

*Research Article*

A Deep Reinforcement Learning Scheduling Approach for Production Lines in Series

Zhijing Zhou^{1,2}, Jiajia Wang¹, Gang Yin², Jiachen Zhou³, Yongxiang Zhang², Jian Guo^{1*}, Junhao Zhang^{1**}

¹College of Mechanical and Electrical Engineering, Wenzhou University, Wenzhou 325035, China

²Zhejiang Haozhonghao Health Products Co., Ltd, Wenzhou, 325400, China

³Academy of Interdisciplinary Studies, Hong Kong University of Science and Technology, Hong Kong, 999077, China

*Corresponding author: 20210701@wzu.edu.cn, Tel.: +86 13700232898

**Corresponding author: zjh1911612032@gmail.com, Tel.: +86 18357812382

Abstract: Mixed-model assembly lines (MMAL) are increasingly replacing traditional single-type lines to meet mass customization needs as product demand diversifies. This paper addresses the series-parallel production line sequencing problem (SPPLSP) within MMAL, addressing the complexity introduced by fluctuating orders, asynchronous machinery, and resource imbalances. This study introduces a deep reinforcement learning (DRL) approach to address these challenges, focusing on the impact of machine selection and job sequencing on order tardiness and energy consumption during transitions. Initially, a mixed-integer linear programming model (MILPM) is expressed with constraints, such as conditions related to serial-parallel production lines and machines, to make a clear problem description. Subsequently, a double deep Q-network (DDQN) algorithm is employed to solve the SPPLSP, utilizing an innovative state and action space representation to maintain dynamic balance across serial and parallel lines and adapt to changes in machine and workpiece data. Additionally, an improved agent reward function integrates a multi-agent hierarchical distributed scheduling structure to approximate the actual objectives. Finally, a simulation-based numerical study, grounded in real-world case scenarios, validates the performance of the proposed method across various scales of problems. The numerical results reveal that the DDQN method consistently achieves faster convergence and superior stability values across all test instances and yields higher-quality Pareto frontier solutions in 49/50 test instances.

Keywords: Double deep Q-network; Mixed-model assembly line; Multi-agent; Series-parallel production line sequencing

1. Introduction

The diversification of market demands has rendered the traditional single-type production lines increasingly insufficient to meet the evolving needs for mass customization (Zhou & Shen, 2018). Consequently, mixed-model assembly lines (MMALs) have gained widespread adoption, enabling manufacturers to produce a variety of products on a single line, thereby enhancing flexibility and responsiveness to market changes (Yelles-Chaouche et al., 2022). As manufacturers strive to balance the efficiency of mass production with product customization, the concept of MMALs has emerged as a key strategy (Gao et al., 2020). However, the effective operation of MMALs is intricately linked to the management of series-parallel production line sequencing problems (SPPLSP) (Xi et al., 2022). This relationship underscores the importance of SPPLSP within the broader context of MMALs, as it directly impacts the ability to optimize production schedules, minimize tardiness, and reduce energy consumption during transitions (Nourelfath & Yalaoui, 2012). The complexity of SPPLSP arises from the need to coordinate both serial and

parallel production lines, which is exacerbated by the fluctuating nature of orders, asynchronous machinery, and resource imbalances (Gholizadeh et al., 2022).

In diversified manufacturing systems, machine setup times and switching energy costs determined by operational sequence modifications are a significant research focus (Yang & Cheng, 2020). Traditional production line research commonly assumes sequence-independent parameters, largely ignoring the transition-phase volatility between distinct processing operations. However, actual workshop dynamics prove that neglecting these sequence-dependent variations severely degrades global energy efficiency as the variety of product types expands (Zhang et al., 2020a).

Traditional production control cannot adequately capture dynamic shopfloor interdependencies in SPPLSP. Although deep reinforcement learning (DRL) (Rai et al., 2021) effectively manages random job arrivals, existing models often overlook complex synchronization constraints and sequence-dependent energy costs in series-parallel configurations. To address this, we introduce a multi-agent cooperative Double Deep Q-Network (DDQN) framework with asynchronous transitions. By decoupling action selection from target evaluation, DDQN eliminates the severe Q-value overestimation inherent in traditional DQN (Van Hasselt et al., 2016), resolving sub-optimal scheduling in high-dimensional search spaces. Furthermore, our routing and sequencing agents collaboratively optimize synchronization constraints and the energy-tardiness trade-off, delivering a more resilient scheduling solution than standard DRL approaches.

The remainder of this paper is organized as follows: Section 2 provides a comprehensive literature review and analysis of the research focus. Section 3 describes the SPPLSP, defines the notations involved, and describes the mathematical model. Section 4 introduces the DDQN-based method for solving the SPPLSP. Section 5 verifies the validity and superiority of the proposed approach through real-world computational experiments. Finally, Section 6 summarizes the research and lists some future research opportunities.

2. Literature Review

2.1 Scheduling in the Series-Parallel Production System

Research on series-parallel production system scheduling generally spans static, dynamic, and hybrid strategies. In static and reliability-focused modeling, Azadeh et al. (2015) incorporated warm standby redundancy in a multi-state flow-shop to optimize system reliability, while Noureldath and Yalaoui (2012) integrated load distribution to minimize operating costs under load-dependent failure constraints. To handle shopfloor disturbances, Li and Peng (2014) modeled stochastic line reconfigurations using Markov processes and universal generating functions. More recently, Xi et al. (2022) applied a Decomposition-Coordination Method to simultaneously optimize machine selection and buffer allocation in unbalanced lines, effectively balancing total cost of ownership with throughput and WIP inventory levels.

To adapt to volatile multi-product environments, scheduling research has increasingly integrated mixed-model optimization with series-parallel configurations (Taube & Minner, 2018). Uncertainty theory has been incorporated to balance assembly line fluctuations (Liu et al., 2021), and localized neighborhood search heuristics have been utilized to efficiently resolve sequence-dependent setup constraints (Yang & Cheng, 2020). This underscores the critical need for efficient algorithms to rapidly solve complex scheduling problems.

Many algorithms, rules, and other scheduling methods have been proposed to search for the best scheduling solution to effectively solve the various scheduling problems mentioned above (Liu et al., 2021; Sikora, 2024). Some related studies of scheduling methods are presented below. Noureldath and Yalaoui (2012) used an integrated scheduling approach that combines decision-making regarding load distribution with tactical production planning to solve the multitask scheduling problem in multistate series-parallel systems. Mosadegh et al. (2017) applied dynamic programming and greedy algorithms to minimize the total expected workload in single-station sequencing. Furthermore, Mosadegh et al. (2020) extended their approach to multi-station

configurations using a hyper-simulated annealing method. Wang et al. (2021) balanced parallel disassembly lines under stochastic times by optimizing workstation counts, workloads, and disassembly profits. Collectively, these methodologies lay a solid mathematical foundation for resolving the complexities of multi-product series-parallel scheduling.

2.2 DRL-based Scheduling in the Production System

Machine learning methods have recently been applied in various research fields due to their superior learning capabilities (Han et al., 2019; Zhang et al., 2020b). Reinforcement learning (RL) and deep RL (DRL) provide a general framework for decision-making optimization in dynamic environments, helping solve combinatorial optimization problems (Chen et al., 2015; Kayhan & Yildiz, 2023). The aim of DRL is to use the scalability of neural networks in high-dimensional spaces to solve problems of larger scale and complexity (Zhao & Melkote, 2024). Some of the seminal related works are shown below. Some scholars have adopted RL and DRL methods for adaptive scheduling decisions to respond to changes in the shop floor environment in real time (Hu et al., 2020; Li et al., 2023). Recent advances have led to the development of a comprehensive methodology and review that systematically outlines the state-of-the-art and future directions of deep reinforcement learning for machine scheduling (Khadivi et al., 2025). Zhao et al. (2020) proposed an assembly sequence planning system (ASPW) based on DRL for production sequencing, which automatically plans the sequence for complex assembly products to improve assembly efficiency. (Mao et al., 2023) introduced a Petri-net-based DQN approach for adaptive disassembly sequence planning in VR maintenance training. Their framework successfully stabilized the dynamic policy search by asynchronously updating dual neural networks and integrating a genetic algorithm fitness function to substitute the conventional long-term return. Paeng et al. (2021) proposed a DRL framework featuring parameter-sharing neural networks to minimize total delays across unseen scheduling tasks without retraining. Fuqing and Yu (2026) integrated multi-task learning with graph edge attention networks to handle highly dynamic workshop environments, while Du et al. (2024) utilized a multi-objective DQN to jointly optimize makespan, crane transportation, and machine setups. Furthermore, the versatility of Double DQN (DDQN) has been extensively validated in adjacent high-dimensional domains, such as the QoS routing protocol by Zhang et al. (2025) and the adaptive edge-cloud task scheduler by Yamsani and Reddy (2026). Collectively, these advancements demonstrate the efficacy of DDQN-based architectures in managing high-dimensional state spaces and complex SPPLSP constraints. Beyond traditional shopfloors, reinforcement learning has proven effective in adjacent dynamic domains; for example, Alla and Thangarasu (2023) combined RL with blockchain for drug supply chain logistics, while Farouk et al. (2026) developed a hybrid A3C-PPO scheduler for high-congestion traffic scenarios. These cross-domain applications highlight the adaptability of RL agents in handling stochastic, constrained environments similar to our SPPLSP.

Compared with traditional meta-heuristics, DRL-based algorithms demonstrate substantial superiority in solution quality and computational efficiency for complex manufacturing scheduling. Modern mass customization trends necessitate the integration of series-parallel configurations with mixed-model assembly lines. Through the aforementioned literature review, several critical research limitations are synthesized as follows: 1) mixed-assembly line and series-parallel production as a production system research hotspot has attracted attention, and some scholars have proposed different scheduling methods for the two systems separately. However, there is no relevant research on these two production systems as a whole, considering the intricate interactions between machine selection, job sequencing, and coordination of serial and parallel production lines. 2) DRL algorithms have been preliminarily applied to the scheduling problems of various production systems, but there are fewer studies on the problems of multi-product and multi-resource collaborative scheduling based on DRL; and 3) combining production sequences with DRL to solve the scheduling problem of series-parallel production lines with mixed-model assembly lines is less studied.

Figure 1 synthesizes the research gap mapping to address these literature limitations. To bridge this gap, we propose a multi-agent cooperative DDQN framework for the SPPLSP that simultaneously addresses shop topology bottlenecks and sequence-dependent setup energy constraints. By dynamically regulating production queues and machine states, our decoupled architecture mitigates stochastic order fluctuations, jointly minimizing total tardiness and switching energy costs to enhance the flexibility and efficiency of mixed-model assembly lines.

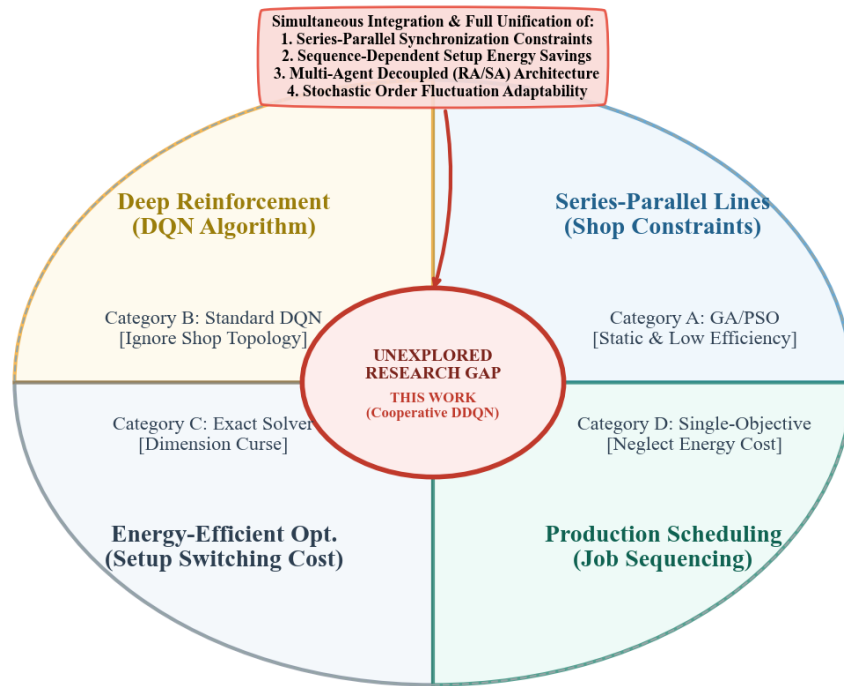


Figure 1 Tech-matrix map of the existing literature and identified research gaps

3. Problem Analysis and Mathematical Formulation

3.1 Problem Analysis

This study considers a mixed-model assembly line integrating multiple parallel component lines with a series-connected assembly line, where dedicated workstations equipped with identical parallel machines schedule j stochastic workpieces across pt product types with distinct deadlines. Although specific machine operations differ, parts follow identical workstation routing sequences, with processing times, sequence-dependent setup switchovers, and energy costs derived from historical manufacturing data. As illustrated in the hierarchical workflow in Figure 2, dynamic scheduling is executed systematically: upon job arrival, the Routing Agent (RA) determines precise facility allocation, while the Sequencing Agent (SA) concurrently processes operation sequencing to eliminate resource starvation.

3.2 Notations

The notations for the indices, constants, and decision variables involved in the mathematical model of the SPPLSP established in this paper are defined as follows:

3.2.1 Indices

1. Product types: $\mathbf{PT} = \{PT_b : b = 1, 2, \dots, n\}$ is the set of n product types.

2. Jobs: $\mathbf{J} = \{J_i : i = 1, 2, \dots, n\}$ is the set of n jobs.
3. Lines: $\mathbf{L} = \{L_l : l = 1, 2, \dots, b\}$ is the set of lines b .
4. Workstations: $\mathbf{S} = \{S_s : s = 1, 2, \dots, c\}$ is the set of c workstations. Because the buffers and first-level attributes correspond to the workstations, they share the same indices as the workstations.
5. Machines: $\mathbf{M} = \{M_m : m = 1, 2, \dots, a\}$ is the set of workstations.
6. Buffers: $\mathbf{MB} = \{MB_s : s = 1, 2, \dots, c\}$ is the set of c buffers.
7. First-level attributes: $\mathbf{AB}^1 = \{AB_s^1 : s = 1, 2, \dots, c\}$ is the set of attributes.
8. Second-level attributes: $\mathbf{AB}^2 = \{AB_e^2 : e = 1, 2, \dots, d\}$ is the set of second-level attributes.

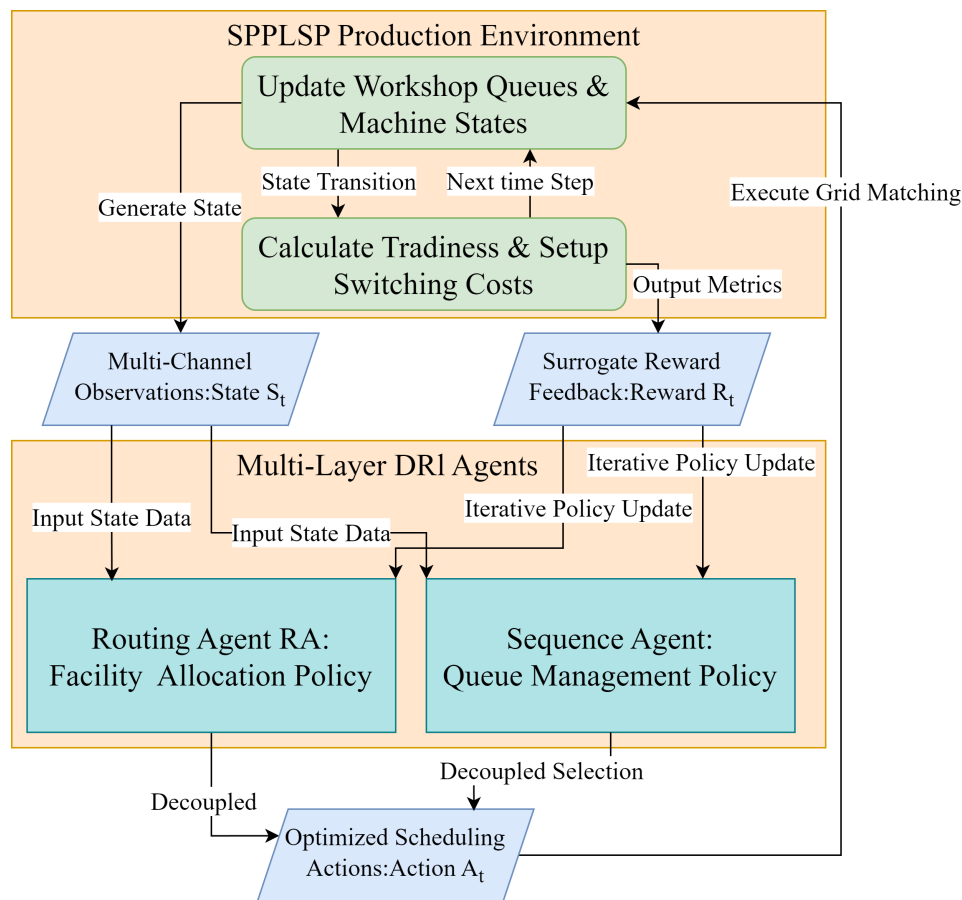


Figure 2 Proposed hierarchical optimization workflow for joint job routing and operation sequencing

3.2.2 Constants and Parameters

1. t_{now} : Current time.
2. N_{Pb} : Number of products PT_b .
3. E_b : Due date (delivery deadline) for product PT_b .

4. pr_b : Proportion of quantity of each part required for product PT_b , where $pr_b = x_{1b} : x_{2b} : \dots : x_{lb}$.
5. $am_{i,l,s,m}$: Available time that machine m to process job i .
6. $at_{i,l,s}$: Arrival time that job i arrives at buffer s corresponding to workstation s on production line l .
7. $wt_{i,l,s}$: Waiting time for job i at buffer s corresponding to workstation s on production line l .
8. $ct_{l,s}^{u,v}$: Machine operation switching time between the first-level attribute AB_s^1 corresponding to the second-level attributes AB_u^2 and AB_v^2 corresponding to workstation s on production line l .
9. $cc_{l,s}^{u,v}$: Machine operation switching energy cost between the first-level attribute AB_s^1 corresponding to the second-level attributes AB_u^2 and AB_v^2 corresponding to workstation s on production line l .
10. $ccl_{s,m}$: The cumulative energy cost associated with switching between consecutive jobs assigned to machine m at workstation s on line l .
11. $pt_{i,l,s,m,e}$: Processing time of job i assigned to machine m at workstation s on production line l and specifying secondary processing attribute e corresponding to processing operation e .
12. $rpt_{i,l}$: Remaining processing time for job i on line l .
13. $rt_{i,l,s}$: Release time of job i at workstation s on line l .
14. SK_i : Slack time of job i .
15. SK_g : Slack time of job group g .
16. $AT_{b,g}$: $AT_{b,g} = (at_{b,g}^{first}, at_{b,g}^{last})$ is the arrival time of job groups, where $at_{b,g}^{last}$ is the arrival time of the last part that arrives in the buffer in job group g with assembled product type PT_b , and $at_{b,g}^{first}$ is the arrival time of the first part that arrives in the buffer in job group g with assembled product type PT_b .

3.2.3 Decision Variables

1. $spt_{i,l,s,m}$: Starting processing time after job i is assigned to machine m in workstation s on production line l .
2. $C_{i,l,s,m}$: Completion time of job i at machine m in workstation s on line l .
3. $pn_{i,l,s,m,t}$: 1 if job i is assigned to machine m in workstation s on production line l at time t ; otherwise, 0.
4. $nb_{i,j}$: 1 if job i is the immediately preceding job of job j on a particular machine, otherwise 0.
5. Jobs sequence: $\mathbf{JQ}_{l,s,m} = \{J_i : i = 1, 2, \dots, n\}$ is the sequence of jobs to be processed assigned to machine m at workstation s on line l .
6. Operations sequence: $\mathbf{ABQ}_{l,s,m} = \{AB_e^2 : e = 1, 2, \dots, d\}$ is the sequence of operations assigned to machine m at workstation s on production line l .

7. Jobs group sequence: $\mathbf{GQ} = \{J_g : g = 1, 2, \dots, n\}$ is the sequence of job groups in the assembly line buffer.
8. $C_{b,g,m}$: The completion time of the g -th job group of product type b on assembly machine m , while E_b is the due date (delivery deadline) for product type b .

3.3 MILP SPPLSP Model Considering Operation Switching

In the SPPLSP, managing the production of diverse products with varying delivery dates and processing requirements is essential for optimizing manufacturing efficiency and minimizing setup changes' energy costs. This study aims to minimize the total tardiness and switching costs by establishing an SPPLSP optimization model that accounts for switching. The objective functions are defined in Equations (1) and (4).

$$f_1 = \sum_{b=1}^{PT} \sum_{g=1}^{GQ} \sum_{m=1}^M |C_{b,g,m} - E_b| \quad (1)$$

$$cc_{i,j}^{u,v} = \begin{cases} cc_{l,s}^{u,v}, & u \neq v \\ 0, & u = v \end{cases} \quad (2)$$

$$cc_{l,s,m} = \sum_{i=1}^J \sum_{j=1}^J nb_{i,j} \times cc_{i,j}^{u,v}, \quad i \neq j, \forall l, s, m \quad (3)$$

$$f_2 = \sum_{l=1}^L \sum_{s=1}^S \sum_{m=1}^M cc_{l,s,m} \quad (4)$$

The first term in Eq. (1) represents the total weighted tardiness across all orders. Eqs. (2) and (3) define the switching energy costs between the operations. The second term in Eq. (1) signifies the total switching energy consumption cost incurred by changes in the setup during the entire production sequence.

The resource constraints of the production system are as follows.

Eq. (5) is the machine processing capacity constraint, i.e., a machine can process at most one job at a given time:

$$\sum_{i=1}^J pn_{i,l,s,m,t} \leq 1, \quad \forall l, s, m \quad (5)$$

The job processing count constraint can be determined by Eq. (6), where each job can only be processed at most once by a workstation, i.e., the job visits each workstation once or 0 times in a given order, and there is no re-entry rework:

$$\sum_{l=1}^L \sum_{s=1}^S pn_{i,l,s} \leq 1, \quad \forall i \quad (6)$$

The shop floor machining process constraints can be derived as Eqs. (7)–(10), which are the relationship constraints of completion time, switching time, waiting time, start processing time, and release time, respectively:

$$C_{i,l,s,m} = spt_{i,l,s,m} + pt_{i,l,s,m,e} \quad (7)$$

$$wt_{i,l,s} = spt_{i,l,s,m} - at_{i,l,s} \quad (8)$$

$$spt_{i,l,s,m} = \begin{cases} at_{i,l,s} + ct_{l,s,m}, & \text{when } \sum_{j=1}^J pn_{j,l,s,m,t} < 1 \\ rt_{j,m} + ct_{l,s,m}, & \text{when } \sum_{j=1}^J pn_{j,l,s,m,t} = 1 \text{ and } nb_{j,i} = 1 \end{cases} \quad (9)$$

$$rt_{i,l,s} = \begin{cases} spt_{i,l,s,m} + pt_{i,l,s,m,e}, & \text{when } \sum_{j=1}^J pn_{j,l,s+1,B,t} < MBS_{l,s+1} \\ rt_{j,l,s+1}, & \text{when } \sum_{j=1}^J pn_{j,l,s+1,B,t} = MBS_{l,s+1} \end{cases} \quad (10)$$

Eq. (11) indicates that the time information is not less than zero:

$$C_{i,l,s,m}, ct_{l,s,m}, wt_{i,l,s}, spt_{i,l,s,m}, pt_{i,l,s,m,e}, rt_{i,l,s} \geq 0, \quad \forall i, l, s, m, e \quad (11)$$

Eq. (12) is the composition of the parts of the group of operations queued for machining in the assembly line's buffer zone. The component composition of each job group to be processed is determined by the proportion of components of the product type to which it belongs:

$$J_g^b = \left(\frac{\{J_{i,1}, AB_{1,e}\}}{x_1^b}, \frac{\{J_{i,2}, AB_{2,e}\}}{x_2^b}, \dots, \frac{\{J_{i,l}, AB_{l,e}\}}{x_l^b} \right) \quad (12)$$

Eq. (13) is the completion time for the job to go through the entire processing line:

$$C_{i,l} = \max(C_{i,l,s,m}) \quad (13)$$

Eqs. (14)–(15) are the arrival times of the job group parts to the buffer:

$$at_{b,g}^{first} = \min \left(\frac{\{C_{i,1}\}}{x_1^b}, \frac{\{C_{i,2}\}}{x_2^b}, \dots, \frac{\{C_{i,l}\}}{x_l^b} \right) \quad (14)$$

$$at_{b,g}^{last} = \max \left(\frac{\{C_{i,1}\}}{x_1^b}, \frac{\{C_{i,2}\}}{x_2^b}, \dots, \frac{\{C_{i,l}\}}{x_l^b} \right) \quad (15)$$

Eqs. (16)–(17) are the waiting time and start processing time constraints in the bus buffer:

$$wt_{b,g} = spt_{b,g} - at_{b,g}^{first} \quad (16)$$

$$spt_{b,g,m} = \begin{cases} at_{b,g}^{last}, & \text{when } \sum_{g=1}^{GQ} pn_{g,m,t} < 1 \\ rt_{h,m}, & \text{when } \sum_{g=1}^{GQ} pn_{g,m,t} = 1 \text{ and } nb_{h,g} = 1 \end{cases} \quad (17)$$

Eqs. (18)–(19) are the slack time constraints:

$$\min(E_b) - t_{now} \leq SK_g \leq \max(E_b) - t_{now} \quad (18)$$

$$\min(E_b - pt_b) - t_{now} \leq SK_i \leq \max(E_b - pt_b) - t_{now} \quad (19)$$

4. DDQN-based SPPLSP Approach

4.1 Overview of the DDQN-based Scheduling Optimization Method

Figure 3 shows the decoupled multi-agent deep reinforcement learning architecture for optimizing SPPLSP using the DDQN algorithm. Within this framework, multi-channel observations from the series-parallel manufacturing environment are dynamically distributed into two parallel state paths. The color-coded RA and SA independently process these state inputs via their respective evaluation and target network pairs. Driven by system metrics feedback, the surrogate reward function serves as a critical process carrier to route step rewards back to both networks, dynamically guiding the policy updates through dedicated asynchronous feedback loops.

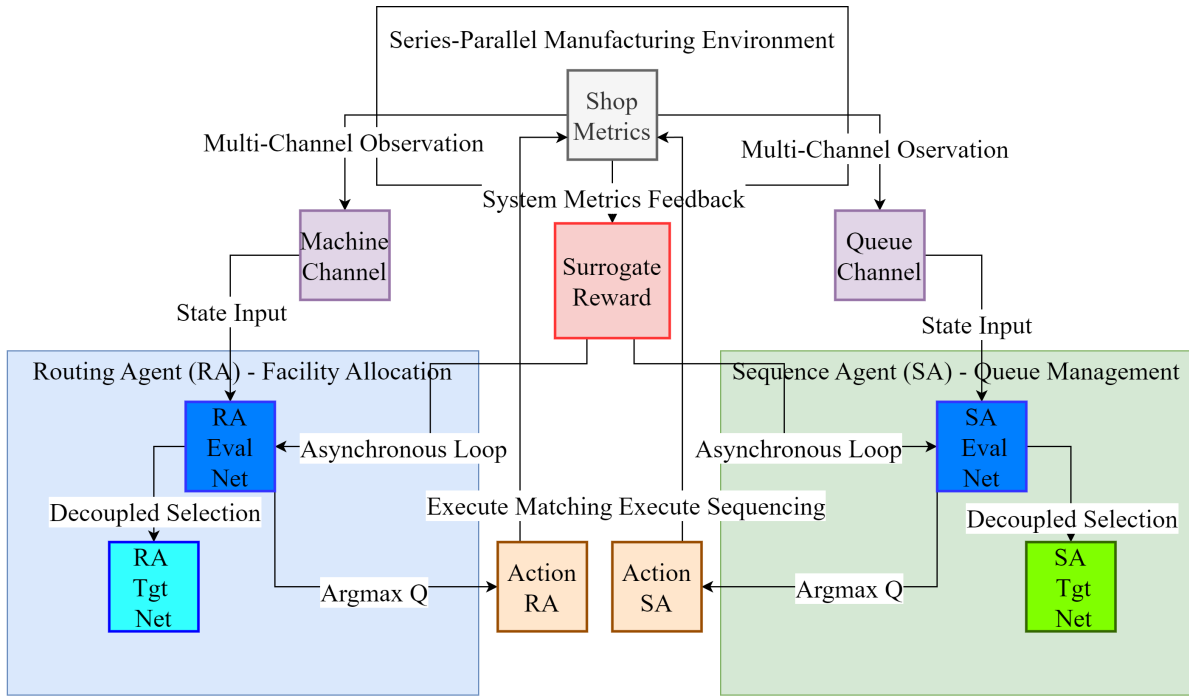


Figure 3 Decoupled multi-agent deep reinforcement learning scheduler architecture

4.2 State Representation

Traditional neural networks in deep reinforcement learning struggle with varying job line sizes in production systems. To address this, many studies use fixed scales, which limits their applicability in dynamic real-world production. However, this study focuses on dynamic production systems with changing job arrivals and completion.

We construct a state space that adapts to changes in the production system using abstract job information. The following representations are relevant: (1) $\mathbf{GQ} = \{J_g : g = 1, 2, \dots, n\}$ represents the current assembly line job group sequence in the system, useful for measuring production completion across production lines and shortage status in the output sequence; (2) $\mathbf{AT} = \{AT_{b,g}\}$ is the set of times these job groups arrive at the final assembly buffer, where $AT_{b,g}$ consists of elements $(at_{b,g}^{first}, at_{b,g}^{last})$.

The state space of the routing agent includes information on processing facilities and all scheduled jobs, which is crucial for managing the production system. For RA, which controls the entire production system, the size of its state space is given by Eq. (20):

$$4 + 5 \times \sum_{l=1}^L \sum_{s=1}^S M_{l,s} \quad (20)$$

To manage the dynamic nature of job arrivals and departures, a state space that adapts to varying job line lengths and captures essential job information is required. A stable state space is constructed using abstracted data, which encapsulates job sequencing and system updates. Several features are segregated into channels based on their type and magnitude, with detailed specifications outlined in Table 1.

4.3 Action Representation

To effectively resolve dynamic combinatorial complexities, the pre-allocation event of each workstation is defined as an operational decision point. The decoupled action space balances real-time cooperative interactions between the Routing Agent (RA) and Sequencing Agent (SA) through specific heuristic combinations. Upon job arrival, the RA framework dynamically synchronizes machine selection via five strategic allocation principles: (1) Earliest Available Machine

to balance immediate facility availability; (2) Minimum Cumulative Working Time to even out cumulative asset workloads; (3) Minimum Setup Time and (4) Minimum Switching Cost to directly suppress successive setup changes and transition-dependent financial constraints; and (5) Random Selection to stochastically sample candidate pools during baseline exploration.

Table 1 State-space features and corresponding representations for the Sequencing Agent (SA)

Channel	Feature	Representation
Number	Jobs in the buffer of workstation on the line	$ JQ $
	Jobs groups in the buffer of assembly line	$ GQ $
Processing type	Second-level attributes of jobs to be selected	AB_e^2
	Operations of machines to be processed	$O_{l,s,e}$
Time	Arrival time of job to be processed on the line	$at_{i,l,s}$
	Processing time of job to be processed on the line	$pt_{i,l,s,e}$
	Remaining processing time of job to be processed on the line	$rpt_{i,l}$
	Slack time of job to be processed on the line	SK_i
	Changing setup time of job to be processed on the line	$ct_{l,s}^{u,v}$

Concurrently, when a facility transitions to an idle state, the SA framework governs local queue processing and product attribute allocation using eight priority rules: (1) Earliest arrival time to maintain strict queue entry sequences; (2) random selection for neutral task sampling; (3) minimum processing time and (4) minimum remaining processing time to optimize localized throughput and mitigate total downstream line residence bottlenecks; (5) minimum setup time and (6) minimum switching cost to restrict immediate switching energy and cost penalties per transition; (7) minimum slack time to prioritize urgent tasks utilizing dynamic delivery tracking indices; and (8) maximum waiting time to actively prevent line resource starvation and material shortages within the assembly buffer.

4.4 Surrogate Reward Function

This section constructs a dual-objective surrogate reward function driven simultaneously by real-time order slack times and operational setup switching energy costs to mitigate the credit assignment problem in multi-agent environments and balance volatile manufacturing objectives (Jing et al., 2024; Wei et al., 2022).

First, the reward function must balance minimizing switching costs with meeting order completion goals. In situations with many jobs nearing completion and varying types, too much focus on attribute consistency can lead to delays for other products. This process solely focuses on the consistency of job assignment attributes, overlooking criticality and urgency. To improve performance, rewards should be aligned with both job completion goals and switching costs. Additionally, rewards should reflect overall system performance, not just individual steps, to solve the credit assignment problem and guide agents toward the final production goals.

Maintaining on-time delivery for all jobs is challenging, particularly for diverse production orders with similar due dates. Therefore, quantifying the extent of job completion delays or earliness and incorporating them into rewards is a more rational approach. For job J_i and job group J_g , using a sigmoid function to transform their slack times SK_i and SK_g into a common

key factor $F_{i,g}$ that represents job or job group urgency helps mitigate outliers (extremely large or small values).

$$F_{i,g} = 1 - \frac{\frac{SK_i}{|SK_i| + \beta} + \frac{SK_g}{|SK_g| + \beta}}{2} \quad (21)$$

$F_i \in (0, 2)$, where SK_i and SK_g represent the slack times for job i and job group g , respectively; β is a user-specified factor used to adjust the curvature of the sigmoid function; as the value of β increases, the sigmoid function becomes flatter and smoother. Based on the considerations above, we propose two reward functions driven by slack time and changing setup energy cost.

For RA, the reward function evaluates all machine states, including availability and cumulative operational information, to assign jobs that minimize the total accumulation of tardiness, earliness, and switching costs. When job J_i arrives at workstation s on line l and completes processing operation $O_e^{l,s}$ on machine m corresponding to second-level attribute AB_e^2 , the estimated slack time $ET(SK_i^{t+1})$ and estimated changing setup energy cost $ET(CC_i^{t+1})$ at time step $t + 1$ are computed.

$$ET(SK_i^{t+1}) = \frac{\sum_{b=1}^{PT} E_b}{|\{PT_b\}|} - t_{now} - \sum_{s=s_{now}+1}^S \frac{\sum_{e=1}^{AB_{l,s}^2} pt_{i,l,s,m,e}}{|\{AB_{l,s}^2\}|} - \frac{\sum_{b=1}^{PT} pt_{b,g}}{|\{PT_B\}|} \quad (22)$$

$$ET(CC_i^{t+1}) = \sum_{s=s_{now}+1}^S \left(\sum_{\substack{i=1, j=1 \\ i \neq j}}^J nb_{i,j} \times \frac{\sum_{u=1, v=1}^{AB_{l,s}^2} cc_{i,j}^{u,v}}{|\{AB_{l,s}^2\}|^2} \right) \quad (23)$$

$$SK_i^{t+1} = E_{b,g} - t_{now} - \sum_{s=s_{now}+1}^S pt_{i,l,s,m,e} - pt_{b,g} \quad (24)$$

$$CC_i^{t+1} = \sum_{s=s_{now}+1}^S \left(\sum_{\substack{i=1, j=1 \\ i \neq j}}^J nb_{i,j} \times cc_{i,j}^{u,v} \right) \quad (25)$$

The reward function is computed as a weighted sum of the actual slack time gain/loss compared to the estimated slack time and the actual switching cost gain/loss compared to the estimated switching cost adjusted with the operational key factor. It can be derived as follows:

$$r(t) = \omega F_i \times (SK_i^{t+1} - ET(SK_i^{t+1})) + (1 - \omega) (CC_i^{t+1} - ET(CC_i^{t+1})) \quad (26)$$

where ω is the weight factor of the time factor.

The objective of SA is to prioritize job sequencing to minimize long-term cumulative tardiness, earliness, and switching costs. Rewards are derived from the performance impacted by queuing: when SA on machine m at workstation s on line l selects job J_i from its queue $JQ_{l,s,m}$ and assigns it under second-level attribute AB_e^2 belonging to $AB_{l,s}^1$, it does so at the expense of extending the queue time for other jobs and increases switching costs.

The estimated wait time for each queued job at all workstations can be determined as follows:

$$ET(wt_i) = \sum_s \sum_{j=1}^J pt_{j,l,s,m,e} \times nb_{i,j} + pt_{b,g} \quad (27)$$

The actual wait time for each queued job in front of all workstations can be expressed as

$$wt_i = \sum_s^S wt_{i,l,s} + wt_{b,g} \quad (28)$$

Finally, the reward function is as follows:

$$r(t) = \omega F_i \times (wt_i - ET(wt_i)) + (1 - \omega) (CC_i^{t+1} - ET(CC_i^{t+1})) \quad (29)$$

4.5 Asynchronous Transition

We employ an asynchronous transition method for optimizing rewards based on operation completion rather than fixed time steps. The transition starts at the decision point at time step t and is predicated on the completion of the operation at time step $t + 1$ (Figure 3). The time step $t + 1$ does not necessarily coincide with the next decision point. During this asynchronous leapfrogging cycle, as mapped out in the decoupled linear timelines of Figure 4, the RA initiates the facility allocation to trigger the machine setup stage, while the SA dynamically synchronizes buffer states and executes queue processing actions. The surrogate reward is routed back via an asynchronous loop through this cooperative timeline to coordinate local execution and achieve global optimization.

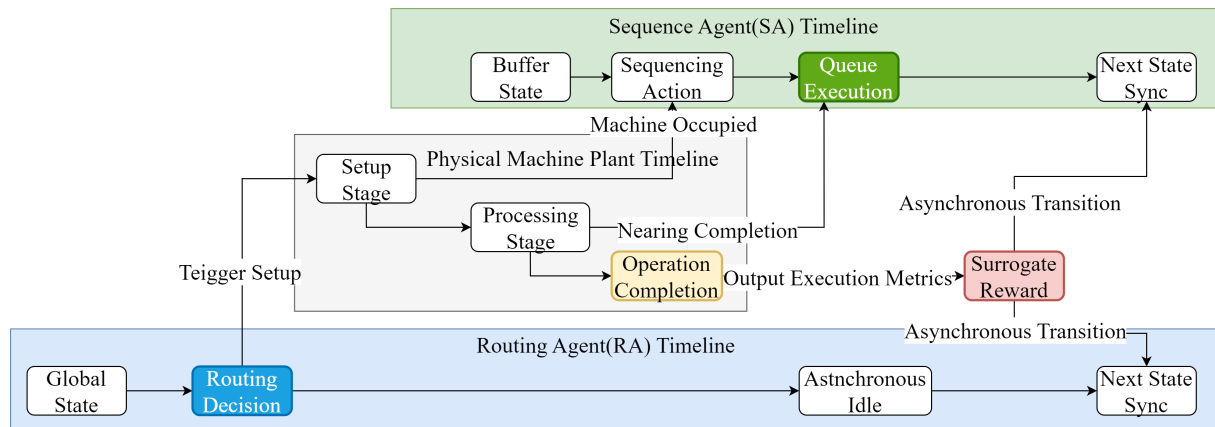


Figure 4 Asynchronous multi-agent state transition and decoupling control timeline

4.6 Parameters and Training

Agents with similar scopes share neural network parameters and experience repositories to reduce training complexity and foster cooperative behaviors (Wen et al., 2021). The remaining DRL hyperparameters are summarized in Table 2.

The production system simulation and the artificial neural network were developed using the Simpy and PyTorch Python libraries, respectively. Training was conducted in PyCharm on a Windows 11 system running on a PC equipped with a graphics processing unit (GPU), 11th Gen Intel(R) Core processor, and 16.0 GB RAM.

5. Case Study

A numerical experiment is conducted using simulation data derived from a real-world smart home manufacturing facility to evaluate the performance of the proposed DDQN algorithm. Section 5.1 details the test scenarios and problem instances, including the parameter settings' rationale. Section 5.2 validates the effectiveness of the algorithm by comparing its results with the exact solutions obtained from the MILP model currently used by the factory. Finally, Section 5.3 presents a comparative analysis with three other benchmark algorithms across various

performance indicators, demonstrating the proposed method's superiority in handling complex scheduling tasks.

Table 2 Parameters for the DRL and ANN

Category	Parameter	RA	SA
DRL related	Discount factor (γ)	0.7	0.7
	Exploration rate (ϵ)	0.35 to 0.2	0.35 to 0.2
	Minibatch size	128	64
	Replay memory size	512	256
	Iterations	500	500
Hyperparameters of the ANN	Input size	7	16
	Output size	1	2
	Channels	1	3
	Hidden layers	$16 \times 16 \times 16 \times 8 \times 8$	$48 \times 36 \times 36 \times 24 \times 24 \times 12$

5.1 Basic Scenario and Information

Figure 5 shows the structurally mapped physical topology and queue-routing flows of the series-parallel manufacturing workshop. The workshop features multiple parallel component lines equipped with dedicated buffers and identical parallel machines executing dyeing, cutting, stamping, and air rod installation operations, which sequentially merge into a centralized final assembly line where color-coded geometric shapes represent distinct product attributes.

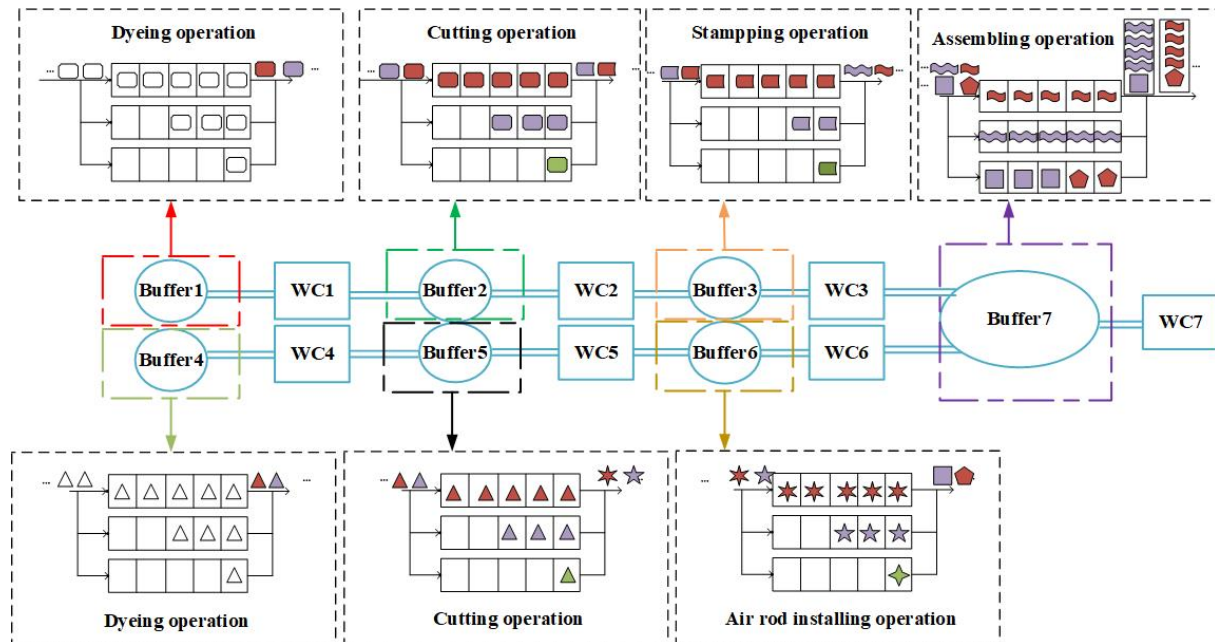


Figure 5 Physical workshop topology and layout of production flow

This experiment selects 10 product types: PT_1 ($NP_1=16$), PT_2 ($NP_2=18$), PT_3 ($NP_3=20$), PT_4 ($NP_4=44$), PT_5 ($NP_5=48$), PT_6 ($NP_6=50$), PT_7 ($NP_7=68$), PT_8 ($NP_8=88$), PT_9 ($NP_9=92$), PT_{10} ($NP_{10}=100$), forming 50 distinct test instance combinations derived from a real-world smart home manufacturing facility. Processing details for each product are listed in Table 3. Categorized by production task quantity, instances are divided into three scales: small

scale (10 instances with $\sum NP \leq 100$), medium scale (20 instances with $100 < \sum NP \leq 200$), and large scale (20 instances with $\sum NP > 200$). Small-scale instances provide a baseline against mathematical optima, whereas medium- and large-scale instances assess computational efficiency under high combinatorial complexity. Following standard DRL benchmarking protocols, each test instance was independently executed 10 times to mitigate variance from neural network initializations and policy exploration, suppressing stochastic outliers and verifying algorithm stability in consistently minimizing tardiness and switching costs.

Table 3 Processing time for all product types (units: min)

Product type	The work center index						
	1	2	3	4	5	6	7
1	32	55	33	38	111	40	55
2	32	75	44	38	122	55	55
3	32	90	55	38	133	60	55
4	32	75	33	38	122	40	55
5	32	90	55	38	133	60	55
6	32	55	55	38	111	60	55
7	32	75	33	38	122	40	55
8	32	55	44	38	111	55	55
9	32	75	55	38	122	60	55
10	32	55	55	38	111	60	55

5.2 Comparison of Scheduling Effects

For each instance scale, one representative case was selected to evaluate the proposed DDQN algorithm against exact solutions from the plant's current MILP model (solved via Gurobi). Both methods were independently executed 10 times per case. A comparative analysis was conducted focusing on two key performance indicators: total tardiness (min) and switching costs (USD). Figure 6 illustrates these outcomes across ten runs for all three scales.

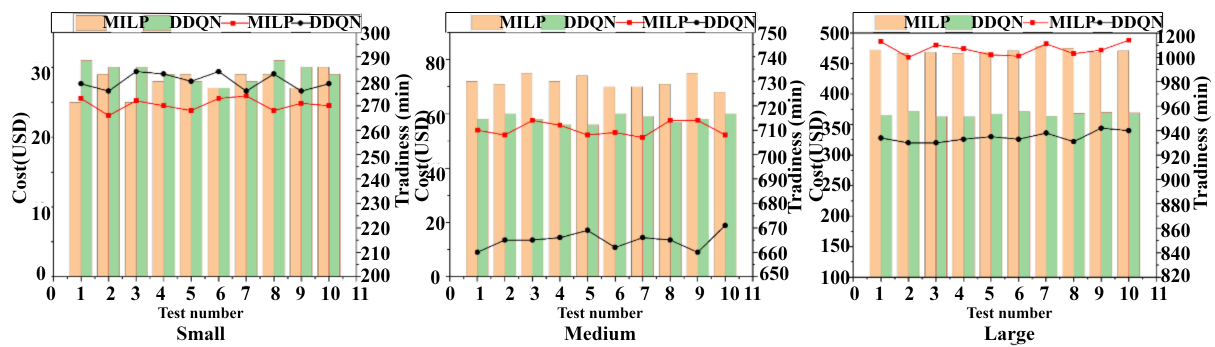


Figure 6 Comparison of tardiness and switching costs between MILP and DDQN

From the computational results, it can be concluded that productivity was enhanced and operational costs were reduced by the DDQN method. Consistent performance across runs demonstrated the stability of both the algorithm and the model. Although the optimization advantage was less pronounced in small-scale problems, its effectiveness grew significantly as the problem scale expanded. Specifically, in solving large-scale instances, switching costs were reduced by 22.4% and delay time was shortened by 6.8% compared to the MILP-based exact solution.

5.3 Algorithm Performance Comparison

Three benchmark algorithms across different scales are selected for comparison to evaluate the proposed DDQN: the foundational DQN framework (Van Hasselt et al., 2016), the highly applicable NSGA-II assembly scheduler (Yuan et al., 2021), and the recently proposed Black-winged Kite Optimization Algorithm (BKA) (Wang et al., 2024). Each algorithm was independently executed 20 times across 50 instances to analyze the cumulative rewards, total tardiness, and total setup costs. Three standard metrics are introduced to rigorously assess solution quality under multi-objective trade-offs where direct comparison is infeasible: the number of Pareto solutions (NS), inverted generational distance (IGD), and hypervolume (HV).

5.3.1 Rewarding Performance

Figure 7 (Supplementary Material) illustrates the cumulative reward training curves for four algorithms across small-, medium-, and large-scale instances, each represented by a single example. The reward per training round is the sum of all tasks. Results indicate that in the small-scale instance, DDQN, DQN, BKA, and NSGA-II algorithms converge to stable cumulative rewards after 61, 37, 91, and 98 rounds, respectively, with stable values of 201, 179, 185, and 175, respectively. In the medium-scale instance, the algorithms converge after 129, 103, 101, and 207 rounds, respectively, with stable values of 402, 358, 371, and 349, respectively. In the large-scale instance, convergence occurs after 87, 68, 93, and 257 rounds, with stable values of 608, 585, 597, and 581, respectively. A higher stable cumulative reward indicates the superior learning effectiveness of the algorithms. Overall, the DDQN-based scheduling algorithm achieved the highest cumulative reward.

An ablation study was conducted to evaluate individual policy contributions, as shown in Figure 8 (Supplementary Material). Compared with the complete framework (Proposed Full DDQN), deactivating the Double DQN component incurs non-negligible overestimation bias, leading to visible strategy oscillations. Eliminating hierarchical reward shaping severely retards early convergence due to sparse supervisory signals. Furthermore, without the decoupled multi-agent coordination or the event-driven asynchronous transition timeline, the systematic reward capacities drop substantially, mathematically validating the synergistic indispensability of all four mechanisms under complex SPPLSP configurations.

5.3.2 Quality of Pareto Solution

The statistical distribution of IGD, NS, and HV metrics is visualized across the composite boxplots in Figure 9 (Supplementary Material) based on 20 independent runs per instance, where the shaded horizontal bands represent the explicit 95% confidence intervals (95% CI) of the proposed DDQN.

For IGD values (left column), the proposed DDQN consistently achieves the lowest values with an expanding performance gap as the problem scale increases. For NS values (middle column), DDQN significantly outperforms competing algorithms in medium- and large-scale scenarios. Regarding HV comparisons (right column), DDQN demonstrates exceptional convergence and stability. Furthermore, a non-parametric Mann–Whitney U test confirms that these localized advancements are statistically significant ($p < 0.01$ or $p < 0.05$ markers), whereas Cliff's delta (δ) effect size analysis validates the proposed framework's global robustness.

The final distributed Pareto frontiers and multi-dimensional performance indices are structurally consolidated in Figure 10 (Supplementary Material) to verify this optimization efficacy more intuitively. As illustrated in Figure 10(a), the proposed cooperative DDQN securely dominates both the DQN and NSGA-II baselines, converging closest to the ideal multi-objective origin. Furthermore, the programmatic radar mapping shown in Figure 10(b) exhibits a systematic expansion toward the outermost perimeter of the DDQN. This dual-perspective visualization confirms that our decoupled coordination architecture consistently guarantees global scheduling robustness and superior convergence characteristics across both tardiness and energy-saving

dimensions within the SPPLSP workshop configuration.

6. Conclusion

This paper presents a DDQN-based deep reinforcement learning approach to solve the series-parallel production line sequencing problem (SPPLSP). The main contributions and findings are as follows: (1) A mixed-integer linear programming (MILP) model is formulated under serial-parallel synchronization and multi-part assembly constraints, simultaneously minimizing completion tardiness and sequence-dependent setup energy costs. (2) A hierarchical multi-agent DDQN framework integrated with dynamic reward shaping is developed, effectively optimizing cooperation between routing and sequencing agents while accelerating policy convergence. (3) Industrial case studies demonstrate that the proposed DDQN algorithm surpasses the plant's existing MILP solver in operational efficiency and cost savings, while showing superior convergence and Pareto solution quality over state-of-the-art benchmarks. (4) Future research will incorporate realistic constraints such as limited buffer capacities and stochastic material supply, alongside deep transfer learning and hybrid algorithms to enhance policy generalization. Future extensions should investigate practical production constraints, including limited buffer capacities and stochastic part supply timelines. Incorporating deep transfer learning across adjacent manufacturing environments represents a promising trajectory to enhance policy generalization, alleviate computational burdens, and reduce data dependencies. Finally, developing hybrid meta-heuristic reinforcement learning frameworks could further elevate global optimization efficacy.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT Authorship Contribution Statement

Zhijing Zhou: Writing, original draft, Investigation. Jiajia Wang: Investigation. Gang Yin: Review & editing. Jiachen Zhou: Formal analysis. Yongxiang Zhang: Data curation. Jian Guo: Review & editing, Formal analysis, Supervision. Junhao Zhang: Review & editing, Supervision.

Data Availability Statement

Data will be made available on request.

Acknowledgements

This work was supported by the Wenzhou Science and Technology Major Project (Grant No. ZG2024005). In addition, we would like to thank all the colleagues and anonymous reviewers who helped to improve the paper.

References

- Alla, K. R., & Thangarasu, G. (2023). Supply chain management of drug products in blockchain using reinforcement learning. *International Journal of Technology*, 14(6), 1256–1265. <https://doi.org/10.14716/ijtech.v14i6.6626>
- Azadeh, A., Shoja, B. M., Ghanei, S., & Sheikhalishahi, M. (2015). A multi-objective optimization problem for multi-state series-parallel systems: A two-stage flow-shop manufacturing system. *Reliability Engineering & System Safety*, 136, 62–74. <https://doi.org/10.1016/j.res.2014.11.009>

- Chen, C., Xia, B., Zhou, B., & Xi, L. (2015). A reinforcement learning based approach for a multiple-load carrier scheduling problem. *Journal of Intelligent Manufacturing*, 26, 1233–1245. <https://doi.org/10.1007/s10845-013-0852-9>
- Du, Y., Li, J., Li, C., & Duan, P. (2024). A reinforcement learning approach for flexible job shop scheduling problem with crane transportation and setup times. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4), 5695–5709. <https://doi.org/10.1109/TNNLS.2022.3208942>
- Farouk, M. M., Chin, C. G., & Roslee, M. (2026). Hybrid reinforcement learning-enabled scheduler for 5G burst traffic. *International Journal of Technology*, 17(3), 866–882. <https://doi.org/10.14716/ijtech.v17i3.8208>
- Fuqing, Z., & Yu, J. (2026). A multi-agent reinforcement learning with multi-task learning and graph edge attention networks for dynamic flexible job shop scheduling. *Applied Soft Computing*, 187, 114277. <https://doi.org/10.1016/j.asoc.2025.114277>
- Gao, K., Huang, Y., Sadollah, A., & Wang, A. (2020). A review of energy-efficient scheduling in intelligent production systems. *Complex & Intelligent Systems*, 6, 237–249. <https://doi.org/10.1007/s40747-019-00122-6>
- Gholizadeh, H., Chaleshigar, M., & Fazlollahtabar, H. (2022). Robust optimization of uncertainty-based preventive maintenance model for scheduling series–parallel production systems (real case: Disposable appliances production). *ISA Transactions*, 128, 54–67. <https://doi.org/10.1016/j.isatra.2021.11.041>
- Han, X., Liu, H., Sun, F., & Zhang, X. (2019). Active object detection with multistep action prediction using deep q-network. *IEEE Transactions on Industrial Informatics*, 15(6), 3723–3731. <https://doi.org/10.1109/TII.2019.2890849>
- Hu, H., Jia, X., He, Q., Fu, S., & Liu, K. (2020). Deep reinforcement learning based AGVs real-time scheduling with mixed rule for flexible shop floor in industry 4.0. *Computers & Industrial Engineering*, 149, 106749. <https://doi.org/10.1016/j.cie.2020.106749>
- Jing, X., Yao, X., Liu, M., & Zhou, J. (2024). Multi-agent reinforcement learning based on graph convolutional network for flexible job shop scheduling. *Journal of Intelligent Manufacturing*, 35(1), 75–93. <https://doi.org/10.1007/s10845-022-02037-5>
- Kayhan, B. M., & Yildiz, G. (2023). Reinforcement learning applications to machine scheduling problems: A comprehensive literature review. *Journal of Intelligent Manufacturing*, 34(3), 905–929. <https://doi.org/10.1007/s10845-021-01847-3>
- Khadivi, M., Charter, T., Yaghoubi, M., Jalayer, M., Ahang, M., Shojaeinasab, A., & Najjarian, H. (2025). Deep reinforcement learning for machine scheduling: Methodology, the state-of-the-art, and future directions. *Computers & Industrial Engineering*, 200, 110856. <https://doi.org/10.1016/j.cie.2025.110856>
- Li, L., Yang, X., Yang, S., & Xu, X. (2023). Optimization of oxygen system scheduling in hybrid action space based on deep reinforcement learning. *Computers & Chemical Engineering*, 171, 108168. <https://doi.org/10.1016/j.compchemeng.2023.108168>
- Li, Y. F., & Peng, R. (2014). Availability modeling and optimization of dynamic multi-state series–parallel systems with random reconfiguration. *Reliability Engineering & System Safety*, 127, 47–57. <https://doi.org/10.1016/j.ress.2014.03.005>
- Liu, X., Yang, X., & Lei, M. (2021). Optimisation of mixed-model assembly line balancing problem under uncertain demand. *Journal of Manufacturing Systems*, 59, 214–227. <https://doi.org/10.1016/j.jmsy.2021.02.019>
- Mao, H., Liu, Z., & Qiu, C. (2023). Adaptive disassembly sequence planning for VR maintenance training via deep reinforcement learning. *The International Journal of Advanced Manufacturing Technology*, 124, 3039–3048. <https://doi.org/10.1007/s00170-021-08290-x>
- Mosadegh, H., Fatemi Ghomi, S. M. T., & Süer, G. A. (2017). Heuristic approaches for mixed-model sequencing problem with stochastic processing times. *International Journal of Production Research*, 55(10), 2857–2880. <https://doi.org/10.1080/00207543.2016.1223897>

- Mosadegh, H., Ghomi, S. M. T. F., & Süer, G. A. (2020). Stochastic mixed-model assembly line sequencing problem: Mathematical modeling and q-learning based simulated annealing hyper-heuristics. *European Journal of Operational Research*, 282(2), 530–544. <https://doi.org/10.1016/j.ejor.2019.09.021>
- Nourelfath, M., & Yalaoui, F. (2012). Integrated load distribution and production planning in series-parallel multi-state systems with failure rate depending on load. *Reliability Engineering & System Safety*, 106, 138–145. <https://doi.org/10.1016/j.ress.2012.06.006>
- Paeng, B., Park, I. B., & Park, J. (2021). Deep reinforcement learning for minimizing tardiness in parallel machine scheduling with sequence dependent family setups. *IEEE Access*, 9, 101390–101401. <https://doi.org/10.1109/ACCESS.2021.3097254>
- Rai, R., Tiwari, M. K., Ivanov, D., & Dolgui, A. (2021). Machine learning in manufacturing and industry 4.0 applications. *International Journal of Production Research*, 59(16), 4773–4778. <https://doi.org/10.1109/OJIES.2024.3431240>
- Sikora, C. G. S. (2024). Balancing mixed-model assembly lines for random sequences. *European Journal of Operational Research*, 314(2), 597–611. <https://doi.org/10.1016/j.ejor.2023.10.008>
- Taube, F., & Minner, S. (2018). Resequencing mixed-model assembly lines with restoration to customer orders. *Omega*, 78, 99–111. <https://doi.org/10.1016/j.omega.2017.11.006>
- Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1). <https://doi.org/10.48550/arXiv.1509.06461>
- Wang, J., Wang, W., Hu, X., Qiu, L., & Zang, H. (2024). Black-winged kite algorithm: A nature-inspired meta-heuristic for solving benchmark functions and engineering problems. *Artificial Intelligence Review*, 57(4), 98. <https://doi.org/10.1007/s10462-024-10723-4>
- Wang, K., Li, X., Gao, L., Li, P., & Gupta, S. M. (2021). A genetic simulated annealing algorithm for parallel partial disassembly line balancing problem. *Applied Soft Computing*, 107, 107404. <https://doi.org/10.1016/j.asoc.2021.107404>
- Wei, Q., Li, Y., Zhang, J., & Wang, F. Y. (2022). VGN: Value decomposition with graph attention networks for multiagent reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 35(1), 182–195. <https://doi.org/10.1109/TNNLS.2022.3172572>
- Wen, G., Fu, J., Dai, P., & Zhou, J. (2021). DTDE: A new cooperative multi-agent reinforcement learning framework. *The Innovation*, 2(4). <https://doi.org/10.1016/j.xinn.2021.100162>
- Xi, S., Smith, J. M. G., Chen, Q., Mao, N., & Zhang, H. (2022). Simultaneous machine selection and buffer allocation in large unbalanced series-parallel production lines. *International Journal of Production Research*, 60(7), 2103–2125. <https://doi.org/10.1080/00207543.2021.1884306>
- Yamsani, N., & Reddy, P. C. (2026). EdgeSched-DQN: An intelligent deep reinforcement learning-based framework for optimized task scheduling in edge-cloud environments. *Array*, 29, 100645. <https://doi.org/10.1016/j.array.2025.100645>
- Yang, W., & Cheng, W. (2020). Modelling and solving mixed-model two-sided assembly line balancing problem with sequence-dependent setup time. *International Journal of Production Research*, 58(21), 6638–6659. <https://doi.org/10.1080/00207543.2019.1683255>
- Yelles-Chaouche, A. R., Gurevsky, E., Brahimi, N., & Dolgui, A. (2022). Minimizing task reassignments under balancing multi-product reconfigurable manufacturing lines. *Computers & Industrial Engineering*, 173, 108660. <https://doi.org/10.1016/j.cie.2022.108660>
- Yuan, M., Li, Y., Zhang, L., & Pei, F. (2021). Research on intelligent workshop resource scheduling method based on improved NSGA-II algorithm. *Robotics and Computer-Integrated Manufacturing*, 71, 102141. <https://doi.org/10.1016/j.rcim.2021.102141>
- Zhang, B., Xu, L., & Zhang, J. (2020a). A multi-objective cellular genetic algorithm for energy-oriented balancing and sequencing problem of mixed-model assembly line. *Journal of Cleaner Production*, 244, 118845. <https://doi.org/10.1016/j.jclepro.2019.118845>

- Zhang, K., He, F., Zhang, Z., Lin, X., & Li, M. (2020b). Multi-vehicle routing problems with soft time windows: A multi-agent reinforcement learning approach. *Transportation Research Part C: Emerging Technologies*, 121, 102861. <https://doi.org/10.1016/j.trc.2020.102861>
- Zhang, M., Liu, Y., Chen, H., & Cai, W. (2025). Double DQN-based efficient quality of service routing protocol in Internet of Underwater Things with mobile nodes. *Ad Hoc Networks*, 175, 103856. <https://doi.org/10.1016/j.adhoc.2025.103856>
- Zhao, C., & Melkote, S. N. (2024). Learning the manufacturing capabilities of machining and finishing processes using a deep neural network model. *Journal of Intelligent Manufacturing*, 35(4), 1845–1865. <https://doi.org/10.1007/s10845-023-02134-z>
- Zhao, M., Guo, X., Zhang, X., Fang, Y., & Ou, Y. (2020). Asp-w-drl: Assembly sequence planning for workpieces via a deep reinforcement learning approach. *Assembly Automation*, 40(1), 65–75. <https://doi.org/10.1108/AA-11-2018-0211>
- Zhou, B. H., & Shen, C. Y. (2018). Multi-objective optimization of material delivery for mixed model assembly lines with energy consideration. *Journal of Cleaner Production*, 192, 293–305. <https://doi.org/10.1016/j.jclepro.2018.04.251>