Data Engineering and Applications

Research Paper

Coordinated Optimization of Multi-Robot Automated Assembly Lines Based on Deep Q-Learning

Julia Duda[1, *] and Tomasz Wiśniewski[2]

[1] Faculty of Mechanical Engineering, University of Applied Sciences in Nysa, Nysa 48-300, Poland
[2] Faculty of Mechanical Engineering, University of Applied Sciences in Konin, Konin 62-500, Poland
*Corresponding author: julia.d@wsz-nysa.pl

Abstract. Current large-scale multi-robot automated assembly line systems face common issues such as real-time coordination and intelligent resource scheduling. Here, a dynamic scalable manufacturing framework based on deep Q-learning is proposed. Deep reinforcement learning, combined with real-time perception, distributed robot control, and fast peer-to-peer communication, is a component of the proposed method. Create an event-driven assembly line simulator and simulate multiple robot teams, each with different workloads. The above experiments show that deep Q-learning scheduling outperforms traditional optimization and heuristic methods. Under different team sizes and task allocations, its throughput increased by up to 18.4%, and the median idle ratio of robots decreased by 32%. In robustness tests under robot failure and task interruption scenarios, the system's success rate exceeded 85%, with only a slight decrease in performance. Moreover, sensitivity analysis indicates that it is relatively stable when changing the reward strategy and learning rate; ablation experiments show that the multi-head attention architecture is the most sample-efficient in industrial scheduling. According to the above research results, learning-driven adaptive control strategies can meet the demands of large-scale and flexible production. This will improve the production efficiency and operational stability of industrial automation in the real world.

Keywords: *Deep Reinforcement Learning, Multi-Robot Scheduling, Real-Time Control, Resource Allocation, Robust Optimization*

Received on 19 October 2025, Accepted on 09 April 2026, Published on 17 April 2026

Introduction

With the continuous changes in global manufacturing, more and more people are beginning to use intelligent, scalable multi-robot assembly systems [1]. Due to changes in people's demands, product lifecycles have also shortened. Therefore, production is more flexible and adaptable [2]. Multi-robot systems can improve production efficiency and allocate resources to different locations [3]. The aforementioned methods can also improve the quality and stability of the production line [4]. However, building an efficient and secure real-time coordination platform is key to ensuring the stable operation of the entire system [5]. In uncertain environments, intelligent scheduling operations and heterogeneous task allocation are the most important issues [6]. Due to the inability of traditional architectures and static optimizers to meet current demands, there is a need to develop next-generation automation methods based on learning [7]. With the development of Industry 4.0, high-tech assembly line intelligent robots, including distributed autonomy and collaborative operations, are receiving increasing attention [8].

In the current large-scale dynamic manufacturing environment, most scheduling and allocation methods have not yet been fully realized, despite the development of robots and artificial intelligence [9]. Heuristic or rule-based optimizers typically assume the environment is deterministic or has fixed resource constraints, and therefore cannot respond promptly to changes in tasks or the system environment [10]. Traditional

reinforcement learning can handle environmental changes, but large state and action spaces encounter the curse of dimensionality problem [11]. Compared to the aforementioned traditional methods, multi-robot systems also have some limitations. For example, it can lead to difficulties in credit allocation and coordination [12]. Table Q-learning and lookup-based strategies are generally not widely applicable, and as the number of tasks or robots increases, the computational cost also rises [13]. Deep learning has been used in reinforcement learning. It has created DQL and other algorithms that can approach optimal strategies in the context of large-scale perception and process state spaces [14]. Nevertheless, low-latency inference and reliable integration with industrial robot controllers still struggle to function effectively in practical applications [15].

This paper proposes a joint optimization method for multi-robot automated assembly lines based on deep Q-learning. The system consists of three components: deep reinforcement learning, real-time perception and robot control, and peer-to-peer communication. These components form a large-scale intelligent coordination framework. The above are the main results: an improved DQL model suitable for complex assembly environments, an integrated action and decision system, and extensive experiments conducted in various scenarios. This study shows that learning-based coordination can improve the reliability and efficiency of industrial robot automation.

Algorithmic Foundations

Fundamentals of Reinforcement Learning in Robotics

Robots can learn to explore the world interactively through reinforcement learning (RL). In its classical form, a reinforcement learning agent repeatedly observes the current state of the environment, selects an action, receives a corresponding reward, and then transitions to the next state. Markov Decision Processes (MDP) are commonly used to model the behavior of systems. The goal of the learning process is to maximize the expected reward over time by modifying the policy function. Q-learning is an early reinforcement learning method based on value, which directly learns to estimate the expected return of state-action pairs. There is no system model or offline supervised data to support it [16].

Q-learning and other reinforcement learning methods can help robotic systems solve adaptation problems in unstructured and dynamic environments. The aforementioned algorithms are based on the temporal difference (TD) learning rule, and through iteration and updates based on observed rewards, they can independently improve the robot's performance in navigation, manipulation, assembly, and other areas. Unlike supervised learning, reinforcement learning is a self-improving feedback loop. The policy is not adjusted based on labels; it is refined through trial and error by the agent during exploration [17]. Since reinforcement learning directly addresses the problem of optimizing long-term rewards, it is more suitable for goal-oriented sequential decision-making compared to unsupervised learning [18].

Traditional reinforcement learning is not suitable for handling large-scale robotic problems, and as the complexity of the environment or the number of agents increases, the "curse of dimensionality" problem becomes more severe. The state-action space grows exponentially, and tabular or simple function methods are no longer applicable. There has already been research using state space abstraction and function approximators for policy and value estimation to address the aforementioned issues. The exploration-exploitation trade-off in physical robots is related to reinforcement learning. Excessive exploration may cause losses and economic damage [19]. On the other hand, autonomous assembly lines and multi-robot systems still rely on reinforcement learning [20].

Deep Q-Learning Model Formulation

Deep Q-Networks (DQN) are a combination of Q-learning and deep learning, and have already applied reinforcement learning to high-dimensional problems, such as those in automated assembly environments [21]. Deep neural networks can be used as powerful function approximators to learn the Q-value function using various sensor data (such as vision, kinematics, and process states), without the need for feature engineering to determine actions. States, actions, and rewards require low-capacity encoding to meet the demands of multi-robot assembly lines. Motion commands, task assignments, and explicit coordination requests constitute the

action space [22]. Task queues, robot positions, resource usage, and system bottlenecks are all components of the state setting.

Design of the reward function: It aims to improve efficiency and throughput and penalizes collisions, idle time, or resource blocking through a penalty mechanism. A good incentive will motivate the agent to achieve the system's goals and strive to achieve them over a longer period. Using mini-batch stochastic gradient descent to train the deep Q-network improves data efficiency, breaks temporal correlation, and samples from the experience replay buffer. The target network can help stabilize Q-learning updates during the training process [23].

The DQN-based framework is suitable for multi-robot environments. Since multiple agents work together, each learner's response to environmental changes is no longer consistent, making convergence and reproduction difficult. Methods to address this issue include prioritized experience replay, shared or parameterized policies, and a combination of centralized training and decentralized execution [24]. By using distributed architectures and curriculum learning schemes, the convergence and robustness of heterogeneous robot teams can be improved. The aforementioned modifications make deep Q-learning feasible and effective when dealing with large-scale and complex assemblies.

Integration with Multi-Robot Control Systems

In order for the robot's new algorithm to be truly usable, it must be integrated with all the software and hardware of the automated assembly line. In practice, the deep Q-learning module is used as the decision center and is connected to robot middleware, such as industrial automation frameworks and the Robot Operating System (ROS) [25]. The reasoning cycle of the learning agent is used to acquire and process data streams from sensors, such as position, force, and visual signals, to create the observation space.

Based on the inference of executing commands, the DQL agent sends them to the robot controller through a fortified communication protocol, and then safely and quickly executes the actions. The module is used to perform low-level command translation and other synchronization operations in multi-robot systems to ensure system stability in the event of scaling or interference. Minimize the perception-decision-action process; otherwise, it will not be suitable for industrial applications.

Industrial applications also require a reliable failover and fallback mechanism to prevent the learning layer from encountering untrustworthy situations or deteriorating sensor conditions. When the system encounters issues or problems arise in the early stages, the control stack may adopt a rule-based fallback mechanism or an old-fashioned scheduler. Finally, due to the modular structure of the system, adding new algorithms and optimizations to the DQL core is relatively simple and does not affect other parts of the assembly system. By using the aforementioned practical integration strategies, deep reinforcement learning is becoming an important component of high-end manufacturing automation.

Real-Time System Architecture

Latency, Throughput, and Control Window Design

The reliability of real-time control is the foundation for the high-speed operation and stable production of the latest automated assembly lines. Limit the system's throughput, control window, and latency through software organization and hardware capacity. Delay (L) is the time interval between the occurrence of an external event and the issuance of the corresponding actuator command. It can be formally written as

$$L = t_{actuation} - t_{sensing} \quad \text{Eq.(1)}$$

In high-mix and high-throughput environments, maintaining synchronization between different production stages requires reducing the overall system latency to prevent signal congestion. As L increases, coordination between robots will decrease, which may lead to unsynchronized operations or bottlenecks in resource sharing.

Throughput (T) is the number of assembly tasks completed per unit of time, indicating the effectiveness of the system's operation:

$$T = \frac{\sum_{k=1}^{M} \mathbb{I}_{\{task_k\ completed\}}}{\Delta t} \quad \text{Eq.(2)}$$

where M is the total number of tasks within time interval Δt, and $\mathbb{I}_{\{.\}}$indicates successful task completion. High throughput is not just about high operational speed; it also involves the collaboration of sensing, reasoning, and execution to reduce deadlocks and idle time.

The deterministic control window (W) is the strict round-trip time required for a complete perception-reasoning-execution cycle and must be designed to be less than the physical cycle limit of the process.

$$W = max\{W_{sense}, W_{infer}, W_{act}\} \quad \text{Eq.(3)}$$

where W_{sense} measures sensor acquisition time, W_{infer} the duration for deep policy computation, and W_{act} the actuation dispatch including communication overhead. Notably, W must often remain under 50 ms to support sub-cycle robot reactions and guarantee real-time safety interlocks.

Each robot's agent issues its control commands as a result of a deep policy function, integrating both instantaneous local sensory inputs ($\mathbf{o}_t$) and summarized system context ($\hat{\mathbf{c}}_t$):

$$\boldsymbol{u}_t = \pi_\theta(\boldsymbol{o}_t, \hat{\boldsymbol{c}}_t) \quad \text{Eq.(4)}$$

Here, π_θ is the trained policy parametrized by θ, with $\hat{\mathbf{c}}_t$ encapsulating workload, upstream progress, and dynamic bottlenecks. Such integration enables predictive adjustments, mitigating the cascading effects of variable task times and network latency.

Figure 1 shows the data flow and control logic pipeline of the entire assembly system, including all key subsystems and their connections. The above structure illustrates how digital communication, sensing, distributed and centralized reasoning, and controller actions form a closed-loop system in real-time.

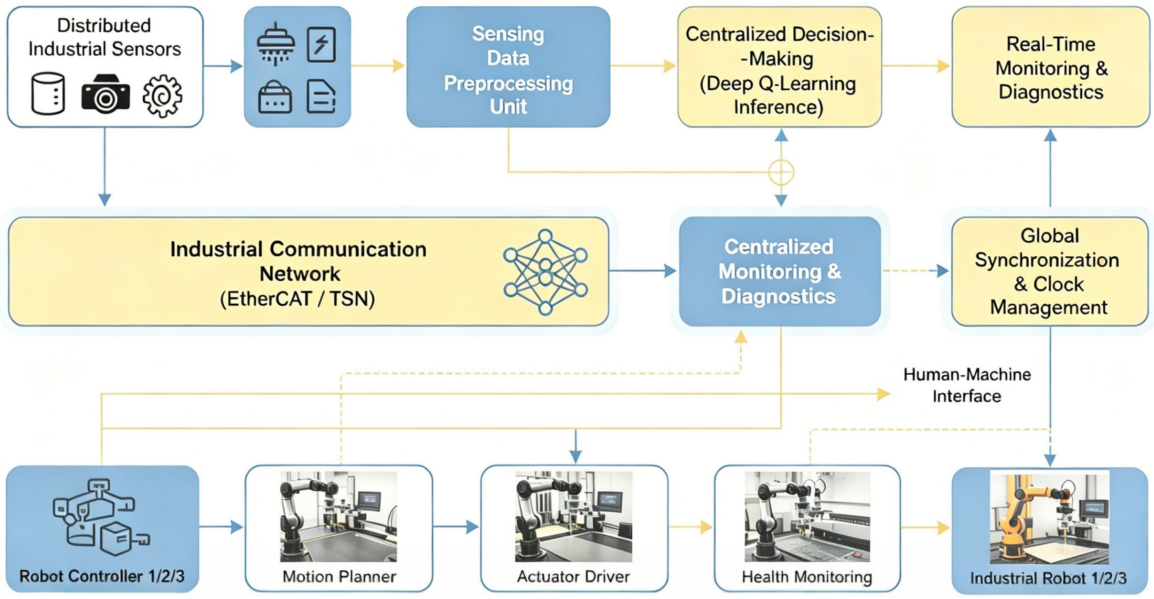

Figure 1. Overview of the multi-robot automated assembly line control system

By formally constraining and adjusting each control window subcomponent, the entire system has been guarantyd to be bounded in terms of jitter and capable of meeting deterministic real-time contracts. Therefore, it can ensure throughput and quality under high-demand conditions.

Communication and Synchronization among Robots

In order to meet the stringent requirements of time-criticality and determinism in industrial automation, a robust and scalable reliable communication backend for multi-robot systems is needed. Each robot i encodes and broadcasts a real-time status vector $\mathbf{x}_i(t)$, comprising its position, task state, communication health, and resource claims. The state collective across all N robots is:

$$\boldsymbol{X}(t) = [\boldsymbol{x}_1(t), \boldsymbol{x}_2(t), \ldots, \boldsymbol{x}_N(t)] \quad \text{Eq.(5)}$$

This collective, updated at sub-second frequencies (e.g., every $5 - 20$ ms), supports global routing decisions, load balancing, and collision avoidance.

As shown below, the total message propagation delay of any logical communication path includes outbound queuing, deterministic network transmission, and inbound parsing delay:

$$D_{comm} = D_{tx} + D_{net} + D_{parse} \quad \text{Eq.(6)}$$

where D_{tx} is the local device transmission latency (affected by packet scheduling and interface bandwidth), D_{net} is the physical network delay (cable length, switch buffering, and protocol stack), and D_{parse} encapsulates deserialization and user-level message extraction.

To ensure all robots reach a coordinated state with guaranteed order, logical clocks ($C_i(t)$) are periodically synchronized using distributed protocols-such as IEEE 1588 Precision Time Protocol-so that, at any instant:

$$\forall i, j: \left|C_i(t) - C_j(t)\right| < \epsilon \quad \text{Eq.(7)}$$

where ϵ is typically held below a tenth of the control window (< 5 ms) to prevent synchronization drift impacting coordinated task execution or handovers.

Safety and Liveness are further enhanced by bounding the per-cycle probability of synchronization faults:

$$P_{sync\text{-}fail} \leq \delta \quad \text{Eq.(8)}$$

Set δ to a relatively high industrial standard (for example, a failure rate of less than one in a million cycles). Message redundancy, hardware watchdogs, and backup supervisory arbitration will provide fault tolerance when communication is lost or the state is outdated.

There are many types of networks. EtherCAT and Time-Sensitive Networking (TSN) monitor the signal network and perform global broadcasting; direct point-to-point channels or multicast coverage are used for resource negotiation and low-latency intent. By using the aforementioned protocols, the platform can ensure reliable, lossless communication and bounded latency between clusters of hundreds to thousands of robots. These protocols enable the platform to provide reliable distributed real-time decision-making for advanced automated assembly.

Scalability and Resource Scheduling

The new automated assembly line must have easily expandable real-time control capabilities. As the number of robots (N) increases, more products and workloads need to be handled. However, good resource scheduling can still ensure the system's throughput and responsiveness. By strategically managing communication, computing, and physical resources, idle time can be reduced, and bottlenecks in the production line can be eliminated.

Let $\mathcal{R} = \{r_1, r_2, \ldots, r_N\}$ denote the set of collaborative robots and $\mathcal{T}_t$ the set of tasks at time t. The assignment variable $x_{i,j}(t)$ is defined as a binary indicator:

$$x_{i,j}(t) \in \{0,1\} \quad \text{Eq.(9)}$$

with $x_{i,j}(t) = 1$ if robot r_i is allocated to task τ_j at time t, and 0 otherwise.

The control strategy aims to reduce the stress on all robots and make their distribution more even:

$$\bar{L}(t) = \frac{1}{N}\sum_{i=1}^{N} L_i(t) \quad \text{Eq.(10)}$$

where $L_i(t)$ is the number of tasks or processing time accumulated by robot r_i, and $\bar{L}(t)$ is the average workload.

A fundamental scheduling constraint requires that each task is assigned to exactly one robot at any moment:

$$\sum_{i=1}^{N} x_{i,j}(t) = 1, \forall \tau_j \in \mathcal{T}_t \quad \text{Eq.(11)}$$

Bottleneck stations are monitored through real-time queue measurements:

$$Q_k(t) = \textit{Number of pending tasks at station } k \textit{ at time } t \quad \text{Eq.(12)}$$

If $Q_k(t)$ exceeds a critical value, the scheduling layer triggers dynamic rescheduling or prioritization to balance loads.

Figure 2 shows the decision loop of deep Q-learning for autonomous optimisation of robot-task assignment in dynamic and uncertain production environments.

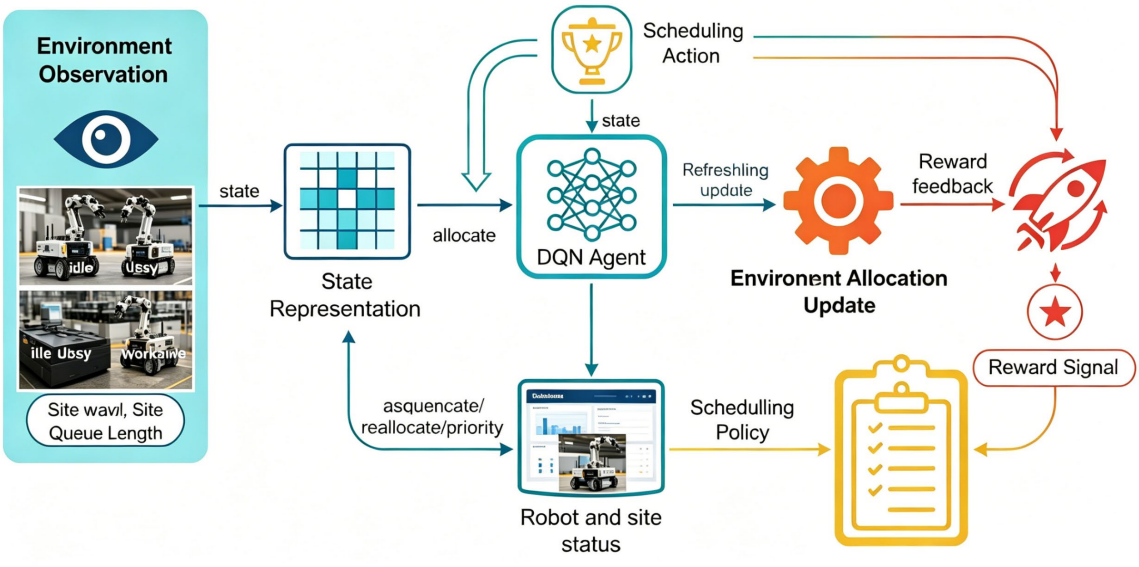

Figure 2. Process flow of the deep Q-learning-based multi-robot coordination and decision-making cycle

Here, the agent continuously adapts the assignment policy π based on observations of robot and station status, ultimately optimizing the expected cumulative reward for both throughput and fairness:

$$\pi^*(s_t) = arg \max_{\pi} \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t, \pi\right] \quad \text{Eq.(13)}$$

Build on the above data-driven scheduling and introduce a real-time communication platform to expand both the quantity and variety of robots and products in a large-scale system without reducing production efficiency.

Experimental Results and Data Analysis

Simulation Environment and Evaluation Protocol

In order to more thoroughly verify the scalability and performance of the new distributed scheduling system, a high-fidelity simulation was conducted using a custom event-driven assembly line simulator. In order to simulate the randomness and volatility in smart manufacturing scenarios, such as random task arrivals, various resource differences, and changes in robot team composition, a specific simulation environment was established. The test platform simulates the sub-assembly process of a car. There are N=8, 16, and 32 robots, with each robot located at up to ten parallel workstations. Due to drive constraints, random error rates, and variable computation delays, a single robot is not ideal. In each execution cycle, communication between agents is synchronized and delay-controlled through a single message bus, with strict checks for resource conflicts and physical collisions. The application of complex queue management and dynamic task prioritization in industrial environments provides solutions to complex problems.

To ensure consistency, each simulation experiment starts with the same random seed. To obtain quantitative results, the averages and standard deviations of 25 independent experiments for each configuration are averaged. In order to meet existing standards, a fixed simulation period of 1200 seconds will be used for performance comparison. In order to conduct a comparative evaluation, many classic and state-of-the-art benchmarks were used. These benchmarks include a centralized MILP scheduler, which optimizes global allocation using mixed-integer linear programming [26], a greedy heuristic algorithm that prioritizes robot

availability and minimizes instantaneous costs [27], a polling allocation scheme for uniform cyclic load balancing [28], and a decentralized deep reinforcement learning (RL) scheduler based on deep Q-learning and multi

Use an indicator system suitable for industrial automation to measure performance. These metrics include production line output (the average number of product units completed per full episode), cumulative reward (the discounted total of global rewards collected by all agents at all time steps), and task completion delay (the average and percentile time taken from the start to the end of a task). The distribution of idle time is the percentage of idle time in robot simulations, which can be used to measure resource utilization. Cooperation efficiency is also defined as the ratio of successful collaborative handovers to the total number of attempted robot-site transition cycles. This indicates the degree of multi-agent cooperation. According to the best practices benchmark for smart manufacturing, robustness can also be measured by the time the system can maintain normal operating conditions after intentional failures or simulated communication loss [29]. Under $p < 0.05$, the two-sided Wilcoxon rank-sum test validated all statistical results. To ensure complete reproducibility, the research community may request the provision of datasets and simulation source code.

Quantitative Analysis and Benchmark Comparison

This section will conduct a comprehensive quantitative analysis of the Deep Reinforcement Learning scheduling framework (DRL). This will include global performance metrics, detailed learning behaviors, real-time shop floor efficiency, and the internal structure of multi-agent collaboration. In order to obtain statistical reference, each experiment is randomly repeated 25 times.

As shown in Figure 3a, the cumulative reward of the DRL scheduler rapidly and steadily increased from approximately 8000 to 16600 over the first 150 training episodes, consistently surpassing the greedy (6000-11100) and MILP (7000-12900) baselines. Therefore, Figure 3b shows that DRL is more stable in learning rewards, with a DRL loss of approximately 0.126, while the MILP loss is around 0.195. As shown in Figure 3c, the number of iterations for MILP (median approximately 230) and the greedy algorithm (median approximately 270) is significantly higher and more variable, with over 92% of DRL training runs converging to the optimal allocation within 180 episodes. This indicates the efficiency and stability of the DRL optimization process.

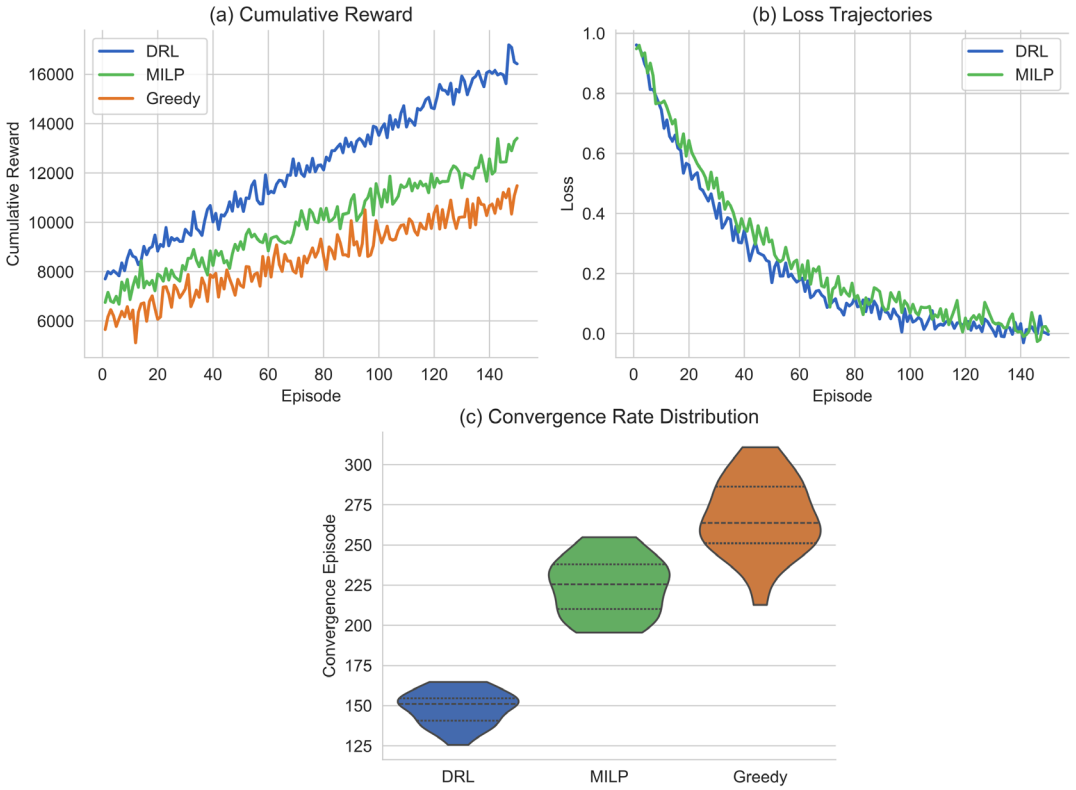

Figure 3. Learning and Convergence Performance (a) Cumulative reward over episodes; (b) Loss trajectories; (c) Convergence rate distribution

As shown in Figure 4a, the system throughput increases almost linearly with the team size. At $N = 8$, 16, 24, and 32, the DRL scheduler achieves throughputs of 162, 305, 453, and 590 units per episode, respectively. In contrast, at $N = 32$, the cyclic baseline and greedy baseline only produced 498 and 531 units, respectively, highlighting the $11.2 - 18.4\%$ improvement brought by coordinated DRL scheduling. The task completion delays under load factors ranging from 0.5 to 1.2 is shown in Figure 4b. This indicates that even under peak demand, the average delay of DRL is far below 60 seconds, while MILP and polling exceed 70 seconds when approaching saturation. This solution can absorb queue surges and ensure timely task completion, as the 95th percentile delay of DRL is 17.6 seconds lower than the optimal baseline of $N = 32$, indicating that this solution has advantages under high load conditions.

Figure 4c further demonstrates that DRL substantially improves resource utilization, reducing the median idle ratio to 7.4% (IQR: $6.3\% - 8.8\%$) versus MILP (11.1%, IQR: $9.9\% - 13.1\%$) and round-robin (12.6%, IQR: $10.2\% - 14.8\%$). Such reductions confirm the DRL scheduler's capability for equitable load balancing across large robot collectives, minimizing wasted operational intervals and sustaining high throughput.

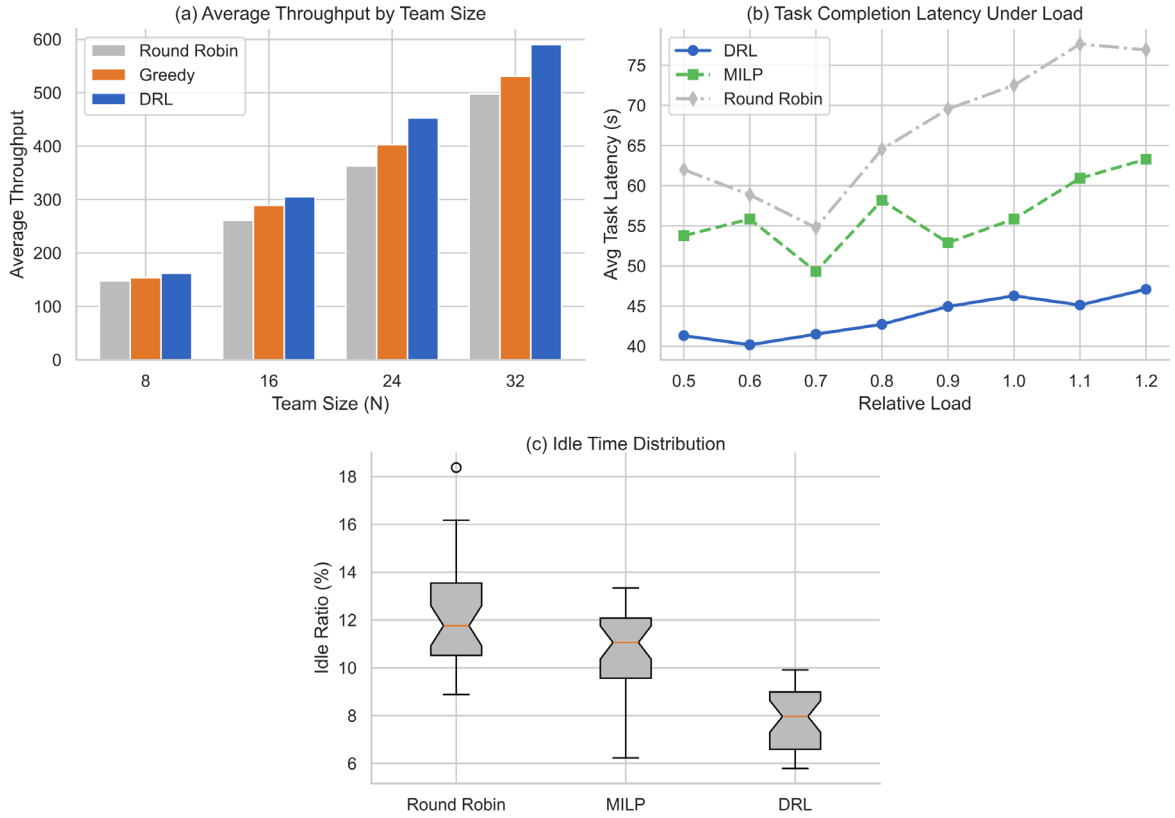

Figure 4. Assembly Line Throughput and Latency (a) Average throughput by team size; (b) Task completion latency under load; (c) Distribution of idle times among robots

The collaborative model and network structure also appear in Figure 5. The Gantt chart shows the planned execution times of different tasks, as shown in Figure 5a. To achieve efficient handovers, the DRL agent shortened the task time periods, thereby reducing the time wasted due to bottlenecks and task overlaps. As shown in Figure 5b, the enhanced coordination network has a rich multi-layer structure, including 7 distinct robot-task node handover paths, 8 direct robot-scheduler connections, and 19 robust robot-robot links. Horizontal communication between peers and vertical communication between system layers occur frequently. The new system has a more complex structure, whereas the previous systems often lacked connectivity or were too decentralized. As shown in Figure 5c, for DRL, almost all robot time blocks have a utilization rate exceeding 85%, with 45% of them exceeding 95%. Therefore, the spatiotemporal resource scheduling is efficient, and idle or stagnant states are extremely rare.

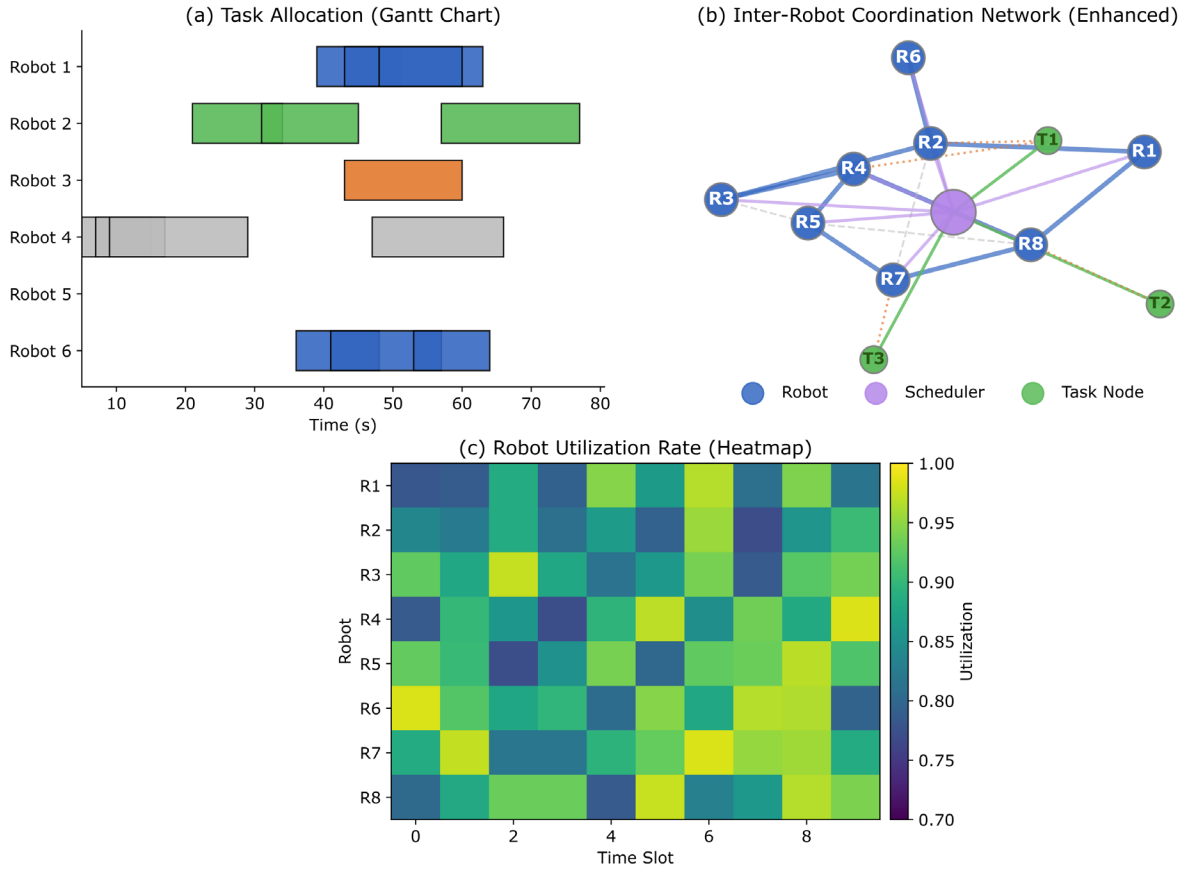

Figure 5. Multi-Robot Task Allocation Visualization (a) Task allocation Gantt chart; (b) Inter-robot coordination network; (c) Heatmap of utilization rates

Statistical analysis (Wilcoxon signed-rank test, $p < 0.05$) confirmed that all observed DRL improvements were statistically significant compared to the baseline. The aforementioned benefits are universal, as they remain stable under different product combinations, robot availability, and random task arrivals. Various feedback will be provided, resources can be quickly reallocated, and open-loop control will be introduced.

Robustness and Adaptability Studies

In industrial multi-robot environments, intelligent scheduling systems need to have good efficiency and be relatively fault-tolerant and insensitive to disturbances or changes in hyperparameters. A series of targeted experiments were conducted to determine the adaptability and fault tolerance of the proposed Deep Reinforcement Learning (DRL) framework under adverse weather conditions.

First, the system resilience is tested under random hardware failure conditions; subsequently, some robots are abruptly stopped during operation to simulate system failures. As shown in Figure 6a, under all failure rates, the DRL scheduler outperforms the heuristic and MILP methods in terms of task completion success rate. Among them, the MILP benchmark dropped to 71.4%, the greedy benchmark dropped to 67.2%, while the average episode success rate of DRL reached 85.7%, even in the case of up to 20% robot failures in a team of 32 units. It is worth noting that the performance of the DRL method declines at relatively high failure rates (91.6% for DRL and 81.2% for the greedy algorithm), but it performs better in adjusting strategies and dynamically reallocating tasks under unstable team compositions [30].

At the task level of industrial production lines, other unexpected disruptions include sudden reductions in work or delays. We introduced random task interruptions of varying frequencies during operation to simulate the aforementioned disturbances. Figure 6b shows that the performance degradation of the DRL agent is the smallest, although all strategies experience some throughput loss as the interruption frequency increases. In the case of a 15% task interruption rate per episode, DRL maintained approximately 83.6% of the original throughput;

MILP achieved about 76.9%, while greedy scheduling only reached around 70.5%. Due to the interference rate of 20%, the throughput of DRL is 74.5%, which is more than 12 percentage points higher than the next best algorithm. According to the above findings, agents can quickly reallocate and reschedule tasks to reduce overall productivity loss [31].

Similarly, they can quickly repair temporary faults in industrial automation systems. Figure 6c shows the distribution of system recovery times, that is, the time interval between fault injection and baseline throughput recovery. The DRL-based system has a median recovery time of 28 seconds and a good recovery time distribution. The median times for the greedy baseline and MILP are 47 seconds and 54 seconds, respectively, both showing slower and more unstable recovery times. Moreover, the recovery interval of DRL is shorter (standard deviation ≈ 5 seconds vs. ≈ 10 for MILP and ≈ 12 for greedy), indicating that DRL recovers to its optimal state more quickly and reliably after disturbances [32].

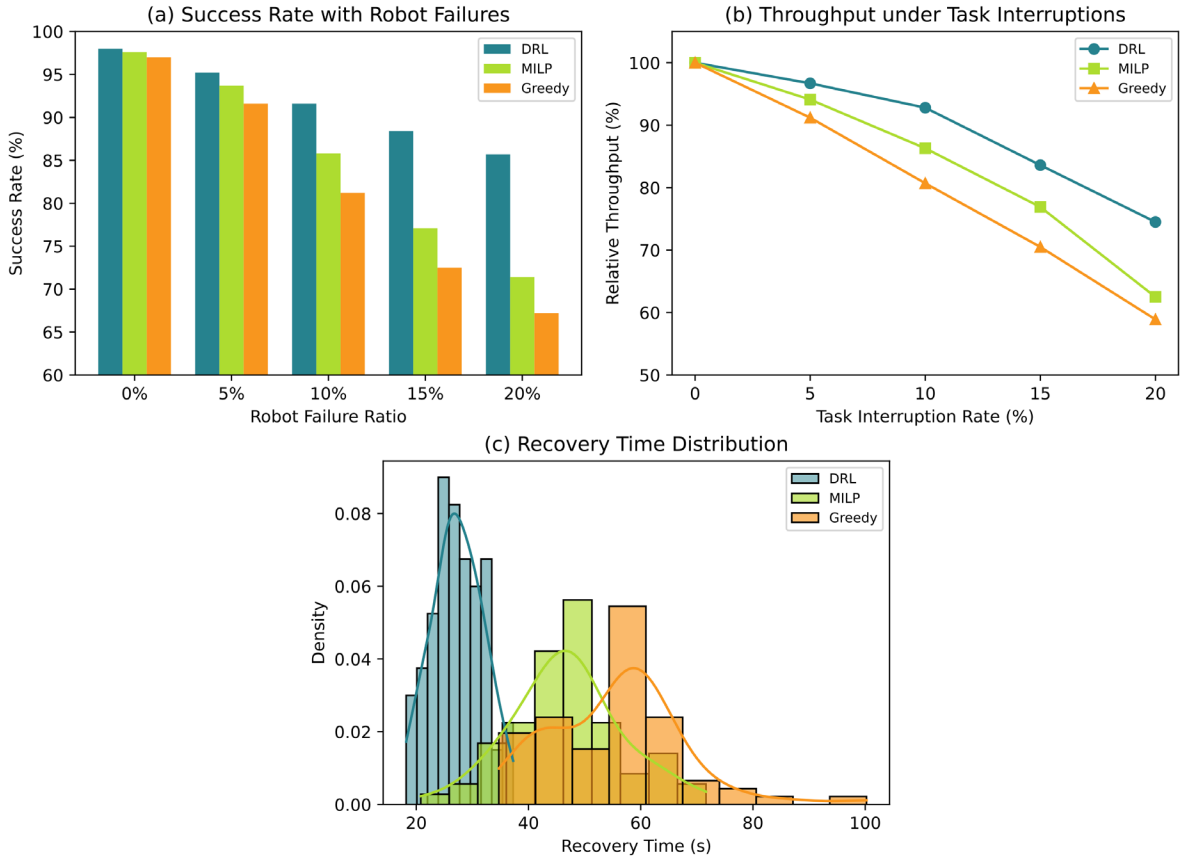

Figure 6. Robustness and Fault Tolerance Analysis (a) Task completion success rate under varying robot failure ratios; (b) Relative throughput with different task interruption rates; (c) Distribution of system recovery times following transient faults

Furthermore, each model was used for sensitivity and ablation analysis. Figure 7a shows the model's performance at various learning rates. It also shows the average throughput and cumulative reward of five repeated trials for each setting. The DRL scheduler, across a wide range of hyperparameters (1×10^{-4} to 5×10^{-4}), achieves an average cumulative reward of over 15,000, and the throughput per episode consistently exceeds 490 tasks. Only when the learning rate is below 5×10^{-5} or above 2×10^{-3} can a significant drop in both metrics and an increase in performance variance be observed; otherwise, the model remains stable.

Examine the impact of reward shaping strategies on efficiency and calibration. As shown in Figure 7b, compared to traditional dense or sparse reward signals (averaging 13,820 and 15,960), adding adaptive stage penalties increased the total reward by up to 12%. The use of a hybrid shaping method with episodic penalties and dense baselines also showed good results. The adaptive shaping method achieved convergence in an average of only 121 training iterations, which is significantly faster than the 174 iterations required by the traditional dense feedback method.

According to the ablation experiments of the network structure, as shown in Figure 7c, there are significant differences in sample efficiency and final state convergence among these four neural designs. The multi-head attention network achieved the highest cumulative reward (average: 16,300; range: 16,100–16,470) and required the least sample convergence (average: 8,900). It also outperforms the classic feedforward network (average: 13,100; 22,000 samples), LSTM (average: 14,580; 18,200 samples), and graph neural network variants (average: 15,060; 11,600 samples). The experimental runs on these two architectures indicate that they have the same advantages in terms of reward and efficiency. Furthermore, in the attention-based model, there is a smaller degree of variance between these runs.

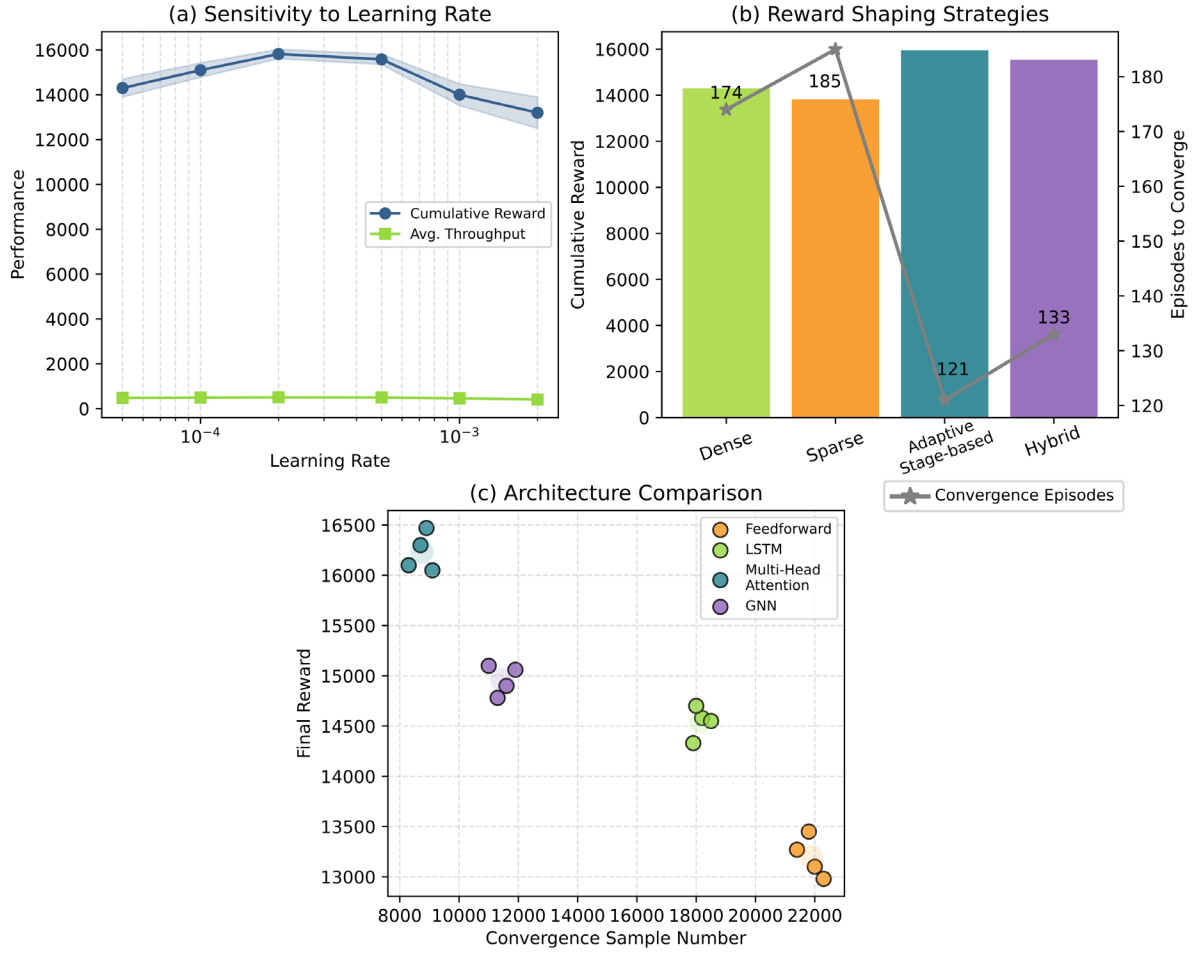

Figure 7. Sensitivity and Ablation Study (a) Cumulative reward and throughput for varying learning rates. (b) Performance comparison of reward shaping strategies. (c) Final reward and sample efficiency across network architectures

Conclusion

To address operational uncertainty, fault tolerance, and adaptability, this paper proposes a stable deep reinforcement learning (DRL) framework to solve the intelligent scheduling problem in industrial multi-robot environments. Through extensive experimental testing, the new technology outperforms traditional methods in many practical challenges. The aforementioned experiments indicate that, in the case of high robot failure rates and unexpected interruptions, the DRL-based scheduler can also improve task success rates and throughput. Moreover, this system is considered suitable for dynamic, safety-critical manufacturing facilities because it is faster and more stable in post-disaster recovery. Sensitivity and ablation experiment further validate the robustness of this method to hyperparameter changes and provide justification for this architecture; multiple attention heads have achieved significant improvements in sample efficiency and convergence quality.

Although some results have been achieved, there are still some drawbacks. The current DRL framework can handle random fluctuations and hardware failures, but the learning phase requires more computation and data,

making immediate deployment under offline data or hardware constraints potentially unfeasible. Moreover, these policy models may be widely applicable to new layouts or task types, but their practical application outside the training scenarios remains unknown. Interpretability and comprehensibility are also issues, as is typical with most model-free methods; although the system has achieved good results, the reasons for its decisions are not as clear as those based on rule-based or optimization methods.

In order to further expand the theoretical foundation and application scope of this study, further research is needed. Future research will explore continuous learning and rapid adaptation mechanisms, so that the system can self-improve with minimal human intervention when environmental changes occur. Using data or transfer learning methods can reduce training costs and improve sample efficiency in bridging the gap between simulation and real-world deployment. Explainable artificial intelligence and model-based reinforcement learning can also be used to enhance the transparency of critical functions and the operational stability. Finally, the framework will be extended to address the complex multi-objective scheduling constraints and large-scale mixed robot fleet issues in next-generation industrial automation systems.

Author Contributions

Barbara Szymańska and Małgorzata Kaczmarek contribute to conceptualization, methodology, software, validation, analysis, investigation, data collection, draft preparation, manuscript editing, visualization, supervision. Elżbieta Woźniak contributes to conceptualization, methodology, software. All authors have read and agreed with the manuscript before its submission and publication.

Funding

This research received no specific financial support from any funding agency.

Institutional Review Board Statement

Not applicable.

References

[1] Wang, L., Pan, Z., & Wang, J. (2021). A review of reinforcement learning based intelligent optimization for manufacturing scheduling. Complex System Modeling and Simulation, 1(4), 257-270. https://doi.org/10.23919/CSMS.2021.0027

[2] Johnson, D., Chen, G., & Lu, Y. (2022). Multi-agent reinforcement learning for real-time dynamic production scheduling in a robot assembly cell. IEEE Robotics and Automation Letters, 7(3), 7684-7691. https://doi.org/10.1109/LRA.2022.3184795

[3] Wang, Z., & Gombolay, M. (2020). Learning scheduling policies for multi-robot coordination with graph attention networks. IEEE Robotics and Automation Letters, 5(3), 4509-4516. https://doi.org/10.1109/LRA.2020.3002198

[4] Zhao, Y., Wang, Y., Tan, Y., Zhang, J., & Yu, H. (2021). Dynamic jobshop scheduling algorithm based on deep Q network. Ieee Access, 9, 122995-123011. https://doi.org/10.1109/ACCESS.2021.3110242

[5] Amirghafouri, F., Neghabi, A. A., Shakeri, H., & Sola, Y. E. (2025). Nature-inspired meta-heuristic algorithms for resource allocation in the internet of things. International Journal of Communication Systems, 38(5), e6141. https://doi.org/10.1002/dac.6141

[6] Chen, Z., Chen, K. C., Dong, C., & Nie, Z. (2021). 6G mobile communications for multi-robot smart factory. Journal of ICT Standardization, 9(3), 371-404. https://doi.org/10.13052/jicts2245-800X.934

[7] Orr, J., & Dutta, A. (2023). Multi-agent deep reinforcement learning for multi-robot applications: A survey. Sensors, 23(7), 3625. https://doi.org/10.3390/s23073625

[8] Chen, S., Zhu, M., Han, S., & Gupta, S. (2025). A deep fusion framework for end-to-end multi-product inventory optimization in e-commerce scenarios. International Journal of Production Economics, 109839. https://doi.org/10.1016/j.ijpe.2025.109839

[9] Liu, Q., Liu, Z., Xiong, B., Xu, W., & Liu, Y. (2021). Deep reinforcement learning-based safe interaction for industrial human-robot collaboration using intrinsic reward function. Advanced Engineering Informatics, 49, 101360. https://doi.org/10.1016/j.aei.2021.101360

[10] Yu, T., Huang, J., & Chang, Q. (2021). Optimizing task scheduling in human-robot collaboration with deep multi-agent reinforcement learning. Journal of manufacturing systems, 60, 487-499. https://doi.org/10.1016/j.jmsy.2021.07.015
[11] Dai, W., Rai, U., Chiun, J., Cao, Y., & Sartoretti, G. (2025). Heterogeneous multi-robot task allocation and scheduling via reinforcement learning. IEEE Robotics and Automation Letters, 10(3), 2654-2661. https://doi.org/10.1109/LRA.2025.3534682
[12] Zhou, L., Zhang, L., & Horn, B. K. (2020). Deep reinforcement learning-based dynamic scheduling in smart manufacturing. Procedia Cirp, 93, 383-388. https://doi.org/10.1016/j.procir.2020.05.163
[13] Lei, K., Guo, P., Wang, Y., Zhang, J., Meng, X., & Qian, L. (2023). Large-scale dynamic scheduling for flexible job-shop with random arrivals of new jobs by hierarchical reinforcement learning. IEEE Transactions on Industrial Informatics, 20(1), 1007-1018. https://doi.org/10.1109/TII.2023.3272661
[14] Wang, Y., Su, S., & Wang, Y. (2025). Attention-augmented multi-agent collaboration for Smart Industrial Internet of Things task offloading. Internet of Things, 31, 101572. https://doi.org/10.1016/j.iot.2025.101572
[15] Urrea, C. (2025). Hybrid Deep Learning-Reinforcement Learning for Adaptive Human-Robot Task Allocation in Industry 5.0. Systems, 13(8), 631. https://doi.org/10.3390/systems13080631
[16] Zhang, M., Lu, Y., Hu, Y., Amaitik, N., & Xu, Y. (2022). Dynamic scheduling method for job-shop manufacturing systems by deep reinforcement learning with proximal policy optimization. Sustainability, 14(9), 5177. https://doi.org/10.3390/su14095177
[17] Chen, H., Lai, Y., Liu, J., & Wanyan, H. (2024). Interpretable cross-layer intrusion response system based on deep reinforcement learning for industrial control systems. IEEE Transactions on Industrial Informatics, 20(7), 9771-9781. https://doi.org/10.1109/TII.2024.3388672
[18] Ong, K. S. H., Wang, W., Hieu, N. Q., Niyato, D., & Friedrichs, T. (2022). Predictive maintenance model for IIoT-based manufacturing: A transferable deep reinforcement learning approach. IEEE Internet of Things Journal, 9(17), 15725-15741. https://doi.org/10.1109/JIOT.2022.3151862
[19] Urrea, C. (2025, April). Hybrid Fault-Tolerant Control in Cooperative Robotics: Advances in Resilience and Scalability. In Actuators (Vol. 14, No. 4, p. 177). MDPI. https://doi.org/10.3390/act14040177
[20] Shakeri, Z., Benfriha, K., Varmazyar, M., Talhi, E., & Quenehen, A. (2025). Production scheduling with multi-robot task allocation in a real industry 4.0 setting. Scientific Reports, 15(1), 1795. https://doi.org/10.1038/s41598-024-84240-3
[21] Guo, J., Tian, P., Wang, Q., Li, J., Tian, C., Zhou, C., ... & Gao, H. (2025). Phenotyping inspection robot system via width-adjustable configuration and depth-sensing-based end-to-end deep reinforcement learning row-following algorithm. Computers and Electronics in Agriculture, 239, 111066. https://doi.org/10.1016/j.compag.2025.111066
[22] Wu, Z., Fan, H., Sun, Y., & Peng, M. (2023). Efficient multi-objective optimization on dynamic flexible job shop scheduling using deep reinforcement learning approach. Processes, 11(7), 2018. https://doi.org/10.3390/pr11072018
[23] Caliskanelli, I., Goodliffe, M., Whiffin, C., Xymitoulias, M., Whittaker, E., Verma, S., & Skilton, R. (2021). Engineering interoperable, plug-and-play, distributed, robotic control systems for futureproof fusion power plants. Robotics, 10(3), 108. https://doi.org/10.3390/robotics10030108
[24] Neves, M., Vieira, M., & Neto, P. (2021). A study on a Q-Learning algorithm application to a manufacturing assembly problem. Journal of Manufacturing Systems, 59, 426-440. https://doi.org/10.1016/j.jmsy.2021.02.014
[25] Wang, H., Bai, Y., & Xie, X. (2023). Deep reinforcement learning based resource allocation in delay-tolerance-aware 5G industrial IoT systems. IEEE Transactions on Communications, 72(1), 209-221. https://doi.org/10.1109/TCOMM.2023.3322736
[26] Wang, H., Lin, W., Peng, T., Xiao, Q., & Tang, R. (2025). Multi-agent deep reinforcement learning-based approach for dynamic flexible assembly job shop scheduling with uncertain processing and transport times. Expert Systems with Applications, 270, 126441. https://doi.org/10.1016/j.eswa.2025.126441
[27] del Real Torres, A., Andreiana, D. S., Ojeda Roldan, A., Hernandez Bustos, A., & Acevedo Galicia, L. E. (2022). A review of deep reinforcement learning approaches for smart manufacturing in industry 4.0 and 5.0 framework. Applied Sciences, 12(23), 12377. https://doi.org/10.3390/app122312377
[28] Zhang, L., Sun, Y., Barth, A., & Ma, O. (2020). Decentralized control of multi-robot system in cooperative object transportation using deep reinforcement learning. IEEE access, 8, 184109-184119. https://doi.org/10.1109/ACCESS.2020.3025287

[29] Zhao, L., Fan, J., Zhang, C., Shen, W., & Zhuang, J. (2023). A DRL-based reactive scheduling policy for flexible job shops with random job arrivals. IEEE Transactions on Automation Science and Engineering, 21(3), 2912-2923. https://doi.org/10.1109/TASE.2023.3271666

[30] Bingol, M. C., & Aydogmus, O. (2025). A Reinforcement Learning Approach to Robust Control in an Industrial Application. Arabian Journal for Science and Engineering, 50(8), 6083-6094. https://doi.org/10.1007/s13369-024-09797-7

[31] Zhou, T., Tang, D., Zhu, H., & Wang, L. (2020). Reinforcement learning with composite rewards for production scheduling in a smart factory. IEEE Access, 9, 752-766. https://doi.org/10.1109/ACCESS.2020.3046784

[32] Jiang, C., Chen, Z., & Guo, Y. (2020). Multi-robot formation control: a comparison between model-based and learning-based methods. Journal of Control and Decision, 7(1), 90-108. https://doi.org/10.1080/23307706.2019.1697970