International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18 (2026) pp. 1-9

https://doi.org/10.70917/ijcisim-2026-0087
Article

Combining Edge Computing to Enhance Real-Time Control and Low-Latency Response of Line Robots

Jianjia Qi *

Heilongjiang Institute of Technology, Harbin 150050, Heilongjiang, China; hgcqjj@163.com

Abstract: In the context of real-time control and latency response scenarios for production line robots, this study focuses on optimizing the "cloud-edge-end" three-layer heterogeneous deployment based on edge computing. Multi-agent reinforcement learning methods are employed for task scheduling optimization, enabling distributed control to reduce resource waste and further improve load balancing. Dynamic event-triggered communication optimization reduces latency while minimizing network resource consumption. Through simulation testing, applying this method to optimize real-time control latency on the cloud platform achieved over 85% higher latency reduction compared to traditional cloud platform optimization tests. The system operates stably in dynamic complex scenarios, providing both theoretical and practical research and exploration into production line robot control optimization.

Keywords: edge computing; production line robots; real-time control; low latency response; multi-agent reinforcement learning; dynamic event triggering

1. Introduction

The Fourth Industrial Revolution continues to advance, and leveraging smart manufacturing as a means to achieve industrial upgrading and efficiency improvements has emerged as a new path forward for modern enterprises [1]. As key executors in smart manufacturing, robots ensure precision while leveraging their high efficiency and stability to replace human labor in industrial automation upgrades, becoming the primary means for enterprises to reduce costs and improve efficiency in production [2-4]. Faced with the trend toward higher flexibility and shorter production lines, the traditional industrial robot control architecture, which relies on cloud systems for centralized computing and control, suffers from drawbacks such as high control latency, low reliability, and difficulties in resource scheduling [5-6]. In the traditional model, the robot's perception information is transmitted over long-distance networks to the cloud for computation and control, and then responds based on the cloud's computational results. This process is further constrained by network bandwidth and cloud service resource limitations [7-9]. When network congestion or service delays occur, it easily leads to command delays or interruptions, resulting in unstable and unreliable control of the entire production line [10-11]. This issue is particularly pronounced in scenarios where production lines demand extremely high response times for commands, thereby hindering the deepening development of smart manufacturing [12].

Edge computing, as a new computing paradigm, focuses on pushing computational and storage resources to the network edge, enabling data to be processed locally in real-time. This significantly reduces end-to-end response latency, decreases reliance on the network, and enhances system real-time performance and reliability [13-16]. Literature [17] indicates that compared to traditional production system architectures supported by centralized data centers, the designed three-layer architecture industrial robot system based on edge computing can provide better real-time performance and network transmission performance, addressing the previously faced issues of high bandwidth costs and prolonged latency. Literature [18] indicates that in the modernization of agricultural practices, drones and ground-based unmanned vehicles serve as both network data collection devices and edge devices within the network. By leveraging edge computing methods, data from numerous devices can be processed in

real-time, effectively addressing traditional network congestion issues. Literature [19] introduces a robotic factory for manufacturing, where the introduction of edge computing methods successfully extends the computing resources, network bandwidth, and storage capacity of the cloud platform to the IoT edge, effectively improving product assembly efficiency and production efficiency. Based on this, edge computing nodes can be introduced into the robot control system on the production line to further reduce the latency of perception-computation-action, alleviate the pressure on cloud computing, and perform local intelligent task scheduling and control.

Additionally, with the synergistic development of mainstream technologies such as artificial intelligence, IoT, and 5G communications, edge computing possesses the capability to support collaboration among various heterogeneous devices, demonstrating good flexibility and scalability in tasks such as multi-task scheduling, dynamic resource scheduling, and communication overhead management [20-23]. Literature [24] clarifies that distributed resource scheduling (DRS) decision-making between terminal devices and edge devices is a fundamental issue facing edge computing. By studying DRS-enabled methods in specific application scenarios, it can better meet requirements such as autonomous computing, scalability, and low latency. Reference [25] designs an intelligent resource scheduling strategy for edge computing based on a hybrid computing framework, which provides good real-time performance, satisfaction, and energy efficiency in smart manufacturing environments, thereby fully meeting the real-time requirements of smart manufacturing. Reference [26] proposes a two-layer hybrid scheduling model based on graph neural networks and deep reinforcement learning to enhance the task scheduling process of robots under edge computing. The proposed model helps improve the efficiency of robot task management and plays an important role in scenarios with high response time requirements. Therefore, integrating edge computing with production line robot control systems to achieve a deeply integrated control architecture, enabling more responsive, efficient scheduling, and a safer system, has become a key focus and hot topic in the field of intelligent manufacturing.

Against this backdrop, this paper focuses on how to apply edge computing to real-time control and low-latency response for production line robots, aiming to break through the performance limitations of current cloud-based control architectures. This effort seeks to further enhance the intelligence and autonomy of production line robots, providing a theoretical and technological foundation for future intelligent manufacturing.

This paper primarily explores the research approach of "edge computing + production line robot control," following a methodology that progresses from hardware architecture to distributed task scheduling algorithms, communication mechanisms, and finally optimization algorithms. Research is conducted across four main aspects: the three-tier "cloud-edge-end" division of labor model, multi-agent reinforcement learning combined with distributed task scheduling models, dynamic event-triggered mechanisms, and annealing optimization algorithms. The control performance of the entire system platform is validated through these investigations.

2. Methods for Improving Real-Time Control and Low-Latency Response Capabilities of Robots

2.1. Edge Computing Node Deployment

Deploying edge computing nodes on production line robots within robot control cabinets is a critical issue affecting the real-time response capabilities of production line robots. This paper proposes a three-tier hierarchical deployment architecture—"local edge-remote edge-cloud edge"—to meet the task execution requirements of production line robots with different response latency needs. Local edge computing nodes are deployed within or near the industrial PC/UPC in the production line robot's control cabinet. The response latency of edge nodes is strictly limited to within 5 meters, handling safety tasks such as robot-to-robot collision detection and emergency braking, which require millisecond-level response times. For local edge computing nodes, it is recommended to use industrial computers or embedded platforms with machine-programmable control capabilities, running real-time operating systems such as VxWorks and Real-Time Linux to ensure the determinism of control tasks. Remote edge computing nodes cover production lines or groups of collaborative robots, handling tasks such as trajectory planning and camera-based visual recognition, which require medium-complexity control tasks with latency ranging from 10 to 50 milliseconds.

In terms of hardware architecture design, the near-end nodes adopt a heterogeneous computing architecture based on ARM Cortex-A72 multi-core processors and FPGA coprocessors. The FPGA specifically handles deterministic signal processing and control algorithms, while the CPU handles general computing tasks. The system is equipped with 8GB or more of DDR4 memory and a 128GB industrial-grade SSD to meet high-speed caching and local storage requirements. The fanless aluminum

alloy housing design complies with the IP65 dustproof and waterproof standard, with an operating temperature range of -40°C to 85°C and vibration resistance of 5G or higher. Communication interfaces include Gigabit Ethernet, high-speed serial ports RS-485/422, CAN bus, and industrial Ethernet interfaces supporting TSN (Time-Sensitive Networking).

In the network connection topology, deterministic network protocols such as PROFINET RT/IRT or EtherCAT are used between the near-end nodes and the robot controller, with a communication cycle controlled below 250μs to ensure real-time transmission of control signals. Critical network links are designed with redundancy and support fast convergence protocols such as RSTP/MSTP, with network failure switchover time controlled within 20ms to meet the high availability requirements of industrial control systems. The data acquisition and processing workflow adopts a unified data access framework based on OPC UA, supporting transparent access and protocol conversion for various industrial bus protocols such as Modbus, PROFIBUS, and DeviceNet. The minimum sampling period for real-time control data is set to 1 ms to ensure precise capture of rapidly changing physical quantities. Process monitoring data is sampled at intervals of 10 ms to 100 ms based on signal change characteristics to balance data acquisition accuracy and system overhead. Edge nodes use a stream processing architecture to establish a data diversion mechanism, allowing critical control data to enter the real-time processing pipeline, while non-critical monitoring data is pre-processed and compressed before being transmitted to the cloud to achieve efficient use of network bandwidth.

The software platform adopts a containerized microservice architecture, with core control functions deployed at the real-time operating system layer to ensure deterministic response, and data analysis and management functions deployed as Docker containers for flexible upgrades and functional expansion. The edge node operating system uses a customized Linux real-time version configured with the PREEMPT_RT patch to control scheduling latency within 50μs. Resource isolation is achieved through cgroups and namespace technology to ensure that critical control tasks are not interfered with by other processes. Node management uses edge computing orchestration systems such as K3s or KubeEdge to support automatic deployment, elastic scaling, and health monitoring of applications, establishing a robust edge computing infrastructure for real-time control and low-latency response capabilities of production line robots.

2.2. Distributed Task Scheduling Algorithm

Task scheduling issues faced by production line robots in edge computing environments exhibit high complexity, and traditional centralized scheduling methods cannot meet the collaborative requirements of distributed edge nodes. We model each edge computing node as an independent agent, using a Markov decision process to describe node interactions, and employ multi-agent reinforcement learning to achieve autonomous learning and collaborative optimization of task scheduling. Agents gradually learn optimal scheduling strategies through environmental interactions, with accumulated experience driving strategy updates, leading to continuous improvements in scheduling decision quality and enhanced system performance. The algorithm is based on the Q-learning framework, with value function updates following the following formula:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \tag{1}$$

In the equation, $Q(s,a)$ represents the value function of the intelligent agent performing action a in state s, reflecting the expected long-term return of the state-action pair. The learning rate parameter α controls the degree of influence of new experiences on the value function update, with values set in the range of 0.1 to 0.3 to balance learning speed and stability. The discount factor γ ranges from 0 to 1, balancing the importance of immediate rewards and future rewards. A larger γ value makes the agent more focused on long-term benefits. The immediate reward r is calculated based on metrics such as task completion time and resource utilization, and s' represents the next state after executing the action.

The agent's action space consists of four basic actions: allocation, migration, storage, and upload, as well as three actions for prioritizing allocation tasks and three actions for reserving memory space during allocation. The state space includes parameters such as resource occupancy at each edge node, load and network status of adjacent edge nodes, work queue status of each edge node, and historical scheduling completion time of the edge node [27]. Each agent's state vector consists of its own resource utilization, the number of tasks awaiting processing, the load status of each adjacent agent node, and the network status between adjacent agent nodes. The action space is the allocation action taken by the agent. The greedy function balances exploration and exploitation, with the initial coefficient for exploration set relatively high to ensure that more agents attempt new scheduling methods in the initial stage. As the number of iterations increases during the learning process, the greedy function coefficient is gradually

reduced to encourage agents to select the optimal scheduling strategy.

The reward function employs a multi-objective composite mechanism to guide agents in learning desired scheduling behaviors, primarily including task completion time rewards, resource utilization efficiency rewards, load balancing rewards, and energy efficiency rewards. The total reward function achieves multi-objective balance through weighted summation [28]. The algorithm incorporates an experience replay mechanism and target network technology to enhance convergence speed. The experience replay buffer stores historical samples for random sampling during training, breaking sample temporal dependencies to improve learning efficiency. The target network periodically replicates the main network parameters to calculate the target Q-values, reducing training instability. Priority experience replay determines sample importance based on time difference errors. Multi-agent coordination is achieved through information sharing and strategy negotiation. Each agent periodically exchanges state information and scheduling decisions, and consistency algorithms achieve global coordination to avoid scheduling conflicts.

2.3. Dynamic Event Trigger Mechanism Design

In edge computing environments, the design of dynamic event-triggered mechanisms plays a critical role in achieving efficient data transmission and real-time control. Traditional time-triggered mechanisms often result in unnecessary communication overhead and response delays, particularly in industrial automation scenarios where balancing real-time performance and resource utilization efficiency is of utmost importance. Therefore, we designed this mechanism based on the principle of determining the timing of data transmission based on changes in system state. Data is only transmitted to edge computing nodes or the cloud when the system state undergoes significant changes or meets specific trigger conditions. This approach reduces redundant data transmission while optimizing network performance under conditions that ensure control accuracy. During the design process, Lyapunov stability theory was applied to verify system stability. The system dynamics can be represented as:

$$x = f(x,u) + g(x,u)w \tag{2}$$

In the equation, x represents the system state, u is the control input, w represents the disturbance, and f and g are the system state transfer function and disturbance influence function, respectively.

By continuously monitoring the current state and comparing it with system analysis, the system adapts to changes in trigger conditions across different control scenarios. Specifically, trigger conditions can be determined based on changes in system state values and the rate of change in control inputs. When these exceed predefined thresholds, data transmission is triggered; otherwise, the current state is maintained to avoid unnecessary communication overhead. This constitutes an adaptive triggering method for network load management, thereby achieving reduced network load and improved system operational efficiency.

Compared to traditional time-based triggering mechanisms, the multi-level triggering judgment mechanism proposed in this paper achieves the lowest network latency while ensuring the stability and performance of the system control process after triggering. The dynamic event triggering algorithm, based on Lyapunov stability theory, ensures system stability and reliability, providing design and development insights for edge computing applications in industrial fields, with significant theoretical and practical value. It can meet the requirements of certain special scenarios where resource demands are low but control requirements demand high real-time performance. By continuously refining the parameters of trigger thresholds and optimizing the algorithms for trigger criteria, the system design achieves better adaptability. This enables it to maintain good performance even under complex conditions such as changes in network operating conditions or variations in workload intensity, providing important guidance for designing robust edge computing structures.

2.4. Resource Allocation Optimization Plan

The problem of multi-task resource optimization for production line robots in edge computing environments is a complex combinatorial optimization problem involving resource constraints and multi-objective optimization. The objective is to maximize the average task completion latency while minimizing system power consumption under limited computational, storage, and network bandwidth constraints. Greedy and heuristic algorithms often get stuck in local optima; while genetic algorithms have global optimization capabilities, they tend to converge slowly and are difficult to tune. To address these issues, this paper proposes a multi-objective resource optimization strategy for production line robots using a hybrid simulated annealing method. This algorithm formulates the task resource allocation problem for multi-task edge computing nodes as a constrained optimization problem, incorporating factors such as average task completion time, node power consumption, task load balancing, and service

quality into the objective function.

Assume that the edge system consists of N computing nodes and M tasks to be processed. The computing capacity of the i th node is C_i, and its storage capacity is S_i. The computing requirement of the j th task is R_j, and its storage requirement is D_j. The resource allocation matrix $X = \{x_{ij}\}$ represents the task allocation relationship, where $x_{ij} = 1$ indicates that task j is assigned to node i for execution, and 0 otherwise. The objective function is defined as:

$$F(X) = \alpha \cdot T(X) + \beta \cdot E(X) + \gamma \cdot L(X) \tag{3}$$

In the equation, $T(X)$ denotes the total system delay, $E(X)$ denotes the total energy consumption, $L(X)$ denotes the load imbalance degree, and α, β, γ are weighting coefficients.

The simulated annealing algorithm draws inspiration from the physical phenomenon of solid annealing, simulating the changes in atomic energy states during metal cooling to search for the global optimal solution to an optimization problem. The algorithm starts from an initial solution and generates neighboring solutions at each iteration, deciding whether to accept them based on the Metropolis criterion, with an acceptance probability of:

$$P = \exp(-\Delta E / T) \tag{4}$$

In the equation, ΔE is the change in the objective function value, and T is the current temperature parameter.

The temperature parameter gradually decreases during the iteration process. The adaptive temperature adjustment strategy designed in this paper is expressed by an exponential decay function:

$$T_k = T_0 \cdot \alpha^k \tag{5}$$

In the equation, T_0 is the initial temperature, α is the cooling coefficient, and k is the iteration count.

A temperature rebound mechanism is introduced to activate the search process. The neighborhood solution generation strategy uses a combination of multiple transformation operations. When dealing with multi-objective optimization problems, the weighted sum method is used to merge multiple objectives into a single objective function.

The algorithm implementation employs an elite retention strategy to prevent high-quality solutions from being lost during random search, and a diversity maintenance mechanism to ensure the uniform distribution of the solution set. The convergence criterion is when the objective function value shows no significant improvement over several consecutive generations or reaches the maximum iteration count. After the algorithm terminates, it outputs the optimal resource allocation scheme.

3. Results and Discussion

3.1. Analysis of Experimental Results

To verify the optimization capabilities of edge computing technology for real-time control of production line robots, a complete test environment for production line robots was established. The system platform primarily consists of three near-end edge nodes, two far-end edge nodes, and a cloud server. The production line robots utilize six-axis industrial robots equipped with various sensors to collect real-time data. The near-end edge nodes are equipped with ARM Cortex-A72 (1.5GHz) x4 processors + Xilinx Artix-7 field-programmable gate arrays, 8GB DDR4 industrial-grade solid-state drives, and 128GB solid-state drives. The remote edge nodes use Intel Core i7-10700 (2.9GHz) processors, 16GB DDR4 industrial-grade solid-state drives, and 512GB industrial-grade solid-state drives. The cloud servers use dual Intel Xeon Gold 6248R (3.0GHz) processors, 128GB industrial-grade solid-state drives, and 2TB fiber-optic high-speed solid-state drives. Interconnection between remote and local nodes uses time-sensitive networking. The communication protocol between local nodes and robot controllers is EtherCAT, with a cycle time of 250 microseconds. Remote and local edge nodes are connected via gigabit industrial Ethernet, using service-level policies to ensure priority transmission of industrial control data. Remote edge nodes communicate with the cloud via a 10-gigabit fiber-optic network for high-speed data transmission.

We compared the response capabilities of the traditional centralized cloud computing model and the edge cloud computing model under different operating conditions. Using three practical operating conditions as test cases: stable production line operation, overload operation, and sudden overload. The test results are shown in Table 1. The edge-based cloud computing model achieved system optimization

in all three operating conditions, with response times improved by over 85%, and even in sudden overload situations, response times were kept under 50 milliseconds, fully meeting the real-time control requirements of the production line's robots.

Table 1. Average response time under different operating conditions (ms).

Architecture type	Normal operation	High load operation	Sudden load
Cloud computing	178.5	246.3	312.7
This article	22.4	36.7	42.3
Optimization ratio (%)	87.5%	85.1%	86.5%

Performance evaluation of distributed task scheduling algorithms shows that scheduling strategies based on multi-agent reinforcement learning demonstrate significant advantages over traditional methods. The comparative data in Table 2 shows that this algorithm achieves optimal performance in four core metrics: average task delay, resource utilization, load balancing, and system energy consumption. In particular, load balancing is improved by more than 30% compared to traditional scheduling algorithms, which is important for maintaining stable system operation under high load conditions.

Table 2. Performance comparison of different task scheduling algorithms.

Algorithms	Average task delay (ms)	Resource use rate (%)	Load balancing degree	Energy Consumption (W)
Polling scheduling	48.6	65.3	0.63	187.2
Greedy scheduling	41.2	72.4	0.58	172.5
Genetic algorithm	35.7	78.1	0.72	165.3
This article	28.3	83.5	0.85	153.8

Additionally, this paper simulates the operation of the resource allocation scheme in edge computing scenarios of different scales, simulating the operational effects in small-scale scenarios (5 nodes and 20 tasks), medium-scale scenarios (15 nodes and 60 tasks), and large-scale scenarios (30 nodes and 120 tasks). The algorithms compared include the greedy algorithm, genetic algorithm, and particle swarm optimization algorithm. Through comparative experiments, the advantages of the simulated annealing resource allocation scheme proposed in this paper can be verified. The algorithm can significantly reduce task execution latency without increasing energy consumption. In large-scale scenarios with stringent resource constraints, the algorithm's characteristics become even more pronounced. To provide higher-quality edge computing resource allocation services that improve the response latency of production line robots and enhance the accuracy of their decision-making, thereby reducing the energy consumption of edge computing systems, which is beneficial for improving the overall operational efficiency of the production line. Therefore, the proposed simulated annealing algorithm effectively avoids getting stuck in local optima during the search process and achieves a reasonable balance between exploration and exploitation by quickly finding solutions close to the global optimum within a large solution space.

Our testing of the aforementioned dynamic event-triggered mechanism also yielded positive results. We tested the network bandwidth consumption under various communication mechanisms during an 8-hour operational period of the robot. As shown in Figure 1, the trends in network bandwidth consumption under the three communication mechanisms—fixed-cycle communication, static event triggering, and dynamic event triggering—are clearly discernible over the 8-hour period. As shown in Figure 1, throughout the entire testing period, the network bandwidth consumption under the dynamic event-triggered mechanism remained near the minimum value and was significantly lower than that under fixed-cycle communication and static event-triggered mechanisms during the most active periods. This is attributed to the flexible and adaptive nature of the dynamic event-triggered mechanism, which can reduce the frequency of information transmission even when there is a high demand for information, and vice versa. The advantage in system resource utilization did not result in a decline in system control performance. During the entire testing process, the average error in path tracking for robots using the dynamic event-triggered mechanism was 0.21 mm, while the average error for path tracking using fixed-cycle communication was 0.19 mm, with a difference of only 0.02 mm. However, the network resource consumption under the dynamic event-triggered mechanism was halved, demonstrating the

mechanism's ability to efficiently utilize the robot's system resources.

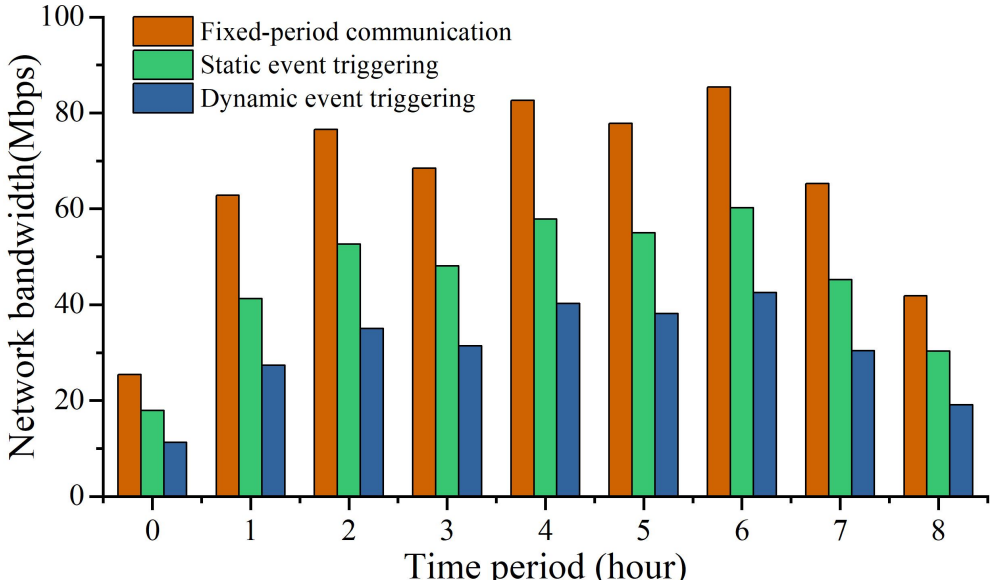

Figure 1. Network bandwidth occupation of different communication mechanisms.

Through analysis of the experimental results, it can be seen that the research scheme proposed in this paper can effectively reduce latency in real-time control scenarios for production line robots while ensuring flexibility, improving the real-time response speed of robots, and further improving the control efficiency of existing production line robots. Additionally, the designed distributed task scheduling algorithm leverages network resource advantages while implementing event-correlated scheduling tasks, resulting in high network utilization. The dynamic event-triggered computing resource allocation strategy reduces computational resource overhead in the system while maintaining control precision. The event-driven computing and communication task scheduling strategy ensures real-time control capabilities of robots while enabling flexible and controllable distributed real-time task scheduling.

3.2. Discussion of Experimental Results

From the analysis of experimental data, the three-tier progressive edge computing technology architecture has shown significant improvements in enhancing the real-time control capabilities of production line robots and reducing latency response times. However, it has also revealed certain shortcomings. The three-tier progressive edge computing architecture can effectively meet the usage requirements of different latency-sensitive tasks in various application environments. The design of near-end edge nodes can achieve millisecond-level response times for safety-critical tasks, fully satisfying the real-time and deterministic response requirements of industrial control systems. However, the hierarchical design of the architecture, while offering greater flexibility, also increases decision-making complexity. Especially regarding task reallocation, when an edge node fails or becomes overloaded, the decision delay in task reallocation can impact overall system performance. Experimental results show that compared to traditional cloud computing architectures, this architecture achieves over an 85% improvement in response time, primarily due to reduced data processing distances and decreased network transmission latency within the architecture. However, there is still some variability in the experiments, and as seen from the experimental data, the response time can still reach below 50ms under sudden load spikes, indicating that there is still room for improvement in the architecture's computational resource allocation and task scheduling strategies.

In terms of task scheduling, the distributed scheduling algorithm based on multi-agent reinforcement learning has effectively addressed scheduling issues. Compared to traditional scheduling algorithms, it achieves significant improvements in average task latency, resource utilization, and load balancing, with load balancing improved by over 30%. Additionally, through continuous learning and interaction among multi-agents, the scheduling decision-making strategy demonstrates strong adaptability during dynamic workload adjustments. However, further improvements are needed in the algorithmic performance of reinforcement learning algorithms, specifically in terms of learning rate and robustness. The learning efficiency of this algorithm fluctuates significantly during the early iteration stages but converges after training, enabling it to better serve practical applications. However, this also increases the complexity of

system deployment and maintenance. Additionally, while the collaborative action of multiple agents effectively enhances the distributed characteristics of system scheduling, it also introduces corresponding communication overhead. Unstable factors in industrial environments may impact the actual system's operation. Furthermore, the application of the dynamic event-triggered mechanism reduces network bandwidth overhead without compromising control accuracy. Compared to the previous method of tracking state changes, the robot's path tracking error increases by only 0.02 mm. With network overhead reduced by nearly 50%, the system performance and network communication overhead of the dynamic event-triggered mechanism are relatively balanced. Furthermore, the determination of thresholds during the design process of the dynamic event-triggered mechanism significantly impacts the system. If the system threshold is too sensitive, it may overlook important state changes. However, when the threshold is set too low, it may result in unnecessary communication traffic, leading to insufficient utilization of network resources. Therefore, how to adaptively adjust based on specific application scenarios remains an important direction for future research.

Future edge computing technology will also support the development of production line robot control systems. For example, the low latency and high reliability of 5G will support the optimization of edge computing technology effects. Research should focus on management solutions for intelligent edge computing nodes. For instance, artificial intelligence can be used to achieve resource prediction, dynamic configuration and self-healing, and automatic performance optimization to enhance the self-management capabilities of edge computing systems. Given the complexity of industrial environments, the development of edge computing technology should prioritize exploring flexible and scalable edge computing architectures that support seamless integration and collaborative processing of heterogeneous edge devices, thereby promoting the widespread adoption of edge computing technology. In terms of algorithm optimization, future research can integrate federated learning technology with multi-agent reinforcement learning algorithms. Distributed training can reduce data flow between nodes and protect industrial data privacy. By using digital twin technology to model the operational status of production line robots, more accurate predictions and simulation calculations can be provided for task assignment and resource optimization, thereby enhancing the effectiveness of algorithm optimization. In the field of security, edge computing nodes will also become an important component of distributed architectures in the future development of edge computing. The risks of cyberattacks and data breaches facing industrial control systems will continue to rise. Therefore, establishing a reasonable defense system and data encryption strategies to ensure the operational security of industrial control systems is of great significance.

4. Conclusion

This paper proposes a “cloud-edge-end” control architecture for production line robots, featuring high real-time control capabilities and low system latency. It introduces an intelligent agent reinforcement learning scheduling algorithm and dynamic event-triggered methods to ensure timely responses and reasonable scheduling. Simulation and experimental results demonstrate that edge computing achieves an average response time that is over 85% faster than traditional cloud computing, with the highest latency maintained below 50ms. Compared to traditional methods, the dynamic event-triggered approach reduces network resource usage by approximately 50%. The multi-agent scheduling method improves load balancing by over 30% under high-load conditions and reduces average latency by 28.5%. This method addresses low-latency and high-reliability control in dynamic and complex environments for production line robots, providing a feasible technical and practical demonstration for smart manufacturing. It also lays the foundation for future improvements and updates in edge intelligence.

References

1. Chen, Y. (2017). Integrated and intelligent manufacturing: Perspectives and enablers. *Engineering*, 3(5), 588-595.
2. Day, C. P. (2018). Robotics in industry-their role in intelligent manufacturing. *Engineering*, 4(4), 440-445.
3. Liu, X. J. (2019). Research toward IoT and robotics in intelligent manufacturing: a survey. Int. J. Mater. Mech. *Manuf*, 7(3), 128-132.
4. Qian, J., Zi, B., Wang, D., Ma, Y., & Zhang, D. (2017). The design and development of an omni-directional mobile robot oriented to an intelligent manufacturing system. *Sensors*, 17(9), 2073.
5. Rahimi, R., Shao, C., Veeraraghavan, M., Fumagalli, A., Nicho, J., Meyer, J., ... & Evans, P. (2017, April). An industrial robotics application with cloud computing and high-speed networking. *In 2017 First IEEE International Conference on Robotic Computing (IRC)* (pp. 44-51). IEEE.
6. Sethuraman, S., Vikram, S., Surya, R. D., & Singh, B. (2024). Applications of Cloud Computing in Industrial Robotics. *In Shaping the Future of Automation With Cloud-Enhanced Robotics* (pp. 181-204). IGI Global.

7. Bogue, R. (2017). Cloud robotics: a review of technologies, developments and applications. *Industrial Robot: An International Journal*, 44(1), 1-5.
8. Liu, J., Xu, W., Zhang, J., Zhou, Z., & Pham, D. T. (2016, June). Industrial cloud robotics towards sustainable manufacturing. *In International Manufacturing Science and Engineering Conference* (Vol. 49903, p. V002T04A017). American Society of Mechanical Engineers.
9. Zhang, N. (2021). A cloud-based platform for big data-driven cps modeling of robots. *IEEE Access*, 9, 34667-34680.
10. Wang, X. V., Wang, L., Mohammed, A., & Givehchi, M. (2017). Ubiquitous manufacturing system based on Cloud: A robotics application. *Robotics and Computer-Integrated Manufacturing*, 45, 116-125.
11. Wang, S., Zhang, C., Liu, C., Li, D., & Tang, H. (2017). Cloud-assisted interaction and negotiation of industrial robots for the smart factory. *Computers & Electrical Engineering*, 63, 66-78.
12. Gupta, A., Dixit, A. K., Kumar, K. S., Lavanya, C., Chakravarthi, M. K., & Gangodkar, D. (2022, December). Analyzing Robotics and Computer Integrated Manufacturing of Key Areas Using Cloud Computing. *In 2022 5th International Conference on Contemporary Computing and Informatics (IC3I)* (pp. 194-199). IEEE.
13. Elbamby, M. S., Perfecto, C., Liu, C. F., Park, J., Samarakoon, S., Chen, X., & Bennis, M. (2019). Wireless edge computing with latency and reliability guarantees. *Proceedings of the IEEE*, 107(8), 1717-1737.
14. Chen, C. L., Brinton, C. G., & Aggarwal, V. (2021). Latency minimization for mobile edge computing networks. *IEEE Transactions on Mobile Computing*, 22(4), 2233-2247.
15. Intharawijitr, K., Iida, K., & Koga, H. (2017). Simulation study of low latency network architecture using mobile edge computing. *IEICE TRANSACTIONS on Information and Systems*, 100(5), 963-972.
16. Shu, C., Zhao, Z., Han, Y., Min, G., & Duan, H. (2019). Multi-user offloading for edge computing networks: A dependency-aware and latency-optimal approach. *IEEE Internet of Things Journal*, 7(3), 1678-1689.
17. Chen, Y., Feng, Q., & Shi, W. (2018). An industrial robot system based on edge computing: An early experience. In USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18).
18. Ayranci, A. A., & Erkmen, B. (2024, May). Edge Computing and Robotic Applications in Modern Agriculture. *In 2024 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)* (pp. 1-6). IEEE.
19. Hu, L., Miao, Y., Wu, G., Hassan, M. M., & Humar, I. (2019). iRobot-Factory: An intelligent robot factory based on cognitive manufacturing and edge computing. *Future Generation Computer Systems*, 90, 569-577.
20. Tang, Z., Jia, W., Zhou, X., Yang, W., & You, Y. (2020). Representation and reinforcement learning for task scheduling in edge computing. *IEEE Transactions on Big Data*, 8(3), 795-808.
21. Avan, A., Azim, A., & Mahmoud, Q. H. (2023). A state-of-the-art review of task scheduling for edge computing: A delay-sensitive application perspective. *Electronics*, 12(12), 2599.
22. Yang, C., Liao, F., Lan, S., Wang, L., Shen, W., & Huang, G. Q. (2023). Flexible resource scheduling for software-defined cloud manufacturing with edge computing. *Engineering*, 22, 60-70.
23. Lin, C. C., Deng, D. J., Chih, Y. L., & Chiu, H. T. (2019). Smart manufacturing scheduling with edge computing using multiclass deep Q network. *IEEE Transactions on Industrial Informatics*, 15(7), 4276-4284.
24. Sahni, Y., Cao, J., Yang, L., & Wang, S. (2022). Distributed resource scheduling in edge computing: Problems, solutions, and opportunities. *Computer Networks*, 219, 109430.
25. Li, X., Wan, J., Dai, H. N., Imran, M., Xia, M., & Celesti, A. (2019). A hybrid computing solution and resource scheduling strategy for edge computing in smart manufacturing. *IEEE Transactions on Industrial Informatics*, 15(7), 4225-4234.
26. Tang, H., Jiao, R., Xue, F., Cao, Y., Yang, Y., & Zhang, S. (2024). Task Scheduling Strategy of Logistics Cloud Robot Based on Edge Computing. *Wireless Personal Communications*, 137(4), 2339-2358.
27. Zhang, F., Deng, R., Zhao, X., & Wang, M. M. (2021). Load balancing for distributed intelligent edge computing: A state-based game approach. *IEEE Transactions on Cognitive Communications and Networking*, 7(4), 1066-1077.
28. Song, M., Lee, Y., & Kim, K. (2021). Reward-oriented task offloading under limited edge server power for multiaccess edge computing. *IEEE Internet of Things Journal*, 8(17), 13425-13438.