INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH
https://doi.org/10.1080/00207543.2025.2533528

# An improved deep Q-network for dynamic flexible job shop scheduling with limited maintenance resources

Wenchao Yi[a,b], Nanxing Chen[a], Yong Chen[a] and Zhi Pei[a]

[a]College of Mechanical Engineering, Zhejiang University of Technology, Hangzhou, People's Republic of China; [b]State Key Laboratory of Intelligent Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan, People's Republic of China

**ABSTRACT**
This study addresses the Dynamic Flexible Job Shop Scheduling Problem with Limited Maintenance Resources (DFJSP-LMR), a critical challenge in modern manufacturing systems. Traditional Flexible Job Shop Scheduling (FJSP) models often fail to account for machine deterioration, maintenance constraints, and dynamic events such as machine breakdowns and urgent tasks. To bridge this gap, we propose a novel mathematical model that integrates preventive maintenance (PM) and limited maintenance resources into the scheduling framework, aiming to minimise the makespan. The model considers machine reliability, maintenance activities, and resource constraints, reflecting real-world production environments. To solve this NP-hard problem, we introduce a hybrid algorithm combining Deep Q-Network (DQN) with a local search (LS) mechanism, enhanced by a Sliding Time Window Algorithm (STW) to manage maintenance resource allocation efficiently. The proposed DQN-LS algorithm is validated through extensive computational experiments, demonstrating superior performance compared to traditional scheduling rules and DQN without LS. The results highlight the algorithm's effectiveness in optimising makespan while balancing maintenance and production scheduling, offering valuable insights for both theoretical research and practical applications in dynamic manufacturing systems.

**ARTICLE HISTORY**
Received 25 December 2024
Accepted 21 June 2025

**KEYWORDS**
Dynamic flexible job shop scheduling problem; maintenance; deterioration; limited resources; deep reinforcement learning

## 1. Introduction

In recent decades, production scheduling has garnered considerable scholarly attention as a fundamental operational research challenge. Defined as the allocation of tasks and resources under constraints such as production sequences, process routes, and machine tools, scheduling aims to optimise a specified performance index. With the increasing complexity of manufacturing systems, shop-floor operations face elevated demands for timely resource allocation in response to machine deterioration, unexpected failures, and maintenance requirements. Consequently, incorporating maintenance planning into FJSP has emerged as a critical research direction, supporting more resilient and efficient production processes.

Contemporary manufacturing increasingly features multi-variety, small- and medium-sized batch production modes to address personalised and diversified customer demands. Early studies established methods and modelling frameworks for job-shop scheduling (Conway, Maxwell, and Miller 1967; Johnson 1954), later extended by meta-heuristic algorithms that tackled large-scale scheduling problems (Yamada and Nakano 1992). Comprehensive analyses were also conducted on the complexity of scheduling problems and associated algorithms (Lawler et al. 1993). As traditional non-flexible scheduling models began to fall short of modern industrial requirements, interest shifted toward FJSP. A variety of multi-objective and hybrid strategies for FJSP were proposed (J. Li, Pan, and Liang 2010; W. J. Xia and Wu 2005), while more recent efforts (Guo, Shi, and Wang 2024; Lei, Guo, and Wang 2024; Lei, Guo, and Zhao 2022) explored the potential of Deep Reinforcement Learning (DRL) to improve dynamic job scheduling performances (Cunha, Madureira, and Fonseca 2020). Dynamic events such as equipment breakdowns, emergency orders, and raw material shortages are increasingly common in production environments, underscoring the need for real-time rescheduling strategies and computerised reactive schedule repair tools (C. Wang and Jiang 2018; Ghaleb, Zolfagharinia, and Taghipour 2020; Tariq, Khan, and But 2024). However,

**CONTACT** Wenchao Yi yiwenchao@zjut.edu.cn College of Mechanical Engineering, Zhejiang University of Technology, Hangzhou 310000, People's Republic of China

This article has been corrected with minor changes. These changes do not impact the academic content of the article.

practical scheduling systems must also account for maintenance activities to address machine deterioration and failures, especially as these factors substantially affect workshop performance. Despite existing research, the role of limited maintenance resources – such as scarce maintenance staff or restricted repair time windows – requires further exploration.

In dynamic flexible job shop environments, maintaining normal production operations hinges on the condition of the machines. Over time, aging equipment inevitably degrades, resulting in decreased reliability, higher failure rates, and job deterioration (prolonged processing times for later tasks) (Da, Feng, and Pan 2016; Pakzad-Moghaddam, Mina, and Tavakkoli-Moghaddam 2014). PM is thus critical for preserving operational efficiency and extending equipment lifespans, yet many studies overlook its potential impact on scheduling outcomes (An et al. 2023). The integrated optimisation of maintenance and production scheduling is increasingly recognised for both its theoretical relevance and considerable practical value, as it addresses real-world constraints and resource limitations. Consequently, research on the Dynamic Flexible Job Shop Scheduling Problem (DFJSP) has grown rapidly. DFJSP extends the conventional FJSP by incorporating dynamic factors such as machine breakdowns, resource fluctuations, and urgent tasks. Incorporating limited maintenance resources within DFJSP further reflects actual production settings, as maintenance staff and scheduling windows are not unlimited. Hence, studying DFJSP that includes maintenance constraints is crucial for bridging the gap between theoretical development and industrial application.

Recent advancements in artificial intelligence have introduced DRL as a powerful method to tackle complex decision-making and combinatorial optimisation problems (Heess et al. 2015; Nair et al. 2015; Zeng, Li, and Bai 2022). By combining deep learning's capability for feature extraction with reinforcement learning's adaptive decision-making framework, DRL has proven useful for scheduling tasks under dynamic and uncertain conditions. While some research has started applying DRL to FJSP (Blundell et al. 2016; Wocker, Ostermeier, and Wanninger 2024), current studies primarily concentrate on conventional objectives and constraints. There remains significant potential to integrate machine deterioration, failure, and maintenance resources into DRL-based scheduling schemes to further enhance practical relevance. Accordingly, DFJSP involving PM and limited maintenance resources represents an NP-hard problem that more accurately mirrors real production environments, thereby delivering results that better inform operational decision-making.

In light of these considerations, the present work extends prior DFJSP models by encompassing limited maintenance resources and proposing a novel approach based on DQN. A LS algorithm is embedded to strengthen search capability in later stages, particularly when fine-tuning solutions. Numerous computational experiments and statistical analyses validate the proposed DQN-LS, demonstrating superior performance compared to conventional approaches. This study seeks to clarify the research significance of DFJSP with constrained maintenance resources while streamlining the literature review and situating DFJSP as a critical problem in evolving, intelligent manufacturing systems. By providing an enhanced problem model and an effective solution framework, this work offers theoretical insights and practical guidelines for real-world dynamic production environments.

The three contributions of this article are as follows:

(1) Previous studies on DFJSP with limited maintenance resources are few, this paper proposes a new solution.
(2) To solve the dynamic flexible job shop scheduling problem with maintenance based on DQN, the minimisation of makespan is taken as the target, and the $5 \times 3$ action space is designed as the compound rule.
(3) A hybrid algorithm based on DQN is designed, including STW algorithm and LS algorithm.

The remainder of this article is structured as follows: Section 2 reviews the current research landscape regarding DFJSP, maintenance, and DRL. Section 3 formulates the mathematical model for DFJSP-LMR. Subsequently, Section 4 introduces the proposed algorithm, DQN-LS algorithm. Section 5 outlines the parameter settings required for optimal algorithm performance and assesses the effectiveness of the proposed improvements. Additionally, the DQN-LS algorithm is compared with both DQN and established scheduling rules. Finally, Section 6 summarises the work conducted and discusses potential future research directions.

## 2. Literature review

This paper focuses on DFJSP, where dynamic events can be categorised into two primary types: resource-related events and job-related events. Resource-related events involve interruptions or changes in the availability or condition of production resources, such as machines, tools, or materials. These events often necessitate adjustments in resource allocation, job rearrangement, or resource substitution. Consequently, effective dynamic scheduling algorithms and techniques are critical for swiftly assessing the impact of these events on existing schedules, identifying necessary adjustments, and implementing them to minimise disruptions while optimising resource utilisation. In this field, the scheduling decisions must frequently adapt to real-time disruptions and uncertain operating conditions, as highlighted by Pinedo (2022). Scholars commonly distinguish three primary approaches to dynamic scheduling: completely reactive scheduling, pre-reactive scheduling, and robust scheduling. Completely reactive scheduling, as described by W. Xia and Wu (2005), continually updates the schedule in response to unexpected events, maintaining a high level of flexibility but often sacrificing long-term stability. In contrast, pre-reactive scheduling, which Gao, Gen, and Sun (2006) discuss, attempts to anticipate typical disruptions by introducing buffers or contingency plans into the initial schedule, thereby reducing the sensitivity of the solution to minor disturbances. Finally, robust scheduling aims to create solutions that remain as effective as possible under a wide range of potential disruptions without frequent modifications, balancing responsiveness and stability over longer horizons, as noted by B. Wang and Yang (2009). These complementary methods, each with distinct strengths and weaknesses, form the foundation for ongoing research into adaptive and intelligent scheduling mechanisms for dynamic FJSP, as illustrated by Kacem, Hammadi, and Borne (2002).

Recent studies on FJSP concentrate on addressing the complexities introduced by dynamic events, such as machine breakdowns. Al-Hinai and ElMekkawy (2011) introduced a hybrid Genetic Algorithm (hGA) designed to enhance stability in the event of unexpected machine failures. This method first generates an optimised schedule under deterministic conditions and then refines it to accommodate random breakdowns. Y. M. Wang, Yin, and Qin (2013) further improved scheduling quality by devising a unique chromosome encoding scheme within a standard Genetic Algorithm, significantly reducing time lost due to machine disruptions. Meanwhile, Gu et al. (2016) employed a multi-objective optimisation approach that ensures swift schedule recovery while maintaining high throughput. Nouiri, Bekrar, and Trentesaux (2018) integrated rescheduling and energy-efficiency goals through Particle Swarm Optimization (PSO), which adapts to real-time breakdowns while measuring energy consumption concurrently. In addition, Buddala and Mahapatra (2019) and Yang, Huang, and Wang (2020) focussed on robustness by utilising multi-stage or AI-based methods, including teaching-learning-based optimisation and extreme learning machines, demonstrating methodologies aimed at mitigating the negative impacts of disruptions. Soofi, Yazdani, and Amiri (2021) highlighted the effectiveness of fuzzy-stochastic techniques in managing dual-resource constraints, especially in large-scale problems. Wei, He, and Guo (2023) combined multi-objective migrating birds optimisation (MBO) with path relinking strategies to achieve a balance between efficiency and stability. Parallel to addressing abrupt machine failures, researchers have begun to consider machine deterioration as a gradual degradation phenomenon that affects processing times and reliability. Wu, Shen, and Li (2019) proposed an elitist quantum-inspired evolutionary algorithm that aims to reduce both makespan and energy consumption from a holistic perspective, addressing how fatigue or wear-and-tear factors can impede system performance. Azadeh, Goodarzi, and Kolaee (2019) examined uncertain processing and sequence-dependent setup times resulting from machine deterioration by integrating simulation, neural networks, and genetic algorithms to enhance schedule quality. Ghaleb, ElMekkawy, and Nourelfath (2021) adopted a more direct approach to real-time integrated production scheduling and maintenance planning, reporting both cost reductions and efficiency gains from the application of a modified hybrid genetic algorithm (mHGA). M. Li, Chang, and Liu (2024) developed a discrete artificial bee colony (DABC) method tailored to varying deterioration rates, which consistently outperformed other baselines regarding metrics such as average makespan and total processing time. Collectively, these studies underscore the necessity of considering the intricate interplay between dynamic events, machine deterioration, and energy consumption in effective DFJSP modelling. The complexities involved in real-time scheduling decisions highlight the need for advanced methodologies capable of dynamically adapting to changes, ensuring resilience and operational continuity in production environments. This approach not only improves productivity but also endows manufacturing systems with the agility required to thrive in increasingly competitive landscapes.

Since resource availability in FJSPs often hinges on maintenance activities, the integration of maintenance tasks directly into the scheduling process has emerged as a critical research direction. Moradi, Ghomi, and Zandieh (2010) proposed a hierarchical approach that flexibly addresses maintenance requirements, minimising system disruptions. Similar work by Da, Feng, and Pan (2016) modelled a single machine's time-dependent failure rate, demonstrating that strategically placed PM can significantly lower overall costs. X. Chen et al. (2020) explored multi-machine configurations through a non-dominated sorting genetic algorithm, emphasising the importance of multi-objective optimisation to account for maintenance costs and machine downtime. Moreover, Huang, Chang, and Arinez (2020) utilised DRL to capture probabilistic dependencies across multiple components, while Baykasoğlu and Madenoglu (2021) illustrated the advantages of a GRASP-based approach that jointly schedules production and maintenance events. Additional efforts by An et al. (2023) and Fang, Li, and Wang (2023) further exemplify the synergy of advanced optimisation and machine learning for robust scheduling, especially when accommodating both new jobs and maintenance on short notice. Many real-world manufacturing systems operate under constrained maintenance resources, such as limited access to skilled crews or critical spare parts. S. Wang and Yu (2010) addressed these constraints through a filtered beam search algorithm, which represents maintenance activities similarly to production jobs, ensuring transparency in resource needs during scheduling. Upasani et al. (2017) developed a distributed planning architecture that coordinates tasks among multiple machines while respecting resource limitations, a particularly beneficial approach for large-scale systems. An et al. (2021) further enhanced this discourse by factoring in heterogeneous repair crew skill levels, stressing the importance of assigning specific tasks to appropriate crews for maximal efficiency. Wocker, Ostermeier, and Wanninger (2024) concentrated on the limited availability of a single maintenance crew, developing a metaheuristic that balances preventive tasks and production flow with minimal disruption. This integrated perspective emphasises the interdependence among maintenance scheduling, job sequencing, and resource constraints within manufacturing systems. Approaches that concurrently address maintenance and job scheduling often outperform those that treat maintenance as a secondary concern. Recognizing the limitations of maintenance resources deepens the understanding of operational dynamics and necessitates the development of advanced coordination strategies to align maintenance with production needs. Current research advocates for dynamic decision-making frameworks for the real-time reallocation of maintenance tasks, which is crucial for alleviating scheduling bottlenecks arising from limited resources. While substantial progress has been made in addressing challenges related to constrained maintenance resources, a notable gap exists in the application of DRL methodologies to these issues.

DRL has emerged as a powerful tool for addressing high-dimensional scheduling problems, particularly those characterised by stochastic elements. Huang, Chang, and Arinez (2020) demonstrated the effectiveness of Double Deep Q-Network(DDQN) in PM within serial production lines, illustrating DRL's capability to autonomously learn and adapt 'group maintenance' or 'opportunistic maintenance' strategies. Liu, Chen, and Jiang (2020) developed an actor-critic DRL framework focussed on selective maintenance in multi-state systems across consecutive missions, reporting notable improvements in system availability. Expanding to multi-component scenarios, N. Zhang and Si (2020) introduced DRL-based condition-based maintenance (CBM), aligning individual component degradation profiles with overall cost-minimisation objectives. Further leveraging DRL, Lamprecht, Wurst, and Huber (2021) demonstrated that DDQN could surpass heuristic benchmarks in flow line systems, while Andriotis and Papakonstantinou (2021) incorporated multi-agent DRL within a partially observable Markov decision process (POMDP). The methodologies proposed by P. Zhang, Zhu, and Xie (2021) and J. Chen and Wang (2023) showcased how both model-based and state-based deep learning approaches can effectively reduce maintenance costs under uncertain or unknown degradation patterns. Lastly, C. Zhang, Li, and Coit (2023) presented an opportunistic maintenance model for load-sharing systems, employing proximal policy optimisation to autonomously manage large-scale tasks. Despite the versatility of DRL in maintenance scheduling, relatively few contributions have directly addressed minimising makespan within DFJSP contexts. Some pioneering works have demonstrated that DRL approaches can indirectly reduce makespan by minimising downtime. For instance, Rodríguez et al. (2022) employed a multi-agent DRL framework for predictive maintenance on parallel machines, effectively preventing failures through timely interventions that shorten overall processing times. Zhao and Smidts (2022) tackled system degradation under uncertain conditions, leveraging reinforcement learning to maximise cumulative rewards correlating with minimal job delays. Similarly, Feng and Li (2022) utilised a neural network-based DRL algorithm for predictive maintenance in multistage systems, achieving improved system throughput and reduced waiting times.

In conclusion, the integration of DRL into flexible job-shop systems signifies a significant advancement in enhancing operational performance, particularly in optimising makespan while addressing constraints related to limited maintenance resources. The fundamentally adaptive nature of DRL facilitates continuous learning and refinement of decision-making processes within dynamic manufacturing environments. The aim of this research is to incorporate the adaptive learning mechanisms inherent in DRL to formulate strategies that not only minimise makespan but also optimise maintenance resource allocation. This dual focus on operational efficiency and resource management is critical for improving the responsiveness and resilience of manufacturing processes. Ultimately, the convergence of DRL's adaptive potential with the challenges posed by limited maintenance resources highlights significant opportunities for enhancing production efficiency, contributing valuable insights for future research and practical applications within the manufacturing sector.

## 3. Mathematical model for DFJSP

### 3.1. Problem definition

Due to the constraints of limited maintenance resources, DFJSP-LMR differs from the traditional FJSP. Consequently, the algorithm design must account for concurrent maintenance activities. This section presents the mathematical model of DFJSP-LMR.

Before defining the problem, it is important to note that, similar to the traditional FJSP model, the DFJSP-LMR model operates under the following assumptions:

(1) Maintenance activities cannot commence on a machine until at least one job has been completed on it.
(2) The number of concurrent maintenance activities is limited to Q at any given time.
(3) During any period t on a single machine, only one maintenance activity can be performed.
(4) The duration of maintenance activities is predetermined and assumed to be constant.
(5) Neither jobs nor PM activities can be interrupted once they have started.
(6) The sequence of operations of each job is fixed.
(7) A job can only be processed on one machine at a time.
(8) Machines can only handle one job or one maintenance activity at a time.
(9) When processing or maintenance is required, the machine status is determined based on the start time.

Based on the above assumptions, the DFJSP-LMR model is defined as follows (Wocker, Ostermeier, and Wanninger 2024; X. Chen et al. 2020). There is a set of $n$ jobs $J = (J_1, J_2, J_3, \ldots, J_n)$ and a set of $m$ machines $M = M_1, M_2, \ldots, M_n$. Each job $J_i$ consists of a sequence of $n_i$ operations $O = O_{i,1}, O_{i,2}, \ldots, O_{i,ni}$, which must be processed sequentially according to specific precedence constraints. This results in two critical sub-problems:

(1) Machine Selection (MS) Problem: This involves selecting an appropriate machine $M_k$ for the operation $O_{i,j}$ from the set of available machines $M_a$.
(2) Operation Sequence (OS) Problem: Since at least one machine can process multiple jobs or operations, it is necessary to optimise the processing sequence of the operations on the machine.

Machine reliability is a pivotal factor in assessing a machine's capability to complete operations. As machines age, deterioration impacts reliability, necessitating the analysis of reliability and the scheduling of corresponding maintenance activities. PM is essential for determining the optimal start time and duration of maintenance required across different reliability intervals. The failure probability distributions for the machines, predominantly electromechanical equipment in our research, closely follow the Weibull distribution. The failure probability density function $f_k(T_k)$ of the machine $M_k$ at time $T_k$ is described as:

$$f_k(T_k) = \frac{\beta_k}{\eta_k} \cdot \left(\frac{T_k}{\eta_k}\right)^{\beta_k - 1} \cdot \exp\left[-\left(\frac{T_k}{\eta_k}\right)^{\beta_k}\right], \quad T_k \geq 0 \tag{1}$$

The reliability $R_k(T_k)$ of $M_k$ at time $T_k$ can be expressed as:

$$\begin{aligned} R_k(T_k) &= 1 - \int_0^{T_k} f_k(T_k)\,\mathrm{d}T_k \\ &= \exp\left[-\left(\frac{T_k}{\eta_k}\right)^{\beta_k}\right], \quad T_k \geq 0 \end{aligned} \tag{2}$$

Since $R_k(T_k)$ is related to the machine's effective age $(T_k)$, it is necessary to analyze $T_k$ before studying the reliability. In practice, a machine cannot be fully restored to an as-good-as-new state after maintenance. Therefore, we assume that the age reduction factor $p_k(p_u, p_v, p_w)$ of $M_k$ is fixed. If the effective age of $M_k$ before performing maintenance is $T_k$, and the effective age after performing maintenance is $T'_k$, the relationship between $T_k$ and $T'_k$ can be expressed as:

$$T'_k = (1 - p_k)T_k \tag{3}$$

In addition, the curves depicting the machine's effective age $T_k$ over calendar time $t$ are illustrated in Figure 1.

To address machine degradation phenomena, we establish a three-tier reliability classification system (area *I*, *II* and *III*) based on failure probability thresholds. As shown in Figure 2, each area corresponds to specific degradation characteristics, with $R_{II}$ and $R_{III}$ denoting the critical reliability thresholds for area *II* and *III* respectively. During the execution of operation $O_{i,j}$, real-time processing parameters are dynamically adjusted according to the machine's cumulative service time $(t)$, enabling

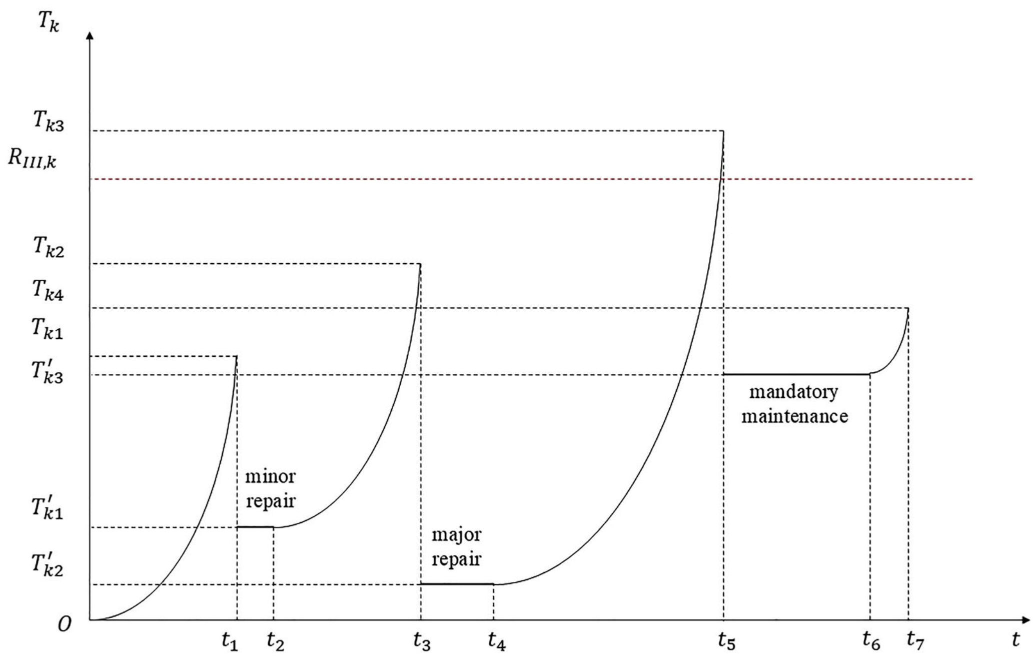


**Figure 1.** The curves of machine effective age after different maintenance.

continuous reliability assessment through function $R_k(t)$. This formulation generates three distinct operational regimes that systematically associate reliability metrics with equipment health states and corresponding maintenance protocols.

(1) If the reliability $R_k(t)$ falls in the $I$ area (normal area), no machine deterioration and maintenance activities are taken.
(2) If the reliability $R_k(t)$ falls in the $II$ area (deterioration area), when the machine is processed, the processing time will become longer because of degrading. There are two types of maintenance in this area: major repair and minor repair.
(3) If the reliability $R_k(t)$ falls in the $III$ area (mandatory maintenance area), the maintenance activity must be performed before $O_{i,j}$.

## 3.2. The mathematical model of DFJSP-LMR

Before building the mathematical model, it is essential to define the necessary notations and decision variables. Table 1 presents the notations used in the FJSP-LMR and the notations for the decision variables.

In this paper, the objective function is to minimise the maximum completion time of all jobs described by Equation (4).

$$\min C_{\max} = \min\left(\max_{i=1,\dots,m} C_i\right) \tag{4}$$

The constraints that need to be considered are outlined as follows:

$$t_{ijk} = \begin{cases} p_{ijk}, & a_{ijk} < R_{\mathrm{II},k} \\ p_{ijk} + \sigma_k\left(a_{ijk} - t_{\mathrm{II}}\right), & R_{\mathrm{II},k} \le a_{ijk} \\ & \quad \le R_{\mathrm{III},k}, \quad \forall\, i,j,k \end{cases} \tag{5}$$

Equation (5) represents the actual processing time $t_{ijk}$ of a job considering the machine deterioration effect, where $t_{II}$ denotes the time associated with machine $M_k$ at which $R_k(T_k) = R_{II,k}$.

$$E_{ijk} = S_{ijk} + x_{ijk} \times t_{ijk} + u_{ijk} \times t_{u,k} + v_{ijk} \times t_{v,k} + w_{ijk} \times t_{w,k} \tag{6}$$

$$E_{ijk} \ge S_{i(j-1)k} \tag{7}$$

Equations (6) and (7) represent the precedence constraints for the sequence of operations for each job.

$$\sum_{k=0}^{m} x_{ijk} = 1, \quad \forall\, i,j \tag{8}$$

Equation (8) represents the assignment of operations, indicating that each operation can be assigned to only one machine.

$$u_{ijk} + v_{ijk} + w_{ijk} \le Q(h) \tag{9}$$

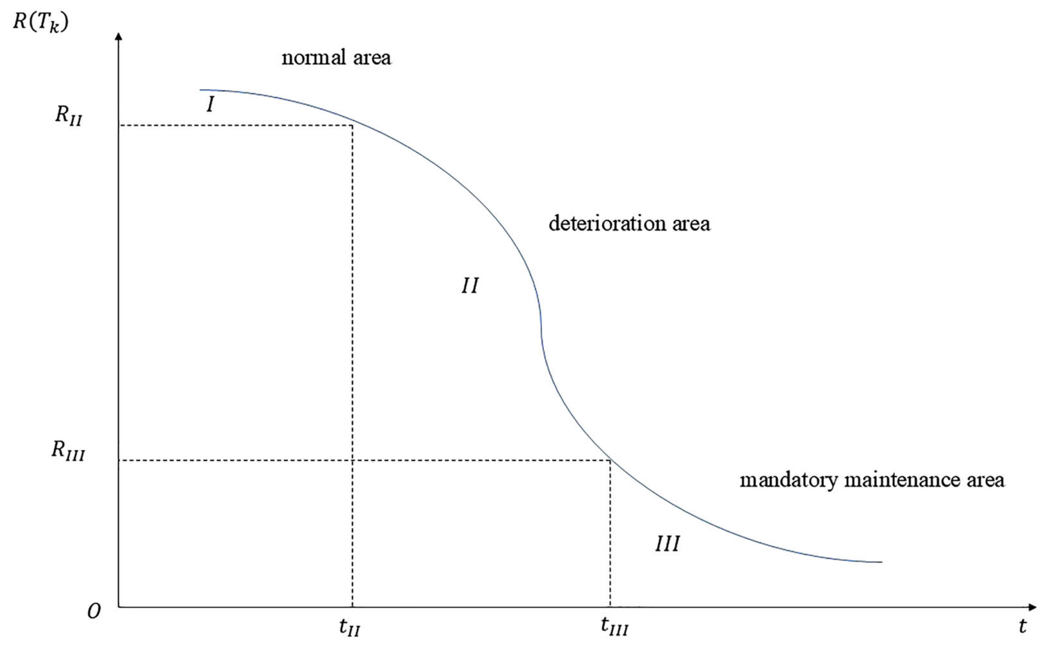


**Figure 2.** The division of areas within different reliability intervals.

Equation (9) indicates that the number of maintenance activities at any given time $h$ does not exceed Q.

$$S_{ijk} + t_{ijk} \leq S_{uvk} + N\left(1 - y_{ijuvk}\right), \quad \forall\, i, j, \quad \forall\, M_k \in M_{ij} \cap M_{uv} \tag{10}$$

$$E_{ijk} \leq S_{uvk} + N\left(1 - y_{ijuvk}\right), \quad \forall\, i, j, \quad \forall\, M_k \in M_{ij} \cap M_{uv} \tag{11}$$

Equations (10) and (11) describe the sequencing relationship on the machines, where $N$ represents a sufficiently large value. When $y_{ijuvk} = 1$, operation $O_{ij}$ is processed before operation $O_{uv}$ on machine $M_k$, meaning the start time of $O_{uv}$ occurs after the completion time of $O_{ij}$.

$$S_{ijk} \geq 0, \quad E_{ijk} \geq 0 \tag{12}$$

Equation (12) states that the start time and end time of each operation are both greater than zero, indicating that the job is available for processing at time zero.

$$p_{ijk} \geq 0 \tag{13}$$

Equation (13) indicates that the completion time of the job must be non-negative.

$$\{x_{ijk}, y_{ijuvk}, u_{ijk}, v_{ijk}, w_{ijk}\} \in \{0, 1\}, \quad \forall\, i, j, u, v, \quad \forall\, M_k \in M_{ij} \cap M_{uv} \tag{14}$$

## 4. DQN-LS algorithm

This section begins with the definition of state features. Subsequently, the proposed dispatching rules (actions) are introduced, followed by the definition of the reward function at each rescheduling point. The STW Algorithm, employed to address the constraints of limited maintenance resources, is then presented. Finally, a LS algorithm is described to prevent unnecessary maintenance.

### 4.1. Definition of state features

In the realm of reinforcement learning, the interplay between the agent, the environment, and the state forms a cohesive framework for decision-making and learning. The agent serves as the learner or decision-maker, actively engaging with the environment, which comprises the context or setting that responds to the agent's actions. As the agent interacts with this dynamic environment, it receives feedback in the form of rewards or penalties, essential for guiding its behaviour toward the achievement of specific goals. Simultaneously, the state represents the agent's current situation within the environment, encapsulating all relevant information necessary for making informed decisions. This state influences the agent's action selection, driving the learning process as the agent adapts its strategies based on the outcomes of its interactions. Thus, the interconnectedness of these components enables the agent to refine its decision-making capabilities through continual learning and adaptation within an interactive setting.

It is crucial that the states captured by the agent contain sufficient information for the agent to effectively learn optimal policies. In order to solve the problem

**Table 1.** Notations used in the FJSP-LMR and the decision variables.

| No. | Notation | Description |
|---|---|---|
| 1 | $i$ | Index of job, $i \in I, I = \{1, 2, \dots, m\}$ |
| 2 | $j$ | Index of operation, $j \in J, J = \{1, 2, \dots, m_i\}$ |
| 3 | $k$ | Index of machine, $k \in K, K = \{1, 2, \dots, n\}$ |
| 4 | $J_i$ | The $i$th job |
| 5 | $O_{ij}$ | The $j$th operation of $J_i$ |
| 6 | $M_k$ | The $k$th machine |
| 7 | $M_{ij}$ | The set of machines for processing $O_{ij}$, $M_{ij} \subset M$ |
| 8 | $C_{max}$ | The maximum completion time of all jobs |
| 9 | $C_i$ | The completion time of $J_i$ |
| 10 | $p_{ijk}$ | The expected processing time for $O_{ij}$ of $J_i$ on $M_k$ |
| 11 | $t_{ijk}$ | The actual processing time for $O_{ij}$ of $J_i$ on $M_k$ |
| 12 | $S_{ijk}$ | The start time of $O_{ij}$ on $M_k$ |
| 13 | $E_{ijk}$ | The end time of $O_{ij}$ on $M_k$ |
| 14 | $\sigma_k$ | The deterioration coefficient of $M_k$ |
| 15 | $Q$ | The maximum maintenance resources |
| 16 | $R_{II,k}$ | The reliability threshold of $M_k$ in the $II$ area |
| 17 | $R_{III,k}$ | The reliability threshold of $M_k$ in the $III$ area |
| 18 | $t_{u,k}$ | The time required for a minor repair on $M_k$ |
| 19 | $t_{v,k}$ | The time required for a major repair on $M_k$ |
| 20 | $t_{w,k}$ | The time required for a mandatory maintenance on $M_k$ |
| 21 | $p_u$ | The age reduction factor of minor repair |
| 22 | $p_v$ | The age reduction factor of major repair |
| 23 | $p_w$ | The age reduction factor of mandatory maintenance |
| 24 | $a_{ijk}$ | The machine age of $M_k$ before processing $O_{ij}$ |
| 25 | $\beta_k$ | The shape parameters of $M_k$ in the Weibull distribution |
| 26 | $\eta_k$ | The scale parameters of $M_k$ in the Weibull distribution |
| 27 | $x_{ijk}$ | 1: $O_{ij}$ of $J_i$ is processed on $M_k$<br>0: $O_{ij}$ of $J_i$ is not processed on $M_k$ |
| 28 | $y_{ijuvk}$ | 1: $O_{ij}$ is processed before $O_{uv}$ on machine $M_k$<br>0: $O_{ij}$ is processed after $O_{uv}$ on machine $M_k$ or $O_{ij}$ and $O_{uv}$ are not processed on the same machine |
| 29 | $u_{ijk}$ | 1: $M_k$ is performing a minor repair<br>0: $M_k$ is not performing a minor repair |
| 30 | $v_{ijk}$ | 1: $M_k$ is performing a major repair<br>0: $M_k$ is not performing a major repair |
| 31 | $w_{ijk}$ | 1: $M_k$ is performing a mandatory maintenance<br>0: $M_k$ is not performing a mandatory maintenance |

**Table 2.** Notations used in DQN.

| No. | Notation | Description |
|---|---|---|
| 1 | $m$ | Number of machines |
| 2 | $n$ | Total number of workpieces |
| 3 | $rj$ | Number of remaining unprocessed workpieces |
| 4 | $OP_i$ | Number of operations for workpieces |
| 5 | $op_i$ | Remaining unprocessed operations for unfinished workpieces |
| 6 | $p_{a,b}$ | Average processing time for the remaining unprocessed operations |
| 7 | $URM_k$ | Utilization rate of machine $k$ |
| 8 | $CRW_i$ | Completion rate of workpiece $i$ |
| 9 | $A_k$ | Current age of machine $k$ |
| 10 | $UC_{job}$ | A set of unfinished jobs |
| 11 | $ATM_k$ | Current available time of machine $k$ |

better, six state features are selected as the input of DQN.Before delving into the specifics of the state features, it is important to introduce some preliminary notations in Table 2.

Based on the notations defined above, at each rescheduling point $h$, the state features can be summarised as follows:

(1) Average remaining processing time of unfinished workpieces, denoted as $\mathrm{ART}_{\mathrm{ave}}(h)$, as described in Equation (15):

$$\mathrm{ART}_{\mathrm{ave}}(h) = \frac{\sum_{i=1}^{n} \sum_{op_i}^{OP_i} p_{i,op_i}}{rj} \tag{15}$$

(2) Average utilisation rate of machines, denoted as $\mathrm{URM}_{\mathrm{ave}}(h)$, as described in Equation (16):

$$\mathrm{URM}_{\mathrm{ave}}(h) = \frac{\sum_{k=1}^{m} \mathrm{URM}_k(h)}{m} \tag{16}$$

(3) Standard deviation of machine utilisation rate, denoted as $\mathrm{URM}_{\mathrm{std}}(h)$, as described in Equation (17):

$$\mathrm{URM}_{\mathrm{std}}(h) = \sqrt{\frac{\sum_{k=1}^{m} (\mathrm{URM}_k(h) - \mathrm{URM}_{\mathrm{ave}}(h))^2}{m}} \tag{17}$$

(4) Average completion rate of workpieces, denoted as $\mathrm{CRW}_{\mathrm{ave}}(h)$, as described in Equation (18):

$$\mathrm{CRW}_{\mathrm{ave}}(h) = \frac{\sum_{i=1}^{n} \mathrm{CRW}_i(h)}{n} \tag{18}$$

(5) Standard deviation of workpiece completion rate, denoted as $\mathrm{CRW}_{\mathrm{std}}(h)$, as described in Equation (19):

$$\mathrm{CRW}_{\mathrm{std}}(h) = \sqrt{\frac{\sum_{i=1}^{n} (\mathrm{CRW}_i(h) - \mathrm{CRW}_{\mathrm{ave}}(h))^2}{n}} \tag{19}$$

(6) Average machine age, denoted as $\mathrm{MA}_{\mathrm{ave}}(h)$, as described in Equation (20):

$$\mathrm{MA}_{\mathrm{ave}}(h) = \frac{\sum_{k=1}^{m} A_k(h)}{m} \tag{20}$$

## 4.2. The proposed dispatching rules

Actions and policies are crucial for the agent's decision-making in reinforcement learning. An action represents a choice made by the agent that affects the state of the environment, which can be either discrete, involving specific moves, or continuous, where parameters are fluidly adjusted. The agent's choice of action is guided by its policy, a strategy that determines which action to take in a given state and can be either deterministic, producing a consistent action, or stochastic, incorporating randomness in action selection. Together, actions and policies enable the agent to effectively navigate and adapt to the environment, facilitating the learning process through exploration and strategy evaluation.

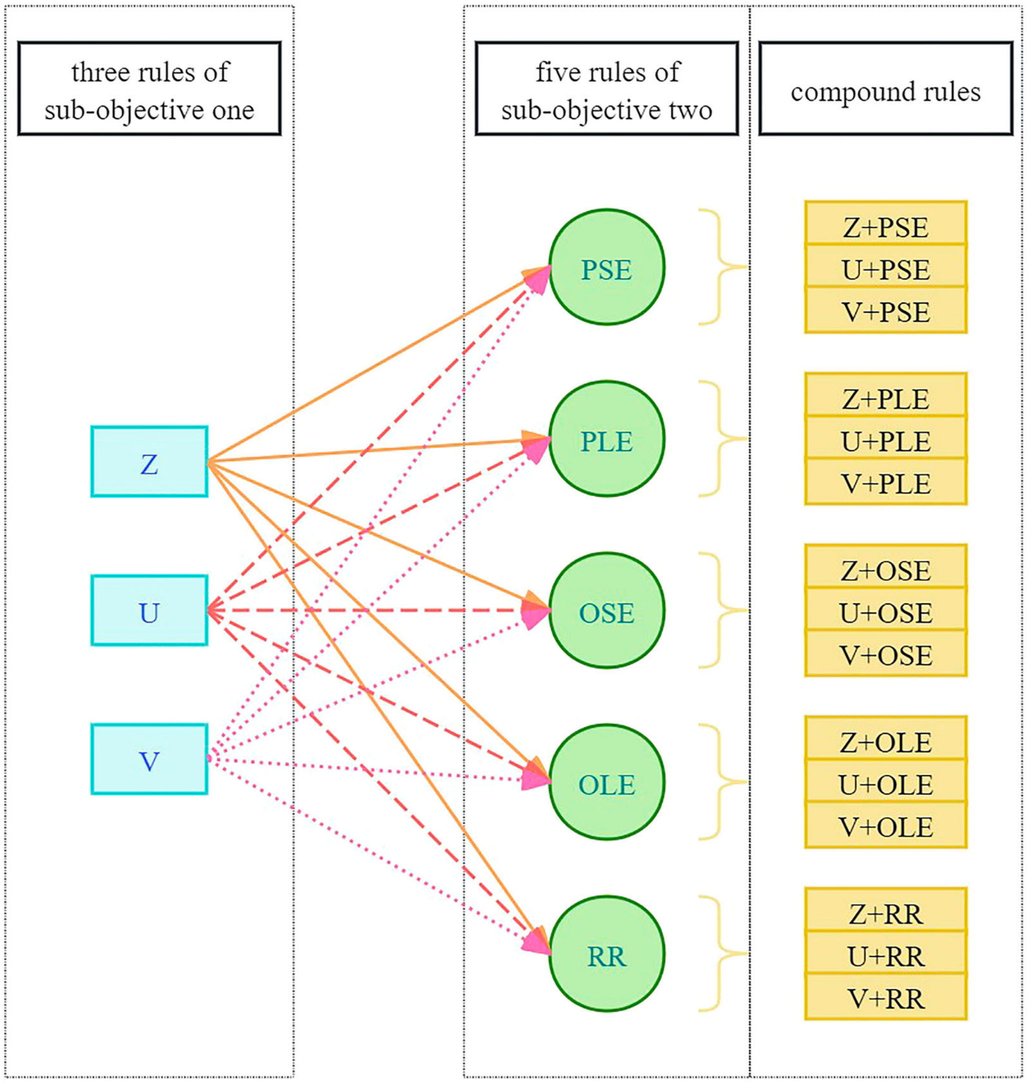


**Figure 3.** The compound scheduling rules.

Different scheduling rules yield different outcomes, and the strategic arrangement of these rules can significantly influence multi-objective performance. When applying a scheduling rule, two sub-objectives must be addressed. The first sub-objective is to determine whether maintenance is required, while the second sub-objective is to select the next job operation and assign it to the most suitable machine for processing. Taking these factors into account, this research proposes a combined rule design approach. Three scheduling rules are developed for sub-objective one, and five scheduling rules are formulated for sub-objective two. By combining the rules of sub-objective one with those of sub-objective two, a total of fifteen compound scheduling rules are generated, as illustrated in Figure 3.

When designing the rules for sub-objective one, it is necessary to first consider whether maintenance is required. There are three possible scenarios: no maintenance ($Z$), minor repair ($U$), and major repair ($V$). In the case of minor repair, the probability $P_u$ is 0.65 and the repair duration is 5. For major repair, the probability $P_v$ is 0.9 and the repair duration is 10. These values are fixed.

After determining the type of maintenance to perform, it is necessary to select both the workpiece and the machine. When designing the first scheduling rule for sub-objective two (PSE), we begin by defining a set of unfinished jobs, denoted as $UC_{job}$. From this set, for each job $J_i$ ($J_i \in \mathrm{UC_{job}}$), the job with the lowest average remaining processing time is selected for execution, as shown in Equation (21). The corresponding formula for the next operation number of $J_i$ is provided in Equation (22). Next, the parameter $ATM_k$ is defined as the current available time of each machine, and $M_{i,\mathrm{op}_i,k}$ represents the available machines for processing each job. The machine with the earliest available time is then selected, as illustrated in Equation (23). In summary, the PSE rule prioritizes selecting the job with the lowest average remaining processing time and assigning it to the earliest available machine.

$$i = \min_{i \in UC_{\text{job}}} \frac{\sum_{op_i}^{OP_i} p_{i,op_i}}{OP_i - op_i + 1}, \quad OP_i > op_i \tag{21}$$

$$op_i = op_i + 1 \tag{22}$$

$$k = \min\left(ATM_k \mid M_{i,op_i,k} \neq 0\right) \tag{23}$$

Similar to PSE, when designing the second rule for sub-objective two (PLE), the job with the longest average remaining processing time is selected for execution, as shown in Equation (24).

$$i = \max_{i \in UC_{\text{job}}} \frac{\sum_{op_i}^{OP_i} p_{i,op_i}}{OP_i - op_i + 1}, \quad OP_i > op_i \tag{24}$$

The third scheduling rule (OSE) and the fourth scheduling rule (OLE) for sub-objective two focus on operations. OSE prioritizes the workpiece with the most remaining operations, while OLE prioritizes the workpiece with fewest remaining operations. These rules are described in Equations

$$i = \max\left(OP_i - op_i + 1\right), \quad OP_i > op_i \tag{25}$$

$$i = \min\left(OP_i - op_i + 1\right), \quad OP_i > op_i \tag{26}$$

For PSE, OSE, and OLE, the subsequent steps follow the same procedure as outlined in Equations (22) and (23). To summarise, the PLE rule selects the job with the longest average remaining processing time and assigns it to the earliest available machine. The OSE rule prioritizes the job with the most remaining operations and assigns it to the earliest available machine, while the OLE rule selects the job with the fewest remaining operations and assigns it to the earliest available machine.

To prevent local optima, the fifth scheduling rule of sub-objective two (RR) is to randomly select a job and a machine. The formulas of RR are defined as Equations (27), (22) and (28).

$$i = \text{Randomly choose } i \text{ from } UC_{job} \tag{27}$$

$$k = \text{Randomly choose } k \text{ from } M_k, M_{i,\text{op}_i,k} \neq 0 \tag{28}$$

The compound scheduling rules (Figure 3) are formed by combining the scheduling rules for sub-objective one and sub-objective two, resulting in the following combinations: Z+ PSE, U+ PSE, V+ PSE, Z+ PLE, U+ PLE, V+ PLE, Z+ OSE, U+ OSE, V+ OSE, Z+ OLE, U+ OLE, V+ OLE, Z+RR, U+RR, V+RR.

## 4.3. Definition of rewards

The reward serves as a crucial feedback signal from the environment provided to the agent following an action. This signal quantitatively reflects the value of the action regarding the agent's goals, allowing the agent to evaluate its effectiveness. Rewards inform the learning process by reinforcing desirable actions that lead to positive outcomes while discouraging actions that yield negative feedback. Thus, rewards play a vital role in shaping the agent's strategy and enhancing its performance within the environment. The agent's objective is to maximise the total reward, making the formulation of the reward function a critical step in the development of an effective reinforcement learning algorithm (Ogunfowora and Najjaran 2023).

This research adopts the minimisation of makespan as the primary scheduling objective, as defined in Equation (4). Thus, the expectation of maximising the cumulative reward $G_t$ (the goal of the RL agent) aligns with the objective of minimising the makespan. Accordingly, the reward function is defined in Equation (29).

$$R(s_t, a_t) = \begin{cases} -\left(T(t+1) - T(t)\right) - 10, & \text{while } U \\ -\left(T(t+1) - T(t)\right) - 20, & \text{while } V \\ -\left(T(t+1) - T(t)\right), & \text{while } Z \end{cases} \tag{29}$$

In this function, $R(s_t, a_t)$ represents the immediate reward for taking action $a_t$ in state $s_t$, while $T(t)$ is the maximum end time across all machines at state $s_t$, and similarly, $T(t+1)$ at state $s_{t+1}$. This reward structure encourages the agent to choose operations with shorter processing times, filling idle machine time and avoiding unnecessary repairs, while also balancing the load across machines.

## 4.4. Sliding time window algorithm

To address the challenge of resource limitations during maintenance scheduling, this paper introduces the innovative the STW algorithm. The steps and details of the STW algorithm are outlined in Algorithm 1 (lines 1–23). The process mainly consists of the following three parts: sorting the maintenance list by the start time of maintenance (line 1), identifying time periods where resources are unavailable due to maintenance (lines 2–8), and inserting new maintenance into the available time slots (lines 9–23).

Additionally, the principle of STW is further illustrated in Figure 4. Primarily, the unavailable maintenance periods are ranked. When a new maintenance task arrives, it first attempts to insert the new maintenance

**Algorithm 1** Sliding Time Window Algorithm (STW)

```
Require:
    $Q = \{(s_1, e_1), (s_2, e_2), \ldots, (s_n, e_n)\}$: set of maintenance
    $M_k$: selected machine
    $T$: current maintenance time
    $q$: limited maintenance resource
    $CT_k$: available time of machine $k$
Ensure:
    maintenance start time
    maintenance end time
    Take $q = 2$ as example
1:  Sort $Q$ according to the first element of tuple $(s_i)$ :sorted $Q$
2:  for $i = 1$ to $n$ in sorted $Q$ do
3:      for $j = i + 1$ to $i + n$ in sorted $Q$ do
4:          if $s_i < e_i$ and $s_j > e_j$ then
5:              $\textit{list of overlap}.\text{append}(\max(s_i, s_j), \min(e_i, e_j))$
6:          end if
7:      end for
8:  end for
9:  Suppose $start\_time \Leftarrow CT_k$, $end\_time \Leftarrow CT_k + T$
10: if $overlap$ is empty or $start\_time > overlap[-1][1]$ or $end\_time < overlap[0][0]$ then
11:     $maintenance\ start\ time \Leftarrow start\_time$, $maintenance\ end\ time \Leftarrow end\_time$
12: else
13:     for $i = 1$ to $m$ in $overlap$ do
14:         if $start\_time > overlap[i][1]$ and $end\_time < overlap[i+1][0]$ then
15:             $maintenance\ start\ time \Leftarrow overlap[i][1]$
16:             $maintenance\ end\ time \Leftarrow maintenance\ start\ time + T$
17:             break
18:         else
19:             $maintenance\ start\ time \Leftarrow overlap[i+1][1]$
20:             $maintenance\ end\ time \Leftarrow maintenance\ start\ time + T$
21:         end if
22:     end for
23: end if
```

into position 1. If feasible, it is inserted; otherwise, it attempts to insert it into position 2, and so on. If the new maintenance cannot be inserted into any of the scheduled maintenance slots, it is scheduled at the end.

## 4.5. Local search algorithm

To minimise redundant maintenance and reduce the objective of minimising makespan, this paper proposes the LS Algorithm. The steps and details of LS are outlined in Algorithm 2 (lines 1–15). Specifically, optimisation is carried out in the following aspects: if machine k undergoes maintenance before scheduling an operation, the maintenance is removed; if the estimated machine age (focusing on the current maintenance($i$), the estimated machine age is calculated from the end of the previous maintenance to the start of the next maintenance) after removing the maintenance is less than the original machine age and is also less than $R_{III,k}$, the maintenance is deleted; otherwise, the maintenance should be retained.

## 5. Computational experiments

The algorithm used in this paper is DQN-LS and is implemented in Python 3.9 based on the PyTorch for simulation experiments on an Intel Core i7 Intel(R) Core (TM) i7-13700H CPU 2.40 GHz with 32 GB. The data used for the training model consists of randomly generated instances of FJSP, taking into account machine breakdowns and maintenance. The benchmark parameters and algorithm parameters are presented in Table 3.

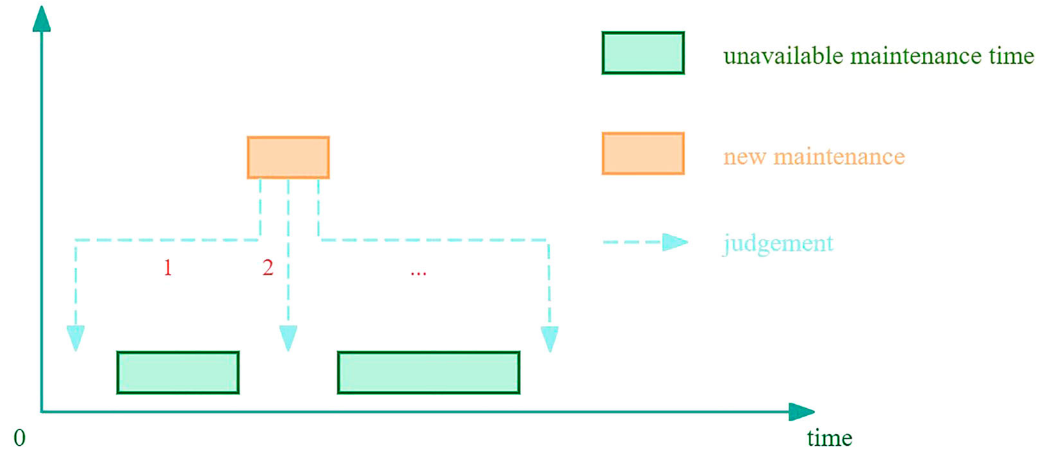


**Figure 4.** Diagram of Sliding Time Window Algorithm.

**Algorithm 2** Local Search Algorithm (LS)

```
Require:
    list of job processing (includes maintenance)
    list of maintenance
    $R_{III,k}$
Ensure:
    new list of job processing (after deleting some maintenance)
1:  for i in list of maintenance do
2:      Find maintenance(i) in list of job processing
3:      if nooperation before i then
4:          Delete maintenance(i) from list of job processing
5:      else
6:          if estimated machine age < original machine age(after deleting i, calculate range to $i + 1$ or i is the last one)
    then
7:              if estimated age < $R_{III,k}$ then
8:                  Delete maintenance(i) from list of job processing
9:              else
10:                 Reserve maintenance(i)
11:             end if
12:         end if
13:     end if
14: end for
15: Generate new list of job processing
```

## 5.1. The reliability threshold parameter analysis

Through experimentation, it was found that the optimal solution was achieved at $R_{III} = 0.80$ in nine cases, while six cases reached optimality at $R_{III} = 0.78$. Notably, for the $15 \times 8$ scale, all five cases yielded optimal solutions at $R_{III} = 0.80$. The detailed data can be found in Table 4. From another perspective, a reduction in $R_{III}$ from its optimal value results in an increase in the makespan. Similarly, an increase in $R_{III}$ beyond its optimal value also leads to a rise in the makespan (as shown in Figure 5).

An analysis of the underlying factors reveals that when $R_{III}$ decreases, the duration for machines transitioning from the *II* area to the *III* area increases. Furthermore, the deterioration effect causes the processing time of machines to rise, further contributing to the increased makespan. Conversely, when $R_{III}$ increases, the transition time for machines moving from the *II* area to the *III* area decreases. This reduction may lead to more frequent forced maintenance for machines after completing workpieces, ultimately resulting in an increased makespan.

**Table 3.** The benchmark parameters and the proposed algorithm parameters.

| No. | Parameters | Range of values |
|---|---|---|
| 1 | Number of jobs | 6,15,20 |
| 2 | Number of machines | 6,8,10 |
| 3 | Number of operations per job | 6 |
| 4 | Processing time of each operation | (0,20] |
| 5 | The time required for a minor repair on $M_k$ | 5 |
| 6 | The time required for a major repair on $M_k$ | 10 |
| 7 | The time required for a mandatory maintenance on $M_k$ | 30 |
| 8 | The age reduction factor of minor repair | 0.35 |
| 9 | The age reduction factor of major repair | 0.1 |
| 10 | The age reduction factor of mandatory maintenance | 0.5 |
| 11 | The reliability threshold $R_{II}$ in the $II$ area | 0.95 |
| 12 | The shape parameters of $M_k$ in the Weibull distribution | (70.00,78.00) |
| 13 | The scale parameters of $M_k$ in the Weibull distribution | (1.60,1.80) |
| 14 | Learning rate | 0.00001 |
| 15 | Replay buffer size | 1000 |
| 16 | Discount factor | 0.9 |
| 17 | Batch size of samples | 128 |
| 18 | Q network update step size | 50 |

Notably, deviations from the optimal $R_{III}$ not only impact makespan but also influence algorithmic stability. To quantitatively assess this relationship, we analyze standard deviation (std) of makespan across three production scales (6 × 6, 15 × 8, and 20 × 10) under varying $R_{III}$ values. For the 6 × 6 configuration, average std values across five cases measure 8.16, 8.41, 7.66, 9.37, 9.89, and 11.26 respectively with increasing $R_{III}$. The 15 × 8 scale displays a similar pattern with std values of 16.32, 16.42, 16.07, 16.07, 20.29, and 21.11. Crucially, both scales exhibit minimised std at $R_{III} = 0.80$, indicating optimal algorithmic stability at this parameter value. The 20 × 10 configuration presents a trend with std measurements of 23.00, 23.78, 23.12, 23.48, 28.01, and 31.49. While $R_{III} = 0.80$ (23.12) doesn't achieve the absolute minimum in this large-scale environment, it maintains superior stability compared to higher $R_{III}$ values, demonstrating preserved robustness despite increased system complexity.

In this set of experiments (Figure 5), the average standard deviation is 18.89, with a standard deviation of 17.22 for $R_{III} = 0.78$ and 17.52 for $R_{III} = 0.80$. Both values are below the average standard deviation for this experiment, indicating greater stability. Based on the aforementioned analysis, subsequent experiments will fix $R_{III}$ at 0.80.

### 5.2. The limited maintenance resources analysis

In the flexible job shop scheduling problem that considers maintenance resources, the potential factor contributing to increased completion time lies in adjusting the number of maintenance activities that can be executed in parallel. In this comparative experiment, limited maintenance resources with $Q = 1, 2, 3, 4$ were compared to unlimited maintenance resources (Table 5).

Figure 6 illustrates how variations in Q affect the makespan, providing examples for scales of 6 × 6, 15 × 8, and 20 × 10. Increasing Q from one to four leads to a significant reduction in the percentage increase of makespan(unlimited maintenance resources as a baseline): for the 6 × 6 scale, it decreases from 33.34% to just 2.82%; for the 15 × 8 scale, it drops from 74.3% to 6.74%; and for the 20 × 10 scale, it falls from 85.31% to 12.73%. An individual examination of these three figures follows. In Figure 6(a), the small-scale case shows a notable decrease in makespan when $Q$ is raised from 1 to 2, dropping from 115.17 to 89.99. However, when Q is further increased to 3 and 4, the makespan changes little, resulting in values of 88.77 and 88.81, respectively. This indicates that maintenance resources are almost maximised at $Q = 2$ for the 6 × 6 scale. In Figure 6(b), the medium-scale case reveals significant potential for optimising the makespan as $Q$ increases from 1 to 3, reducing it from 302.46 to 184.79. The makespan remains nearly constant between $Q = 3$ and $Q = 4$, with a value of 185.23 at $Q = 4$. In Figure 6(c), the large-scale case shows a continuous decrease in makespan as $Q$ increases from 1 to 4, with respective values of 344.93, 237.48, 211.37, and 209.83. Thus, at the 20 × 10 scale, further increases in $Q$ may still provide opportunities for additional optimisation of the makespan.

For the 6 × 6 configuration, the average standard deviations are measured at 17.48, 12.04, 7.66, and 8.15 as $Q$ increases from 1 to 4, with minimum variation observed at $Q = 3$ indicating peak scheduling stability. This phenomenon suggests that at small scales, the system reaches an optimal stability threshold at $Q = 3$, beyond which additional maintenance resources fail to further reduce scheduling variability. With limited total tasks, excessive maintenance capacity ($Q = 4$) may create idle resources that introduce operational instability through irregular maintenance scheduling patterns. This aligns with the earlier makespan observation where $Q = 2$ already achieved near-optimal duration, indicating that smaller systems saturate their resource efficiency earlier.

This contrasts with the medium-scale and large-scale systems where standard deviations demonstrate progressive reductions, dropping respectively from 28.09 to 14.51 (15 × 8 scale) and from 30.77 to 21.98 (20 × 10 scale) with equivalent $Q$ value increases. The inverse correlation between $Q$ magnitudes and standard deviations across larger configurations reveals enhanced system stability through resource scaling, contrasting with

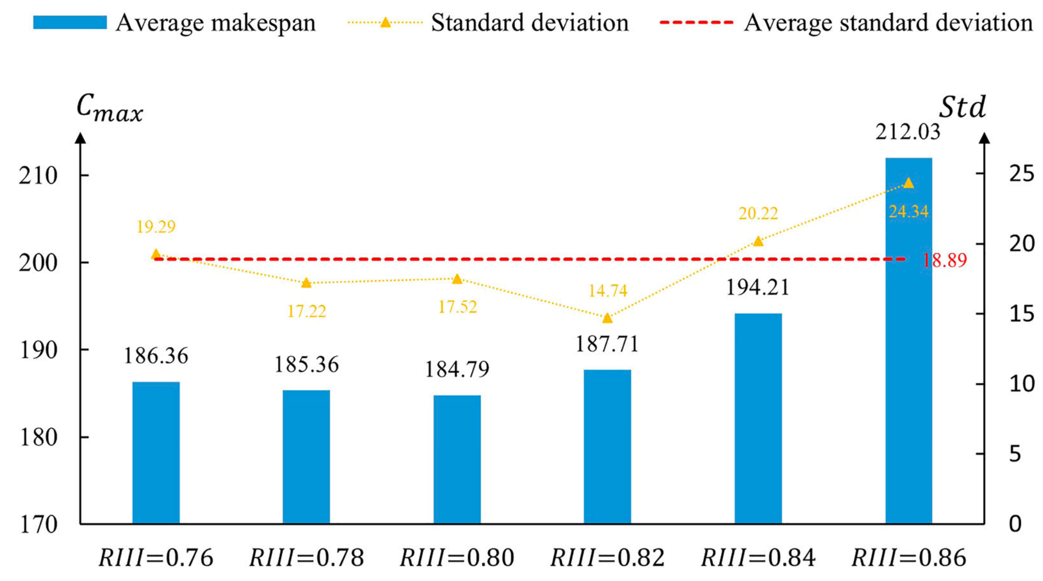


**Figure 5.** The parameter $R_{III}$ analysis for the '15 × 8, $Q = 3$, case 1'.

**Table 4.** Mean value and standard deviation of makespan by different $R_{III}$ when $Q = 3$.

| | $R_{III} = 0.76$ | | $R_{III} = 0.78$ | | $R_{III} = 0.80$ | | $R_{III} = 0.82$ | | $R_{III} = 0.84$ | | $R_{III} = 0.86$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | mean | std | mean | std | mean | std | mean | std | mean | std | mean | std |
| 6 × 6 | | | | | | | | | | | | |
| case1 | 93.35 | 5.81 | 89.32 | 7.29 | **88.82** | 7.60 | 90.37 | 10.56 | 93.74 | 11.73 | 97.15 | 12.51 |
| case2 | 105.69 | 10.18 | 103.65 | 12.42 | **102.10** | 5.04 | 106.11 | 8.75 | 107.80 | 9.98 | 118.88 | 12.55 |
| case3 | 95.35 | 9.40 | **94.11** | 8.94 | 94.14 | 10.38 | 97.40 | 9.91 | 102.30 | 12.50 | 107.86 | 9.14 |
| case4 | 104.08 | 9.50 | **102.34** | 7.60 | 102.36 | 10.57 | 104.24 | 9.53 | 106.24 | 8.17 | 117.12 | 10.71 |
| case5 | 103.03 | 5.93 | **102.41** | 5.82 | 102.94 | 4.71 | 105.69 | 8.09 | 106.33 | 7.09 | 125.24 | 11.37 |
| 15 × 8 | | | | | | | | | | | | |
| case1 | 186.36 | 19.29 | 185.36 | 17.22 | **184.79** | 17.52 | 187.71 | 14.74 | 194.21 | 20.22 | 212.03 | 24.34 |
| case2 | 187.16 | 18.22 | 183.18 | 14.84 | **183.17** | 12.69 | 190.04 | 20.08 | 196.82 | 16.81 | 249.93 | 24.30 |
| case3 | 187.71 | 14.13 | 186.66 | 19.10 | **184.47** | 11.70 | 188.08 | 13.09 | 198.91 | 19.74 | 250.85 | 14.66 |
| case4 | 188.15 | 16.17 | 185.83 | 15.49 | **184.54** | 19.92 | 188.99 | 14.85 | 196.84 | 23.20 | 230.17 | 22.46 |
| case5 | 188.13 | 13.77 | 184.07 | 15.45 | **183.46** | 18.53 | 188.72 | 17.58 | 200.25 | 21.46 | 260.58 | 19.78 |
| 20 × 10 | | | | | | | | | | | | |
| case1 | 216.55 | 24.23 | **206.78** | 17.89 | 211.37 | 20.66 | 221.71 | 26.58 | 230.61 | 22.63 | 251.42 | 29.19 |
| case2 | 215.32 | 22.80 | **208.55** | 22.23 | 217.21 | 28.11 | 224.55 | 31.15 | 228.01 | 30.20 | 247.82 | 37.05 |
| case3 | 216.60 | 20.80 | 208.01 | 20.37 | **200.98** | 17.08 | 224.74 | 23.02 | 241.40 | 33.67 | 266.51 | 29.12 |
| case4 | 220.16 | 24.45 | **211.10** | 28.62 | 216.82 | 26.78 | 221.78 | 18.48 | 236.39 | 27.34 | 245.57 | 30.85 |
| case5 | 219.46 | 22.73 | 222.62 | 29.77 | **218.66** | 22.96 | 223.91 | 18.19 | 247.14 | 26.22 | 273.81 | 31.22 |

the small-scale system's performance ceiling at $Q = 3$. Medium and large-scale systems demonstrate monotonically decreasing standard deviations with increasing $Q$ values. The persistent improvement in scheduling stability reflects how larger systems inherently contain more maintenance-task interactions and resource contention points. Higher $Q$ values enable parallel maintenance execution that progressively mitigates these frictions, each incremental resource addition helps decouple interdependent maintenance activities, thereby reducing schedule variability. Notably, the 20 × 10 system shows the most gradual stability improvement, suggesting that ultra-large systems require substantial resource investments to overcome their complex interdependency networks. These findings collectively demonstrate that scale fundamentally modulates the relationship between maintenance resources and schedule robustness, with system size determining both the optimal $Q$ threshold and the marginal returns on resource investments.

However, it is important to note that increasing $Q$ requires additional maintenance staff. Consequently, manufacturers must carefully evaluate the maintenance costs against the potential benefits of reducing makespan increases.

To validate the feasibility of the decision-making process, Figure 7 illustrates the Gantt charts for case 1 of '6 × 6 $Q = 3$ $R_{III} = 0.80$' and case 1 of '20 × 10 $Q = 3$ $R_{III} = 0.8$', respectively. As observed from the figures, the constraint of limited maintenance resources $Q = 3$

**Table 5.** Mean value and standard deviation of makespan by different $Q$ when $R_{III} = 0.80$ (In this table, 'inc' represents increment percent).

| | $Q = 1$ | | | $Q = 2$ | | | $Q = 3$ | | | $Q = 4$ | | | unlimited $Q$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | mean | std | inc | mean | std | inc | mean | std | inc | mean | std | inc | mean |
| 6 × 6 | | | | | | | | | | | | | |
| case1 | 115.17 | 18.96 | 33.34% | 89.99 | 10.09 | 4.18% | 88.82 | 7.60 | 2.77% | 88.81 | 8.54 | 2.82% | 86.37 |
| case2 | 133.96 | 17.37 | 32.93% | 109.91 | 10.00 | 9.06% | 102.10 | 5.04 | 1.31% | 102.51 | 9.38 | 1.71% | 100.78 |
| case3 | 125.42 | 16.57 | 38.02% | 101.72 | 12.12 | 11.94% | 94.14 | 10.38 | 3.59% | 93.76 | 6.30 | 3.17% | 90.87 |
| case4 | 133.04 | 15.85 | 34.25% | 109.99 | 12.69 | 10.99% | 102.36 | 10.57 | 3.29% | 101.48 | 9.80 | 2.41% | 99.10 |
| case5 | 146.19 | 18.62 | 47.95% | 115.98 | 15.28 | 17.38% | 102.94 | 4.71 | 4.18% | 103.98 | 6.73 | 5.24% | 98.81 |
| 15 × 8 | | | | | | | | | | | | | |
| case1 | 302.46 | 27.96 | 74.30% | 200.15 | 18.97 | 15.34% | 184.79 | 17.52 | 6.49% | 185.23 | 17.33 | 6.74% | 173.53 |
| case2 | 297.85 | 29.09 | 71.45% | 197.55 | 13.97 | 13.72% | 183.17 | 12.69 | 5.44% | 185.04 | 15.28 | 6.52% | 173.72 |
| case3 | 308.10 | 23.65 | 75.17% | 204.37 | 13.99 | 16.19% | 184.47 | 11.70 | 4.88% | 185.19 | 13.05 | 5.29% | 175.89 |
| case4 | 293.03 | 28.12 | 67.74% | 200.22 | 21.63 | 14.61% | 184.54 | 19.92 | 5.64% | 184.96 | 13.32 | 5.88% | 174.69 |
| case5 | 294.38 | 31.65 | 68.08% | 195.09 | 17.68 | 11.39% | 183.46 | 18.53 | 4.75% | 184.03 | 13.57 | 5.07% | 175.15 |
| 20 × 10 | | | | | | | | | | | | | |
| case1 | 344.93 | 35.32 | 85.31% | 237.48 | 25.56 | 27.58% | 211.37 | 20.66 | 13.56% | 209.83 | 17.39 | 12.73% | 186.13 |
| case2 | 354.63 | 38.85 | 75.22% | 244.34 | 23.78 | 20.72% | 217.21 | 28.11 | 7.32% | 214.98 | 27.43 | 6.22% | 202.40 |
| case3 | 369.29 | 21.38 | 91.99% | 250.69 | 32.54 | 30.33% | 200.98 | 17.08 | 4.48% | 199.41 | 18.37 | 3.67% | 192.35 |
| case4 | 354.68 | 28.83 | 81.87% | 244.91 | 29.21 | 25.58% | 216.82 | 26.78 | 11.18% | 208.06 | 28.61 | 6.69% | 195.02 |
| case5 | 378.07 | 29.49 | 84.53% | 246.77 | 34.88 | 20.45% | 218.66 | 22.96 | 6.73% | 217.63 | 18.09 | 6.22% | 204.88 |

is strictly adhered to throughout the entire processing period. The makespan for these cases is 75 and 203, respectively.

To provide a more intuitive view of the feasibility and effectiveness of the decision-making process, Figure 8 presents the variation in machine age values for each machine in case 1 of '6 × 6 $Q = 3$ $R_{III} = 0.80$'. The coloured lines in these figures correspond to the respective jobs in the Gantt charts, while the black dashed lines indicate that the machine is idle. Additionally, the red horizontal line represents the threshold $R_{III}$ for the machine, and the dark yellow horizontal line represents the threshold $R_{II}$. Specifically, when the machine age is less than $R_{II}$, the machine is in normal area; when the machine age is greater than $R_{II}$ but less than $R_{III}$, the machine is in deterioration area; and when the machine age exceeds $R_{III}$, the machine is in mandatory maintenance area.

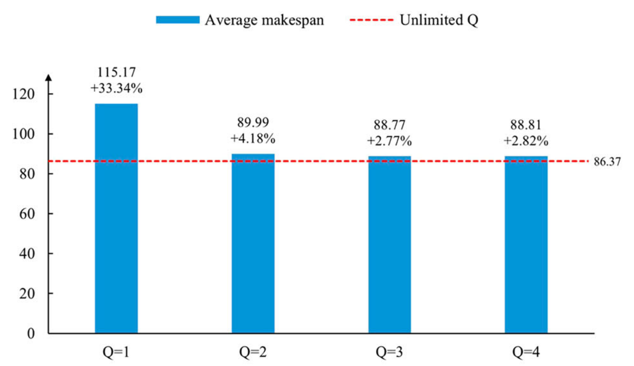
As explained in Section 3.2, when the machine age is in *II* area, the processing time increases due to the deterioration effect. Conversely, when the machine age is in *I* area, the processing time remains unchanged. For example, in Figure 8(a), $R_{II,1} = 11.31$, and the coordinate point (12, 11) indicates that at time 12, the machine age is 11. This demonstrates that machine 1 has not yet reached the deterioration effect, and thus its processing time remains unchanged at 13. In Figure 8(c), $R_{II,3} = 11.90$, and the coordinate point (12, 12) indicates that at time 12, the machine age of machine 3 is 12. At this point, machine 3 has reached the deterioration effect, causing its processing time to increase from the original 6 to 6.03. Through the analysis of experimental data and results, the feasibility and effectiveness of the proposed method's decisions can be demonstrated.

## 5.3. Comparisons with other well-known dispatching rules

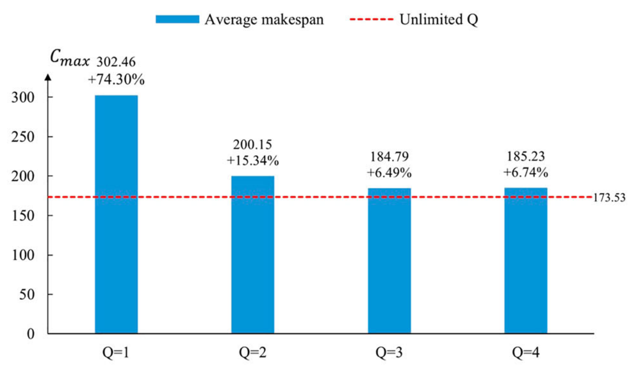
To further validate the effectiveness of DQN-LS, a comparison was made with four well-established scheduling rules: First-In-First-Out (FIFO), Most Remaining Time (MRT), Shortest Processing Time (SPT), and Longest Processing Time (LPT). The FIFO rule prioritizes the job that arrived first for the next operation. MRT focuses on the job that has the longest remaining processing time, while SPT selects the operation with the shortest processing time from the current jobs. Conversely, LPT chooses the operation from the job with the longest processing time among the available ones.

Additionally, the proposed algorithm was evaluated against a DQN without local search and Proximal Policy Optimization(PPO). For comparison purposes, a completely random rule was also included, where an unprocessed operation is randomly chosen and assigned to an available machine.

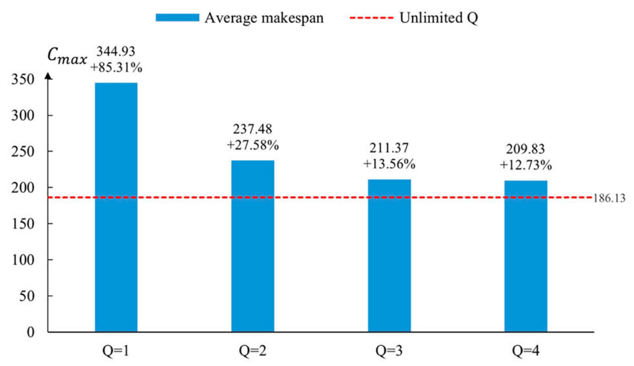
It is crucial to highlight that the aforementioned scheduling rules (with the exception of the random rule) do not explicitly specify the processing machine, rendering them inadequate for addressing DFJSP-LMR discussed in this paper. To mitigate this limitation, operations selected using the established four scheduling rules were assigned to the earliest available machine to minimise total tardiness and allow for a fair comparison. Moreover, for MRT, SPT, and LPT, the processing time for



(a) The limited maintenance resources analysis for the "6 × 6 $R_{III}$ = 0.80 *case* 1"



(b) The limited maintenance resources analysis for the "15 × 8 $R_{III}$ = 0.80 *case* 1"

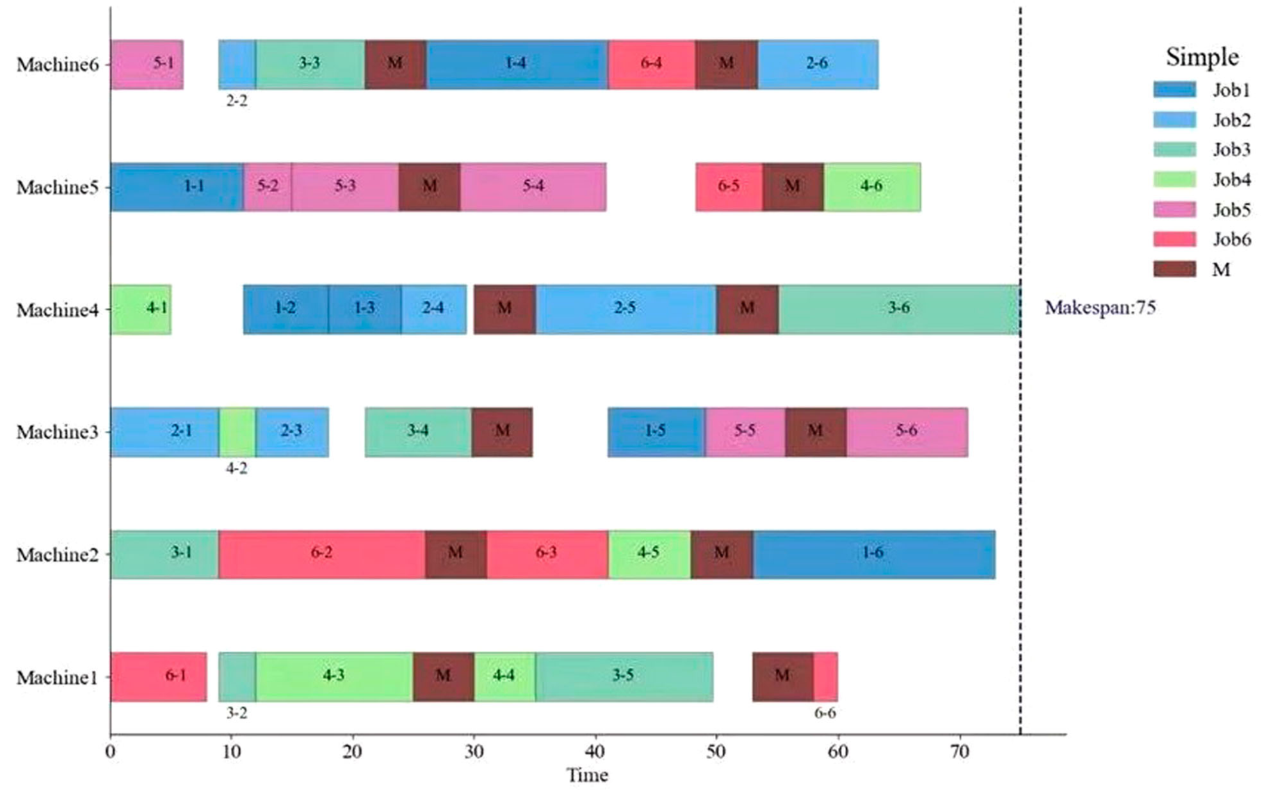


(c) The limited maintenance resources analysis for the "20 × 10 $R_{III}$ = 0.80 *case* 1"

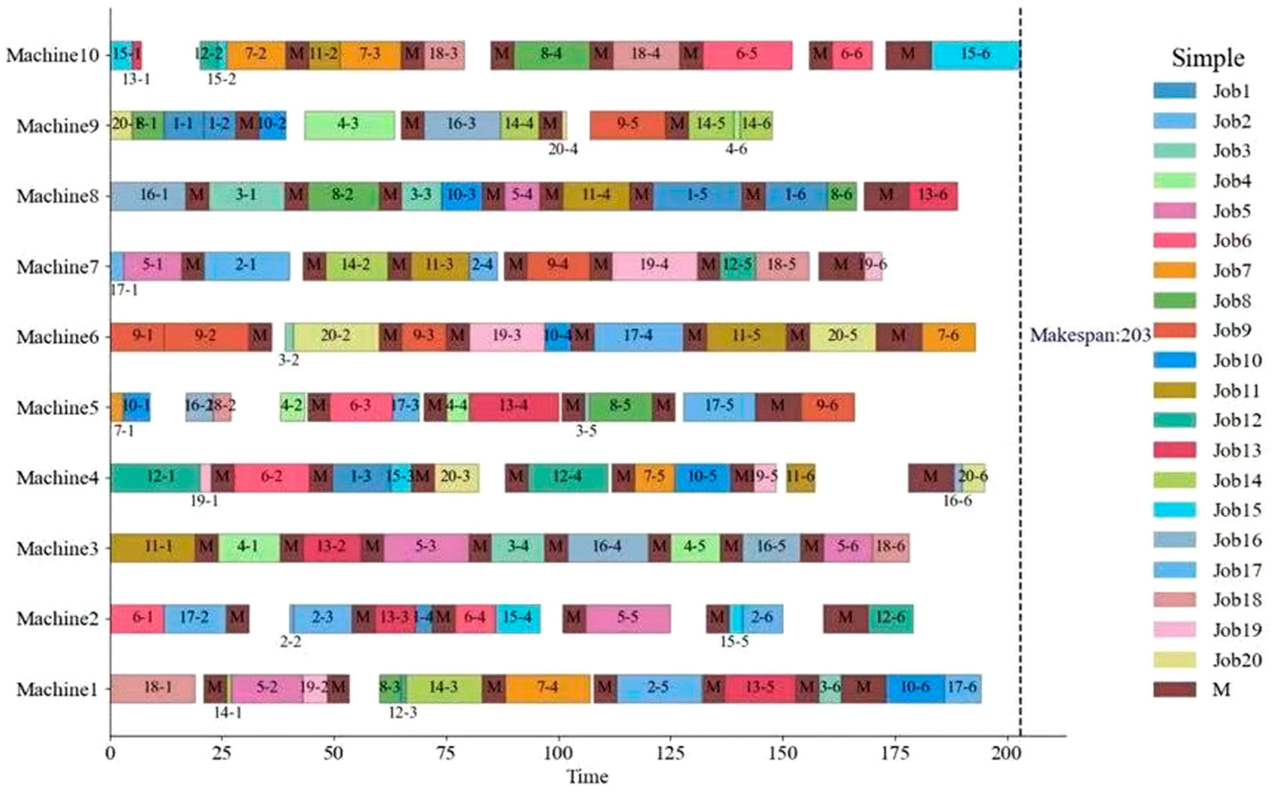
**Figure 6.** The limited maintenance resources analysis.

an operation was estimated by averaging its processing times across all available machines.

The comparative analysis of scheduling methods across 20 independent replicas per instance reveals distinct performance patterns. As documented in Table 6, the proposed DQN-LS algorithm demonstrates superior scheduling effectiveness by achieving optimal makespan solutions in 11 out of 15 test cases, which not only significantly outperforms both DQN and conventional scheduling rules, but also validates the method's generalizability. Notably, the hybrid approach exhibits scale-dependent characteristics: In small-scale configurations, DQN-LS achieves the lowest average standard deviation, thereby indicating superior schedule stability through effective local search optimisation. However, medium and large-scale instances present an inverse relationship between solution quality and stability: while DQN-LS maintains its makespan superiority, its standard deviation values nevertheless show moderate increases compared to classical rules. This performance dichotomy suggests the algorithm's adaptive strength in balancing solution optimality and robustness across different problem sizes, where its neural network component effectively navigates complex solution spaces.

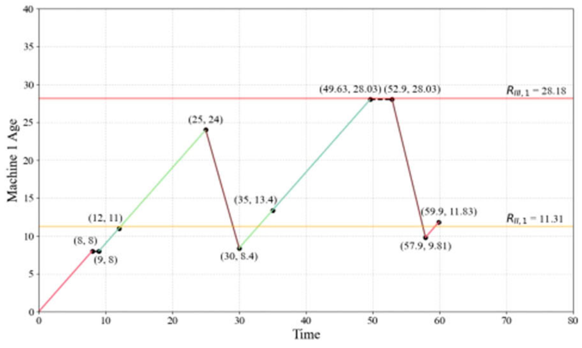


(a) Gantt Chart for "$6 \times 6\ Q = 3\ R_{III} = 0.80$ case 1"

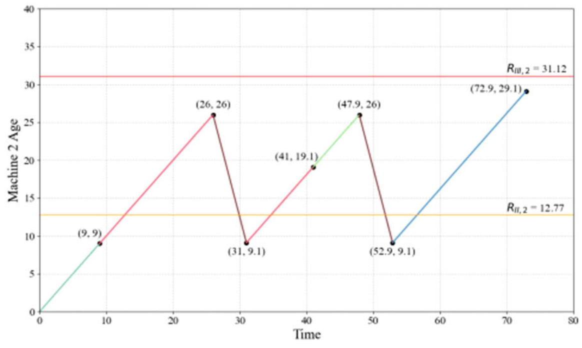


(b) Gantt Chart for "$20 \times 10\ Q = 3\ R_{III} = 0.80$ case 1"

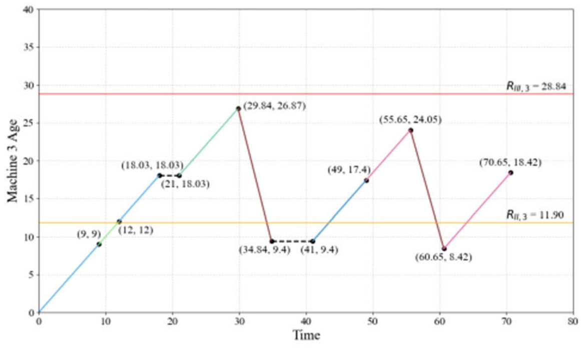
**Figure 7.** Gantt charts for different scales, M represents maintenance.



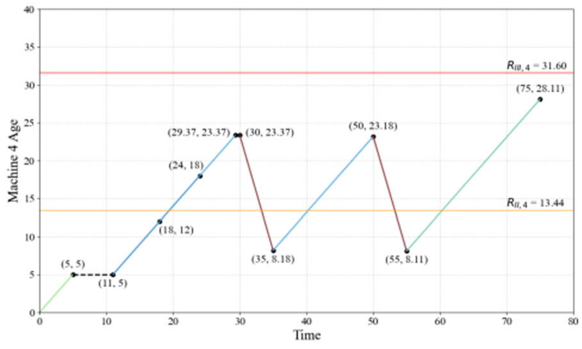
(a) The trajectory plot for Machine 1 age



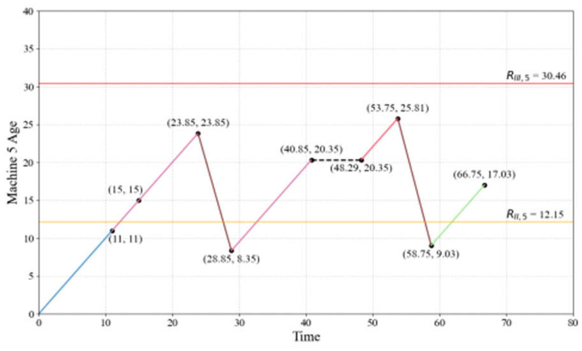
(b) The trajectory plot for Machine 2 age



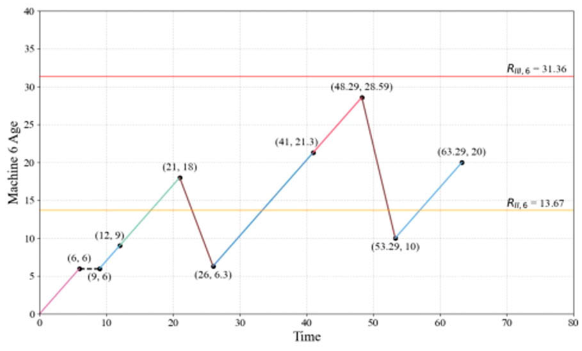
(c) The trajectory plot for Machine 3 age



(d) The trajectory plot for Machine 4 age



(e) The trajectory plot for Machine 5 age



(f) The trajectory plot for Machine 6 age

**Figure 8.** The trajectory plot for Machine age.

## 5.4. *Application of the proposed approach*

This section presents a real-world case study to demonstrate the applicability and effectiveness of the proposed approach. The study examines a small manufacturing workshop, structured as a flexible job shop, specialising in customised automotive components. The workshop operates 7 machines and handles 12 jobs, each consisting of 6 operations, all to be completed within a planning horizon of 3 days. Jobs consist of a sequence of operations that can be performed on multiple machines based on their availability and capability. The machine failure model follows the Weibull distribution, with unique shape and scale parameters assigned to every machine. Detailed data for this case study can be provided upon request by contacting the corresponding author.

The primary objective of this case study is to minimise the makespan while satisfying processing constraints, including limited worker availability, and adhering to delivery deadlines. Therefore, the number of maintenance workers in this instance is fixed at three.

**Table 6.** Mean value and standard deviation of makespan by different algorithms and dispatching rules when $Q = 3$ and $R_{III} = 0.80$.

| | DQN-LS | | DQN | | PPO | | FIFO | | LPT | | SPT | | MRT | | RR | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Mean | Std | Mean | Std | Mean | Std | Mean | Std | Mean | Std | Mean | Std | Mean | Std | Mean | Std |
| 6 × 6 | | | | | | | | | | | | | | | | |
| case1 | **88.82** | 7.60 | 95.82 | 10.85 | 105.90 | 15.68 | 90.00 | 8.04 | 125.35 | 10.51 | 133.22 | 5.70 | 96.37 | 12.91 | 153.86 | 11.21 |
| case2 | **102.10** | 5.04 | 109.95 | 4.51 | 121.75 | 14.22 | 104.44 | 7.75 | 140.57 | 10.32 | 156.57 | 9.12 | 102.13 | 6.78 | 167.23 | 11.72 |
| case3 | **94.14** | 10.38 | 99.25 | 11.40 | 121.76 | 16.25 | 96.39 | 11.09 | 127.21 | 17.85 | 132.80 | 9.47 | 94.17 | 4.77 | 160.90 | 11.44 |
| case4 | 102.36 | 10.57 | 109.95 | 9.20 | 140.31 | 15.78 | 104.23 | 12.24 | 160.34 | 9.50 | 162.20 | 9.50 | **100.06** | 8.11 | 180.48 | 12.41 |
| case5 | **102.94** | 4.71 | 112.14 | 7.95 | 134.77 | 18.55 | 106.61 | 5.18 | 153.00 | 14.07 | 151.43 | 13.36 | 104.08 | 8.29 | 185.00 | 14.89 |
| 15 × 8 | | | | | | | | | | | | | | | | |
| case1 | **184.79** | 17.52 | 206.84 | 16.21 | 215.97 | 37.13 | 193.21 | 19.94 | 247.20 | 11.89 | 234.88 | 8.72 | 184.86 | 5.23 | 300.06 | 11.95 |
| case2 | 183.17 | 12.69 | 206.12 | 14.09 | 230.13 | 29.88 | **181.20** | 6.46 | 263.51 | 12.32 | 276.33 | 10.11 | 182.42 | 6.55 | 300.96 | 17.92 |
| case3 | **184.47** | 11.70 | 206.75 | 14.68 | 229.27 | 43.79 | 186.37 | 9.69 | 270.35 | 12.72 | 285.63 | 8.38 | 184.97 | 5.72 | 320.42 | 20.66 |
| case4 | **184.54** | 19.92 | 208.47 | 20.55 | 221.44 | 31.48 | 186.32 | 20.62 | 245.56 | 8.66 | 267.32 | 11.75 | 185.90 | 12.21 | 299.52 | 14.14 |
| case5 | **183.46** | 18.53 | 208.17 | 18.70 | 207.69 | 40.72 | 188.90 | 11.45 | 251.64 | 11.58 | 276.25 | 10.95 | 183.83 | 11.66 | 307.92 | 17.03 |
| 20 × 10 | | | | | | | | | | | | | | | | |
| case1 | **211.37** | 20.66 | 247.67 | 19.52 | 236.34 | 39.50 | 213.10 | 7.40 | 282.84 | 14.46 | 325.02 | 10.73 | 212.47 | 6.77 | 342.90 | 17.89 |
| case2 | 217.21 | 28.11 | 248.89 | 25.26 | 255.77 | 36.09 | 213.93 | 7.17 | 282.65 | 17.43 | 316.70 | 14.55 | **213.29** | 9.62 | 336.99 | 22.50 |
| case3 | **207.25** | 22.30 | 240.30 | 17.11 | 249.91 | 55.96 | 210.35 | 6.85 | 279.81 | 9.33 | 317.39 | 15.63 | 213.66 | 6.96 | 335.37 | 23.64 |
| case4 | 216.82 | 26.78 | 249.23 | 26.73 | 279.32 | 48.34 | 213.41 | 9.98 | 302.07 | 14.31 | 297.70 | 10.60 | **208.43** | 11.00 | 343.67 | 17.36 |
| case5 | **218.66** | 22.96 | 260.70 | 22.95 | 265.48 | 56.12 | 219.14 | 9.71 | 308.79 | 13.56 | 296.87 | 16.27 | 225.44 | 9.20 | 354.16 | 21.86 |

**Table 7.** Analysis of Algorithms on Instance Data.

| $Q = 3$ | average(min) | std | due date(days) |
|---|---|---|---|
| DQN-LS | 1747.27 | 7.12 | 2.43 |
| DQN | 1864.42 | 6.72 | 2.59 |
| PPO | 2620.38 | 44.53 | 3.64 |
| FIFO | 1812.05 | 20.66 | 2.52 |
| LPT | 2350.74 | 7.59 | 3.26 |
| SPT | 2404.76 | 5.55 | 3.34 |
| MRT | 1834.97 | 18.01 | 2.55 |
| RR | 2895.78 | 15.29 | 4.02 |

To rigorously evaluate its performance, the proposed algorithm, DQN-LS, was subjected to 20 parallel simulations and compared with other algorithmic rules in terms of makespan and stability.

As shown in Table 7, the results demonstrate that the makespan achieved by DQN-LS is superior to other algorithmic rules in this instance. Specifically, the completion time for DQN-LS is 1747.27 minutes, which translates to approximately 2.43 days, meeting the delivery deadline. In terms of stability, although DQN-LS is not the best-performing algorithm, its standard deviation value is relatively low, indicating that the proposed algorithm exhibits consistent and reliable performance.

The experimental data highlights the comparative performance of DQN-LS against other algorithms. For instance, while FIFO achieves a slightly shorter completion time of 1812.05 minutes, its standard deviation is significantly higher than that of DQN-LS, suggesting less stability. Similarly, other algorithms like PPO and RR show higher completion times and greater variability, further emphasising the effectiveness of DQN-LS in balancing makespan minimisation and stability.

These results emphasise the capability of DQN-LS to effectively optimise scheduling under constrained conditions, including limited worker availability and strict delivery deadlines. The algorithm not only achieves a minimised makespan but also demonstrates a relatively low variability in performance, ensuring both efficiency and stability. These qualities make DQN-LS a promising and practical approach for addressing complex real-world scheduling challenges.

## 6. Conclusion and outlook

Considering that traditional FJSP models are no longer suitable for the limited practical production environment of maintenance workers, this paper incorporates limited maintenance resources as constraints and establishes a mathematical model for DFJSP-LMR aimed at minimising the makespan, based on FJSP. This model fully considers the situation where the production workshop has varying amounts of maintenance resources.

To solve the DFJSP-LMR model, this paper proposes a STW algorithm and conducts experiments using a combination of DQN and LS methods. In addition, 15 composite scheduling rules are proposed to select unprocessed jobs and schedule the earliest available machines for processing. Numerical experiments are conducted under different scales and cases to validate the effectiveness and generality of the proposed DQN.

In future work, other objectives, such as production costs and energy consumption, are also worth considering. Production and maintenance are interrelated, and the costs incurred by them also need to be studied. How to arrange scheduling to better reduce costs is one of the primary concerns for factories.

At the same time, it is important to note that the proposed DQN-LS has not been compared with other advanced RL methods. Therefore, we will investigate other advanced methods, including A2C, TRPO, and DDQN, and compare their performance with DQN-LS.

## Disclosure statement

No potential conflict of interest was reported by the author(s).

## Funding

This work was supported by the National Natural Science Foundation of China under Grant Nos. [52005447, 72271222]; Zhejiang Provincial Natural Science Foundation of China under Grant Nos. [LQ21E050014, LGG22G010002]; and Open Research Topics of National Key Laboratory of Intelligent Manufacturing Equipment and Technology under Grant Nos. [IMETKF2024019].

## Notes on contributors

*WenChao Yi* received the B.Sc. and Ph.D. degrees in Industrial Engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2011 and 2016, respectively. She is currently a Lecturer with the College of Mechanical Engineering, Zhejiang University of Technology, China. Her current research interests include evolutionary algorithms, and its application in scheduling.

*Nanxing Chen* is a Master's student in the Department of Industrial Engineering at Zhejiang University of Technology. His current research interests include optimisation algorithms and job shop scheduling problems.

*Yong Chen* was born in Changsha, Hunan Province, China in 1973. He received the B.S. degree, M.S. degree and Ph.D. degree in Mechanical Engineering from Zhejiang University, Hangzhou, in 2000. Since 2009, he has been a Professor with the Institute of Industrial Engineering, Zhejiang University of Technology, Hangzhou. His research interests include intelligent system planning and algorithms.

*Zhi Pei* received the B.S. and Ph.D. degrees in industrial engineering from Tsinghua University, Beijing, China, in 2005 and 2011, respectively. He was a Visiting Professor at North Carolina State University, Raleigh, USA, in 2015. He is currently a Full Professor with the Department of Industrial Engineering, Zhejiang University of Technology, China. His current research interests include manufacturing system modelling, machine scheduling, nonlinear optimisation, and queueing theory.

## Data availability statement

The data that support the findings of this study are available from the corresponding author[Y. Chen], upon reasonable request.

## ORCID

*Zhi Pei* http://orcid.org/0000-0001-6808-1490

## References

Al-Hinai, N., and T. ElMekkawy. 2011. "Robust and Stable Flexible Job Shop Scheduling with Random Machine Breakdowns Using a Hybrid Genetic Algorithm." *International Journal of Production Economics* 132 (2): 279–291. https://doi.org/10.1016/j.ijpe.2011.04.020.

An, Y., X. Chen, K. Gao, Y. Li, and L. Zhang. 2023. "Multiobjective Flexible Job-Shop Rescheduling with New Job Insertion and Machine Preventive Maintenance." *IEEE Transactions on Cybernetics* 53 (5): 3101–3113. https://doi.org/10.1109/TCYB.2022.3151855.

An, Y., X. Chen, Y. Li, J. Zhang, and J. Jiang. 2021. "Flexible Job-Shop Scheduling and Heterogeneous Repairman Assignment with Maintenance Time Window and Employee Timetable Constraints." *Expert Systems with Applications* 186:115693. https://doi.org/10.1016/j.eswa.2021.115693.

Andriotis, C. P., and K. G. Papakonstantinou. 2021. "Deep Reinforcement Learning Driven Inspection and Maintenance for Deteriorating Engineering Systems." *Reliability Engineering & System Safety* 212:107551. https://doi.org/10.1016/j.ress.2021.107551.

Azadeh, A., A. H. Goodarzi, and M. H. Kolaee. 2019. "An Efficient Simulation–Neural Network–Genetic Algorithm for Flexible Flow Shops with Sequence-Dependent Setup Times, Job Deterioration and Learning Effects." *Neural Computing and Applications* 31:5327–5341. https://doi.org/10.1007/s00521-018-3368-6.

Baykasoğlu, A., and F. S. Madenoglu. 2021. "Greedy Randomized Adaptive Search Procedure for Simultaneous Scheduling of Production and Preventive Maintenance Activities in Dynamic Flexible Job Shops." *Soft Computing* 25:14893–14932. https://doi.org/10.1007/s00500-021-06053-0.

Blundell, C., B. Uria, A. Pritzel, Y. Li, A. Ruderman, J. Leibo, J. Rae, D. Wierstra, and D. Hassabis. 2016. "Model-Free Episodic Control." *arxiv preprint* arxiv:1606.04460, https://doi.org/10.48550/arXiv.1606.04460..

Buddala, R., and S. S. Mahapatra. 2019. "Two-Stage Teaching-Learning-Based Optimization Method for Flexible Job-Shop Scheduling under Machine Breakdown." *The International Journal of Advanced Manufacturing Technology* 100:1419–1432. https://doi.org/10.1007/s00170-018-2805-0.

Chen, X., Y. An, Z. Zhang, and Y. Li. 2020. "An Approximate Nondominated Sorting Genetic Algorithm to Integrate Optimization of Production Scheduling and Accurate Maintenance Based on Reliability Intervals." *Journal of Manufacturing Systems* 54:227–241. https://doi.org/10.1016/j.jmsy.2019.12.004.

Chen, J., and Y. Wang. 2023. "A Deep Reinforcement Learning Approach for Maintenance Planning of Multi-Component Systems with Complex Structure." *Neural Computing and*

*Applications* 35:15549–15562. https://doi.org/10.1007/s00521-023-08542-9.

Conway, R., W. L. Maxwell, and L. W. Miller. 1967. "Theory of Scheduling." In *Computer Science*. Reading, MA: Addison-Wesley.

Cunha, B., A. M. Madureira, B. Fonseca, and D. Coelho. 2020. "Deep Reinforcement Learning as a Job Shop Scheduling Solver: A Literature Review." *Advances in Intelligent Systems and Computing* 923:350–359. https://doi.org/10.1007/978-3-030-14347-3_34.

Da, W., H. Feng, and E. Pan. 2016. "Integrated Preventive Maintenance and Production Scheduling Optimization on Uniform Parallel Machines with Deterioration Effect." In *2016 IEEE International Conference on Industrial Engineering and Engineering Management*, 951–953. Shanghai, China: IEEE.

Fang, X., J. Li, and Y. Wang. 2023. "Learning to Schedule Job Shop Scheduling Problem with Maintenance Time Using Graph Node Embedding and Deep Reinforcement Learning." In Vol. 12709 of *Fourth International Conference on Artificial Intelligence and Electromechanical Automation (AIEA 2023)*, 1271–1279. Nanjing, China: SPIE. https://doi.org/10.1117/12.2684742.

Feng, M., and Y. Li. 2022. "Predictive Maintenance Decision Making Based on Reinforcement Learning in Multistage Production Systems." *IEEE Access* 10:18910–18921. https://doi.org/10.1109/ACCESS.2022.3151170.

Gao, J., M. Gen, and L. Sun. 2006. "Scheduling Jobs and Maintenances in Flexible Job Shop with a Hybrid Genetic Algorithm." *Journal of Intelligent Manufacturing* 17:493–507. https://doi.org/10.1007/s10845-005-0021-x.

Ghaleb, M., A. ElMekkawy, and M. Nourelfath. 2021. "Real-Time Integrated Production-Scheduling and Maintenance-Planning in a Flexible Job Shop with Machine Deterioration and Condition-Based Maintenance." *Journal of Manufacturing Systems* 61:423–449. https://doi.org/10.1016/j.jmsy.2021.09.018.

Ghaleb, M., H. Zolfagharinia, and S. Taghipour. 2020. "Real-Time Production Scheduling in the Industry-4.0 Context: Addressing Uncertainties in Job Arrivals and Machine Breakdowns." *Computers & Operations Research* 123:105031. https://doi.org/10.1016/j.cor.2020.105031.

Gu, Z., L. Li, J. Zheng, and G. Liu. 2016. "A Flexible Job-Shop Rescheduling Method by considering the Machine Equipment Availability." In *Proceedings of the 2016 28th Chinese Control and Decision Conference (CCDC)*, 4898–4902. China: IEEE. https://doi.org/10.1109/CCDC.2016.7531479.

Guo, P., H. Shi, Y. Wang, and J. Xiong. 2024. "Multi-Objective Scheduling of Cloud-Edge Cooperation in Distributed Manufacturing via Multi-Agent Deep Reinforcement Learning." *International Journal of Production Research* 1–25. https://doi.org/10.1080/00207543.2024.2329316.

Heess, N., G. Wayne, D. Silver, T. Lillicrap, T. Erez, and Y. Tassa. 2015. "Learning Continuous Control Policies by Stochastic Value Gradients." *Advances in Neural Information Processing Systems* 28. https://arxiv.org/abs/1510.09142.

Huang, J., Q. Chang, and J. Arinez. 2020. "Deep Reinforcement Learning Based Preventive Maintenance Policy for Serial Production Lines." *Expert Systems with Applications* 160:113701. https://doi.org/10.1016/j.eswa.2020.113701.

Johnson, S. M. 1954. "Optimal Two- and Three-Stage Production Schedules with Setup Times Included." *Naval Research Logistics Quarterly* 1 (1): 61–68. https://doi.org/10.1002/nav.3800010110.

Kacem, I., S. Hammadi, and P. Borne. 2002. "Approach by Localization and Multiobjective Evolutionary Optimization for Flexible Job-Shop Scheduling Problems." *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 32 (1): 1–13. https://doi.org/10.1109/TSMCC.2002.1009117.

Lamprecht, R., F. Wurst, and M. F. Huber. 2021. Reinforcement Learning Based Condition-Oriented Maintenance Scheduling for Flow Line Systems. In *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*, 1–7. Palma de Mallorca, Spain: IEEE. https://doi.org/10.1109/INDIN45523.2021.9557373.

Lawler, E. L., J. K. Lenstra, A. Kan, and D. Shmoys. 1993. "Sequencing and Scheduling: Algorithms and Complexity." *Handbooks in Operations Research and Management Science* 4: 445–522. https://doi.org/10.1016/S0927-0507(05)80189-6.

Lei, K., P. Guo, Y. Wang, J. Zhang, X. Meng, and L. Qian. 2024. "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.

Lei, K., P. Guo, W. Zhao, Y. Wang, L. Qian, X. Meng, and L. Tang. 2022. "A Multi-Action Deep Reinforcement Learning Framework for Flexible Job-Shop Scheduling Problem." *Expert Systems with Applications* 205:117796. https://doi.org/10.1016/j.eswa.2022.117796.

Li, M., C. T. Chang, and Z. Liu. 2024. "A Discrete Artificial Bee Colony Algorithm and Its Application in Flexible Flow Shop Scheduling with Assembly and Machine Deterioration Effect." *Applied Soft Computing* 159:111593. https://doi.org/10.1016/j.asoc.2024.111593.

Li, J., Q. Pan, and Y. Liang. 2010. "An Effective Hybrid Tabu Search Algorithm for Multi-Objective Flexible Job-Shop Scheduling Problems." *Computers & Industrial Engineering*59:647–662. https://doi.org/10.1016/j.cie.2010.07.014.

Liu, Y., Y. Chen, and T. Jiang. 2020. "Dynamic Selective Maintenance Optimization for Multi-State Systems over a Finite Horizon: A Deep Reinforcement Learning Approach." *European Journal of Operational Research* 283:166–181. https://doi.org/10.1016/j.ejor.2019.10.049.

Moradi, E., S. M. T. F. Ghomi, and M. Zandieh. 2010. "An Efficient Architecture for Scheduling Flexible Job-Shop with Machine Availability Constraints." *The International Journal of Advanced Manufacturing Technology* 51:325–339. https://doi.org/10.1007/s00170-010-2621-7.

Nair, A., P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. D. Maria, V. Panneershelvam, et al. 2015. "Massively Parallel Methods for Deep Reinforcement Learning." *arxiv preprint* arxiv:1507.04296. https://arxiv.org/abs/1507.04296.

Nouiri, M., A. Bekrar, and D. Trentesaux. 2018. "Towards Energy Efficient Scheduling and Rescheduling for Dynamic Flexible Job Shop Problem." *IFAC PapersOnLine* 51 (11): 1275–1280. https://doi.org/10.1016/j.ifacol.2018.08.357.

Ogunfowora, O., and H. Najjaran. 2023. "Reinforcement and Deep Reinforcement Learning-Based Solutions for Machine Maintenance Planning, Scheduling Policies, and Optimization." *Journal of Manufacturing Systems* 70: 244–263. https://doi.org/10.1016/j.jmsy.2023.07.014.

Pakzad-Moghaddam, S. H., H. Mina, and R. Tavakkoli-Moghaddam. 2014. "An Approach for Modeling a New Single Machine Scheduling Problem with Deteriorating and Learning Effects." *Computers & Industrial Engineering* 78:33–43. https://doi.org/10.1016/j.cie.2014.09.021.

Pinedo, M. L. 2022. "Scheduling." In *Scheduling*. American: Springer. https://doi.org/10.1007/978-3-031-05921-6.

Rodríguez, M. L. R., S. Kubler, A. de Giorgio, M. Cordy, J. Robert, and Y. Le Traon. 2022. "Multi-agent Deep Reinforcement Learning Based Predictive Maintenance on Parallel Machines." *Robotics and Computer-Integrated Manufacturing* 78:102406. https://doi.org/10.1016/j.rcim.2022.102406.

Soofi, P., M. Yazdani, M. Amiri, and M. A. Adibi. 2021. "Robust Fuzzy-Stochastic Programming Model and Meta-Heuristic Algorithms for Dual-Resource Constrained Flexible Job-Shop Scheduling Problem under Machine Breakdown." *IEEE Access* 9:155740–155762. https://doi.org/10.1109/ACCESS.2021.3126820.

Tariq, A., S. A. Khan, W. H. But, A. Javaid, and T. Shehryar. 2024. "An Iot-Enabled Real-Time Dynamic Scheduler for Flexible Job Shop Scheduling (FJSS) in an Industry 4.0-based Manufacturing Execution System (MES 4.0)." *IEEE Access* 12:49653–49666. https://doi.org/10.1109/ACCESS.2024.3384252.

Upasani, K., M. Bakshi, V. Pandhare, and B. K. Lad. 2017. "Distributed Maintenance Planning in Manufacturing Industries." *Computers & Industrial Engineering* 108:1–14. https://doi.org/10.1016/j.cie.2017.03.027.

Wang, C., and P. Jiang. 2018. "Manifold Learning Based Rescheduling Decision Mechanism for Recessive Disturbances in Rfid-Driven Job Shops." *Journal of Intelligent Manufacturing* 29:1485–1500. https://doi.org/10.1007/s10845-016-1194-1.

Wang, B., and X. Yang. 2009. "Robust Model for Job Shop Scheduling with Uncertain Processing Times." In *2009 Chinese Control and Decision Conference*, 2484–2489. Guilin, China: IEEE. https://doi.org/10.1109/CCDC.2009.5192445.

Wang, Y. M., H. L. Yin, and K. D. Qin. 2013. "A Novel Genetic Algorithm for Flexible Job Shop Scheduling Problems with Machine Disruptions." *International Journal of Advanced Manufacturing Technology* 68:1317–1326. https://doi.org/10.1007/s00170-013-4923-z.

Wang, S., and J. Yu. 2010. "An Effective Heuristic for Flexible Job-Shop Scheduling Problem with Maintenance Activities." *Computers & Industrial Engineering* 59 (3): 436–447. https://doi.org/10.1016/j.cie.2010.05.012.

Wei, L., J. He, Z. Guo, and Z. Hu. 2023. "A Multi-Objective Migrating Birds Optimization Algorithm Based on Game Theory for Dynamic Flexible Job Shop Scheduling Problem." *Expert Systems with Applications* 227:120268. https://doi.org/10.1016/j.eswa.2023.120268.

Wocker, M. M., F. F. Ostermeier, T. Wanninger, R. Zwinkau, and J. Deuse. 2024. "Flexible Job Shop Scheduling with Preventive Maintenance Consideration." *Journal of Intelligent Manufacturing* 35:1517–1539. https://doi.org/10.1007/s10845-023-02114-3.

Wu, X., X. Shen, and C. Li. 2019. "The Flexible Job-Shop Scheduling Problem considering Deterioration Effect and Energy Consumption Simultaneously." *Computers & Industrial Engineering* 135:1004–1024. https://doi.org/10.1016/j.cie.2019.06.048.

Xia, W., and Z. Wu. 2005. "An Effective Hybrid Optimization Approach for Multi-Objective Flexible Job-Shop Scheduling Problems." *Computers & Industrial Engineering* 48:409–425. https://doi.org/10.1016/j.cie.2005.01.018.

Xia, W. J., and Z. M. Wu. 2005. "An Effective Hybrid Optimization Approach for Multi-Objective Flexible Job-Shop Scheduling Problems." *Computers & Industrial Engineering* 48:409–425. https://doi.org/10.1016/j.cie.2005.01.018.

Yamada, T., and R. Nakano. 1992. "A Genetic Algorithm Applicable to Large-Scale Job-Shop Problems." In *PPSN*. Vol. 2, 281–290.

Yang, Y., M. Huang, Z. Yu Wang, and Q. Bing Zhu. 2020. "Robust Scheduling Based on Extreme Learning Machine for Bi-Objective Flexible Job-Shop Problems with Machine Breakdowns." *Expert Systems with Applications* 158:113545. https://doi.org/10.1016/j.eswa.2020.113545.

Zeng, Z., X. Li, and C. Bai. 2022. "A Deep Reinforcement Learning Approach to Flexible Job Shop Scheduling." In *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 884–890. Wuhan, China: IEEE. https://doi.org/10.1109/SMC53654.2022.9945107.

Zhang, C., Y. F. Li, and D. W. Coit. 2023. "Deep Reinforcement Learning for Dynamic Opportunistic Maintenance of Multi-Component Systems with Load Sharing." *IEEE Transactions on Reliability* 72 (3): 863–877. https://doi.org/10.1109/TR.2022.3197322.

Zhang, N., and W. Si. 2020. "Deep Reinforcement Learning for Condition-Based Maintenance Planning of Multi-Component Systems under Dependent Competing Risks." *Reliability Engineering and System Safety* 203:107094. https://doi.org/10.1016/j.ress.2020.107094.

Zhang, P., X. Zhu, and M. Xie. 2021. "A Model-Based Reinforcement Learning Approach for Maintenance Optimization of Degrading Systems in a Large State Space." *Computers & Industrial Engineering* 161:107622. https://doi.org/10.1016/j.cie.2021.107622.

Zhao, Y., and C. Smidts. 2022. "Reinforcement Learning for Adaptive Maintenance Policy Optimization under Imperfect Knowledge of the System Degradation Model and Partial Observability of System States." *Reliability Engineering & System Safety* 224:108541. https://doi.org/10.1016/j.ress.2022.108541.