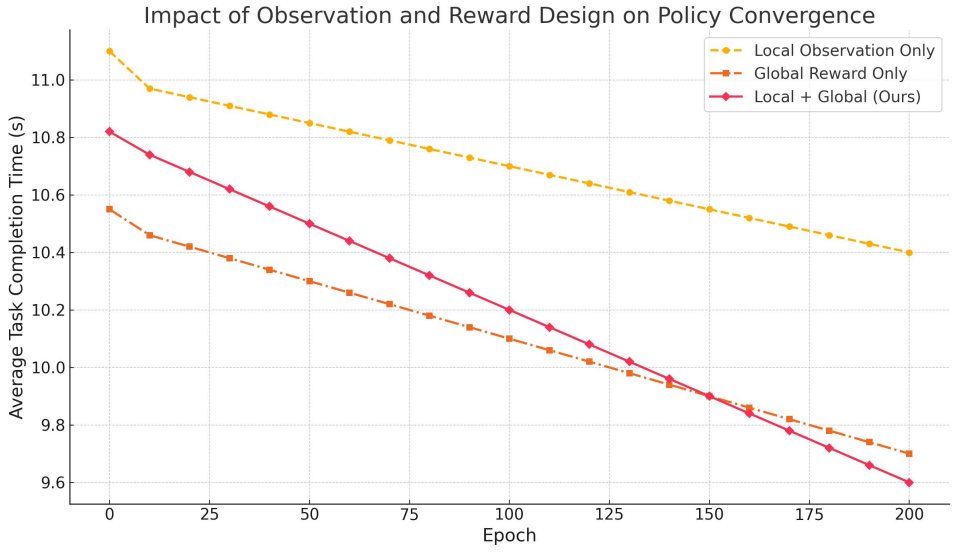


Figure 2. The impact of local observation and global reward design on policy convergence

As shown in Figure 2, when using only local observations, the model exhibits noticeably slow convergence throughout the training process. The average task completion time shows a limited decrease from its initial value and levels off in the later stages. This indicates that, in the absence of global system awareness, agents rely primarily on their own experience. As a result, it becomes difficult to achieve globally optimal task scheduling and resource allocation, which limits the overall system performance improvement.

In contrast, the strategy using only global rewards shows better convergence in the early phase compared to the local observation scheme. The average task completion time drops more rapidly, suggesting that global feedback helps guide agents to adjust their policies toward system-wide optimization. However, due to the lack of fine-grained local perception, agents may fail to respond adequately to local states. This leads to a situation where performance still has room for improvement even after policy convergence.

When local observations are combined with global rewards, the model achieves the best convergence trend. The average task completion time continues to decrease and reaches its minimum at epoch 200. This result demonstrates that the joint modeling of local and global information significantly enhances agent collaboration. It improves responsiveness to local environments while maintaining alignment with global optimization goals. Overall, this design ensures stable policy convergence and substantially improves scheduling efficiency and resource utilization in the distributed system.

4) Experiment on improving multi-node scheduling performance based on graph attention mechanism

Next, this paper also analyzes the ablation experiment of the multi-node scheduling performance based on the graph attention mechanism, and the experimental results are shown in Figure 3.

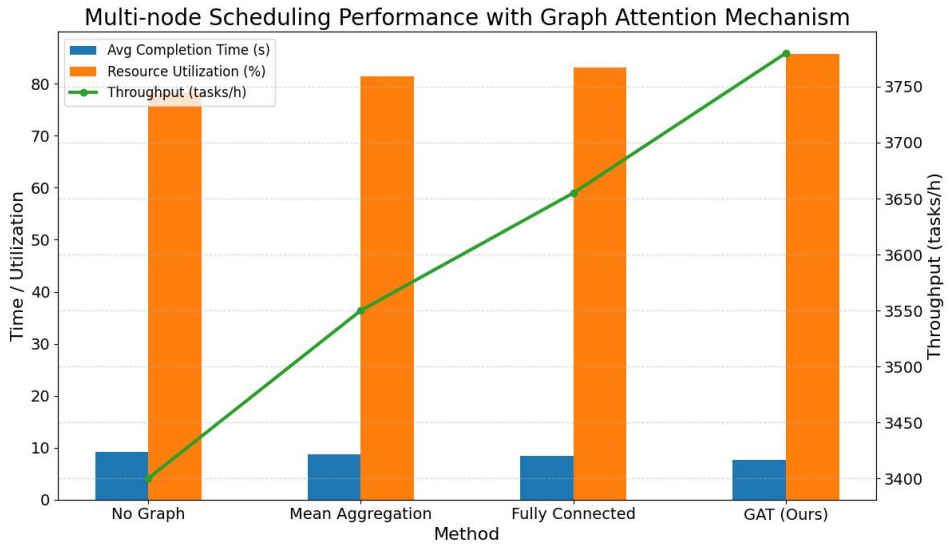


Figure 3. Multi-node Scheduling Performance with Graph Attention Mechanism

As shown in Figure 3, introducing the Graph Attention Network (GAT) significantly improves overall system performance in multi-node scheduling. Compared with strategies that do not use graph structures or rely on mean aggregation or fully connected architectures, GAT achieves the lowest average task completion time at 7.6 seconds. This indicates that, after perceiving local topological information, GAT can make more accurate scheduling decisions, reducing task queuing and waiting time.

In terms of resource utilization, the GAT method also performs best, reaching 85.7%, which is much higher than the 78.2% observed without graph structures. This suggests that the attention mechanism effectively models dependencies among agents, enabling more reasonable coordination and allocation of resources. It helps avoid idle or overloaded nodes. Although the fully connected approach shows some improvement, it

lacks an explicit weighting mechanism. Unlike GAT, it cannot assign higher importance to key neighbors, resulting in performance gaps.

The trend in system throughput further supports these findings. As the graph modeling capability improves, task processing capacity steadily increases. The GAT-based strategy ultimately reaches a peak of 3,780 tasks per hour. Overall, the GAT scheduling strategy outperforms other methods not only on individual metrics but also in terms of system-wide efficiency and collaborative decision-making among agents. This demonstrates its strong potential for application in complex distributed environments.

5) Loss function changes with epoch

Finally, this paper gives a graph of the loss function changing with epoch, as shown in Figure 4.

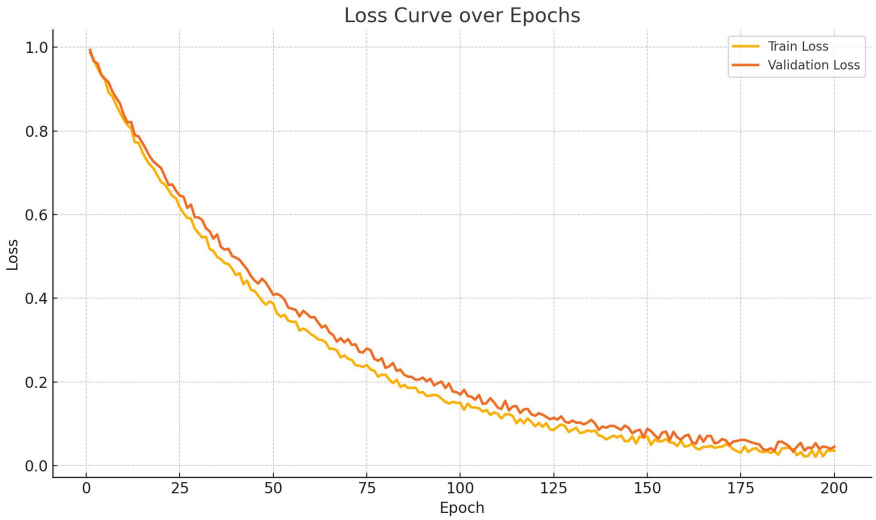


Figure 4. Loss function drop graph

As shown in Figure 4, both the training loss and validation loss exhibit a steady downward trend throughout the training process. This indicates that the model is progressively learning effective strategies and gradually approaching the optimal solution. The loss decreases rapidly during the first 100 epochs, suggesting that the model quickly captures the fundamental patterns of the task in the early stages.

As training continues, the rate of loss reduction slows down and stabilizes around epoch 150. At this point, the gap between training loss and validation loss becomes minimal. This suggests that the model performs well not only on the training set but also generalizes effectively to unseen data, with no significant signs of overfitting. The stable convergence process reflects the effectiveness of model training and the robustness of policy optimization. Moreover, the overall fluctuation in validation loss remains small, indicating that the model maintains high consistency and robustness across multiple training rounds. It is not affected by external disturbances that could cause performance instability. Overall, the observed loss trend demonstrates good convergence and stability of the proposed method, providing a solid foundation for improving task scheduling performance in subsequent applications.

5. Conclusion

This paper addresses the problem of multi-node collaborative optimization in distributed systems and proposes a scheduling method based on multi-agent reinforcement learning. Under a centralized training and decentralized execution framework, the method introduces a graph attention mechanism and joint reward design. These components enhance the efficiency of information exchange among agents and improve policy

convergence quality. Experimental results under various settings show that the proposed method outperforms traditional heuristic algorithms and classical single-agent reinforcement learning models in key metrics such as task completion time, resource utilization, and system throughput. This validates its adaptability and effectiveness in complex scheduling scenarios. To further improve the learning capability and generalization performance of the multi-agent system, a graph-based modeling module is incorporated into the policy network. Multi-layer attention computation enhances agents' sensitivity to the states of neighboring nodes. The experimental results demonstrate that this mechanism significantly improves collaborative scheduling among agents, guiding the entire system toward a globally optimal state. In addition, a comparative analysis of local observation and global reward mechanisms confirms their combined contribution to stable policy convergence.

The paper also conducts sensitivity experiments on key hyperparameters, including learning rate and training epochs. Results indicate that appropriate hyperparameter settings not only improve learning efficiency but also play a critical role in the final performance of the learned policy. Overall, the proposed method demonstrates strong scalability and generality. It is applicable to scheduling optimization tasks in various dynamic distributed environments, offering practical algorithmic support and theoretical insights for intelligent scheduling systems. Future work will expand in three directions. First, we will study the convergence mechanisms of multi-agent policies under communication constraints in large-scale heterogeneous environments. Second, we will explore collaborative learning methods in privacy-sensitive distributed scenarios by integrating mechanisms such as federated learning. Third, we aim to deploy the current model on real-world edge computing platforms for system-level integration and engineering validation, enhancing the practical applicability and scalability of the proposed algorithm.

References

[1] Jayanetti, Amanda, Saman Halgamuge, and Rajkumar Buyya. "Multi-agent deep reinforcement learning framework for renewable energy-aware workflow scheduling on distributed cloud data centers." IEEE Transactions on Parallel and Distributed Systems 35.4 (2024): 604-615.

[2] Wang, Xiaohan, et al. "Dynamic scheduling of tasks in cloud manufacturing with multi-agent reinforcement learning." Journal of Manufacturing Systems 65 (2022): 130-145.

[3] Jing, Xuan, et al. "Multi-agent reinforcement learning based on graph convolutional network for flexible job shop scheduling." Journal of Intelligent Manufacturing 35.1 (2024): 75-93.

[4] Gui, Yong, et al. "Collaborative dynamic scheduling in a self-organizing manufacturing system using multi-agent reinforcement learning." Advanced Engineering Informatics 62 (2024): 102646.

[5] Jiang, Jiechuan, and Zongqing Lu. "I2Q: A fully decentralized Q-learning algorithm." Advances in Neural Information Processing Systems 35 (2022): 20469-20481.

[6] Du, Jianbo, et al. "MADDPG-based joint service placement and task offloading in MEC empowered air−ground integrated networks." IEEE Internet of Things Journal 11.6 (2023): 10600-10615.

[7] Chen, W., Ye, K., Wang, Y., Xu, G., & Xu, C. Z. (2018, December). How does the workload look like in production cloud? analysis and clustering of workloads on alibaba cluster trace. In 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS) (pp. 102-109). IEEE.

[8] Lu, C., Ye, K., Xu, G., Xu, C. Z., & Bai, T. (2017, December). Imbalance in the cloud: An analysis on alibaba cluster trace. In 2017 IEEE International Conference on Big Data (Big Data) (pp. 2884-2892). IEEE.

[9] Everman, B., Rajendran, N., Li, X., & Zong, Z. (2021). Improving the cost efficiency of large-scale cloud systems running hybrid workloads-A case study of Alibaba cluster traces. Sustainable Computing: Informatics and Systems, 30, 100528.

[10] Gong, C., Ma, M., Zeng, L., Yang, Y., Ge, X., & Wu, L. (2024, May). Delay Analysis of Multi-Priority Computing Tasks in Alibaba Cluster Traces. In IEEE INFOCOM 2024-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) (pp. 1-6). IEEE.

[11] Castro, F., Ma, H., Nazerzadeh, H., & Yan, C. (2021). Randomized FIFO mechanisms. arXiv preprint arXiv:2111.10706.

[12] Alhaidari, F., & Balharith, T. Z. (2021). Enhanced round-robin algorithm in the cloud computing environment for optimal task scheduling. Computers, 10(5), 63.

[13] Chang, Ying, and Qinghua Zhu. "Optimized Resource Scheduling for Open Source Mobile Network Heterogeneous Computing Engines: A Comparative Study of DRF and H-DRF Algorithms." 2024 International Conference on Information Technology, Comunication Ecosystem and Management (ITCEM). IEEE, 2024.

[14] Zhang, Lixiang, et al. "Distributed real-time scheduling in cloud manufacturing by deep reinforcement learning." IEEE Transactions on Industrial Informatics 18.12 (2022): 8999-9007.