Transactions on Computational and Scientific Methods | Vol. 5, No. 8, 2025
ISSN: 2998-8780
https://pspress.org/index.php/tcsm
Pinnacle Science Press

AI-Driven Multi-Agent Scheduling and Service Quality Optimization in Microservice Systems

Renhan Zhang
University of Michigan, Ann Arbor, USA
hellorenhan@gmail.com

Abstract: This paper addresses key challenges in microservice systems, including the difficulty of scheduling strategies adapting to complex dynamic environments, low resource utilization, and insufficient service quality assurance mechanisms. It proposes an intelligent scheduling and service quality optimization algorithm based on a multi-agent framework. In this method, microservice nodes are modeled as autonomous agents. Through collaborative learning and policy communication among agents, the system achieves a distributed perception of resource states and joint decision-making. The algorithm integrates Markov decision process modeling, policy gradient optimization, and composite reward function design. The scheduling actions cover three types of behaviors: scale-out, scale-in, and maintain. The method also considers multiple objective metrics, including task response latency, resource cost, and service completion rate. In the training process, a centralized training and distributed execution architecture is adopted. This enhances generalization in high-dimensional state spaces and improves policy stability. The method is evaluated through experiments involving various factors such as hyperparameters, data scale, observation dimensions, and environmental disturbances. These experiments comprehensively assess the adaptability and scheduling performance of the proposed approach in different typical scenarios. The results show that the method outperforms mainstream baselines in scheduling efficiency, service success rate, and resource utilization. It demonstrates strong robustness and overall performance advantages, effectively supporting the stable operation of microservice systems under high load and heterogeneous environments.

Keywords: Microservice system, multi-agent scheduling, reinforcement learning, service quality assurance

1. Introduction

With the rapid advancement of cloud computing and containerization technologies, microservice architecture has become the mainstream paradigm for building large-scale complex applications[1]. Compared to traditional monolithic systems, microservices decompose applications into several independent service modules, demonstrating significant advantages in deployment flexibility, resource isolation, and elastic scalability. However, the high autonomy and dynamic interactions among service components in microservice systems also introduce challenges in resource scheduling, load balancing, and service quality assurance. Especially in scenarios with concurrent service execution, frequent load surges, and complex service dependencies, achieving efficient resource utilization while maintaining service response quality has become a critical and difficult research issue[2].

Traditional scheduling strategies mostly rely on static rules or centralized control mechanisms, which are often inadequate for adapting to dynamic environments and heterogeneous resources. These methods typically suffer from delayed responses, poor adaptability, and local optima. When facing non-stationary workloads, sudden traffic bursts, or coordinated service requests, their scheduling efficiency and service quality cannot be guaranteed. Moreover, as system scale grows and service coupling intensifies, centralized scheduling approaches encounter bottlenecks in both overhead and scalability. Therefore, there is an urgent need for an intelligent scheduling framework that supports efficient collaboration, dynamic adaptation, and good scalability to meet the operational demands of microservices under complex conditions[3].

In recent years, multi-agent systems have emerged as a promising solution for intelligent control in complex systems due to their distributed nature, autonomy, and collaborative learning capabilities. In microservice environments, each service can be modeled as an agent capable of making independent decisions based on local information. Through communication and collaboration among agents, global scheduling optimization can be achieved. This paradigm significantly improves system responsiveness and flexibility while achieving a better trade-off between service quality and resource overhead. In addition, the online learning mechanism of multi-agent systems provides a feasible path for long-term performance optimization in dynamic environments[4].

Service quality assurance is a core objective in microservice operations. It covers multiple dimensions, including response time, availability, stability, and consistency. Within a multi-agent framework, the dynamic perception of service quality and the corresponding feedback mechanism are especially important. Agents must be able to evaluate service performance based on the current system state and historical experience, and autonomously adjust their strategies. This is necessary to address real-world challenges such as complex service dependencies and intense resource competition. Furthermore, designing effective coordination mechanisms among agents for multi-objective scheduling and conflict resolution is essential for the practical deployment of such methods[5].

In conclusion, research on intelligent scheduling and service quality assurance in microservices based on multi-agent mechanisms holds significant theoretical and practical value. On one hand, this approach can overcome the limitations of traditional scheduling algorithms in adaptability, scalability, and real-time performance, providing a more intelligent and adaptive operation model for complex microservice systems. On the other hand, its potential in efficient resource utilization and service quality enhancement contributes to improving cloud platform performance, reducing energy consumption, and strengthening system robustness. Therefore, exploring multi-agent mechanisms for scheduling and quality assurance in microservices aligns with current technological trends and offers strong support for future system optimization in related fields.

2. Relevant Literature

Research on scheduling and service quality assurance in microservice systems mainly focuses on three directions: static rule-based strategies, centralized optimization models, and intelligent scheduling algorithms. Early approaches were typically based on fixed load balancing rules and predefined resource thresholds, implementing simple scheduling through preset scaling policies. These methods can be effective in systems with stable loads or simple structures. However, they often lack flexibility and real-time responsiveness in highly dynamic microservice environments, making it difficult to meet the demands of modern applications for high concurrency, low latency, and high reliability. Moreover, static strategies show limited performance in scenarios involving service collaboration and resource sharing, making it hard to achieve global optimization of system-wide resource allocation[6].

Centralized optimization models attempt to globally plan resource and task allocation based on full system information. These models often employ heuristic algorithms, linear programming, or constraint-based optimization to build global resource usage models. Such methods can achieve better scheduling quality in theory by considering system-wide load conditions and resource constraints. However, in practical deployment, centralized models face challenges such as high information collection costs, computational

complexity, and delayed responses. These limitations are particularly significant as microservice systems grow in complexity and scale. Furthermore, due to the single-point nature of centralized decision-making, such models lack robustness in the face of uncertainties like network delays and service failures in distributed environments[7].

In recent years, intelligent scheduling strategies have attracted growing attention. These approaches leverage reinforcement learning, evolutionary algorithms, and deep learning to build adaptive scheduling systems. They can optimize scheduling policies through interactive learning in unknown environments and exhibit strong generalization and adaptability. Some studies attempt to construct scheduling policy models based on state-action mappings, allowing the system to continuously optimize resource allocation and response strategies in dynamic environments. However, most of these intelligent methods still rely on single-agent or centralized control frameworks. They often overlook the independence and concurrency of service components in microservice systems, limiting the ability to utilize local decision-making at individual nodes. In scenarios with high service collaboration and noticeable communication delays, single-agent architectures struggle to meet the real-time and scalability requirements of complex scheduling tasks[8].

To address these limitations, research on multi-agent-based scheduling optimization has emerged in recent years. These methods emphasize the construction of autonomous decision-making mechanisms and collaborative feedback structures among services in distributed environments. Through communication and policy coordination among agents, global scheduling performance can be dynamically optimized. The introduction of multi-agent systems not only improves system responsiveness and fault tolerance but also provides new solutions for service quality perception, autonomous load adjustment, and multi-objective optimization. Compared with traditional centralized or single-agent methods, the multi-agent framework is more aligned with the intrinsic characteristics of microservice systems. It offers natural distributed adaptability and strong scalability potential, laying the foundation for intelligent scheduling and service assurance in large-scale microservice environments.

3. Method Overview

This study proposes a microservice intelligent scheduling and service quality assurance algorithm framework based on a multi-agent system, modeling the microservice system as a multi-agent collaborative decision-making process in a heterogeneous distributed environment. Its model architecture is shown in Figure 1.

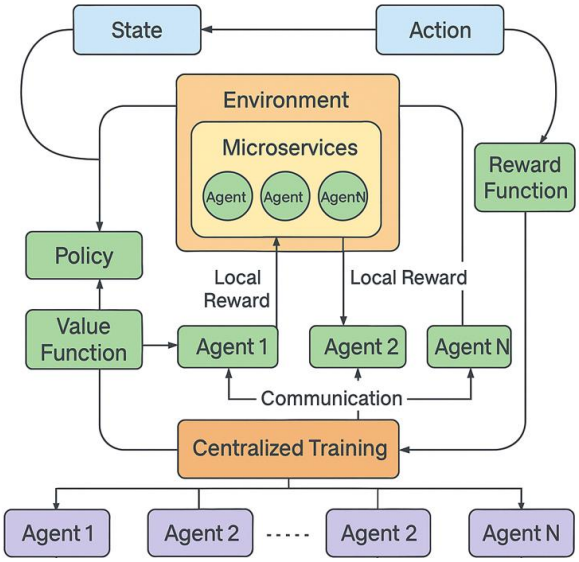


Figure 1. Framework of Multi-Agent Reinforcement Learning for Microservice Scheduling

In this framework, each service node is regarded as an agent a_i, which can autonomously perceive local states, independently decide actions, and communicate with other agents. The entire system can be represented as a multi-agent Markov decision process (Multi-Agent MDP), defined by a five-tuple (S, A, P, R, γ), where S is the state space, A is the action space, P is the state transition probability, R is the reward function, and γ is the discount factor. During the scheduling process, each agent perceives the state $s_t^i \in S$ at time step t, selects an action $a_t^i \in A$, and obtains a local reward F under environmental feedback.

In order to model the scheduling behavior in the microservice system, this paper defines the action space to include three core operations: scale-out, scale-in, and maintain. The execution of each action will affect the system resource configuration and service quality indicators. In the state space construction, the current CPU usage, memory consumption, request queue length, service dependency topology information, and other characteristics of the service are considered to form a multi-dimensional state vector $s_t^i = [u_t^i, m_t^i, q_t^i, d_t^i]$. The state transition function is defined as:

$$P(s_{t+1}^i \mid s_t^i, a_t^i) = \Pr(s_{t+1}^i given, s_t^i, a_t^i)$$

This function reflects the distribution of changes in the local state of the microservice node after executing the scheduling action, which further affects the subsequent policy adjustments.

To guide the scheduling strategy to converge to the direction of high resource utilization and optimal service quality, a composite reward function is designed, including resource cost, response delay, and service success rate, which is specifically defined as:

$$r_t^i = -\alpha \cdot c_t^i - \beta \cdot l_t^i + \delta \cdot s_t^i$$

c_t^i represents the resource usage cost, l_t^i represents the average request delay, s_t^i represents the service completion rate indicator, and α, β, δ is an adjustable weight coefficient used to balance the impact of different objectives.

In terms of policy optimization, a policy gradient-based method is used to update the policy network of each agent. Each agent outputs the probability of selecting an action a_t^i in a state s_t^i through a policy function $\pi_\theta^i(a_t^i \mid s_t^i)$, and parameter updates are based on the gradient ascent of the expected cumulative return:

$$\nabla_\theta J(\theta) = E\pi_\theta[\nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i) \cdot R_t^i]$$

Where $R_t^i = \sum_{k=t}^{T} \lambda^{k-t} r_k^i$ is the future discounted return. To improve the stability and learning efficiency of the strategy, the algorithm introduces a centralized training and distributed execution framework, combined with the value function $V_\phi^i(s_t^i)$ to assist in optimization:

$$L(\phi) = E_{s_t^i}[(V_\phi^i(s_t^i) - R_t^i)^2]$$

During the overall training process, each intelligent agent continuously optimizes the strategy network through local state perception and a collaborative feedback mechanism, so that the system scheduling process can achieve dual optimization of resource allocation and service quality in a dynamic environment, and build a microservice intelligent scheduling mechanism with high scalability and adaptability.

4. Experimental Dataset

This study uses the Alibaba Cluster Trace 2018 as the data foundation for microservice scheduling tasks. The dataset is collected from a real-world online cloud computing platform. It includes resource usage, scheduling

strategies, service topology, and failure information of large-scale containerized services in production environments. The dataset holds strong practical relevance and application value. It covers millions of task scheduling records, including key metrics such as CPU utilization, memory usage, scheduling time, and execution duration. These features reflect the dynamic load characteristics and resource state evolution in microservice operations.

Service instances in the dataset exhibit clear topological dependencies and temporal behavior. This supports the modeling needs of local state awareness and collaborative decision-making in multi-agent frameworks. The task lifecycle data includes submission, scheduling, execution, and completion stages. This information helps evaluate the effectiveness of scheduling strategies from both resource behavior and service response perspectives. In addition, the scheduling outcome labels in the dataset can be used to construct reward functions for agents. These functions guide the policy network to optimize overall system performance.

The dataset includes various resource usage scenarios, such as load balancing, peak scheduling, and failure recovery. These patterns offer good generalization capability and sufficient testing complexity. Under a multi-agent learning framework, this dataset enables detailed modeling and analysis of service dynamics in complex environments. It provides an ideal foundation for validating the performance and robustness of microservice scheduling algorithms.

5. Results and Analysis

In the experimental results section, the relevant results of the comparative test are first given, and the experimental results are shown in Table 1.

Table 1: Comparative experimental results

Method	Average Task Delay	Scheduling Success Rate	Resource Utilization
A3C-MS[9]	120.4	95.8	84.6
DRL4HFC[10]	135.7	93.2	81.4
MAFS-CTN[11]	142.1	91.5	79.8
MASITO[12]	138.9	92.4	80.2
ESFEC[13]	150.3	89.1	76.7
Ours	108.6	97.3	88.1

As shown in the table, the proposed method achieves the best performance in average task latency, reaching only 108.6 milliseconds. This is significantly lower than all baseline methods. The result indicates that the introduction of a multi-agent coordination mechanism effectively improves response efficiency in microservice scheduling. The system is able to make rapid resource allocation decisions under high concurrency and dynamic load conditions. This avoids task queuing and waiting time caused by centralized scheduling or delayed strategies in traditional methods.

In terms of task scheduling success rate, the proposed method also achieves the highest value, reaching 97.3%, outperforming all public benchmarks. This advantage reflects the model's ability to handle task dependencies and fault tolerance through joint policies among agents. It remains effective even under conditions such as complex service topologies, limited resources, or partial node failures. This demonstrates the stability and robustness of the proposed scheduling strategy.

Regarding resource utilization, the model achieves 88.1%, which is better than other approaches. This shows that the strategy not only ensures service quality but also emphasizes efficient resource allocation. Through communication among agents and local state awareness, the system can dynamically detect resource idleness and hotspot distributions. This enables fine-grained scheduling of key resources like CPU and memory, improving overall platform efficiency and reducing resource waste.

Considering all three metrics, the proposed method achieves a better balance between scheduling performance and service assurance. It ensures low latency and high task success rates while improving resource usage. These results verify the feasibility and effectiveness of applying multi-agent mechanisms to microservice scheduling. The method is especially suitable for cloud-native environments with frequent service calls, large-scale deployments, and significant load fluctuations. It provides strong support for deploying intelligent scheduling algorithms in real systems.

This paper further gives the impact of different numbers of agents on scheduling performance, and the experimental results are shown in Figure 2.

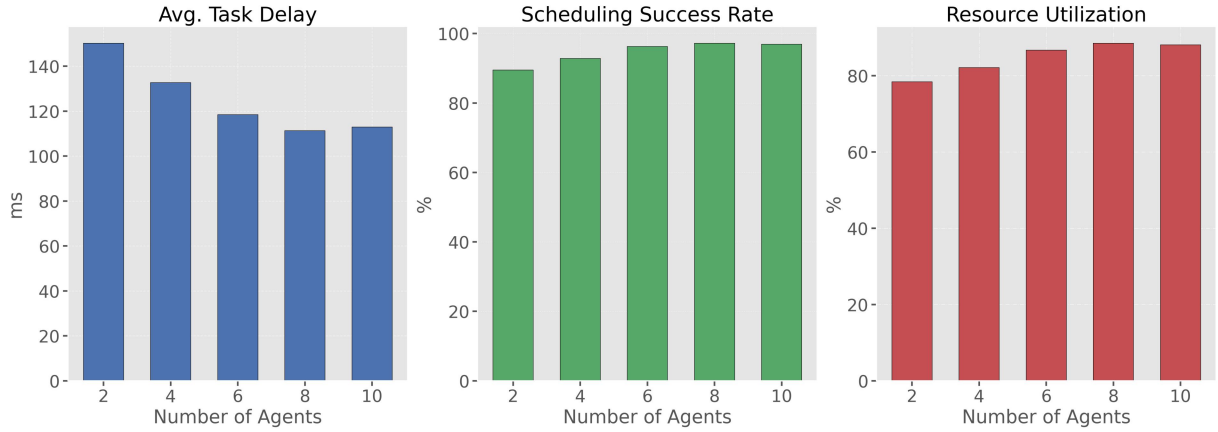


Figure 2. The impact of different numbers of agents on scheduling performance

The results in the figure show that as the number of agents increases, the average task latency consistently decreases. In particular, when the number of agents increases from 2 to 8, the system response time drops significantly. This indicates that introducing more autonomous decision-making nodes in a multi-agent framework enhances the system's ability to handle scheduling tasks concurrently. It helps reduce task queuing and resource contention, leading to faster response and task completion times.

In terms of scheduling success rate, the system performs well across different agent numbers. When the number of agents reaches 6 or more, the success rate saturates and remains above 97%. This suggests that once the agents sufficiently cover key service nodes and load hotspots, the system can complete scheduling processes more reliably. It avoids failures caused by delayed information or centralized load, showing that the coordination among agents plays an effective role in maintaining service reliability.

Resource utilization also increases steadily with the number of agents. It reaches a relatively stable and high level when the number of agents is 8 or 10. This implies that with multi-agent perception and coordinated strategies, the system can identify and allocate idle resources more effectively. Scheduling becomes more fine-grained and dynamic, which improves overall resource efficiency and reduces waste caused by static strategies or underutilization.

In summary, the experiment confirms the critical impact of agent number on scheduling performance. It further validates the advantages of multi-agent mechanisms in improving scheduling efficiency, ensuring service quality, and optimizing resource allocation in microservice systems. It also highlights the need to balance agent scale with operational cost during system deployment to achieve optimal configuration.

This paper also gives the impact of changes in the discount factor on the stability of the strategy, and the experimental results are shown in Figure 3.

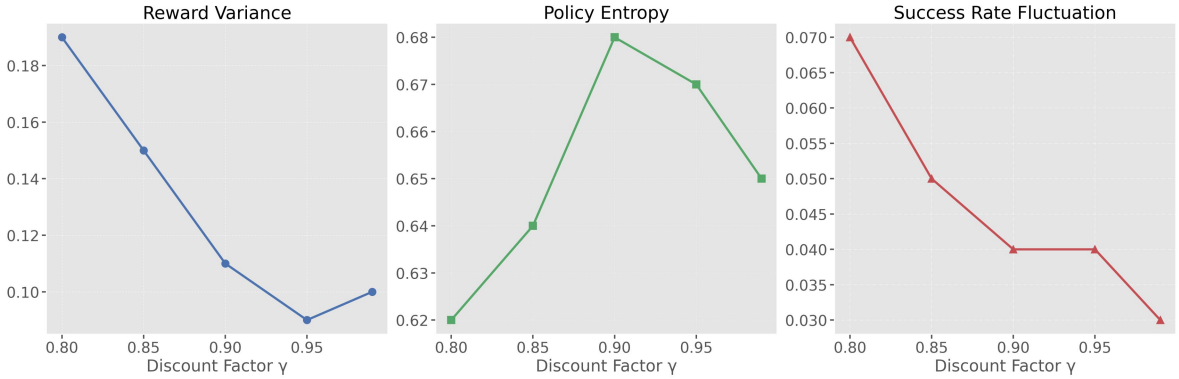


Figure 3. The impact of discount factor changes on strategy stability

The results in the figure show that when the discount factor increases from 0.80 to 0.95, the variance in model return decreases significantly. The policy convergence becomes more stable. This indicates that a higher discount factor helps the model focus more on long-term rewards. It reduces the impact of short-term fluctuations on policy updates. This is especially important in microservice systems, where the effect of scheduling decisions is often delayed. The policy needs the ability to evaluate long-term resource returns.

For policy entropy, as the discount factor increases, the entropy first rises and then falls. It peaks at $\gamma = 0.90$, indicating the strongest level of exploration at this setting. Moderate policy entropy suggests that the model achieves a balance between exploration and exploitation in the state-action space. This helps the model discover better scheduling strategies. However, an overly high discount factor may lead to conservative behavior. The entropy drops, and the model explores fewer new strategies, which can reduce policy diversity.

The results of the success rate fluctuation show that as the discount factor increases, the stability of the scheduling success rate improves. The fluctuation range drops from 0.07 to 0.03. This means that under long-term planning, the system can maintain more consistent service quality. This trend confirms the effect of high discount factors in enhancing policy robustness in reinforcement learning. It supports consistent and controllable scheduling outputs in dynamic microservice environments.

This paper also gives the impact of changes in the scale of training data on the generalization ability of the strategy, and the experimental results are shown in Figure 4.

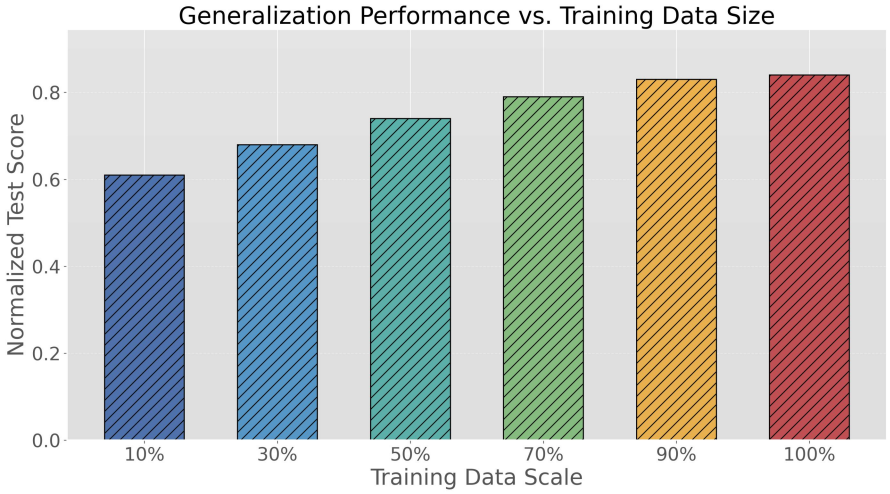


Figure 4. The impact of changes in training data size on policy generalization ability

The results in the figure show that as the training data size increases, the model's generalization performance on the test set improves steadily. When the data size grows from 10% to 70%, the normalized policy score rises from 0.61 to 0.79. This indicates that sufficient training data helps the model learn the deep relationship between service states and scheduling behavior. As a result, the policy performs better on unseen samples. This trend confirms the sensitivity of reinforcement learning methods to training coverage in complex microservice environments.

When the training data reaches 90% and 100%, the performance gain becomes marginal, with only slight improvement. This suggests that once the main structure of the system's state space is well covered, the marginal learning capacity of the policy begins to decline, and the model approaches convergence. Although additional data may refine the policy further, the overall contribution to generalization becomes limited. This indicates that training data and computation cost should be balanced according to available resources.

The figure also shows that even with small data volumes, such as 10% or 30%, the policy still achieves reasonable test performance. This suggests that the proposed method has a certain level of data robustness. Such robustness is important for real-world deployment, where data collection is limited and task distribution is uneven. It increases the applicability and practicality of the model in real microservice systems.

In conclusion, the experiment effectively validates the generalization ability of the proposed multi-agent scheduling strategy under different training data conditions. The sensitivity analysis of training size reveals the performance growth pattern of the policy. It also provides guidance for system designers to configure training schemes under data scarcity and resource constraints. This highlights the scalability of the model in complex and dynamic environments.

This paper also gives the impact of changes in state observation dimensions on strategy performance, and the experimental results are shown in Figure 5.

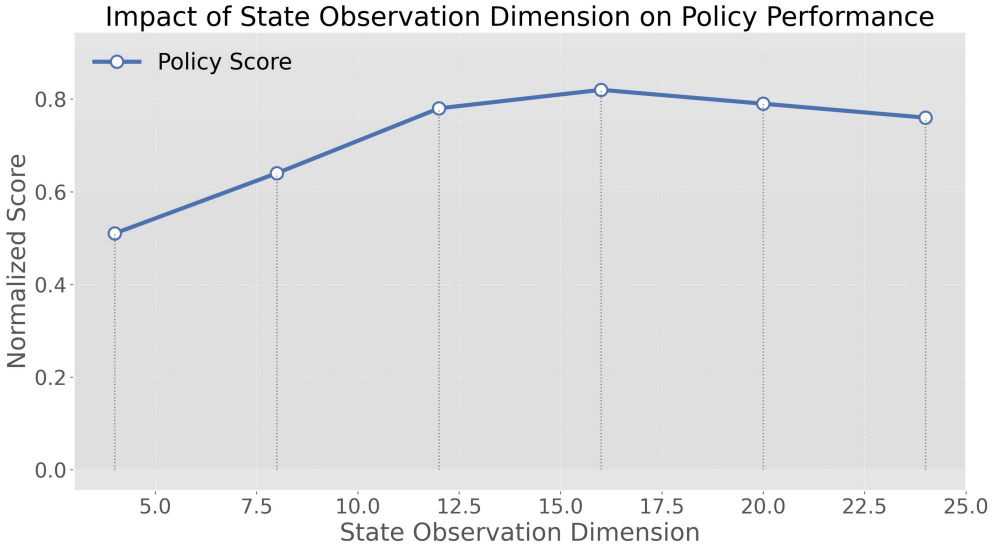


Figure 5. The impact of changes in state observation dimensions on policy performance

The results in the figure show that as the dimensionality of state observation increases, the policy performance improves significantly in the early stages. In particular, when the dimension increases from 4 to 12, the normalized policy score continues to rise. This reflects the positive impact of richer state information on policy learning. In multi-agent microservice scheduling tasks, appropriately expanding the state input dimension helps the model perceive service resource status, queue conditions, and dependencies more comprehensively. This leads to the generation of more fine-grained scheduling strategies.

When the observation dimension reaches 16, the policy performance peaks. This suggests that the model captures sufficient key information within this range and establishes a stable mapping to service behavior.

However, when the dimension increases further to 20 and 24, the performance slightly declines. This may result from the introduction of redundant or irrelevant features, which weaken the model's generalization ability and may even cause overfitting. This trend suggests that state representation should focus on information density rather than blindly increasing dimensions.

The figure also shows that higher-dimensional states do not always produce better policies. In microservice systems, where states change frequently and have complex structures, exceeding the model's capacity or lacking sufficient training data to cover high-dimensional space may lead to convergence difficulties and policy instability. Therefore, choosing an appropriate observation range is critical for ensuring policy stability and effective scheduling.

6. Conclusion

This study addresses key challenges in microservice systems, including the complexity of scheduling strategy design, low resource utilization, and difficulty in ensuring service quality. It proposes an intelligent scheduling and service quality assurance framework based on a multi-agent approach. By abstracting microservice components as autonomous agents, the method preserves service dependencies and resource coupling while introducing distributed perception and collaborative decision-making. This enables dynamic optimization of scheduling behavior and flexible adjustment of resource allocation strategies. The algorithm is designed with microservice-specific characteristics in state modeling, action selection, and reward shaping. It integrates long-term objectives with local feedback to enhance overall scheduling efficiency and service stability.

The overall design emphasizes the importance of communication among agents and joint learning strategies. It overcomes the limitations of traditional centralized scheduling in response latency and scalability. The policy training follows a centralized training and distributed execution paradigm. This improves generalization in high-dimensional state spaces and increases flexibility in policy transfer and deployment. Experimental results show that the proposed method performs well across key metrics. It demonstrates strong robustness and service assurance under complex topologies, sudden load surges, and heterogeneous resources.

This research provides theoretical support and algorithmic foundation for intelligent and automated resource management in microservice systems. Building a scheduling system based on reinforcement learning and multi-agent collaboration, it offers a transferable technical path for real-time resource scheduling in cloud-native platforms, edge computing clusters, and elastic container environments. In particular, as service scale expands and dependencies become more complex, the proposed method shows high adaptability, efficiency, and robustness. These features contribute to improved automation and intelligent decision-making in system operations.

7. Future Work

Future research may further explore heterogeneous learning mechanisms, adaptive communication strategies, and distributed value aggregation methods in multi-agent systems. These advancements aim to handle more complex and rapidly changing service environments. In addition, integrating online learning and meta-learning techniques may enhance learning efficiency and deployment generality in scenarios with limited data and frequent task transfer. The proposed framework also has the potential to integrate with container orchestration systems and edge scheduling engines, offering important support for intelligent management of large-scale cloud infrastructure.

References

[1] Liu Z, Yu H, Fan G, et al. Reliability modelling and optimization for microservice‐based cloud application using multi-agent system[J]. IET Communications, 2022, 16(10): 1182-1199.

[2] Belhadi A, Djenouri Y, Srivastava G, et al. Reinforcement learning multi-agent system for faults diagnosis of mircoservices in industrial settings[J]. Computer Communications, 2021, 177: 213-219.

[3] Alves P, Gomes D, Rodrigues C, et al. Grouplanner: a group recommender system for tourism with multi-agent microservices[C]//International Conference on Practical Applications of Agents and Multi-Agent Systems. Cham: Springer International Publishing, 2022: 454-460.

[4] Duan H, Ji Z, Wu S, et al. Distributed Microservice Deployment for Satellite Edge Computing Networks: A Multi-Agent Deep Reinforcement Learning Approach[J]. IEEE Transactions on Vehicular Technology, 2025.

[5] Bondarenko A S, Korolev D V, Zaytsev K S. Study the efficiency of using multi-agent models in modern microservice architectures[J]. International Journal of Open Information Technologies, 2024, 12(8): 48-55.

[6] Jagutis M, Russell S, Collier R W. Using multi-agent microservices (mams) for agent-based modelling[C]//International Workshop on Engineering Multi-Agent Systems. Cham: Springer Nature Switzerland, 2023: 85-92.

[7] Verkhova G V, Akimov S V, Prisyazhnyuk S P. Distributed Multi-agent Modeling of Complex Systems[C]//2021 XXIV International Conference on Soft Computing and Measurements (SCM). IEEE, 2021: 63-66.

[8] Krishnan R, Durairaj S. Reliability and performance of resource efficiency in dynamic optimization scheduling using multi-agent microservice cloud-fog on IoT applications[J]. Computing, 2024, 106(12): 3837-3878.

[9] Wang Y, Tang T, Fang Z, et al. Intelligent Task Scheduling for Microservices via A3C-Based Reinforcement Learning[J]. arXiv preprint arXiv:2505.00299, 2025.

[10] Kallel A, Rekik M, Khemakhem M. DRL4HFC: Deep Reinforcement Learning for Container-Based Scheduling in Hybrid Fog/Cloud System[C]//ICAART (2). 2024: 231-242.

[11] Wang H, Liu Y, Li W, et al. Multi-agent deep reinforcement learning-based fine-grained traffic scheduling in data center networks[J]. Future Internet, 2024, 16(4): 119.

[12] Ben Sada A, Khelloufi A, Naouri A, et al. Multi-agent deep reinforcement learning-based inference task scheduling and offloading for maximum inference accuracy under time and energy constraints[J]. Electronics, 2024, 13(13): 2580.

[13] Jeong Y, Maria E, Park S. Towards energy-efficient service scheduling in federated edge clouds[J]. Cluster Computing, 2023, 26(5): 2591-2603.