Politecnico di Torino

Politecnico di Torino

School of Management and Production Engineering

Master of Science Course in Engineering and Management

Application of Reinforcement Learning to Dynamic Job Shop Scheduling

Supervisors:
Prof.ssa Giulia Bruno
Niccolò Giovenali

Candidate:
Carola Tralongo

Academic year 2025/2026

Abstract

The increasing complexity and uncertainty of modern manufacturing systems have significantly challenged traditional production planning and control methods. In particular, the Dynamic Job Shop Scheduling Problem (DJSSP) has emerged as a critical issue due to the presence of unpredictable events such as stochastic job arrivals, variable processing times, machine breakdowns, and frequent order changes.

In this context, Reinforcement Learning (RL) has gained increasing attention as a promising paradigm for dynamic scheduling, thanks to its ability to model scheduling as a sequential decision-making process and to learn adaptive policies through interaction with the environment. This thesis investigates the application of reinforcement learning techniques to the Dynamic Job Shop Scheduling Problem, with the objective of assessing their effectiveness, robustness, and managerial relevance compared to traditional scheduling approaches.

The work focuses on analyzing and comparing multiple reinforcement learning frameworks, including value-based, policy-based, graph-based, and multi-agent approaches, across different dynamic scenarios. Particular attention is devoted to how these methods handle uncertainty, scalability, real-time decision-making, and generalization to varying system configurations.

The thesis highlights the advantages of reinforcement learning from a managerial perspective, showing that RL-based scheduling systems outperform traditional dispatching rules and remain competitive with metaheuristic methods, especially in highly dynamic environments. Once trained, reinforcement learning agents are able to generate scheduling decisions with low computational effort, making them suitable for real-time industrial applications. The findings indicate the potential of reinforcement learning methodologies, demonstrating improved robustness to disturbances, better resource utilization, and enhanced flexibility in adapting to changing production conditions.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Purpose of the Thesis

The main focus of this thesis is the application of Reinforcement Learning (RL) methods to address the Dynamic Job Shop Scheduling problem (DJSSP).
The objective is to analyze and evaluate different reinforcement learning frameworks and support their suitability for environment in which change is inherent, typically characterized by unpredictable conditions such as new job arrivals, variable processing times, and machine breakdowns.
In contrast to much of the existing literature, that focuses on specific benchmarks or investigates the application of a single approach, this thesis aims to provide a broader perspective.
This thesis offers an overview of how existing reinforcement learning methods perform across different dynamic settings and highlights the advantages these approaches can offer compared to traditional scheduling techniques.
The specific purpose of this thesis is to motivate the adoption of reinforcement learning methods in dynamic scheduling environments, with particular emphasis on their practical benefits for managerial decision-making.
This work aims to contribute to the fields of reinforcement learning and job shop scheduling by demonstrating the propensity of reinforcement learning algorithms to outperform traditional heuristics and exact methods under conditions of uncertainty and provide evidence of the applicability and scalability of reinforcement learning approaches in real-world production systems. Taken together, these findings support the view that RL-based scheduling systems are not only theoretically effective but also ideal for industrial deployment.

1.2 Structure of the Thesis

The thesis is organized into five chapters, guiding through the definition and integration of reinforcement learning within the job shop scheduling problem and the benefits deriving from its application.

- Chapter 1: Provides a general overview of the study, introducing the purpose and context of the thesis.
- Chapter 2: Covers the theoretical background, presenting the foundational concepts of the work. This chapter offers an overview of Industry 4.0, throughout its evolution and development, and then introduces scheduling theory and the scheduling problem, discussing key principles and their impact. It covers the fundamentals of artificial intelligence and machine learning, followed by a detailed overview on reinforcement learning algorithms and their functionalities.
- Chapter 3: Focuses on the State of the Art, synthesizing the existing relevant literature on reinforcement learning and the dynamic job shop problem, setting the basis for the contributions of this work.
- Chapter 4: Describes the contributions of this thesis, highlighting the advantages of reinforcement learning in dynamic scheduling environments.
- Chapter 5: Summarizes the thesis and highlights the conclusions

This thesis provides a comprehensive exploration of the application of reinforcement learning to scheduling under dynamic conditions, progressing from theoretical concepts to a detailed analysis of its methodology and contributions. The thesis concludes with a discussion on the motivations for companies to adopt reinforcement learning for scheduling optimization.

Chapter 2

Theoretical background

2.1 Industry 4.0

2.1.1 Overview

Industry 4.0 marks a significant step forward in manufacturing and industrial operations, leading to a scenario characterized by the integration of digital technology within physical systems.
The concept of Industry 4.0 refers to the latest stage in the ongoing evolution of industrial production. It was first handled by the German government, with the potential to reshape production processes and redefine the communication between humans and machines, as well as the interactions among suppliers, manufacturers, and customers.
Historically, industrial development has evolved through several distinct stages: mechanization that enabled large-scale production and laid the foundations for industrialization (Industry 1.0), electrification and the rise of assembly-line manufacturing (Industry 2.0), and the emergence of automation driven by electronics and IT (Industry 3.0), all of which transformed industrial performance and productivity.
Industry 4.0 stands out for its integration of intelligent and adaptive systems capable of autonomous operations, continuous optimization, and real-time decision-making. It embodies the convergence of cyber-physical systems (CPS), the Internet of Things (IOT), big data analytics, and artificial intelligence (AI). By emphasizing interoperability, data transparency, decentralization, modularity, and flexibility, Industry 4.0 aims to create smart factories where interconnected machines and systems communicate, self-optimize, and autonomously adapt to changing demands, with minimal human oversight.
Although the adoption of Industry 4.0 brings numerous advantages, including

increased efficiency, customization, and enhanced innovation, it also presents significant challenges that must be addressed, such as high implementation costs, cybersecurity risks, workforce skill gaps, and operational complexity.
Ultimately, Industry 4.0 represents a major milestone in industrial evolution, offering significant potential for the development of smarter production systems, which must be balanced with effective management of the risks involved and leveraged to drive further progress in both society and industry.

2.1.2 Core Technologies

Cyber-Physical Systems (CPS)
Cyber Physical Systems (CPS) are mechanisms where physical objects, humans and software are closely connected to create new levels of interactions. The real world and its physical objects consistently interacts with the virtual world to exchange information and indicate the best decision [1].

Internet of Things (IoT)
The IoT (Internet of Things) refers to a networked world. Machines, mobiles and humans are embedded with electronic sensors, actuators or other digital devices to generate data. The purpose of this network is to collect, exchange and analyze data for improvements or changes in the environment of the network. [1].

Big Data and Analytics
Big data refers to large and complex datasets that can not be efficiently managed or analyzed using traditional data-processing tools. Analytics encompasses the techniques and methods used to analyze such data.

Artificial Intelligence (AI) and Machine Learning (ML)
Artificial intelligence (AI) refers to the capability of computer to perform tasks that typically require human intelligence, such as reasoning, learning, and decision-making. Machine learning (ML) is a subset of AI that enables systems to learn patterns from data and improve their performance without being explicitly programmed.

Cloud Computing and Systems
Cloud Computing describes the remote computing via wireless networks, where services are visualized through scalable resources over the internet. It is also characterized through its extremely fast response and the external storage of data, mostly combined with Big Data [1].

Additive Manufacturing (3D Printing)
Additive Manufacturing (AM), also known as 3D-printing, it is mainly suitable for the individual production of small batch sizes with high complexity. It enables the direct production of digital 3D models and therefore fits into the digital value chain [1].

Augmented Reality (AR) and Virtual Reality (VR)
Another key technology is Augmented or Virtual Reality (AR, VR), where reality is supported and mixed by the use of digital 3D models. AR is mostly used in the assembly of complex components or as a technical 3D-documentation in real time, often as an aid to predictive maintenance. Supporting time, errors and cost reduction can be achieved [1].

2.1.3 Benefits and Challenges

Benefits

By integrating advanced digital technologies, Industry 4.0 provides a wide range of benefits that are reshaping the way industries operate, enhancing efficiency, flexibility, and innovation.
One of the advantages deriving from the application of Industry 4.0 is the promotion of mass-customization, which ensures highly tailored scale production while maintaining cost-effectiveness. Moreover, the combination of data analytics, artificial intelligence, and predictive maintenance reduces both equipment failures and production defects, along with their costs, improving overall quality. Industry 4.0 also fosters sustainability by improving energy efficiency and waste reduction, thereby supporting environmental goals.
A major benefit is its ability of exploiting real-time monitoring and decision making to optimize production processes, improve resource utilization, and minimize downtime.

Challenges

While the previous description may appear promising, understanding its related challenges is crucial to ensure the effective application of Industry 4.0, as real world risks must be addressed.
One of the primary concerns in shifting toward smart manufacturing is the substantial investment required to adapt equipment, infrastructure and human capital to the new environment. Industry 4.0 demands highly skilled professionals, making professional training essential to bridge the workforce skill gap.
The rise of interconnected system amplifies the risk of cyber attacks and data manipulation, representing a critical challenge in this context. Implementing reliable cybersecurity procedures is essential to secure sensitive data and protect

operational stability.
Furthermore, the combination of interconnected systems, highly flexible and customized production environment, and data-driven decision-making makes production planning and control hard to manage, adding complexity and potentially leading to delays, errors, or incorrect decisions.

2.2 Scheduling

2.2.1 Overview

Scheduling is a core component of operations management and production planning focused on efficiently allocate resources and optimize processes and timelines. It represents a fundamental coordination process among machines, workforce, and materials to ensure order fulfillment, timely production, and cost reduction.
Essentially, scheduling involves making decisions within given constraints while optimizing operational targets, such as production rate, machine and resources utilization, and total completion time.
Scheduling techniques range from traditional approaches, such as dispatching and priority rules, to mathematical model, heuristic algorithms and AI-based tools. Continuously evolving to meet the changing demands of the industry, recent methods aim to address uncertainty and dynamism of modern production systems while aligning with organizational objectives.

2.2.2 Traditional Scheduling Approaches

Traditional deterministic scheduling methods are characterized by parameters such as processing times, job sequences, machine availability, and due dates, that are assumed to be known and constant.
These methods, summarized in Table 2.1, perform effectively in stable and controlled environments, like production plants, where processes follow predictable patterns and input variability is minimal. This certainty simplifies the development and implementation of scheduling models and the identification of optimal solutions.
The most widely used approaches are priority or dispatching rules, which determine the job sequence according to specific criteria such as shortest processing time, earliest due date, or first-come-first-served. The predictability of parameters also facilitates mathematical programming techniques, including linear, integer, and mixed-integer formulations, to be applied. They aim to find optimal schedules under fixed conditions.
Gantt charts and related tools are also employed to visualize and analyze scheduling outcomes, supporting planning and resource allocation rather than generating schedules themselves.

Moreover, heuristic methods provide fast, near-optimal solutions by applying problem-specific strategies, when exact optimization becomes computationally difficult.
Collectively, these techniques form the foundation of job shop scheduling and serve as a reference point for more advanced, data-driven, and adaptive approaches (metaheuristics, AI, real-time adaptive scheduling) in modern environments.

Category	**Examples**	**Function**
Dispatching Rules	Shortest Processing Time (SPT) Longest Processing Time (LPT) Earliest Due Date (EDD) Least Slack Time (LST) First-Come-First-Served (FCFS) Critical Ratio (CR)	Job sequencing
Heuristic methods	List scheduling Local re-sequencing	Near-optimal solutions
Mathematical programming	Linear programming Integer programming Mixed-integer programming	Optimal schedule
Visualization tools	Gantt charts	Visualization and analysis

Table 2.1: Scheduling tools and techniques

2.2.3 Dynamic Scheduling

Unlike deterministic scheduling, dynamic scheduling faces variability in the process. It adapts production schedules in real-time to unexpected events such as demand changes, machine breakdowns, or resource unavailability. These models continuously update scheduling decisions based on new available information, thereby ensuring flexibility and responsiveness.
Among the most common scheduling paradigms figure robust, pre-reactive, and completely reactive approaches. Robust scheduling aims to generate schedules that maintain reasonable performance regardless of uncertainties. This is typically done by adding buffer or slack times, or by incorporating stochastic elements into the objective function or constraints. However, it is challenging to account for all possible disturbances, and this strategy may lead to inefficient use of resources. Pre-reactive scheduling involves designing an initial scheduling plan and then modifying it when changes or disruption occur, which is essentially a rescheduling process. Most pre-reactive methods are straightforward and focus on productivity as the optimization goal, but they often overlook the fact that the new scheduling plan may deviate significantly from the original one. Additionally,

frequent rescheduling can increase computational burden and negatively impact manufacturing systems. On the other hand, fully reactive scheduling is widely adopted in real-world production due to its small computational effort and easy implementation. Nevertheless, its effectiveness strongly depends on the objective function, the shop floor configuration and instance-specific factors, meaning it does not always yield optimal results [2]. In dynamic environments, parameters such as job arrivals, processing times, or machine availability often follow probabilistic distributions, preventing the elaboration of a fixed schedule in advance.
To address this complexity, adaptability becomes crucial. Advanced approaches, including metaheuristics (such as genetic algorithms or simulated annealing), and increasingly, artificial intelligence and machine learning methods, need to be exploited to support real-time decision-making and optimization.
Dynamic scheduling plays a crucial role in industries characterized by intrinsic uncertainty, such as healthcare, logistics, and manufacturing, where rescheduling of tasks may be required due to unpredictable demand, emergencies, or shifting priorities.

2.3 Scheduling Problem

2.3.1 Overview

The Job Shop Scheduling Problem (JSSP) is a classical combinatorial optimization problem that arises in production systems and industrial resource allocation. Its goal is to determine an optimal sequence for executing a set of jobs on available machines in order to minimize a specific performance criterion, typically the total completion time.
Formally, consider a job shop as a set of machines $M = \{M_1, M_2, \ldots, M_m\}$ and a set of jobs $J = \{J_1, J_2, \ldots, J_n\}$. Each job J_i consists of a sequence of operations $O_i = \{O_{i1}, O_{i2}, \ldots, O_{in_i}\}$, which must be performed in a predefined order. Every operation O_{ij} is assigned to a specific machine and owns a known processing time p_{ij}. The scheduler must determine start times and sequences for all operations across the machines while respecting precedence and resource constraints [3].
Therefore, the problem consists of n $\times$ m operations in total, under precedence and resource constraints. The main limitations include precedence constraint that each job operation must follow, capacity constraint, meaning that each machine can process only one operation at a time, and no preemption, thus, once an operation starts, it cannot be interrupted.
Let t_{ij} denote the starting time of operation O_{ij}, and C_{ij} its completion time. The makespan, $C_{\max}$, represents the maximum completion time across all operations. The makespan is usually utilized as the optimization criteria for the JSSP. Hence, the objective of a scheduler handling the JSSP is represented by the following function [4]:

$$C_{\max} = \min\left(\max\left(t_{ij} + p_{ij}\right)\right), \quad \forall J_i \in J,\ O_{ij} \in O_i$$

Other frequently adopted objective functions include lateness and tardiness, which measure how well the schedule meets due dates. Lateness is defined as the difference between a job's completion time and its due date, and can therefore take both positive and negative values, depending on whether the job is late or early. Tardiness, instead, represents only the positive part of lateness, penalizing delays while assigning zero cost to jobs completed on time.

2.3.2 Categories of Scheduling Problems

The job scheduling problem can be applied to a variety of scenarios and has therefore been classified in several ways within the literature. These classifications are generally based on factors such as the number and configuration of machines,

the precedence or priority constraints among jobs, the characteristics of processing times, and the objectives pursued during scheduling. Understanding these categories is essential for selecting the most suitable scheduling approach and identifying the expected solution outcomes for the problem under consideration.

Single-machine scheduling

Single-machine scheduling represents the simplest structure class within scheduling. This category is characterized by a single resource responsible for processing a set of jobs, one at a time. Its simplicity makes it an essential foundation for theoretical study. Despite the apparently straightforward setting, the optimization of single-machine problems can become challenging when additional constraints such as job precedence relations or due dates are taken into account. These elements significantly increase the problem's combinatorial complexity and are often used as starting point to a more complex analysis.

Parallel machine scheduling

Parallel machine scheduling involves a set of machines capable of processing different jobs at the same time. In this more complex setting, machines may or may not be identical or related, and can also differ in characteristics such as processing times and capacity. To address this additional complexity, specialized scheduling algorithms and optimization techniques are often required to allocate jobs efficiently and optimize performance criteria.

Job shop scheduling problem (JSSP)

Job shop scheduling allows jobs to be processed according to individual sequencing requirements, with each job following its own specific processing route. As jobs may visit certain machines at predefined times, routing complexity increases, resulting in intricate scheduling constraints. To overcome these challenges, advanced solution methods, such as reinforcement learning, are often required. This makes job shop scheduling one of the most flexible yet computationally demanding environments in production scheduling, serving as a benchmark for evaluating new optimization techniques.

Flexible job shop scheduling problem (FJSSP)

Flexible job shop scheduling is a more complex extension of classical JSSP. It introduces an additional level of flexibility to the problem by allowing each job to be processed on one of several alternative machines, unlike traditional JSSP

where every operation must be executed on a specific machine. This formulation requires two decision stages: appropriate machine selection and identification of the scheduling sequence while considering priority constraints, machine availability, and other operational restrictions. The resulting combinatorial optimization problem grows significantly in complexity, often necessitating the use of advanced or hybrid optimization techniques.

Dynamic job shop scheduling problem (DJSSP)

Dynamic job shop scheduling represents another variant of traditional JSSP that explicitly accounts for real-time uncertainties in the production environment. These may include unexpected job arrivals, stochastic processing time, machine breakdown, and other disruptions that require the schedule to continuously adapt to new information. As a result, the scheduling process becomes an ongoing decision-making activity rather than a one-time optimization task. This dynamic nature significantly increases the complexity of the problem, often requiring fast, reactive, and robust strategies to maintain acceptable performance in highly variable and unpredictable conditions.

2.3.3 Complexity and challenges

The Job Shop Scheduling Problem (JSSP) is known to be NP-hard, meaning that as the size of the problem grows, the mathematical complexity of its combinatorial optimization increases. For this reason, much of the existing research has focused on static scenarios and meta-heuristic algorithms, including genetic algorithm, particle swarm optimization, and simulated annealing, to obtain viable solutions. However, unforeseen disruption may arise in manufacturing systems, making the initial plan infeasible. Even minor disturbances may invalidate a schedule, especially with the rise of smart manufacturing, where high adaptability is essential to maintain efficiency.
Moreover, meta-heuristic algorithms represent a challenge because the problem structure needs to be modified whenever conditions change, increasing the complexity of managing modern production uncertainties, such as machine breakdowns or order changes.
As companies increasingly strive to meet the demand of customer-centered markets, offering greater order flexibility has led to more frequent modifications and cancellations.
For the above mentioned reasons, the development of dynamic scheduling strategies capable of effectively addressing such evolving challenges has become necessary [2].

2.3.4 Solution Approaches

Over the years, a wide range of solution methodologies have been proposed to address the Job Shop Scheduling Problem (JSSP), each characterized by different levels of optimality and flexibility. As shown in Table 2.2, these approaches can be grouped into four main categories, ranging from exact techniques to more advanced ones.

Exact methods explore the entire solution space to find the optimal solution with mathematical certainty. However, their computational complexity grows exponentially with the problem size, making them suitable only for small-scale models that require optimality. Despite the accuracy, they are hence impractical for larger, real-world applications due to their computational time and intensity.

Heuristic methods offer a more efficient alternative. While they do not guarantee optimality, they provide fast and easily implementable solutions by not exploring all possibilities in the solution space. They ensure acceptable, but suboptimal results in a reasonable amount of time, making them appropriate for large scale problems where speed is is favoured to optimality.

Meta-heuristic approaches further extend heuristics by incorporating robust mechanisms that balance local and global exploration of the solution space, escaping local optima and producing good quality solutions. Generally, meta-heuristics are inspired by natural or physical processes. They usually offer greater robustness and flexibility than simple heuristics, although at the cost of increased computational time. Therefore, they are well suited for medium or large scale problems where good quality solutions are required but exact methods are too slow.

Although the previous techniques have been proven effective in various contexts, their performance can decline when dealing with dynamic, real-world, large scale problems in which more adaptable methods are required.
As such, recent research has focused on advanced approaches, referring to hybrid methods that combine multiple strategies or reinforcement learning methods capable of exploiting sequential decision-making mechanisms and real-time feedback. These methods often outperform traditional methods, particularly in complex real-world scenarios that demand adaptability and customization. Despite their computational cost, they are crucial in situations where achieving high performance outweighs complexity constraints.

Category	Resolution methods
Exact methods	Branch and Bound (B&B)
	Branch and Cut (B&C)
	Branch and Price (B&P)
	Integer Linear Programming (ILP)
	Constraint Programming (CP)
Heuristic methods	Priority Dispatching Rules (PDR)
Metaheuristic methods	Local Search (LS)
	Tabu Search (TS)
	Genetic Algorithms (GAs)
Advanced methods	Machine Learning
	Reinforcement Learning
	Multi-agent Systems
	Neural Networks
	Hybrid Methods

Table 2.2: Classification of solution methodologies

2.4 Reinforcement Learning

2.4.1 Introduction to Machine learning

Machine learning (ML) is a branch of artificial intelligence that gives computers the ability to learn and make decisions autonomously, without the need for direct programming, through statistical methods. It is based on the principle that computers are capable of analyzing data, detect patterns, and form conclusions with limited human involvement [5].
Machine learning can be divided into three main categories (Figure 2.1):

- Supervised learning exploits labeled dataset to train models in which each input is paired with a corresponding output. Through this process, the model learns to map the input to the output, enabling forecast of outcomes and precise classification of new, unseen data [5]. Common supervised learning algorithms include decision trees, neural networks, support vector machines (SVMs), and gradient boosting techniques. Supervised learning is generally divided in two main categories: regression, aiming to understand the relationship between variables, and classification, grouping data into distinct classes.

- Unsupervised learning involves training the machine with unknown input samples or labels to achieve clustering of unlabeled dataset and tracing patterns. Typical applications include customer segmentation and techniques for big

data visualization.

- Reinforcement learning follows a trial-and-error learning paradigm in which an agent interacts with a dynamic environment to achieve a specific goal without explicit instructions. By receiving feedbacks on its actions in the form of rewards or punishments, it reinforces favourable outputs and discourages detrimental ones [5]. This approach is well-suited for sequential decision-making problems, where each action influences future states and cumulative rewards.

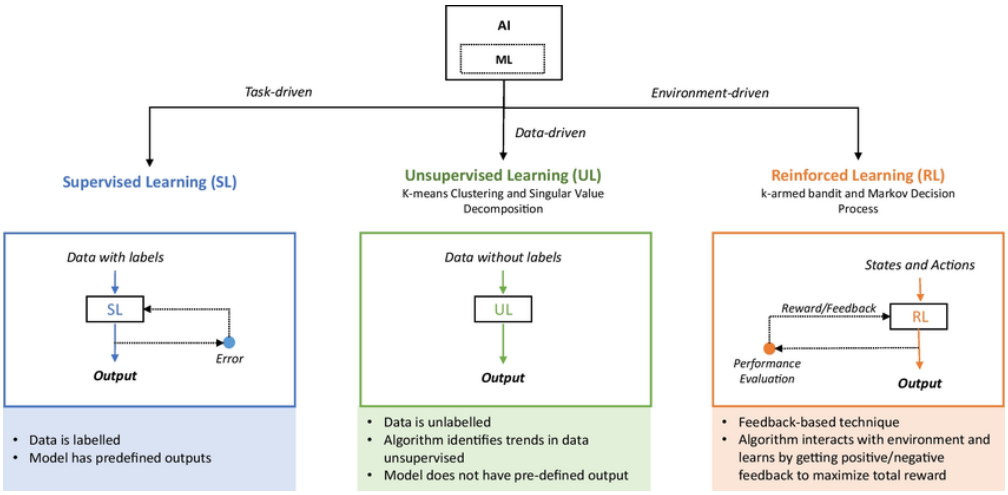


Figure 2.1: Machine learning categories [6]

2.4.2 Fundamentals of Reinforcement Learning

Reinforcement learning (RL) is a subcategory of machine learning that deals with the behaviour of a learning agent in a dynamic environment with the goal of maximizing cumulative rewards. RL distinguishes itself from supervised learning because it conducts learning through a reward mechanism rather than using labeled data. An RL system is fundamentally based on these core elements:

- Agent: Decision-maker that interacts with the environment
- Environment: External system that provides feedbacks based on the agent's interactions
- State (S): Representation of the current situation of the environment
- Action (A): Set of all possible moves that the agent can make at any given state

- Reward (R): Feedback signal indicating the success or failure of the action taken
- Policy (π): Strategy that defines how the agent selects the next action based on the current state

Reinforcement learning problems are typically formulated as Markov Decision Processes (MDPs), which is a mathematical framework to model sequential decision-making.
MDP can be represented as a five tuple $\langle S, A, P, R, \gamma \rangle$ [7]:

- S represents the set of all observable states
- A represents the set of all possible actions
- P is the state transition function, it represents the probability of transitioning from the current state to the next state after executing the action
- R represents the immediate reward
- γ is a reward discount factor, it represents the degree of impact of future rewards on current rewards.

Figure 2.2 illustrates the action selection mechanism of the reinforcement learning agent based on the system state.
The agent selects the current optimal action A by observing the environment state S. The environment feeds back the agent's reward according to the action taken, the agent then receives the reward value R, and the environment transitions to the next decision moment.

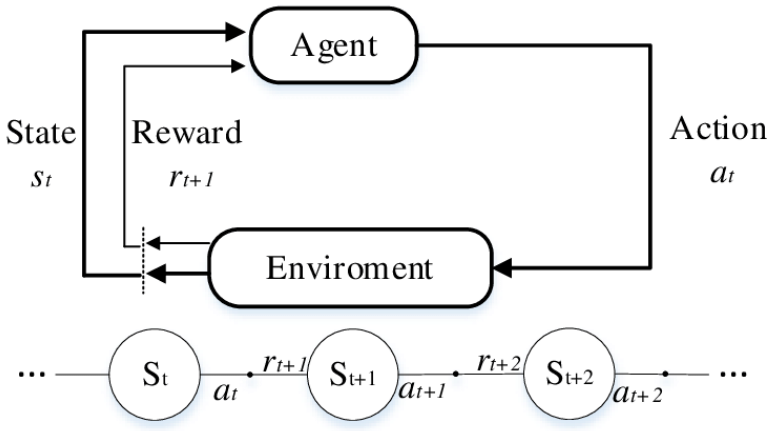


Figure 2.2: State-Action Diagram [8]

Through the continuous interaction of this process, the agent can collect a large amount of interaction data, and use them to update the model to obtain the optimal policy, thus realizing adaptive policy optimization [2].

2.4.3 Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) is a branch of artificial intelligence that combines reinforcement learning with deep learning techniques, such as deep neural networks, to address high-dimensional and complex problems.
Their ability to model and interact with complex systems, makes these methods well suited for several real-world applications.
The DRL framework includes various approaches, such as deep Q-networks (DQN), policy gradient methods (such as Proximal Policy Optimization), actor-critic architectures, among others.

In the context of the job-shop scheduling problem, Deep Reinforcement Learning enables real-time scheduling decisions and policy adjustments based on the current system state, particularly when dealing with unforeseen events [2].
Accordingly, DRL can be classified into the following categories:

- Value-Based Methods, such as Q-learning and Deep Q-networks (DQN), focus on learning an optimal value function through function approximation. Thus, the agent estimates state-action values and derives the policy implicitly by selecting the action with the highest estimated value.

- Policy Gradient Methods, such as REINFORCE, directly optimize a parameterized policy using gradient-based techniques, without relying on a separate value function.

- Actor-Critic Methods combine both policy-based and value-based approaches. The actor is responsible for action selection, while the critic estimates the value function.
  The value function is used to improve the actor's policy parameters reducing the variance of the policy gradient estimates.

- Trust-region and proximal methods further constrain policy optimization. By limiting the magnitude of policy updates and policy changes, these methods ensure improvement and robust performance.

Together, these categories represent different approaches to policy optimization in deep reinforcement learning, encompassing both indirect methods, where the policy is derived from value estimates, and direct methods, where the policy is optimized explicitly.

2.4.4 Application of RL

Reinforcement learning has been actively integrated into many real-world systems for its ability to handle complex decision-making problems.

Robotics and Autonomous systems

Robotics and autonomous systems are one of the main application area of reinforcement learning.
In industrial contexts, robotic arms are employed to perform manipulation tasks that require high precision. Moreover, robots and drones can be integrated with reinforcement learning to learn how to move and avoid obstacles in complex environments.
Reinforcement learning also applies to autonomous driving, where it is used for adaptive cruise control and lane changing.
Reinforcement learning techniques can be also employed in traffic light optimization, to reduce congestion and improve traffic flow in large urban areas.

Gaming

Gaming and simulation environments provide a controlled setting for training reinforcement learning algorithms to learn complex strategies, which can be transferred to real-world applications such as autonomous game-playing bots.

Finance and Trading

Reinforcement learning is applied to algorithmic trading and portfolio management, where the agent can interact with changing market conditions and learn when to execute trades rather than relying on predefined strategies.

Healthcare

Healthcare represents a promising application area for reinforcement learning.
It can be used in adaptive drug dosing, where treatments are optimized based on patient responses, and it can also be helpful in hospital resource allocation, including staff scheduling and bed management, to maintain high care standards.

Energy and Infrastructure

Reinforcement learning has been increasingly adopted in energy and infrastructure management, such as smart grid load balancing, where agents optimize energy distribution in response to fluctuating demand, and data center cooling optimization, to reduce energy consumption while maintaining performance. It has also been

used in renewable energy systems for storage control to support efficient integration of renewable sources into power grids.

Chapter 3

State of the Art

3.1 Introduction

Dynamic Job Shop Scheduling Problems (DJSSP) constitute a complex subset of production scheduling problems due to their complexity and the presence of unpredictable events , such as new job arrivals, machine breakdowns, and uncertain processing times. Unlike static scheduling scenarios, dynamic environments should be able to adapt to changes in the state of the system while remaining efficient. Traditional scheduling approaches often struggle to cope with such conditions, especially when real-time responsiveness is required.
In recent years, Reinforcement Learning has emerged as a promising paradigm to address the DJSSP, thanks to its ability to learn adaptive scheduling policies directly from interactions with the environment. By modeling scheduling as a sequential decision-making problem, RL based approaches can dynamically adjust decisions in response to system disturbances and evolving production states.
This chapter provides the review of research papers based on RL approaches for solving the Dynamic Job Shop Scheduling Problems. The selected papers are analyzed according to their modeling strategies, learning algorithms, and design choices.
The objective of this review is to highlight the main research trends, compare optimization paradigms, and identify current challenges that motivate the contribution of this thesis.

3.2 Paper Selection Methodology

The literature search was performed using Scopus and Web of Science databases. The search primarily focused on recent publications to capture the latest methodological developments, while also considering previous works to further support the

application of reinforcement learning in dynamic job shop scheduling.
The query employed for the search was "Reinforcement Learning" AND "Dynamic Job Shop Scheduling", to specifically identify papers addressing scheduling problems under dynamic conditions through learning-based decision-making approaches, which returned a total of 45 records.
A screening process was carried out based on the year of publication, relevance of the title, and a preliminary analysis of the abstract and full text. Studies that did not explicitly address dynamic job shop environments or did not propose or evaluate reinforcement learning–based solutions were excluded. As a result, 17 papers were selected for detailed analysis, as shown in Figure 3.1.

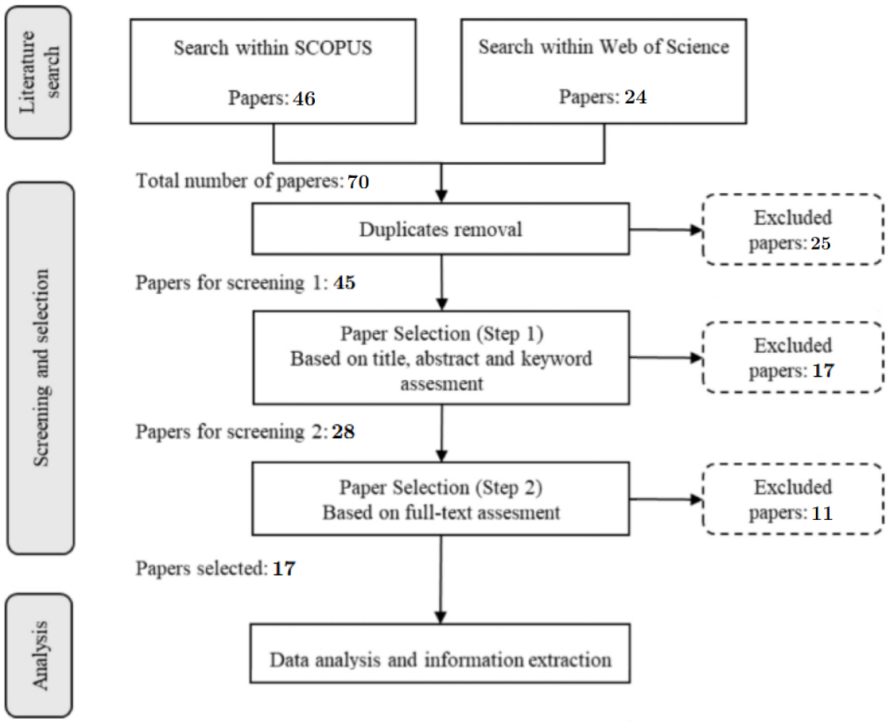


Figure 3.1: Paper selection methodology

3.3 Literature Review

The 17 selected paper are summarized in the Appendix.
The table reports the year of publication, the research focus, the proposed algorithm, and additional relevant factors such as state and action space, results and contributions of the study. This chapter reorganizes and synthesizes the information from the reviewed literature.

Single-agent value-based reinforcement learning approaches represent some of the earliest attempts to apply deep reinforcement learning to DJSSP. In these studies, the scheduling problem is typically formulated as a Markov Decision Process (MDP), where the agent observes the state of the system and selects the dispatching actions to be processed in order to optimize performance objectives, such as makespan or machine utilization [7] [9].
A common feature of these works is the use of Deep Q-Networks (DQN) to learn scheduling policies under stochastic job arrivals and processing times.
Wu et al. (2024) [7] demonstrate that DQN-based algorithm for dynamic dispatching under stochastic job arrivals outperform classical priority rules in dynamic environments and extend this framework by modeling uncertainty in processing times, improving robustness. Wu and Yan (2023) [9] address scalability issues related to fixed-size state representations by introducing a spatial pyramid pooling mechanism, capable of handling variable-sized job shop instances.
These approaches consistently outperform traditional dispatching rules, highlighting the potential of learning-based decision-making in dynamic settings. However, despite their effectiveness, single-agent approaches suffer from scalability issues related to centralization and action space exponential growth, limiting their applicability to large scale or highly dynamic production systems.

To overcome some of these challenges, subsequent studies explore **policy gradient methods** for the dynamic job shop scheduling, particularly Proximal Policy Optimization (PPO) [2] [10] [11]. These approaches directly optimize stochastic policies, hence they are better suited to environments characterized by frequent disturbances and consequently offer improved training stability.
Common across this category is the ability to better handle frequent order arrivals and time-varying system states. Yuan et al. (2025) [2] propose a PPO-based scheduling framework that effectively manages stochastic job arrivals and machine disruptions, while Chen et al. (2024) [10] integrate Long Short-Term Memory (LSTM) networks into the PPO architecture to capture temporal dependencies in the production process, leveraging historical information when making scheduling decisions. Compared to value-based methods, PPO-based approaches demonstrate

superior performance in highly dynamic scenarios, although they remain sensitive to reward design and hyperparameter tuning, which can limit their applicability.

More recent studies model DJSSP using **graph-based** reinforcement learning, leveraging Graph Neural Networks (GNNs) and Graph Attention Networks (GATs) [12] [13] [14] [15] [16]. In these, jobs, machines, and operations are represented as nodes, and precedence constraints and resource-sharing relationships as edges, allowing the model to capture complex structural dependencies.
A key commonality of these approaches is their ability to generalize across job shop instances of varying sizes by exploiting relational information. Liu et al. (2024) [13] propose a GAT-based DRL framework that dynamically prioritizes critical operations through attention mechanisms, while Su et al. (2023) [15] introduce a heterogeneous GNN architecture designed to improve scalability and generalization. Additional improvements are reported through the use of auxiliary learning strategies and optimization techniques such as evolution strategies (Liu et al., 2024) [14]. However, these methods introduce significant architectural and computational complexity, raising concerns about real-time feasibility in industrial settings.

Multi-agent reinforcement learning (MARL) approaches have also been considered in the analysis as they aim to decentralize decision-making by assigning scheduling responsibilities to multiple interacting agents [17] [18] [19] [20]. This paradigm aligns with the structure of modern manufacturing systems characterized by distributed control and flexibility. Qin et al. (2023) [18] show that decentralized agents can outperform centralized heuristics in environments subject to frequent disturbances. Nevertheless, MARL approaches face common challenges related to non-stationarity during training, coordination complexity among agents, and convergence stability, which complicate their deployment in real-world applications.

Hybrid and simulation-based frameworks integrate reinforcement learning with advanced modeling techniques such as digital twins, discrete event simulation (DES), and transformer-based neural architectures to enhance realism and modeling capabilities [21] [22] [23]. These studies aim to bridge the gap between simulation-based research and real-world applicability.
Digital twin–based approaches allow DRL agents to interact with high-fidelity virtual representations of the shop floor, enabling realistic training and evaluation (Zhang et al., 2026) [21]. Moreover, DES-based approaches provide controlled simulation environments for modeling stochastic dynamics.
Transformer-based models have also been introduced to capture long-range temporal dependencies and complex event sequences in dynamic scheduling problem, as demonstrated by Song et al. (2023) [23].

While these approaches show promising results, their high computational requirements and implementation complexity remain open challenges.

3.4 Open Issues and Research Gaps

The literature review clearly notice the growing interest in reinforcement learning as a promising paradigm to address uncertainty in dynamic environments.
Traditional approaches are mainly employed as baselines, while hybrid frameworks are increasingly capturing the interest of the research.

Reinforcement learning has demonstrated to be able to adapt to unexpected events, however some limitations in the existing models have emerged.
First, the majority of work focus on single objective formulations, typically aiming to minimize makespan or tardiness, with limited consideration of other relevant performance indicators. Expanding the scope to multiple optimization objectives and a broad range of disturbance events can improve the decision logic and generality of the model for various production settings.
Second, simulation environments are often oversimplified and even though some models have size-agnostic properties, extremely large-scale instances may pose computational challenges.
The development and evaluation of multi-objective models that better reflect the comprehensive requirements of industrial systems can contribute to better handling complex and dynamic scheduling scenarios.

Chapter 4

Advantages of Reinforcement Learning for Dynamic Job Shop Scheduling

This chapter examines the value of applying Reinforcement Learning (RL) to the Dynamic Job Shop Scheduling Problem (DJSSP) from a management perspective. This exploration builds upon the theoretical principles and prior research discussed in earlier chapters.
Classic scheduling methods, as priority dispatching rules, heuristics and metaheuristics, have been extensively studied in industrial contexts. However, the increasing dynamism of modern production systems questions their efficacy.
The aim is to assess how reinforcement learning performs in scheduling environments and to provide management with reasons to favour RL over alternative scheduling methods.

4.1 Analysis of Reinforcement Learning Applications

This section treats the selected literature as a series of case study to support the analysis. The objective is to interpret the way reinforcement learning faces scheduling decisions in dynamic job shop environments.
For each case, the analysis focuses on the managerial value generated by the application of reinforcement learning, highlighting the benefits these techniques can offer when compared to classic scheduling methods.

Case study: A novel deep reinforcement learning framework based on digital twins for dynamic job shop scheduling problems [21]

This paper investigates the application of reinforcement learning to a real-world scheduling problem within a water heater inner tank welding production line. The system shares traits with typical dynamic job shop settings, featuring several parallel machines at each processing stage, material handling constraints, limited buffer capacities, stochastic processing times, and unexpected disturbances like machine breakdowns.

In this case, the simulated workshop environment is based on a real company water heater welding line and consists of three parallel welding machines combined with two robotic arms and two gantry manipulators, as auxiliary resources. Jobs corresponds to water heater inner tanks.
For each evaluation episode order size is different, varying between 10 and 1000. Two operations per job are required and routing is flexible, meaning that each job can be processed by any of the available machines.
Job processing times are derived from historical data and follow a uniform distribution between 58 and 62.
The impact of machine breakdowns on the production line is taken into account, showing that the approach can proactively reschedule operations minimizing the effect that such disruption may have.
For example, Machine 3 fails at t=150 s, requiring additional 250 s to be repaired. The predictive capabilities of the DT allow rescheduling to be initiated before the full impact of the failure is realized: tasks originally scheduled on Machine 3 are promptly reassigned to other machines available, enabling on-time delivery.

The production process evolves continuously over time, requiring online decisions at each scheduling event. This mirrors the operational complexity found in many modern manufacturing systems, where schedules need to adjust to real-time shop-floor conditions.

Rule-based dispatching methods,commonly utilized in production lines, struggle to cope with variability and often lead to poor load balancing and waste of available resources. Metaheuristic methods typically require long time to compute and need to be reconfigured when system conditions change, making them impractical for real-time use. This forces managers to face a trade-off between quality and responsiveness, which often ends up in prioritizing fast but suboptimal solutions. Introducing a scheduling system designed to be flexible enough for actual production environments is a way to address this trade-off.

The authors introduce a dynamic scheduling framework that combines digital twin technology with deep reinforcement learning. A high-fidelity digital twin of the welding line is built, incorporating realistic characteristics such as processing times, buffer interactions, and material transport logic. The digital twin acts as the training environment for a Double Deep Q-Network (DDQN) agent, which learns machine selection through offline interaction with the simulated production line. Once trained, the agent is deployed online to support real-time scheduling decisions. Offline learning and online execution are important, as they allow intensive training to be conducted without interfering with the system operations, but guaranteeing fast decision-making during execution.

The proposed approach is compared against a wide range of baseline methods, including heuristic dispatching rules, and metaheuristic methods such as genetic algorithms and particle swarm optimization. The tests show that the DDQN-based approach consistently achieves lower makespan values than other methods. It also remains efficient as problem size increases. It is important to notice that, for large-scale instances, the proposed method generates scheduling decisions faster than metaheuristic approaches, which can require hours of computation. This difference highlights a major benefit of reinforcement learning: once trained, the agent is able to make rapid scheduling decisions which make reinforcement learning suitable for modern production systems, requiring fast real-time responses as the system evolves.

Beyond improvements in throughput, this approach consents to distribute workload more evenly across parallel machines. This balance increases overall production efficiency, and also reduces excessive wear on machines, contributing to more sustainable asset management and longer equipment lifecycles.

The system stability in case of disruptions is particularly noteworthy. Through the use of real-time data from the digital twin, the proposed framework can detect deviations between planned and actual production and proactively reschedule tasks in response to machine failures. In simulated failure scenarios, the DDQN-based approach notably reduces the time needed to complete the jobs compared to traditional scheduling rules, mitigating the negative effect disruptions have on performance. This ability directly addresses a critical management concern: maintaining delivery reliability and operational continuity in case of uncertainty.

However, some criticalities are also present. The construction and maintenance of a high-fidelity digital twin requires a significant initial investment in modelling, data integration, and system engineering. In addition, the action space is tailored to the specific structure of the welding line taken into consideration, potentially limiting transferability to other production systems without retraining or redesign.

Case study: A deep reinforcement learning approach for dynamic job-shop scheduling problem considering time variable and new job arrivals [11]

This paper introduces a deep reinforcement learning approach to the dynamic job shop scheduling problem under two common operational disturbances: variable processing times and new jobs arrival. The context is that of mass customization, where production schedules need frequent updates because they are exposed to uncertainty and new customer requests.

In this case, evaluation is conducted on multiple benchmark instances of different scales, each representing different system configurations, while dynamic events are introduced during execution to simulate realistic operating conditions.
The number of machines and jobs is instance dependent and spans between 6 and 50 for jobs and 6 and 20 for machines.
In the dynamic experiments, benchmark instances of different size are taken into account with three scales of new job arrival (small, medium, and large), and the completion rate of the original scheduling operation is set to 25%, 50%, and 75%, for a total of 72 new instances. For each instance, processing times follow two different distributions (Uniform and Gaussian). For variant following the Gaussian distribution, 3 different standard deviations are considered, to show how much the processing time deviates from its mean value, while keeping total processing time unchanged. For those following the Uniform distribution, 3 different bounding factors w influence the processing time of each operation and total processing time is limited to the range (1-w;1+w).

From a management perspective, the main challenge is to keep performance steady when conditions change and the scale of the scheduling instance varies. For this reason, the authors design a single-agent reinforcement learning framework based on Proximal Policy Optimization (PPO) that aims to reduce reliance on manually designed state features and to improve generalization across different problem sizes.

The focus is on how the scheduling state is represented. The scheduling instance is presented as a matrix that shows, for each job and machine, information such as machine assignment and remaining processing time. This matrix is updated over time using rules that indicate whether an operation is being processed, is idle, has been completed, or is waiting for a machine. The matrix is then turned into an image and processed by convolutional neural networks, which learn how jobs and machines interact.

In addition, a Spatial Pyramid Pooling Fast (SPPF) module enables the model to handle scheduling instances of different sizes without changing the network structure. This helps to address a major limitation: the need for extensive manual feature design and the difficulty of applying the same model when the number of jobs or machines changes. This adaptable representation is not tied to a specific problem size, ensuring flexibility and easier implementation across different production environment.

Instead of directly selecting specific operations, the reinforcement learning agent chooses from a set of twelve known priority dispatching rules (PDRs). This allows the RL policy to adapt its dispatching logic to the current environment state. Managerially, this hybrid approach is quite useful as it preserves the interpretability and simplicity of dispatching rules, while also making them more effective by adjusting the rule choice to real-time conditions. Moreover, machine idleness is penalized within a predefined scheduling period, aligning the learning process with the common business objective of reducing makespan through improved utilization and reduced waiting times.

The evaluation is based on a set of public benchmark instances and on new dynamic variants that introduce uncertainty and job insertions. In static experiments, the model outperforms individual dispatching rules and shows competitive results compared with other recent DRL approaches, with growing performance gaps when the instance size increases. In dynamic experiments, two findings are relevant in a management context: the system handles well new jobs arrivals, and remains steady under processing time uncertainty. When new jobs are inserted, reusing a trained policy rather than training a new one achieves better results, particularly when the policy is trained on large-scale instances. Under stochastic processing times, the trained policy maintains a consistent advantage compared to dispatching baselines, indicating that the learned decision process is less affected by change than fixed rules.

At the same time, training time can be significant and computing limits might slow down the system. Also, new jobs arrivals rely on predefined insertion triggers that simplify real-world arrival processes, and actions are restricted to a set of dispatching rules, limiting adaptability.

Case study: Deep reinforcement learning based proximal policy optimization algorithm for dynamic job shop scheduling [2]

This study introduces a deep reinforcement learning method for the dynamic job shop scheduling problem in smart manufacturing environments, taking into account interruptions like machine breakdowns and changes in orders. Traditional scheduling approaches become quite inadequate in these systems, where frequent disturbances can rapidly invalidate established plans.
The core challenge lies in maintaining operational efficiency while reacting quickly to unexpected events, without incurring in high computational costs and instability associated with frequent rescheduling.

In this case, a dynamic job shop scheduling problem is simulated in an intelligent workshop environment. The system consists of multiple jobs and machines, where each job follows a fixed sequence of operations processed on predefined machines. System configurations are defined by randomly generating benchmark instances using the parameters shown in Table 4.1, where the number of machines ranges from 5 to 15, the number of jobs from 10 to 100, and each job comprises 5 to 10 operations. Processing times and due dates follow predefined distributions.
Dynamic conditions are considered, including machine failures, job rework, and order changes during execution.

Parameter	The parameter values
Number of machines for processing	(5,10,15)
Number of workpieces	(10,30,100)
The number of operations per workpiece	(5,10,10)
The machining time of each process on the machine	U (Pachpor et al., 2017, Wang, 2020)
Delivery due time (DDT) of workpiece	U(1,3)
Weighted coefficient weights	$\alpha = 0.5, \beta = 0.5$

Table 4.1: Dynamic job-shop environmental parameters [2]

The job shop is modelled using a state space that captures job progresses, machine status, and overall shop-floor characteristics, including remaining processing times, machine utilization, delay rates, and statistical indicators of how evenly the workload is spread. These features are aggregated using statistical pooling techniques, allowing the trained policy to manage scheduling instances of varying scales. Although this approach requires domain knowledge for feature definition, it

allows the reinforcement learning agent to perceive the global production state in a structured and efficient way.

The action space is defined as a set of priority dispatching rules. Rather than selecting specific operations, the agent chooses among known scheduling rules, such as shortest processing time or earliest due date. In this way, the agent learns which is the most suitable heuristic for varying environmental conditions. This approach is managerially appealing because it maintains interpretability and operability, while overcoming the limitations of depending on a single rule for all scenarios.

Experimental results show that the scheduler outperforms individual dispatching rules and slightly improves upon DQN-based reinforcement learning approaches, especially as problem size increases. Genetic algorithms can perform competitively on small static instances, but their performance worsens on large-scale because the solution space grows exponentially. In contrast, the proposed PPO method maintains stable performance under machine failures and order changes by dynamically adjusting rule selection in response to evolving system states. Importantly, once trained, the scheduling policy generates decisions quickly, with execution times only marginally higher than simple heuristics and significantly lower than rescheduling methods that use metaheuristics.

The evidence suggests that reinforcement learning can serve as an effective real-time decision support in dynamic job shop environments. Its main advantage lies in offering flexible scheduling that responds effectively to disruptions while remaining computationally feasible.
Overall, reinforcement learning is a convenient and practical alternative to classical scheduling methods when production environments are highly dynamic and responsiveness is a critical business requirement.

Case study: Dynamic job shop scheduling under multiple order disturbances using deep reinforcement learning [20]

This study examines the dynamic job shop scheduling problem under multiple order disturbances. Unlike much of the existing literature, frequent and overlapping disruptions are considered. From a managerial perspective, this problem mirrors contemporary manufacturing systems operating in competitive and customer-driven markets, where production schedules have to be constantly adjusted while keeping delivery and efficiency requirements.

In this case, different scenarios in which the algorithm is trained, validated and tested are generated starting from the same parameters shown in Table 4.2.

Processing times follow a uniform distribution and disturbances, including new order arrival, order cancellations and due-date change, are added into the system following a Poisson distribution with mean of 25 or 100, respectively more or less frequently.

Parameter	Value
Number of machines	5, 10
Number of initial orders	30
Total number of order disturbances (the arrival of a new order, order cancellation, changes in order due dates)	10, 50
Time interval of successive new order disturbances following a Poisson distribution	25, 100
Due date tightness	1.0, 2.0
Processing time for each operation	Unif[0,50]

Table 4.2: Parameter settings in different scenarios [20]

A multi-agent deep reinforcement learning framework based on Independent Proximal Policy Optimization is introduced. Each agent is equipped with an Actor–Critic network and is trained independently, but they all share experience data. This setup favours decentralized but collaborative learning, enhancing exploration and reducing correlation of training data.

The state is designed as a five-channel, two-dimensional image representation that captures both global system information and local operational details. To handle changes in the system size due to order fluctuations, a spatial pyramid pooling layer is included within the neural network. This allows for feature extraction regardless of input size. The available actions comprise various dispatching rules, allowing the agents to select the most appropriate scheduling logic depending on the current system state. The aim is to minimize both total tardiness and makespan, with adjustable coefficients to reflect different business priorities.

Experiments are conducted across 72 diverse simulated production setups, varying the number of machines, disturbance frequencies, deadlines tightness, and how different goals were weighted. The results demonstrate that the proposed IPPO based approach works better than traditional PPO algorithms and individual dispatching rules in most scenarios. This suggests that reinforcement learning can

deliver consistent and robust scheduling results across many different operational conditions, even when no single dispatching rule performs well, which is fundamental for the management. The multi-agent structure further enhances robustness by improving exploration and reducing sensitivity to disturbances.

The application of reinforcement learning is, hence, convenient for managers dealing with dynamic job shop environments with frequent and multiple disturbances. It helps with making decisions that adapt to the environment and considers multiple goals, something that traditional scheduling methods struggle to provide.

Case study: Harnessing heterogeneous graph neural networks for Dynamic Job-Shop Scheduling Problem solutions [12]

This paper presents a deep reinforcement learning framework that incorporates heterogeneous graph neural networks to tackle the dynamic job-shop scheduling problem. Conventional methods often use fixed-size matrices or manually designed feature vectors. Instead, the authors model the scheduling environment using a disjunctive graph representation, where operations are nodes, and different types of edges represent precedence and machine-sharing constraints. From a management viewpoint, this model directly addresses a critical limitation of traditional scheduling systems: the inability to account for complex dependencies and adapt when the structure of the system changes.

In this case, several instances of size 6x6 and 10x10 are used for training. Evaluation is conducted on Demirkol and Taillard benchmarks, with instances that range between 20x15/50x20 and 15x15/100x20, respectively. Operations per jobs equals the number of machines and processing times are benchmark specific. Dynamic experiments are then conducted on several instances, differing in problem scale: 6x6, 8x8, and 10x10. These scenarios integrate unexpected conditions of the shop flor, by adding machine breakdowns and stochastic job arrivals, with processing times following a uniform distribution between 1 and 10.

A benefit emerges from the size-agnostic nature of this approach. The model is trained on small and medium-sized instances using a new Bootstrap Curriculum Learning strategy. Then, it is directly applied to larger, unseen problems without the need of retraining.
Experimental results on public benchmarks demonstrate that this application performs better than dispatching rules, genetic algorithms, and other DRL-based schedulers. It also shows less performance variance. This means the scheduling system can handle factory growth, changes in product mix, or resource restructuring without incurring in the cost of model redesign or parameter tuning.

In scenarios characterized by machine breakdowns and stochastic job arrivals, the method maintains good performance and demonstrates strong robustness compared to metaheuristics and rule-based approaches. This stability is particularly valuable for managers, as it reduces schedule volatility, lowers delivery risks, and supports more reliable planning under uncertainty. Some considerations have to be taken into account, such as the computational cost of training and the limited interpretability of learned policies, which may affect adoption in regulated or resource-constrained environments.

For managers overseeing dynamic manufacturing systems, reinforcement learning can become a versatile and useful decision-making support, capable of handling the real complexities of job-shop environments.

Case study: Probing an LSTM-PPO-Based reinforcement learning algorithm to solve dynamic job shop scheduling problem [10]

This study explores the application of a reinforcement learning approach, combining long short-term memory (LTSM) networks with proximal policy optimization, to tackle the dynamic job shop scheduling problem. The authors concentrate on production environments where machine breakdowns, urgent order placements, and changing shop-floor situations are common. What sets this study apart is its approach to the temporal dependencies found in real production systems, an aspect often overlooked by classic scheduling methods and other scheduling strategies.

The proposed LSTM-PPO framework offers a scheduling architecture capable of considering both current and historical job shop information. An LSTM network is integrated into the actor-critic framework of PPO, allowing the scheduling agent to learn temporal patterns in machine utilization, workpiece progression, and disturbance propagation. This temporal understanding enables the agent to foresee the effects of disruptions, rather than reacting to them. For managers, this translates into a more stable and consistent scheduling and less need for rescheduling, especially in environments characterized by frequent disruptions.

In this case, production settings are based on a simulated environment. It consists of three types of machines: lathing, milling, and grinding machine. Three types of workpieces need to be processed: W1, W2, and W3. Each workpiece requires three processes to be executed in a predefined order: turning $\rightarrow$ milling $\rightarrow$ grinding. The processing times of each operation are presented in Table 4.3 and the capacity of each machine is 6.

Parameters	Process route	Times
Workpiece type W1 process and time	Turning-milling-grinding	{O1: 100, O2: 240, O3: 230}
Workpiece type W2 process and time	Turning-milling-grinding	{O1: 370, O2: 260, O3: 200}
Workpiece type W3 process and time	Turning-milling-grinding	{O1: 120, O2: 200, O3: 230}

Table 4.3: Processing times per type of jobs [10]

Twenty workpieces are randomly generated for preliminary training, and arrival and delivery dates are shown in Table 4.4.

Order	Job ID	Type	Arrival time	Due date
001	1–4	W1	0	1500
002	5–12	W2	0	3300
003	13–20	W3	700	3300

Table 4.4: Order information

Machine failures and emergency order insertions are selected to verify the effectiveness of the algorithm in dynamic conditions. In this scenario, the number of machines in the system is augmented to 6 with 20 jobs to be processed. The probability of machine failure is between 5% and 55%, recovery time of each machine is the same, and the processing can be started after recovery.
For example, Machine 3 fails at t=1350 s, and workpieces 9, 7, 18, and 10 originally planned to be processed on it need to be adjusted. Hence, the scheduling decision assigns the workpieces to Machine 4 for processing and Machine 3 returns to process at t=2150 s, to which follows rescheduling.
Moreover, an urgent workpiece (J21) is inserted at t=580 s and needs to be completed at t=1600 s. Production plans are adjusted based on requirements: first processing of workpiece J6, originally planned on Machine 2, is changed to enable the first processing of the new inserted workpiece J21. Subsequently, the entire process is rescheduled considering the current situation and the urgent delivery times of other workpieces.

The scheduling problem is formulated as a sequential decision-making process where the agent selects among multiple dispatching rules, depending on the current situation. The results show that the LSTM-PPO method converges faster and produces better outcomes than genetic algorithms, DQN-based reinforcement learning,

and traditional dispatching rules, in terms makespan and tardiness. This improvement is particularly noticeable when machine failures are more likely, indicating how well RL handles major disruptions.

Thereby, the LSTM-PPO scheduler can absorb disturbances with minimal deviation from the original production plan, maintaining high machine utilization and meeting deadlines. This capability holds particular value for the management because it can help reduce the operational cost of disruptions, support customer satisfaction, and alleviate the need for rescheduling.
The argument that reinforcement learning is a convenient and effective scheduling paradigm for dynamic job shops is supported, especially when it can model time-related factors. By leveraging memory to learn, the scheduling choices are more aware of their context, more stable and efficient than classical methods, fitting with the operational and strategic needs of production managers.

Case study: Smart scheduling of dynamic job shop based on discrete event simulation and deep reinforcement learning [22]

The case setting is a dynamic job shop with random job arrivals, where decisions are made at event-based decision points, such as when a machine becomes idle and multiple jobs are waiting. Managers, in this case, deal with a common scheduling problem: the choice between dispatching rules that are fast but often suboptimal, and complex optimization methods, like metaheuristics, that improve quality solution, though being usually too slow and costly to be rerun whenever disturbances occur. The authors combine discrete event simulation (DES) with deep reinforcement learning and train a PPO agent in a simulation environment that can be adjusted for different production settings.

In this case, 10 jobs are initially present in the shop floor. The arrival of new jobs follows a Poisson distribution. The parameters of the production setting are shown in Table 4.5.

Parameter	Value
Number of initial jobs (n_{init})	10
Number of machines (m)	$\{8, 10, 15\}$
Number of operations in a job (op_nb)	$Randint[1, m]$
Number of newly arrived jobs (n_{new})	$\{20, 50, 100\}$
Average value of exponential distribution between two successive job arrivals (E_{new})	$\{20, 30, 40\}$
Processing time of an operation (PT_{ij})	$U\,[10, 50]$
Due date tightness factor (f_i)	$\{1, 1.2, 1.5\}$

Table 4.5: Parameter setting of production configuration [22]

The number of operations per job equals at most the number of machines. Processing time of each operation follows a uniform distribution. The due date tightness factor of jobs is randomly selected from the set 1,1.2,1.5. The PPO algorithm is initially trained in a simulated job shop environment with 8 machines and 20 new jobs, where the average value of inter arrival time is 20. To further verify the generalization ability of the algorithm, 27 cases are randomly generated based on different parameter combinations.

After being trained , the agent makes real-time dispatching decisions with low computing power, which is crucial when unexpected problems happen often and responsiveness is a performance requirement, such as cases in which due-dates are close or specific service levels are required.
The action space is defined as the selection among different dispatching rules. So, the RL model learns which rule to apply based on current conditions. This is helpful because it permits managers to connect the agent's actions to the interpretability of known scheduling logics (FIFO, SPT, LPT,..), making the system easier to check, explain to stakeholders, and fit into existing dispatching systems.

Scalability and portability are emphasized by building the state representation from statistical descriptions of jobs in the system and queue,rather than a fixed structure tied to a specific number of jobs or machines. This characteristic highlights how the model is less tied to a specific configuration, reducing the risk that it becomes unsuitable if the product mix, arrival rates, or routing change. The study further proposes a centrally trained scheduling policy that can be replicated and assigned to individual machines for decentralized control. This structure still allows for central governance, in terms of key performance indicator alignment, policy updates, and monitoring.

The policy shows generalization across different configurations without needing to be retrained, which is important for the management since real systems rarely allow continuous retraining cycles or extensive retuning. When compared with heuristic and hyper-heuristic methods, the PPO approach shows better balance between decision quality and computation time: methods like Genetic Algorithms may deliver good results, but they require repeated optimization under frequent arrivals. Similarly, learning-based heuristics, such as Genetic Programming (GP) or hyper-heuristics, may need to be retrained for each specific situation to remain competitive. While PPO can operate fast and come closer to higher-quality results in dynamic conditions, improving the flow and offering a more stable operational control under uncertainty.

The performance of the RL policy depends on the accuracy and maintenance of

the simulation environment, since training occurs in DES. Therefore, organizations need sufficient process data, reliable arrival/processing-time models, and the ability to update the simulation as changes occur.

Case study: Dynamic Job-Shop Scheduling via Graph Attention Networks and Deep Reinforcement Learning [13]

This paper addresses dynamic job-shop scheduling in an environment where disruptions, such as machine breakdowns and stochastic job arrivals, can occur. Hence, scheduling becomes an ongoing operational control function. The study posits that many conventional methods presuppose stable inputs, a condition rarely met in actual manufacturing systems, where disruptions can not be known in advance.
This makes the DJSSP a governance and performance challenge: managers must safeguard throughput and performance while adapting to environmental changes that emerge during execution.

In this case, training and validation are conducted on several generated instances with problem size of 10×10. The processing times of the operations follow a uniform distribution between 1 and 10.
For testing in a static environment, two known public benchmarks are considered: experiments included tests on 16 Taillard's benchmarks, with problem sizes ranging from 15×15 to 100×20, and on 16 Demirkol's benchmarks, with problem sizes ranging from 20×15 to 50×20.
For testing in the dynamic environment, several instances with three different problem sizes, including 6×6, 8×8, and 10×10, are used to evaluate the performance of the model. The dynamic environment involves two unexpected events, machine breakdown and stochastic job arrival. The processing time of the operation follows the uniform distribution between 1 and 10.

This research shows reinforcement learning's business potential by framing scheduling as a sequence of decisions. The system learns what action to take next based on the current shop floor situation. This helps move away from rigid, rule-based dispatching, which is fast but often not optimal. Instead, it moves towards a smart dispatcher that gets better over time by learning from experience. An actor-critic model is trained with PPO, which also makes sense from a management perspective as PPO is known for being stable to train and relatively simple to put into practice. The real advantage consists in less dependence on human expertise and less exposure to system variability.

The study construct a disjunctive graph to represent the state, which can represent both precedence and machine-sharing constraints. This matters because scheduling systems often fail due to incomplete visibility, where some approaches only capture a portion of the constraint framework. This can lead to reasonable decisions locally, but create bottlenecks or conflicts globally. In this way, the model can represent complex interdependencies. Also, the state adapts to environmental changes. For a manager, this mirrors how disruptions actually appear: they do not just cause delays, they reshape priorities and the solution space of the next decision.

An interesting benefit, from a managerial point of view, is the use of Graph Attention Networks (GATs), as they help in prioritization. Dispatchers focus on urgent queues, critical machines, or operations that could cause bottlenecks. GATs formalize this behaviour by learning which nodes and connections deserve attention and which can be ignored. This means the system can focus its decisions on what will have the biggest impact on performance, reduces the chance that irrelevant or misleading information drive the schedule. This can contribute not only to better performance but also to greater robustness when the system is stressed.

The method works for different problem sizes: it is trained on smaller instances and still performs well on larger benchmark families. This reduces the risk of being restricted only to specific configurations. In real factories, product mix and volume shift and managers can not waste resources in rebuilding the scheduling logic every time the environment changes.
Performance is validated on public benchmarks (TA and DMU) with various sizes and distributions, showing it can adjust to different data. In tests, it keeps errors low and performs comparably to other scheduling methods and a metaheuristic. In the dynamic experiments, the method keeps errors and performs comparably to both dispatching rules and a metaheuristic baseline.

Managers should understand that reinforcement learning success depends on how well the state representation captures the real constraints and if the learning model can prioritize the most critical issues. When these aspects are designed properly, this approach can offer adaptive scheduling that leads to improved responsiveness, and more stable performance even when production conditions shift.

Case study: A deep reinforcement learning model for dynamic job-shop scheduling problem with uncertain processing time [7]

In this case, the dynamic job-shop scheduling problem, specifically with uncertainty in processing times, is examined. In actual manufacturing systems, processing times fluctuates due to factors like machine state, worker expertise, material variability,

and unexpected disruptions. From a management perspective, such unpredictability creates particular difficulties, as it leads to frequent rescheduling. Traditional approaches require either complete reoptimization or extensive parameter retuning when uncertainty appears, making them expensive and slow to deploy in real-time operational settings.

The scheduling method is evaluated on standard benchmark instances of various sizes which are listed in Table 4.6.

Benchmarks	Source
ft10 (10 × 10)	Fisher (1963)
la01 (10 × 5), la06 (15 × 5), la11 (20 × 5), la16 (10 × 10),	
la21-la25 (15 × 10), la26-la30 (20 × 10), la31-la35 (30 × 10), la36-la40 (15 × 15)	Lawrence (1984)
ta21-ta22 (20 × 20), ta31-ta32 (30 × 15), ta41-ta42 (30 × 20), ta51-ta52 (50 × 15)	Taillard (1993)

Table 4.6: Benchmark instances [7]

In order to validate the generality of the method, instances with uncertainties for rescheduling are generated based on the benchmarks, considering the uncertainty ratio and the degree of uncertainty of operation processing time in the generation of new instances. Specifically, the ratio of uncertain operations is set to 25%, 50%, 75% and 100% and the uncertainty is measured by Normal and Uniform distribution with three levels of deviations.

The proposed DRL approach, based on PPO, is specifically designed to reuse previously learned scheduling knowledge when processing times change. Instead of discarding the existing policy, the system can retrain or fine-tune the policy using new experience generated in uncertainty situations. This is an important point for managers as it implies that the investments in training are not sunk costs. Over time, the scheduling system gets better at handling uncertainty with less additional computational effort.

A clear advantage of this method lies in the simplicity and generality of the state representation. The authors introduce a solution matrix that directly captures the current scheduling order. This choice has two implications. First, it reduces the dependency on specialized modelling knowledge, lowering the barrier for industrial adoption. Second, the representation is less dependent on static information. This makes the learned policy strong even when processing times change, which aligns

with real-world rescheduling.

The model selects job operations among paired priority dispatching rules (PDRs). PDRs are common in manufacturing systems, so they are easy to validate operationally. Instead of replacing PDRs, the DRL agent learns when to use different ones, acting as a controller that adapts decision logic to the system's current state. This improves acceptance and trust, as managers can interpret scheduling decisions in terms of known rules.

Rather than directly optimizing makespan, the reward is based on machine idle time. This means that learning is encouraged in a way that supports operational efficiency by reducing idle time, which is a major goal for the management. This approach also allows for faster and more stable learning, which is fundamental for scheduling systems that need to stabilize within reasonable time frames to be useful in real-world operations.

The authors examine three approaches: training a new policy from scratch, reusing a trained policy without adaptation, and applying an existing policy followed by further training. The results show that reusing a policy notably decreases training time (approximately by 27%), while achieving competitive solution quality, particularly true for small and medium scale instances.
If an immediate response is required, an already trained policy can be used instantly. If more precision is needed and time allows, that policy can be gradually improved without full retraining. This offers management greater flexibility in decision-making under time pressure.

The experimental evidence suggests that the scheduling policy maintains its effectiveness across various uncertainty levels and probability distributions (both Normal and Uniform). This finding implies that the scheduling system is not linked to a specific disturbance, decreasing risks for managers.
Moreover, the reused policy often achieves better performance than its initial training, suggesting that exposure to uncertainty improves the scheduler's decisions over time. Therefore, reinforcement learning schedulers are able to learn from disruptions, transforming variability into a source of improvement.

Case study: Dynamic job-shop scheduling using graph reinforcement learning with auxiliary strategy [14]

This paper tackles dynamic job-shop scheduling in an environment where disruptions often occur. In modern manufacturing systems, especially those operating

under high product variety and small batch sizes, machines may fail unexpectedly and processing times may vary, leading to the managerial dilemma of choosing between high-quality solutions and speed and cost of the execution.
Traditional scheduling methods sometimes attempt to anticipate unexpected events by inserting slack times or buffer capacity, at the cost of how efficiently resources are managed. Fully reactive methods, often built on priority-based dispatching rules, are fast and simple to apply, but their results can vary widely over time.
Reinforcement learning is presented as an approach to overcome this trade-off by supporting continuous, event driven decisions rather than relying on repeated rescheduling.

In this case, a simulation environment is used to solve the DJSP, which is based on instances either from public benchmarks of OR-Library or from randomly generated ones following the Taillard's method.
The performance of P3OR is initially assessed on the standard instances from the OR-Library, including ft06, abz5–9, la01–40, orb01–10, swv01–20, yn1–4, ta01–80, in which the number of jobs and machines vary from 6 to 100 and 5 to 20.
To improve the generalization ability, the agent is then trained on randomly generated instances where small size instances, such as 6x6 and 10x10, are generated following ABZ instances, in which the machine sequence of each job is randomly drawn from a uniform distribution and the processing time follows a uniform distribution U[50,100], and the medium instances, such as 20x15 and 30x20, are generated following Taillard's instances, where the processing times are sampled from U[1,99].
For these instances, the order of machines is randomly changed to simulate uncertainties such as machine breakdown.
When machine breakdowns are taken into account, two parameters are introduced: mean time between failures (MTBF) and mean time to repair (MTTR), to depict failure information. The MTTR of each machine and the number of total breakdowns are constant.
The performance is evaluated in a real production scenario, considering a large workshop of machine tool plant in Hangzhou. The production line mainly produces grinding machine parts, including the machine body, grinder bench, grinding head cover, grinding head base, headstock housing, feeding carrier. The system is equipped with six machines: grinder, scribing machine, boring lathe, drilling machine, milling machine, and locksmith. Ten types of jobs are randomly selected from orders, as described in Table 4.7, producing an instance of 10x6 size.

The proposed GRL-AS framework trains the scheduling policy offline using simulated DJSP instances and then deploys the trained agent online to to make scheduling decisions under real-time constraints. After deployment, the agent can

Job name (Number)	Operations of each job (processing time)
Machine body(1)	mill(28)→bore(87)→scribe(51)→drill(86)→locksmith(95)→grind(69)
Grinder bench(2)	locksmith(51)→scribe(98)→drill(96)→mill(31)→grind(84)→bore(80)
Grinding head cover(3)	mill(49)→scribe(31)→drill(73)→grind(44)→bore(38)→locksmith(96)
Grinding head base(4)	mill(89)→grind(71)→scribe(42)→drill(59)→locksmith(38)→bore(82)
Headstock housing(5)	bore(66)→scribe(84)→drill(56)→locksmith(34)→mill(98)→grind(89)
Feeding carrier(6)	scribe(91)→bore(55)→drill(96)→mill(69)→grind(93)→locksmith(87)
Base plate(7)	locksmith(84)→mill(58)→grind(47)→scribe(31)→drill(63)→bore(72)
Connecting part(8)	grind(23)→bore(26)→mill(37)→scribe(31)→locksmith(86)→drill(51)
Gearbox casing(9)	bore(98)→mill(62)→scribe(85)→drill(96)→grind(51)→locksmith(81)
Rotary table(10)	locksmith(34)→bore(52)→scribe(64)→drill(81)→mill(39)→grind(43)

Table 4.7: Operation information of selected jobs [14]

directly respond to changes. This propriety is central to industrial adoption, since managers often cannot incur high costs, long delays, or uncertain outcomes every time a disruption occurs.

Representing the shop floor as a disjunctive graph allows the scheduling agent to capture both precedence constraints and conflicts deriving from the use of shared machines. More importantly, the Mixed Graph Transformer Network (MGTN) is designed to process graphs of different sizes through attention mechanisms and spatial pyramid pooling. In this way, scheduling is not tied to fixed settings and remains efficient as production volumes, product mixes, or shop configurations vary. This approach limits the need to specifically tune the system and reduces the cost of maintaining multiple scheduling solutions across plants or production lines.

In manufacturing contexts, data and interaction costs are not negligible. Simulated environments require modelling effort, computing resources, and expert validation. P3OR improves training efficiency by separating the policy update, thereby reducing interference during learning. This translates in more stable policies, and lower experimentation costs. The training phase is based on learning effectively from limited samples, which is essential when moving from simulated to real operation environments.

This method significantly outperforms dispatching rules, metaheuristics, and existing DRL approaches, especially on large-scale instances. Particularly relevant for managers as large problem sizes represent closely actual production process. The framework maintains lower scheduling error as the frequency and severity of disruptions increase, indicating more predictable performance.

One point supporting the application of such approach is its performance under machine breakdown scenarios. As the failure severity increases, the RL scheduler performs better than dispatching rules across all tested instances. This demonstrates that the learned policy does not memorize fixed patterns, instead, it adapts its choices as resource availability changes. Reinforcement learning helps managers maintain throughput and delivery date stable even when equipment performance declines, common in heavily utilized production systems.

The approach was implemented in a real industrial setting, not only in simulation.Under machine-breakdown disturbances, the RL-based scheduler produces a shorter makespan than both the value-based and policy-based DRL baselines.This real-world application matters because it supports not only the claim of better algorithm performance but, more importantly, its practical feasibility.The results indicate that the proposed approach can be implemented in practical production settings with different machine types and complex routing. This supports the view that reinforcement learning can move beyond academic studies and serve as a workable tool for production management decisions.

Rather than being only simulated, the approach is applied on an actual industrial environment. In this scenario, the RL-based scheduler achieves a lower makespan than both value-based and policy-based DRL baselines under machine breakdowns. This real-world application is significant as it supports reinforcement learning practical feasibility. It shows that the proposed approach can be integrated into realistic production environments with different machines and complex routing, strengthening the argument that reinforcement learning is a viable managerial tool.

Case study: Dynamic Job-Shop Scheduling Based on Transformer and Deep Reinforcement Learning [23]

This study addresses dynamic job-shop scheduling in an environment that mirrors modern manufacturing realities: as production systems are no longer static, exchange of information is available in real time, and scheduling decisions must be updated continuously as conditions change. Traditional scheduling methods, such as exact mathematical models, metaheuristics, or fixed dispatching rules,

have clear limits in this context. As reinforcement learning supports adaptive and data-driven scheduling decisions that evolve with the system, it can be shown as mean to overcome these limitations.

In this case, 14 scheduling instances from the OR-Library dataset are selected to validate the method, including ft06, ft10, swv01, swv06, abz5, abz7, la01, la06, la21, la31, orb01, orb02, yn01, and yn02, which cover different scheduling scales. The dynamic factor considered is that the job arrives randomly, as the release time of each job is set randomly.

This framework lets the reinforcement learning agent decide which known dispatching rule apply at each decision step, based on the current system state.
The state representation strengthens the approach relevance in managerial decision-making. The shop floor is modelled as a disjunctive graph and the system reflects both structural constraints, including precedence and machine sharing requirements, and real-time operational information, like job status, waiting time, remaining operations, and completion rate. This view allows the scheduling agent to consider local choices alongside their effects on overall system behaviour. As a result, the scheduler develops a more global understanding of system congestion, bottlenecks, and conflicts. From managers, this reduces the risk of locally optimal decisions that can lead to poor global outcomes.

This study integrates prioritized experience replay, transformer-based feature extraction, and a Data Envelopment Analysis (DEA) based evaluation framework that support a key managerial insight: the value of reinforcement learning is not limited to the algorithm itself, but in how it is integrated with domain knowledge, representation design, and the abstraction of decision-making.
By combining interpretability, adaptability, and performance, this framework demonstrates that RL can function as an intelligent component over traditional scheduling logic.

Case study: A spatial pyramid pooling-based deep reinforcement learning model for dynamic job-shop scheduling problem [9]

In this case, an important managerial component is addressed: the reuse of scheduling policies when the size of the problem changes.
In real production environments, dynamic events such as rush orders or new job arrivals change the dimensionality of the scheduling problem. Most classical scheduling methods force managers to retrain the models, which is a barrier to industrial adoption.

The method is evaluated on standard benchmark instances of different sizes, including ft06, abz7, orb01, la21-la40, ta21-ta22, ta31-ta32, ta41-ta42, ta51-ta52, and dmu16. New instances for rescheduling are also generated based on standard benchmarks with different sizes.
Specifically, three datasets are generated based on the scale of newly arrived jobs. In each datasets, eight standard instances are used to generate new instances where the number of machines is instance dependent and the processing time of job operations follows a uniform distribution from 1 to the maximum processing time of the instance. The arriving time of new jobs is based on the completion rate of these standard instances which is 25%, 50% and 75%, respectively. The parameters of the generated instances are shown in Table 4.8.

Base instances	Small-scale datasets	Medium-scale datasets	Large-scale datasets
ft06(6 × 6)	one job arrivals	2 job arrivals	4 job arrivals
orb01(10 × 10)	one job arrivals	3 job arrivals	6 job arrivals
la21(15 × 10)	one job arrivals	3 job arrivals	5 job arrivals
la26(20 × 10)	one job arrivals	3 job arrivals	6 job arrivals
abz7(20 × 15)	one job arrivals	3 job arrivals	6 job arrivals
ta21(20 × 20)	one job arrivals	3 job arrivals	6 job arrivals
dmu16(30 × 20)	one job arrivals	5 job arrivals	10 job arrivals
ta61(50 × 20)	one job arrivals	5 job arrivals	10 job arrivals

Table 4.8: Parameters of generated instances [9]

One important contribution of this study for managers is the development of a scheduling policy that remains applicable regardless of system size. Using Spatial Pyramid Pooling (SPP), the proposed DRL framework can handle state representations of different dimensions. In practice, this supports policy reuse across different production scenarios, which reduces related costs and limits disruption to the organization.This approach allows managers to rely on a single learning-based scheduler that generalizes across scales.

The study introduces a state representation that minimizes dependence on domain expertise, which is a major benefit. Rather than relying on handcrafted features or disjunctive-graph node attributes, which require extensive trial-and-error and expert knowledge, the state is decomposed into a machine-time matrix. These

matrices are treated as image channels, that allow for the use of standard deep learning methods without extensive feature engineering. For managers, this reduces the cost and the risk associated with deploying advanced scheduling systems, as the performance of the model depends less on features and design choices.

Rather than presenting a single solution, decision-making strategies are evaluated under different time and performance constraints. This reflects real decision-making,in which the most suitable choice depends on urgency, system state, and available resources.
The authors distinguish between four scheduling modes: priority dispatching rules (PDR), trained policy, retrained policy, and reused policy. The results show that while PDRs and trained policies offer immediate response times, reused and retrained policies can yield superior scheduling performance when more time is available.

The concept of reused policy is particularly important from a managerial perspective. By initializing rescheduling from an already trained policy and then continuing training on new instances, the scheduler effectively accumulates operational knowledge over time. Experimental results show that reused policies converge faster and often outperform retrained policies. This implies that reinforcement learning can function as an asset, improving its effectiveness as the production environment evolves, rather than starting over after each disruption.

The reward design also supports managerial alignment. The reward function is constructed to be negatively related to makespan, making the learning objective consistent with operational performance metrics.

The ability to reuse learned policies across varying problem sizes, balance speed and accuracy in rescheduling, and continuously improve through experience makes reinforcement learning a powerful alternative to classical scheduling methods. For managers operating in environments with frequent order changes and time pressure, reinforcement learning offers flexible and robust decision-making support.

Case study: A deep multi-agent reinforcement learning approach to solve dynamic job shop scheduling problem [17]

In modern manufacturing systems, machines requirements are starting to demand for local intelligence and real-time data elaboration, while global decision-making system often lack detailed visibility of local conditions. In unpredictability cases, centralized scheduling may not be an efficient solution, motivating the use of multi-agent reinforcement learning (MARL) as a more realistic control paradigm.

In this case, a simulated environment is presented. The job shop consists of 10 machines and a unique event set is randomly generated at each run, which lasts 2000 units of time, with an average of 56, 64, and 72 job arrivals under three different scenarios characterized by different utilization rates (70%, 80%, 90%), respectively.
Each iteration has identical events, such as job's operation sequence, arrival and processing times, and due dates.
Not all the jobs exist in the system at time 0, in fact, new jobs arrive stochastically across the 2000 time-unit horizon, changing the problem size. This instance change, in terms of queue size and remaining operations, requires rescheduling. Congestion varies by scenario, according to the selected utilization level.

An explicit alignment with shop-floor organizational structure emerges. In this study, each machine is modelled as an autonomous scheduling agent that selects the jobs sequence in its queue. Distributing decision authority across the units while maintaining coordination through centralized training, avoids the rigidity of centralized schedules and the inefficiencies deriving from uncoordinated local decisions.

A core management challenge in decentralized scheduling is keeping coordination consistent across units. Independent local decisions can easily lead to system inefficiencies. Centralized training with decentralized execution allows agents to learn cooperative behaviour using shared experience during training, still remaining autonomous during execution. Sharing parameters tends to stabilize training and reduce complexity, making the approach more scalable and manageable from an operational perspective.

Instead of relying on immediate or global reward signals, a knowledge-based reward-shaping method is proposed, linking rewards to job slack use and queueing delays. This mechanism assigns the penalty for late completion in proportion to the amount of time each machine holds the job. This approach improves learning and supports cooperative behaviour that aligns with organization business objectives.

In scheduling, the impact of a machine decision may become visible only later in the operations process. By explicitly modelling how the scheduling decision's influence changes over time, this approach consents a more precise attribution of outcomes to specific decisions.This resembles managerial performance evaluation practices that consider full process lifecycle rather than isolated actions.

The scheduling policy is able to adapt to fluctuating workloads and random job

arrivals without retraining, reflecting in lower overhead because the same scheduling policy can be applied across different production volumes and demand patterns.

The experimental results support the value of the framework presented in this study. Under dynamic conditions with moderate to high utilization, the deep MARL approach consistently outperforms priority dispatching rules and recent DRL-based benchmarks in minimizing cumulative tardiness. The results indicate that performance remains optimal as the system becomes more congested, precisely where scheduling quality matters most.

Genetic algorithms often perform well on large, static problem instances, but their high computational cost limits their use when schedules need to be updated frequently in dynamic settings. By contrast, the deep MARL approach supports faster decisions, which can enable near real-time control. This suggests that reinforcement learning is not a substitute for mathematical optimization, instead, it offers a good balance between decision quality and responsiveness.
This application aligns closely with how modern factories operate, requiring scheduling systems capable of supporting agile, data-driven, and customer-oriented manufacturing operations.

Case study: Evolution strategies-based optimized graph reinforcement learning for solving dynamic job shop scheduling problem [15]

Many reinforcement learning approaches demonstrate impressive results under controlled conditions, however managers require scheduling systems to behave properly also in uncertain conditions. In this study, a scheduling framework that remains effective under stochastic processing times and machine breakdowns is explicitly addressed, considering two of the most common sources of variability.

In this case, the model is trained on randomly generated static instances, where the number of machines is 10, there are 10 jobs to be processed, and the processing time of operations follows a uniform distribution U[1,99]. In addition, the order of machines for the processing of a job is randomly assigned to make different machines handle different operations of a job.
The trained model is then tested on known public benchmark instances of various sizes (FT, LA, ABZ, ORB, SWV, YN, TA, and DMU instances).
Dynamic experiments are also performed on some of the above mentioned benchmark instances, specifically LA16–20 (10×10), DMU01–05 (20×15), TA41–45 (30×20), DMU36–40 (50×20), with the addition of stochastic processing time and machine breakdowns.

Different standard deviations σ (reported levels $\sigma = 1,2,3$) generate different degrees of random processing time disturbance. The actual processing time is equal to the benchmark specific processing time plus some noise controlled by σ.
When machine breakdowns are considered, machines have to pause due to failure and the job can continue to be processed after repair. The mean time between failures (MTBF) and the mean time to repair (MTTR) are two parameters associated with this kind of disruptions, and are assumed to be same for all machine, following the exponential distribution.

Representing the shop floor as a disjunctive graph allows the scheduling policy to consider resource constraints and precedence relations at the same time. For managers, this means making scheduling decisions that are consistent and more context aware, especially in complex and large-scale systems.

The replacement of gradient-based deep reinforcement learning with Evolution Strategies (ES) for policy optimization consents to optimize policy learning using population-based perturbations and direct performance evaluation. This approach makes the learning process steadier and less sensitive to modelling choices.
This shift has important implications for managers. With ES, the scheduling system can be trained by treating makespan as the direct objective function. This simplifies the development process and reduces the need for domain experts for fine-tuning.
In operational terms, this reduces the effort needed to put the method into practice and increases confidence that the learned policy matches actual business objectives.

This method demonstrates that it can be generalized to different problem sizes and distributions, achieving better results than priority dispatching rules and DRL methods on large benchmark instances.
When stochastic processing times and machine breakdowns are introduced, the GRL approach consistently outperforms the best-performing PDRs, also showing lowering variance in outcomes. This suggests that the learned policy improves performance while also lowering operational risk. For managers, lower variance can be as valuable as improved performance because it increases predictability and can improve customer satisfaction..

The analysis strengthens the case for reinforcement learning adoption. In settings where schedules must be revised often, metaheuristic methods can become too slow to return correct results in time, while the GRL approach tends to balance solution quality with the need for quick responses. This positions reinforcement learning as a valid option between fast heuristics and slow optimization algorithms.

The proposed ES based GRL framework focuses on generalization, responsiveness to uncertainty, and ease of deployment. This focus suggests how reinforcement learning can move beyond experimental methods and work as a decision support for the system. For managers facing volatile and large volumes of data environments, this approach offers a valid alternative to existing solutions.

Case study: Dynamic production scheduling towards self-organizing mass personalization: A multi-agent dueling deep reinforcement learning approach [18]

Dynamic events such as stochastic processing times and machine breakdowns can not be anticipated in full. Direct training in highly dynamic environments can be unstable, require large amounts of data, and difficult to manage. By learning policies from static scheduling instances and deploying them in dynamic execution environments, the authors demonstrate adaptability during operations. This separation between learning and execution fits common industrial limits, where retraining during production is impractical and expensive.

In this case, 17 instances from Hurink_eData are selected to conduct static training. These instances involve different combinations of jobs (between 7 and 20) and machines (between 4 and 15). The average processing time across all the operations of all jobs is also varied.
The algorithm performance is tested under dynamic settings, applying the trained scheduling policies for each static instance to its corresponding dynamic instance under stochastic processing time and machine breakdown.
The method is tested under three different probabilities, 10%, 5%, and 3%, of machine's actual processing time deviating from planned processing time, and also three different machine breakdown probabilities, 3%, 2%, and 1%, are considered. Machine's repair time is assumed equal to a processing time randomly selected from all possible processing time in the scheduling instance.

Treating each machine as an autonomous agent has clear implications on how the organization is structured and managed. Single-agent scheduling approaches require global state information and suffer from scalability issues, in this case a partially decentralized architecture is proposed. Each machine acts based on local observations, and coordination mechanisms align decisions to support overall system performance. This design improves scalability, as machines can be added or removed without redesigning the entire control system. For managers, this provides greater flexibility when reconfiguring production lines or expanding capacity.
The proposed cooperation mechanism ensures that machines collectively resolve conflicts and allocate operations in a way that benefits overall system performance.

This shows that autonomy needs to be balanced with coordination to avoid suboptimal outcomes.
Using immediate rewards based on utilization and a final penalty based on total idle time and makespan, the learning process pushes agents to balance between short-term efficiency and long-term system performance.

Across a diverse set of benchmark instances, the learned policies outperform heuristic rules under both stochastic processing times and machine breakdowns. More importantly, performance advantages persists under different levels of disturbance, indicating that the learned policies work reliably not only in specific conditions and supporting the deployment of this type of scheduling systems in uncertain environments.

The ability to apply policies learned in static environments directly to dynamic execution without retraining can lower deployment costs and faster the application of such systems across multiple production settings, supporting scalability.

The broader managerial implication of this paper is that reinforcement learning can function as a control method for self organizing manufacturing networks. The proposed framework supports self-configuration through decentralized agents, improves performance through learned policies, and maintains service level through adaptive responses to disruptions, which aligns with the objectives of Industry 4.0.

Case study: Combining Reinforcement Learning Algorithms with Graph Neural Networks to Solve Dynamic Job Shop Scheduling Problems [16]

Reinforcement learning shows potential for dynamic job shop scheduling, but dependence on manually designed state features limits how well it transfers across different shop floor configurations.

Authors treat dynamic job shop scheduling as problem where the quality of decisions depends on how accurately the system observes and interprets the production environment. Traditional reinforcement learning methods often represent shop floor states with selected indicators, derived from experts knowledge. Although effective in specific settings, this can carry human bias into the learning process and limit the model's ability to generalize across different job shop structures. For managers seeking scheduling systems that can scale and be reused across settings, reliance on domain specific feature engineering remains a major obstacle to industrial deployment.

In this case, a dynamic job shop scheduling problem with randomly arriving jobs and job weights is considered. Experimental instances are generated following a Poisson distribution and consist of two groups of job–machine configurations, 10×10 and 20×15. Each job consists of a sequence of operations processed on fixed machines under classical job shop constraints.

By integrating graph neural networks (GNNs) with deep reinforcement learning, this study proposes a shift toward representation learning guided by the structure of the environment. The job shop is modelled as a disjunctive graph, where nodes represent operations and edges capture both precedence and machine-sharing constraints. This formulation matches the inherent relational structure of production systems, allowing the scheduling system to percieve the production environment in a way that is independent from handcrafted rules and expert knowledge.

Using GNN-based feature extraction, the system can learn autonomously learn which aspects of the job shop state are relevant for decision-making. This finding has two clear implications for managers. It reduces reliance on specialized expertise, consequently it lowers the time and cost required to set up the systems for new factories or production lines. Second, it improves model portability because the same learning architecture can be applied to different job shop instances without redesigning the state space. This fits the aims of smart factories, where fast reconfiguration and adaptation are crucial.

Another important contribution lies in the analysis of dynamic job arrivals and job priorities, represented through weighted completion-time objectives. This reflects realistic production environments in which jobs differ in priority and where scheduling decisions need to account for the urgency of customer demand. By embedding job weights directly into the graph representation and the reward function, the model can make choices that reflect managerial priorities, such as customer satisfaction, service differentiation, and order urgency.

The reward design proposed is also relevant when considered from a management perspective. Rather than depending only on incremental rewards, often used in reinforcement learning, the authors define a global reward function based on the deviation from the best completion time observed. This strategy directs the agent toward higher-quality scheduling decisions and discourages stagnation around suboptimal policies.
The empirical results show that this reward structure performs better as the problem size increases, which suggests it is well suited to large, complex production systems where local heuristics may often fail.

The experiments provide evidence that GNN based reinforcement learning outperforms both rule-based scheduling and standard DRL methods. Notably, the only difference between the benchmark DDQN and the proposed GNN-DRL model lies in how the shop floor state is represented. Fast convergence and solution quality achieved by this model highlight the strategic importance of representation learning in this type of scheduling. For decision-makers, this suggests that allocating resources to more expressive perception models may yield higher returns.

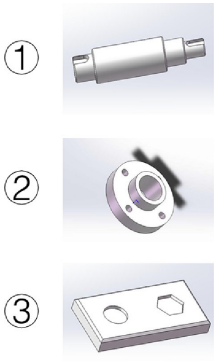
As production systems grow in size and become more complex, manually designing state features becomes impractical. The graph-based representation scales with the number of jobs and machines, so the scheduling model can be applied to larger instances without an exponential increase in complexity. This property is essential in factories operating under Industry 4.0 paradigms, where production systems are expected to evolve continuously.

Although GNN-based models rely less on experts, they tend to require more computation and modelling complexity. This implies that industrial deployments should balance detailed representations with real-time performance requirements.

Case study: Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems [19]

This study analyzes decentralized multi-agent scheduling that treats each machine as an independent agent with its own decision-making ability. This mirrors the real organizational structure of manufacturing systems, where machines, units, and storage systems may operate independently but must coordinate their actions. For managers, this design makes the system less fragile by avoiding failures that can derive from centralized scheduling.

In this case, a realistic dynamic manufacturing workshop is designed. It is composed of six different machines, including three lathes and three millers. There are also two small warehouses equipped with automated storage and retrieval system, to store and pick up raw materials and finished products. Moreover, each machine is equipped with a buffer capacity of 4 jobs, for temporary storage of WIP. There are three types of job in the experimental test, including shaft, flange and plate. Figure 4.1 illustrates the routing of each job type, and the available machines for each operation.



Job type	Process route	Available machines
Shaft	Turning→ Milling	(M_1,M_2,M_3), (M_4,M_5,M_6)
Flange	Turning→ Milling	(M_1,M_2,M_3), (M_4,M_5,M_6)
Plate	Milling	(M_4,M_5,M_6)

Figure 4.1: Routing of jobs and available machines for operations [19]

Customers orders within a certain period of time are collected and used as a test case to evaluate the scheduling efficiency and learning ability of the method. The details of the test case are shown in Table 4.9, where twenty orders with different arrival time and due time are considered.
To test the scheduler's performance to handle disturbance events after training, two experiments are designed, taking into account both order insertion and machine failure.

Index	Job type	Arrival time	Due time	Ideal workload time
1	Shaft	10	151	35,12
2	Flange	45	171	32,10
3	Plate	70	199	43
4	Plate	95	209	38
5	Plate	125	266	47
6	Flange	150	282	30,14
7	Shaft	175	307	33,11
8	Plate	190	340	50
9	Flange	200	368	39,17
10	Plate	210	297	29
11	Flange	235	373	33,13
12	Plate	260	392	44
13	Plate	270	384	38
14	Shaft	295	412	29,10
15	Flange	320	425	26,9
16	Shaft	350	464	28,10
17	Plate	370	475	35
18	Plate	385	496	37
19	Shaft	430	571	31,16
20	Flange	495	651	34,18

Table 4.9: Test case [19]

For example, the arrival time of jobs 8 in the test case is set at t=200 s, which equals that of job 9. Also, the arrival time of job 17 is set at t=385 s, consistent with that of job 18. This represents two disturbance events of order insertion at 200 s and 385 s. In addition, it is assumed that Machine 4 fails at t=300 s and has no ability to perform the processing task until t=400 s. During this period, rescheduling is required, as Machine 4 cannot be assigned tasks until it is repaired and returns to normal operation.

The introduction of a self-organizing negotiation mechanism based on Contract Net Protocol (CNP) is relevant design choice, from a managerial control perspective. In this approach, machines bid for tasks based on current capacity, available buffers, and status. This mechanism lets the system determine its current state in real time. Empirical results indicate that this design reduces scheduling response time by about 70–80%, which is important for factories operating under tight delivery requirements.

Each agent learns to map the observed conditions to specific production decisions, improving this mapping over time through interaction with the environment and updating its choices based on the outcomes. In contrast to heuristic or GA schedulers, whose decision rules remain fixed once training ends, the PPO based scheduler can update its policy during deployment. Hence, the system can improve with use, learning from disruptions instead of becoming unstable because of them.

The composite reward function in this study combines tardiness, workload balance, and profit into a single learning signal. This approach builds managerial priorities into the learning process itself, rather than adding them later through adjustments.

In a distributed reinforcement learning architecture, scalability improves because decisions are made locally and coordination is handled through negotiation among agents rather than a centralized decision process. This property matters for factories that plan modular expansion, frequent reconfiguration, or cloud–edge collaboration strategies.

Comparisons between GP based and DQN based methods further highlights that reinforcement learning offers practical advantages to management. GP-based methods converge to static dispatching rules and lose adaptability after convergence. DQN-based methods may show unstable learning and high variance when exposed to disturbances. In contrast, PPO-based reinforcement learning tends to converge more steadily, learn fast, and allocate workload more evenly. This can lower operational risk and reduce swings in performance, outcomes that managers often value as much as performance improvements.

This paper shows that reinforcement learning can keep operational decisions aligned with managerial objectives. The scheduling system adjusts its schedule when priorities change or disruptions occur, as it operates on its own by negotiating, making decisions, and adjusting to new conditions.

This paper shows that reinforcement learning can keep operational decisions aligned with managerial objectives. The scheduling system does not requires manual intervention when priorities change or disturbances occur. On the contrary, it autonomously negotiates, decides, and adapts.

Overall, this study clearly supports the use of reinforcement learning in dynamic job shop scheduling.
The findings suggest that reinforcement learning schedulers perform better than traditional approaches not only because they can produce better solution, but also because they change the way scheduling decisions are made.
For managers working in volatile, high-mix production settings, this can be seen as a step toward more stable and intelligent production control.

4.1.1 Simulated Job Shop Environments and Real Production Systems: Case Study Comparison

The case study presented in the following paragraph examines real production data derived from a high-mix manufacturing company whose identity is kept confidential. Its production system illustrates the typical complexities of flexible job shop scheduling, producing batches of components that require multiple machining operations on parallel machines that can have overlapping capabilities.

Structure of the System

The system is composed of seven machines, each capable of performing different operations. Due to operational similarities, these machines can be divided into two groups:

- Roughing machines (F2_1, F2_2, F2_3), responsible for initial shaping and excess material removal.
- Finishing machines (F2_4, F2_5, F1_2, F1_3), dedicated to precise finishing, completion of parts and final quality control processes.

The system comprises a series of operations, each with specific requirements and machinery. Every operation can be assigned to a machine category, as highlighted in Table 4.10.
The dataset defines 16 jobs to be processed, each requiring a series of operations in a predefined sequence. Since no job necessitates all 13 operations defined, the number of operations per job may vary from 6 to 12. Each job has individual arrival times and due dates, which may be difficult to be satisfied on time. As a result, delay is embedded in the system.
Table 4.11 lists the machines eligible for performing different operations. It is important to notice that machines are distinguished as primary and secondary, where the latter are indicated in brackets (Table 4.11) and considered as less preferred. The company underlines the utilization of preferred machines that reflect maintenance considerations, process continuity and stability, with the possibility of repeating operations on the same machine when possible, to reduce setup times. These constraints illustrate the complexity of such systems and highlight typical uncertainties that can be found in the Dynamic Job Shop Problem.

ID	**Operation type**	**Machine category**
Fre_20	Lavorazioni 2d Pre Terrazzamento	Roughing
Fre_30	Terrazzamento	Roughing
Fre_31	Apertura Raggi Pre Semifinitura	Roughing
Fre_40	Semifinitura	Roughing
Fre_42	Apertura Raggi Seconda Semifinitura	Finishing
Fre_43	Seconda Semifinitura	Finishing
Fre_21	Lavorazioni 2d	Roughing
Fre_45	Apertura Raggi Pre Finitura	Finishing
Fre_50	Finitura	Finishing
Fre_51	Completamento Raggi Negativi	Finishing
Fre_60	Riprese 2d	Finishing
Fre_70	Tracciatura	Finishing
Fre_22	Lavorazioni 2d Post Finitura	Finishing

Table 4.10: Operations definition

	Operations												
	Fre_20	**Fre_30**	**Fre_31**	**Fre_40**	**Fre_42**	**Fre_43**	**Fre_21**	**Fre_45**	**Fre_50**	**Fre_51**	**Fre_60**	**Fre_70**	**Fre_22**
Machines	F2_1	F2_1	F2_1	F2_1	F2_4	F2_4	F2_1	[F2_1]	[F2_1]	[F2_1]	[F2_1]	[F2_1]	[F2_1]
	F2_2	F2_2	F2_2	F2_2	F2_5	F2_5	F2_2	[F2_2]	[F2_2]	[F2_2]	[F2_2]	[F2_2]	[F2_2]
	F2_3	F2_3	F2_3	F2_3	F1_2	F1_2	F2_3	[F2_3]	[F2_3]	[F2_3]	[F2_3]	[F2_3]	[F2_3]
					F1_3	F1_3	F2_4	F2_4	F2_4	F2_4	F2_4	F2_4	F2_4
							F2_5	F2_5	F2_5	F2_5	F2_5	F2_5	F2_5
							F1_2	F1_2	F1_2	F1_2	F1_2	F1_2	F1_2
							F1_3	F1_3	F1_3	F1_3	F1_3	F1_3	F1_3

Table 4.11: Machine allocation

Case Study Comparison

The majority of case studies examined from the literature has its basis on standard benchmark instances to simulate the production environment. This is a common choice as the objective is to validate the performance of the various algorithms in contexts where stability and scalability can be proven. Benchmark-based simulations allow for controlled and reproducible experiments in order to compare different scheduling approaches.

For those cases based on benchmarks, the job shop structure is typically simplified, as routing flexibility is limited, and machines are often modelled as single and not parallel resources. This is a major distinction from case studies based on actual production data, since simulating reality means taking into account a certain degree of complexity. These cases are often characterized by parallel machines, and jobs can be processed on alternative routes.

When uncertainty is introduced, benchmark-based cases can better approximate aspects of actual production systems. In fact, the simulation of stochastic job arrivals or unpredictable machine breakdowns reflects events that can happen in reality, allowing the scheduling models to account for real uncertainties and increasing their potential of deployment in actual production environments.

Despite their nature, benchmark-based systems share characteristics with the industrial case studies, such as resource sharing constraints, and evolving production states. In dynamic scenarios, the introduction of unexpected events leads to the need of continuous adaptation of scheduling decisions to new system conditions, resulting in rescheduling in both simulated and real environments.

However, while simulated environments provide a valuable abstraction of real job shop dynamics, industrial case studies introduce additional layers of complexity that are difficult to capture within benchmark-based models.
Uncertainty modelling, in particular, differs between the two approaches.

In benchmark-based environments, uncertain events are typically parametrized, following statistical distributions. Instead, uncertainty in actual industrial systems is endogenous and may be influenced by operational factors, making it more difficult to model and predict. In addition, real production-based environments are characterized by more operational constraints that increase the complexity of scheduling decisions.

Table 5.2 summarizes the main aspects of the considered case studies, showing differences and commonalities between benchmark-based and real production-based job shop systems.

	Benchmark-Based	**Real Data-Based**
Environment	Generated job shop models based on standardized benchmark instances with predefined routing and processing times	Real production systems derived from actual shop-floor configuration, machines, and routing constraints
Instance Source	Randomly generated benchmark instances	Real production data
Number of Machines	From 6 to 20	From 3 to 7
Number of Jobs	From 5 to 100	From 10 to 1000
Machine Assignment	Mostly Fixed	Mostly Flexible
Constraints	Limited operational constraints	Priorities and machine-sharing constraints

Table 4.12: Comparison between benchmark-based and real production-based job shop environments

4.2 Adoption of Reinforcement Learning in Dynamic Job Shop Scheduling

Managers are primarily focused on financial stability, operational reliability, scalability, and the investment cost.
Motivating why reinforcement learning is a preferred solution over other methods in operational terms is important, however management decisions involve more than just technical best practices. The following section will lay out the advantages of such applications.
Reinforcement learning should not be evaluated solely as an advanced optimization technique, but as a tool that changes how an organization plans, sustain, and evolves its scheduling processes.
Looking at the previous analysis, the benefits of RL are particularly relevant when translated into cost, risk, and business continuity components, especially in environments characterized by fast and unexpected changes.

4.2.1 Cost efficiency

A central point in the choice of the application is the total cost associated with the scheduling decision model. This includes computational effort, human intervention, system downtime and maintenance needed.
Throughout the previous analysis, RL approaches demonstrate to cut down scheduling execution time, often up to 70–80%, compared to classic optimization methods. More importantly, RL avoids the need for continous rescheduling when unexpected events emerge in the system.
From a cost perspective, this allow to:

- reduce infrastructure requirements
- rely less on domain experts for parameter tuning and rescheduling
- avoid interruptions of the production flow when disturbances occur

Approaches that reduce handcrafted state features lower engineering and reconfiguration costs when the shop floor structure changes, underlying the adaptability of the system.
Thereby, RL transfer scheduling costs from recurring operational expenses to an upfront investment in system design and training.

4.2.2 Operational Robustness and Risk Mitigation

In real world production systems uncertainty is inherent.
Prioritizing scheduling solutions that limit exposure to operational risk, particularly in terms of late deliveries, unstable production plans, and slow reactions to disruptions, is therefore essential.
RL agents are trained through repeated interaction with dynamic environments,so they learn how to handle variability in the system. The analysis consistently points that reinforcement learning schedulers show strong robustness to uncertainties.
When disruptions occur, RL based systems typically experience more limited performance degradation compared to static scheduling methods. Plus, they continue to operate without requiring full rescheduling.
From a risk management perspective, this behaviour is crucial.
Traditional scheduling approaches often demand production to be paused while a new schedule is computed, which raises the risk of missed deadlines and contractual penalties. Reinforcement learning, instead, mitigates this risk by allowing continuous operations and adapt to system variations. This leads to improvements in meeting due dates and reduces the likelihood of repercussions across the production system.
Hence, RL functions as a risk buffer, improving schedule reliability and performance in unpredictable conditions.

4.2.3 Investment and Asset Utilization

Scheduling decisions directly influence throughput, resource utilization, and revenue generation.
The previous analysis reports consistent improvements in throughput and workload balance across resources, as well as increased machine utilization, due to improved scheduling decisions that reduce queues and bottlenecks. These improvements let firms obtain more value from existing assets without additional capital investment.
Many studies indicate that RL based solutions perform almost as well as metaheuristic methods, operating at a fraction of the computational time, making it possible to use them in real-time situations.
Additionally, the fact that trained RL policies can be reused and scaled shows the return on the investment these methods have. Once trained, policies can often be applied to production scenarios with different settings or scaled to larger instances without the need for redesign, significantly reducing marginal deployment costs.

4.2.4 Flexibility and Strategic Fit

Beyond its operational and financial benefits, reinforcement learning aligns closely with wider organizational and strategic objectives. Modern manufacturing strategies often prioritize flexibility, decentralization, and digitalization, all of which can be achieved by applying RL-based scheduling systems.
Some of the analyzed studies embed reinforcement learning within multi-agent architectures, where machines or production units function as independent decision-makers. This decentralized mechanism reduces dependence on central control structures and improves local responsiveness.
From an organizational view, RL-based scheduling can provide effective assistance in:

- Mass customization strategies
- Distributed control
- Continuous improvement through learning

In fact, reinforcement learning supports continuous improvement by learning from data. Unlike static scheduling systems, RL-based schedulers evolve over time, adapting to changes in demand, resource availability, and operational constraints. This learning capacity turns scheduling into a strategic asset that supports long-term competitiveness.

4.3 Results

Reinforcement learning represents a sound management approach for dynamic job shop scheduling. Its advantages go beyond improved performance metrics to cost effectiveness, reliability, scalability, and strategic fit.
Employing this method represents a shift from reactive optimization toward proactive and adaptive decision-making.
The choice of reinforcement learning based scheduling methods is valuable not only for improving performance indicators, but also for reducing operational costs and mitigating scheduling related risks.
This approach lets decisions be made in real-time avoiding constant rescheduling, translating in limited production interruptions and delays.
Its robustness under uncertainty strengthens reliability in case of disruptions, and the management can extend the benefits of such applications across multiple production systems with marginal additional cost, because of its scalability and broad use.
Classical scheduling methods may remain suitable in stable and predictable environments, but their limitations become more evident in systems characterized by uncertainty and complexity.
Reinforcement learning offers managers a flexible, adaptive, and financially justified approach that supports both short-term operational goals and long-term strategic objectives.

Chapter 5

Conclusions

This thesis examined the application of Reinforcement Learning to the Dynamic Job Shop Scheduling Problem, a complex challenge in modern manufacturing systems characterized by uncertainty, variability, and frequent disruptions. The objective was to assess whether reinforcement learning can provide effective, robust, and scalable scheduling solutions in environments where traditional approaches often fail to maintain performance.

The core contribution of this work lied in the qualitative and managerial analysis of selected reinforcement learning applications from the case studies. The results emerging from the analyzed literature indicate that reinforcement learning approaches outperform traditional dispatching rules and remain competitive with metaheuristic methods in dynamic environments. In particular, RL-based schedulers demonstrate a strong ability to adapt to stochastic job arrivals, machine breakdowns, processing time variability, and order changes without requiring frequent rescheduling or extensive human intervention.

A major advantage of reinforcement learning is its capability to generate fast scheduling decisions, once training is completed. This characteristic makes reinforcement learning especially suitable for real-time production environments, where responsiveness is a critical performance requirement. Moreover, several approaches showed improved workload balancing, higher machine utilization, and robustness to disturbances, all of which are extremely relevant for the management scheduling choice. Hybrid solutions further enhanced interpretability and practical feasibility.

Several limitations continue to persist. Many of the analyzed cases rely on simulated environments, which may not fully capture the complexity and unpredictability of real industrial systems. Also, the performance of reinforcement learning models is highly dependent on the design of the reward function, state

representation, and training configuration, which often require significant expertise. Training phases can also be computationally intensive and time-consuming, particularly for large-scale problems. Furthermore, issues related to generalization across different production settings and scalability to more complex and heterogeneous systems remain partially unresolved.

Looking forward, future research should focus on bridging the gap between theoretical models and real-world industrial implementation. Promising directions include the development of hybrid approaches combining reinforcement learning with optimization-based or heuristic methods, the integration of multi-objective formulations to simultaneously address cost, tardiness, energy consumption, and sustainability metrics, and the enhancement of generalization across different production environments. Additionally, experimental validation in real manufacturing systems would provide valuable insights into the practical feasibility, economic impact, and organizational implications of reinforcement learning based scheduling solutions.

In conclusion, this thesis demonstrates that reinforcement learning constitutes a powerful and flexible framework to address the complexities of dynamic job shop scheduling. While important limitations remain, this work suggests that RL-based approaches hold significant potential as advanced decision-support tools in modern manufacturing systems characterized by uncertainty, complexity, and the need for real-time adaptability. By combining computational intelligence with managerial insight, reinforcement learning can contribute to the evolution of more resilient and responsive production systems within the broader context of digital and smart manufacturing.

	Benchmark-Based	**Real Data-Based**	**Real Production Case**
Environment	Simulated	Real production system	High-mix production environment
Instance Source	Benchmark instances with predefined routing and processing times (Taillard, Demirkol, ...)	Real production data based on actual shop-floor configuration	Real production data
Number of Machines	From 6 to 20	From 3 to 7	7
Number of Jobs	From 5 to 100	From 10 to 1000	16
Machine Assignment	Mostly Fixed	Mostly Flexible	Flexible, multiple machines per operation
Constraints	Limited	Priorities and machine-sharing constraints	Primary and secondary machines, Primary preferred

Table 5.1: Comparison between benchmark-based and real production-based job shop environments

	Benchmark-Based	**Real Data-Based**	**Real Production Case**
Environment	Simulated	Real production system	High-mix environment
Instance Source	Standard benchmarks (Taillard, Demirkol...)	Real production data	Real production data
Number of Machines	6–20	3–7	7
Number of Jobs	5–100	10–1000	16
Machine Assignment	Mostly fixed	Mostly flexible	Multiple machines per operation
Constraints	Limited	Priorities and machine sharing constarints	Primary and secondary machines (primary preferred)

Table 5.2: Comparison between benchmark-based and real production-based job shop environments

Appendix

Year	Title	Authors	Uncertainties	Problem	Description	Agent	Algorithm	Action space	State space	Reward function	Num. Objective	Obj. Indicator	Conclusions	Limitations	Future research	Contributions
2026	*A novel deep reinforcement learning framework based on digital twins for dynamic job shop scheduling problems*	*Wenquan Zhang, Zhaoxin Peng, Fei Zhao, Bo Feng, Xuesong Mei*	Processing times and changes in material feeding rates	*DJSSP*	Use of a DT technology to faithfully replicate actual production processes, providing effective means for training and validating scheduling algorithms under uncertain conditions, effectively addressing the issue of model completeness.	Sigle-Agent	**Double Deep Q-Network (DDQN) algorithm:** it decouples action selection from Q-value estimation. One network selects the action with the highest Q-value, while a separate target network is used to compute the target Q-value. This separation reduces the bias caused by overestimating action values, improving both the accuracy and stability of learning.	The action space corresponds to machine numbers, meaning that each action corresponds to selecting a specific machine to dispatch the next workpiece to. The agent decides which machine the gantry robot should move a task to, transferring it from the current spot welder to the input buffer of the next spot welder in the processing sequence.	High-fidelity digital twin (DT) model faithfully replicate the production process, including the operational condition of each machine and utilization level, the current share of output relative to the total production, and the progress of the job it is executing. Also,it incorporates the arrival times and priorities of incoming workpieces, the processing times and the current availability of machines.	The goal is to minimize the maximum completion time. Since the completion time is strongly correlated with machine utilization, the scheduling objective can be reformulated in terms of a machine utilization function: therefore, the larger the reward obtained at each scheduling decision point, the smaller the idle time and the higher the machine utilization.	Single-objective	Completion time	The results demonstrate the effectiveness of the DT-DRL algorithm compared to heuristic methods, due to the utilization of DT technology to model real-world scenarios and the definition of states, actions and reward features shows strong generality and flexibility.	Dependance on a well-designed reward function.	Integrate the constructed real-world scenarios with other policy-based algorithms to overcome the need for large training datasets	Introduction of a highly efficient and innovative DT-DRL scheduling framework for the first time in real production scenarios, effectively addressing real-time dynamic scheduling decision problems. Design of specialized states, actions, and rewards tailored for the job shop, effectively capturing pivotal aspects, enhancing the understanding and decision-making ability of the RL agent regarding the issue.
2026	*A deep reinforcement learning approach for dynamic job-shop scheduling problem considering time variable and new job arrivals*	*Haoyang Yu*, Wenbin Gu, Na Tang, Zhenyang Guo	Processing times and new jobs arrival	*DJSSP*	This paper aims to optimize makespan and scheduling efficiency in dynamic job shop environments by proposing a novel state representation method that expresses state features as multi-channel images to simplify the state characterization process. It leverages a CNN with spatial pyramid pooling (SPP) for feature extraction from scheduling instance matrices.	Sigle-Agent	Use of the **PPO algorithm (clipped version)** which improves policy gradient methods by making updates more stable and efficient. To prevent unstable or overly large updates, it clips policy changes within a small range, ensuring the agent learns reliably while still exploring effectively.	The action space consists of **12 priority dispatching rules (PDRs)**, each based on a distinct heuristic for selecting which job to schedule next.	Each state is represented by a scheduling matrix, the odd-numbered column represents the machine index indicating the machine to which each operation is assigned, and the even-numbered column represents the remaining processing time on each operation.	A dense and continuous reward function is designed to evaluate the cumulative effect of the total processing time and the idle time of machines, aligning rewards with the scheduling objective of minimizing makespan. At the end of each episode, the reward is calculated as the difference between the total scheduling time area and the total processing time, so that smaller idle times and faster job completion yield higher rewards.	Single-objective	Makespan	The approach effectively handles varying-scale instances without handcrafted features and achieves strong performance in both static and dynamic settings. These results confirm the method's robustness, adaptability and potential for real-world intelligent scheduling applications.	The current setup assumes fixed trigger points for job arrivals and does not account for machine failures.	Handle more realistic dynamic uncertainties, potentially via multi-agent RL.	A novel rescheduling mechanism is introduced to efficiently integrate new job arrivals, enhancing adaptability and robustness without the need for retraining.
2025	*Deep reinforcement learning based proximal policy optimization algorithm for dynamic job shop scheduling*	*Minghai Yuan*, Qi Yu, Lizhi Zhang, Songwei Liu, Zichen Li, Fengque Pei	Machine failures and order changes	*DJSSP*	The paper proposes a DRL-based scheduling method using PPO for dynamic job-shop problems. The goal is to maximize the cumulative reward by enabling the agent to adopt optimal strategies when interacting with the environment, considering machine failures and order changes.	Sigle-Agent	The method uses an Actor–Critic framework based on **PPO.** At each decision point the Actor network outputs a probability distribution over the discrete set of scheduling rules, selecting the most suitable dispatching strategy for the current state. While, the Critic network evaluates the quality of that state, providing feedback that stabilizes learning and ensures gradual policy improvement. PPO's clipped update strategy prevents overly large policy changes, maintaining stable convergence.	The action space consists of **14 heuristic scheduling rules** that the agent can select whenever a machine is idle and multiple jobs are waiting.	The state space captures the real-time status of the job shop at each decision point, dynamically updated. Job-level features track current operation progress, remaining processing time, waiting time and delay risk,and also workload and availability of machines.	The reward function is defined at each decision point. The model is trained with different objectives in separate setups (in one training configuration, the reward function is defined to minimize average tardiness and in an other to minimize average processing time). The aim is to maximize the cumulative reward.	Single-objective	Avg tardiness / Avg processing time	By comparing different sized scheduling cases, the DRL approach can adapt to real-time dynamic shop floor production scheduling with higher efficiency than multiple single scheduling rules, DQN and genetic algorithms.	Single-objective optimization and the performance relies on how accurately the training environment reflects real conditions.	Incorporate more disturbance events and optimization objectives	The use of per-decision dense reward functions aligned with scheduling objectives , as it mitigates the sparse-reward problem common in job-shop scheduling.
2025	*Dynamic job shop scheduling under multiple order disturbances using deep reinforcement learning*	*Zhiyuan Sun*, *Wenmin Hon*, *Longlong Gao*, *Chenchen Zhu*, *Qiongshuai Lyu*	Dynamic order arrivals, cancellations or order due date changes	*DJSSP*	The paper proposes a multi-agent IPPO algorithm for dynamic job-shop scheduling under multiple order disturbances, aiming to minimize tardiness and makespan under multiple order disturbances. Agents learn adaptive policies from rich state features and diverse dispatching rules, with reward functions aligned to objectives.	Multi-Agent	The **IPPO** algorithm is a multi-agent PPO framework where each agent has two parallel Actor–Critic networks (new and old policy) and uses CNNs with Spatial Pyramid Pooling to handle variable input sizes caused by uncertainties.	The action space is defined as a **set of 16 dispatching rules**, which the agent selects from at each decision step. The diversity of dispatching rules is increased to enable the agent to fully learn the ability to selecting dispatching rules adaptively.	System state is represented using five feature channels, each encoded as a 2D matrix where rows correspond to operations and columns to orders. Two channels track overall system progress , one records the processing time of all unfinished operations and is updated to 0 when completed, while the other records completed operations by filling in their processing time once finished. Three local channels capture the remaining processing time of the operation currently being machined, the processing time of operations waiting in the queue, and the waiting time of queued operations.	The model is trained separately for two single-objective goals. When minimizing total tardiness, if tardiness decreases the agent receives a positive reward. Over an episode, the cumulative reward becomes the negative of the final total tardiness , so maximizing reward is equivalent to minimizing total tardiness. When minimizing makespan, the less time machines stay idle the higher the reward. Summing these rewards over the episode results in an inversely proportional value to the makespan.	Single-objective	Tardiness/ Makespan	The approach outperforms traditional PPO and dispatching rules across diverse scenarios, showing strong optimization and adaptability.	High computational demands for very large-scale problems.	Incorporate more disturbance events and improve the design of system states, action spaces and reward functions in complex production environments.	This paper introduces a novel approach using Independent Proximal Policy Optimization (IPPO) multiagent algorithm that has not been previously explored in this context, the proposal of a solution for multiobjective dynamic scheduling problems under multiple order disturbances and comprehensive analysis and comparison of experimental results across various parameter settings and scenarios.
2025	*Harnessing heterogeneous graph neural networks for Dynamic Job-Shop Scheduling Problem solutions*	*Chien-Liang Liu, Po-Hao Weng , Chun-Jan Tseng*	Random job arrivals and machine breakdowns	*DJSSP*	The goal is to develop a model that not only captures the complex interconnections, but also dynamically adapts to the evolving nature of real-world manufacturing systems. It introduces a specialized Hypergraph Neural Network (HGNN) architecture that captures the complex relationships and dependencies among jobs, machines, and operations. The model is size-agnostic, allowing it to generalize across different problem scales without retraining, making it highly flexible and scalable.	Sigle-Agent	Integration of **heterogeneous graph neural network (HGNN) with Proximal Policy Optimization (PPO)**. Introduction of Bootstrap Curriculum Learning (BCL) training that progressively introduces the model to more challenging scheduling problem cases.	The agent selects one operation to schedule on an available idle machine, as the action space consists of all currently feasible operation–machine assignments based on machine availability, job precedence, and system dynamics. If neither a machine is available nor any eligible operation exists, a null action is recorded, but such transitions are excluded from training to improve learning efficiency and consistency across static and dynamic settings.	The state space is represented as a disjunctive graph, where nodes represent operations and arcs capture relationships. Each node includes features such as status, processing time, remaining operations, degree of completion, and estimated completion time. The HGNN processes these features using GCNs for fixed precedence relations and GATs for dynamic machine conflicts.	The reward is designed to minimize makespan. At every decision step, the reward is calculated as the difference between the previous makespan and the current makespan (i.e. positive when the action leads to earlier completion). A discount factor is applied so future reductions are still valued, allowing the cumulative return to align with global makespan optimization rather than local improvements.	Single-objective	Makespan	Theresults demonstrate strong performance across both static and dynamic environments. Its size-agnostic design enables generalization from small training instances to larger unseen cases, while the disjunction-graph-based MDP formulation and adaptive HGNN architecture allow it to react intelligently to real-time disruptions.	Extremely large-scale instances may still pose computational challenges	Include setup and transportation times, moving closer to fully industrial deployment.	Novel Graph Embedding Technique,integration of HGNN and DRL,Bootstrap Curriculum Learning (BCL),Extensive Experimental Validation
2024	*Probing on LSTM-PPO-Based reinforcement learning algorithm to solve dynamic job shop scheduling problem*	*Wei Chen, Zequn Zhang, Dunbing Tang , Changchun Liu, Yong Gui, Qingwei Nie, Zhen Zhao*	Machine failure and emergency order insertion	*DJSSP*	To capture temporal dependencies, an **LSTM network** that can retain and interpret past information, enabling the agent to understand how the system evolves over time is used. While **PPO** optimizes the scheduling strategy, allowing the system to autonomously learn effective scheduling policies directly from real-time environment feedback.	Sigle-Agent	To capture temporal dependencies, an **LSTM network** that can retain and interpret past information, enabling the agent to understand how the system evolves over time is used. While **PPO** optimizes the scheduling strategy, allowing the system to autonomously learn effective scheduling policies directly from real-time environment feedback.	The action spece is defined as **6 scheduling rules**, and the agent selects the most appropriate rule based on the current job shop state and the attributes of workpieces in the buffer. When a machine becomes idle, the agent evaluates the input features and outputs a probability for each action, choosing the one with the highest probability as the next scheduling decision.	It includes features describing the status of all remaining jobs, while the machine state reflects overall system utilization alongside real-time attributes of individual machines. Since the various state attributes can differ significantly in scale, all information is normalized to the range [0,1], enhancing the model's ability to generalize across different job shop configurations.	For each action, the reward reflects the change relative to the previous state: a decrease in maximum completion time or total tardiness results in a positive reward, while an increase incurs a penalty. During training, the agent collects state-action trajectories, receives the corresponding reward values, and uses them with the critic network to update the policy via PPO, allowing the scheduling strategy to progressively maximize cumulative reward over time.	Multi-objective	Completion time and Total tardiness	It demonstrates fast solution speed, strong robustness to disturbances and superior performance compared with traditional scheduling methods, achieving lower completion times and total tardiness. The algorithm shows that it can adaptively make scheduling decisions under varying dynamic conditions.	Relies on a predefined set of scheduling rules and the effectiveness of the reward function depends on the choice of weighting parameters.	Explore hybrid or multi-agent reinforcement learning strategies to better handle complex, dynamic scheduling	A novel LSTM-PPO network is proposed, combining an LSTM network to capture temporal dependencies in job shop states with PPO for stable and efficient policy optimization, design of a multi-objective reward function
2024	*Smart scheduling of dynamic job shop based on discrete event simulation and deep reinforcement learning*	*Ziqing Wang, Wenzhu Liao*	Random job arrivals	*DJSSP*	This paper proposes a smart approach based on proximal policy optimization (PPO) to solve dynamic job shop scheduling problem with random job arrivals.	Sigle-Agent	**PPO-clip algorithm** based on DES: The actor and critic networks are updated multiple times. PPO-clip constrains policy updates by clipping the probability ratio, thereby preventing excessively large deviations from the previous policy and ensuring stable learning. After the update, the agent interacts with the DES environment using the updated policy, and this process is repeated until convergence.	The action space consists of **11 dispatching rules (DRs)** representing different job selection strategies in the scheduling process. At each decision point, the scheduling agent selects one rule from this set to determine which job from the waiting queue will be processed next.	It includes system-level features (number of jobs in the system and queues) and job-specific features such as time till due date, remaining operations , remaining processing time , next processing time, and slack time , as well as queue-related variables . These features are normalized and statistically summarized , forming a 30-dimensional state vector that represents the workshop's dynamic environment.	The reward function focuses on minimizing tardiness by using changes in total slack time as a feedback signal. The reward at each step is calculated as the difference in total slack time between consecutive decision points, scaled by a factor $\beta = 0.05$. This design provides continuous feedback, enabling the agent to learn scheduling strategies that proactively reduce job delays and improve production timeliness.	Single-objective	Tardiness	The model learns adaptive scheduling strategies that minimize mean tardiness and outperform traditional dispatching rules and heuristic methods. The PPO-based scheduler demonstrated strong stability, convergence, and generalization, showing potential for real-world implementation.	Dependence on feature design and computational demanding during training.	Focus on multi-objective optimization and multi-agent decision-making for distributed control.	The model defines a detailed MDP model with 30 state features and a reward function aimed at minimizing mean tardiness, along with a reward-scaling strategy for stable convergence.
2024	*Dynamic Job-Shop Scheduling via Graph Attention Networks and Deep Reinforcement Learning*	*Chien-Liang Liu, Po-Hao Weng , Chun-Jan Tseng*	Machine breakdowns and stochastic job arrivals	*DJSSP*	This article proposes deep reinforcement learning and graph neural networks to solve the DJSSP. A disjunctive graph, designing specific node features that reflect the unique characteristics of JSSP with machine breakdowns and stochastic job arrivals is used. The method can dynamically adapt to the occurrence of disruptions, ensuring that it accurately reflects the current state of the environment. The attention mechanism is used to prioritize crucial nodes while discarding irrelevant ones.	Sigle-Agent	The paper proposes a **Proximal Policy Optimization (PPO) based reinforcement learning framework enhanced by Graph Attention Networks (GATs)**: the agent observes a disjunctive graph representation of the production system and learns scheduling policies that minimize makespan. The GAT architecture allows the model to capture complex dependencies between operations and machines, while the PPO framework ensures stable policy updates through a clipped loss.	The action space includes all feasible operations available for scheduling at that moment. When no machines or operations are ready, the agent performs a null action (ignored during training).	The system state is modeled as a disjunctive graph, where nodes represent operations and arcs describe relationships. Each node contains five key features (status, processing time, remaining operations, degree of completion, and estimated completion time) capturing both job progress and machine conditions. In dynamic settings with random job arrivals and machine breakdowns, these features evolve over time, making the state fully time-dependent.	The reward is defined as the change in makespan between consecutive time steps, as the reduction from the previous to the current makespan. By using this incremental feedback, the agent learns to make scheduling decisions that cumulatively shorten production time. With a discount factor $\gamma = 1$, maximizing cumulative reward becomes equivalent to minimizing the final makespan.	Single-objective	Makespan	The model effectively captures job–machine relationships, while DRL enables adaptive decision-making under uncertainty. The method demonstrates strong generalization and robustness, achieving high-quality scheduling results across both static and dynamic conditions. Its size-agnostic design allows training on small instances and direct application to larger problems without retraining.	The current architecture might not capture all structural complexities of large-scale production systems.	Refinements to the GAT structure to improve performance.	A unified, size-agnostic model architecture that can be trained on small instances and directly applied to larger ones without retraining. Integration of GAT with an actor–critic PPO algorithm, enabling the agent to focus attention on the most informative nodes and learn high-quality scheduling policies.

Year	Title	Authors	Uncertainties	Problem	Description	Agent	Algorithm	Action space	State space	Reward function	Num. Objective	Obj. Indicator	Conclusions	Limitations	Future research	Contributions
2024	*A deep reinforcement learning model for dynamic job-shop scheduling problem with uncertain processing time*	*Xinquan Wu, Xuefeng Yan, Donghai Guan, Mingqiang Wei*	Uncertain processing time	*DJSSP*	The purpose of this paper is to improve the convergence and efficiency of DRL-based scheduling for large-scale DJSPs under uncertain processing times, using an enhanced PPO framework.	Sigle-Agent	**PPO framework with a hybrid prioritized experience replay (PER) mechanism** to improve learning efficiency and stability in complex, high-dimensional JSSP. The hybrid PER approach combines uniform sampling (to maintain diversity) with priority-based sampling (to emphasize valuable experiences). Priority is determined by the temporal-difference (TD) error, where samples with higher TD-errors are replayed more frequently.	The action space is constructed using **12 priority dispatching rules (PDRs)**. This design prevents illegal scheduling actions and maintains decision diversity without complex masking operations.	The proposed method introduces a feasible solution matrix as the state representation.This matrix directly encodes scheduling order information for all jobs and operations in a normalized 2D format. This approach eliminates the need for expert-designed features, reduces computational complexity, and captures both dynamic scheduling progress and historical decisions.	By computing rewards from the change in total idle time after each scheduling action, the method achieves a dense reward signal. Since the total processing time is constant, maximizing accumulated rewards directly corresponds to minimizing the makespan, ensuring full consistency between learning and scheduling goals.	Single-objective	Makespan	The reward function based on machine idle time improves learning efficiency and reduce the need for manual feature design, ensuring a clear relationship between rewards and makespan minimization. Experimental results show that the method outperforms DRL schedulers in both static and dynamic environments, effectively adapting to uncertain processing times.	Primarily handles uncertainties tied to processing times.	Focus on developing deep neural networks capable of generating fixed-length state representations, allowing the DRL scheduler to manage variable-scale dynamic scheduling problems.	This paper introduce a feasible solution matrix as a state representation that avoids handcrafted features while capturing dynamic job and machine information for better generalization. A machine idle time–based reward function is designed. To improve convergence and sample efficiency, a hybrid prioritized experience replay method is proposed, prioritizing positive TD-error samples with power-annealed batch sizes. The method demonstrates superior performance and stability across benchmark instances of varying sizes and under dynamic conditions with uncertain processing times. Its modular and size-agnostic framework allows easy integration with different DRL algorithms, making it a flexible and effective approach for both static and dynamic job shop scheduling problems.
2024	*Dynamic job-shop scheduling using graph reinforcement learning with auxiliary strategy*	*Zhenyu Liu, Haoyang Mao, Guodong Sa, Hui Liu, Jianrong Tan*	Stochastic processing time and machine breakdowns	*DJSSP*	Balance solution efficiency and dynamic adaptability for DJSP using a graph reinforcement learning framework.	Sigle-Agent	**Mixed Graph Transformer Network (MGTN)** is used to encode variable-sized disjunctive graphs into fixed-length state representations. A **Phasic Proximal Policy Optimization with Rollback (P3OR)** algorithm then trains the policy in alternating phases: a stabilized PPO-based policy update phase and an auxiliary phase that transfers critic features to the actor, improving sample efficiency and training stability.	The action space is defined as a set of **8 dispatching rules**. These rules represent heuristic strategies commonly used in scheduling and are divided into static and dynamic, such as *SPT*, *LPT*, *FIFO* and *SWT*, *LWT*, *LWKR*.	The state is represented using a disjunctive graph. Each node corresponds to an operation, with two dummy nodes marking the start and end of the entire schedule. The weight assigned to each node represents the processing time of that operation. The graph includes two types of connections: conjunctive arcs, which are directed and define the fixed precedence relations between operations of the same job, and disjunctive edges, which are undirected and connect operations that share the same machine. The scheduling process involves determining the directions of these disjunctive edges, resolving machine conflicts.	The objective is to minimize the makespan. To provide continuous feedback, average machine utilization is adopted as the reward objective, defined as the ratio of total effective processing time to the product of makespan and the number of machines. By maximizing machine utilization, the RL agent learns to minimize idle time and indirectly achieve a shorter makespan, allowing the reward function to offer denser and more stable guidance while remaining consistent with the original scheduling objective.	Single-objective	Makespan	The GRL-AS framework demonstrates strong robustness and generalization for DJSPs in both static and dynamic environments. It effectively handles variable instance sizes and avoids interference between objectives. Evaluations on benchmark and real-world cases show that it outperforms traditional and DRL-based methods, highlighting its potential for efficient and adaptable scheduling in smart manufacturing.	Depends on the diversity of training instances and the computational cost can be high for large-scale problems.	Extending the framework to other DJSP variants and multi-objective criteria.	The scheduling framework presented enables to automatically adjust optimal policies without rescheduling. A novel graph representation module (MGTN) is put forward for extracting state embeddings with static or dynamic events from disjunctive graphs of different sizes, effectively improving generalization. P3OR improves sample utilization by sharing parameters of the policy and value function while decoupling their respective training process. The proposed approach exhibits superior performance beyond other traditional and state-of-art DRL algorithms on both static and dynamic benchmarks.
2023	*Dynamic Job-Shop Scheduling Based on Transformer and Deep Reinforcement Learning*	*Liyuan Song, Yuanyuan Li, Jiacheng Xu*	Random job arrivals	*DJSSP*	This paper aims to address dynamic job-shop scheduling by combining transformer-based state modeling with deep reinforcement learning for adaptive and efficient decision-making.	Sigle-Agent	**Deep Dueling Double Q-Network (D3QN)** with prioritized experience replay and a **transformer-based feature extraction module.** D3QN estimates action values accurately and stabilizes learning in high-dimensional environments, while prioritized replay ensures that the agent focuses on high-value experiences. The transformer captures relationships between operations in the disjunctive graph, extracting meaningful state features.	Actions correspond to selecting from a **14 dispatching rules** that determine which job to schedule next. Data Envelopment Analysis (DEA) is used to evaluate and select the most suitable rules based on multiple objectives like makespan and machine utilization.	The state space is represented using a disjunctive graph that incorporates both static and dynamic information. Nodes represent operations, with features including job and operation IDs, machine assignment, node status, waiting time, remaining operations, and job completion rate. This structured representation captures long-term dependencies and dynamic changes in the scheduling environment, allowing the agent to make informed, adaptive decisions.	The reward function converts the long-term objective of minimizing makespan into an immediate reward by maximizing machine utilization at each scheduling step. This approach allows the agent to evaluate the impact of individual decisions on overall performance. Cumulative rewards reflect incremental improvements in machine usage, guiding the learning process and improving the quality of scheduling decisions.	Single-objective	Makespan	It introduces a dynamic exploration–exploitation strategy and demonstrates that the approach improves scheduling efficiency. The transformer-based feature extraction and the DEA-selected action space significantly enhance performance, consistently outperforming traditional dispatching rules, genetic algorithms, and other DRL methods.	The model is single-agent and focuses on minimizing makespan as a single objective.	Extend to multi-agent environments and incorporate multi-objective optimization beyond makespan.	It contributes a novel state representation using disjunctive graphs, eliminating manual feature engineering, and an action space designed through data envelopment analysis to dynamically select the most effective dispatching rules. The framework also integrates a dynamic exploration–exploitation strategy with an elite mechanism to balance learning efficiency and robustness. The use of a transformer for feature extraction enhances global dependency modeling, outperforming traditional GNN and matrix approaches. Extensive experiments confirm that the proposed method significantly improves scheduling efficiency and generalization.
2023	*A spatial pyramid pooling-based deep reinforcement learning model for dynamic job-shop scheduling problem*	*Xinquan Wu, Xuefeng Yan*	New orders insertion	*DJSSP*	A deep reinforcement learning (DRL) scheduling model based on spatial pyramid pooling networks to solve the DJSP with variant sizes is proposed.	Sigle-Agent	Actor–critic architecture trained with the **PPO algorithm**. The policy network incorporates a Spatial Pyramid Pooling (SPP) module that transforms variable-sized state representations into fixed-length feature vectors, enabling a size-agnostic framework capable of handling dynamic job arrivals and machine changes.	**Twelve priority dispatching rule (PDR)** define the action space including well-known heuristics such as SPT, MWKR, FIFO, and their opposite-priority counterparts, are used to enhance diversity and robustness in scheduling decisions. The paired PDR design ensures a fixed, size-independent action space that generalizes across different problem instances	The paper proposes a state representation for dynamic job shop scheduling that avoids handcrafted features by encoding the environment as an instance matrix. This matrix is decomposed into a machine matrix and a remaining time matrix, forming two image-like channels that evolve with the scheduling process. A third zero matrix is added to create a three-channel input suitable for CNN-based feature extraction. The method reflects operation status and includes a merging mechanism for newly arrived jobs, enabling flexible rescheduling.	The function integrates processing time and idle time across both real and imaginary machines, with the latter serving as a flexible coefficient to adjust the relationship between accumulated reward and makespan. This design ensures a negative linear dependency between the two objectives, making reward maximization equivalent to makespan minimization. By capturing scheduling efficiency through total time-area representation, the method enhances training stability and interpretability.	Single-objective	Makespan	The study proposes a DRL-based scheduling framework for DJSP that effectively handles dynamic events such as new job arrivals. Experimental results show that the method outperforms traditional PDR and existing DRL approaches in both static and dynamic settings. Among rescheduling strategies, the reused policy offers the best balance between performance and computational efficiency, achieving rapid decision-making while adapting effectively to changing environments.	Focuses on single-objective optimization.	Improving the model's performance and extending it to more complex job shop scheduling scenarios.	Experiments show that the PPO scheduling agent surpasses traditional dispatching and heuristic methods, demonstrating robust adaptability and generalization. The approach also supports distributed control, where trained agents can be deployed across machines for intelligent, self-optimizing manufacturing under Industry 4.0.
2023	*A deep multi-agent reinforcement learning approach to solve dynamic job shop scheduling problem*	*Renke Liu, Rajesh Piplani, Carlos Toro*	Random job arrivals	*DJSSP*	This paper aims to enable near real-time and scalable dynamic job shop scheduling under volatile production environments using a deep multi-agent reinforcement learning framework.	Multi-Agent	**Multi-agent reinforcement learning (MARL)**, where each machine acts as an autonomous agent. A centralized training–decentralized execution scheme with parameter sharing is adopted to ensure consistent learning among agents while maintaining decentralized decision-making during operation.	The action space simply corresponds to the state space, in which actions are pointed to four slots of candidate jobs (possibly more than one slot direct to one job). The slot with the highest state-action value would be selected to be processed. Subsequently, the mapping from job-specific features to the direct selection of jobs can be learned.	The state space is a 5x5 matrix consisting of the information of both queuing and arriving jobs (if any). The top four rows could contain duplicated elements when the number of queuing jobs is less than four. The state space incorporates detailed information about up to four candidate jobs, selected according to four well-known sequencing rules: SPT, LWKR, MS, and WINQ. Each candidate job is described through five features representing processing time, expected completion, slack, next-queue waiting time, and detention time.	The proposed chronological joint action structure reconstructs each job's production history, allocating penalties to agents proportionally to their contribution to the job's delay. Rewards are computed upon job completion by tracking queue times and applying weighting based on each job's slack time, ensuring higher penalties for detaining urgent jobs. This design effectively links local scheduling decisions with global performance, fostering cooperative behavior among agents and improving learning convergence.	Single-objective	Tardiness	By combining centralized training, decentralized execution and parameter sharing, the proposed method effectively adapts DRL to real-world factory conditions. Novel state–action representations and knowledge-based reward shaping enable agents to manage dynamic environments while maintaining scalability. Experimental and ablation results confirm that the approach significantly outperforms existing algorithms and traditional dispatching rules in both efficiency and robustness.	The handcrafted state features rely on expert knowledge, reducing generalizability.	Improve model performance involving deeper neural architectures, automated data preprocessing, and feature selection to manage complex industrial data.	Integrated deep MARL algorithm to perform dynamic job shop scheduling. Specialized state and action representations that can tackle size-agnostic scheduling tasks, which are scalable in dynamic environments, Chronological joint-action framework to alleviate the credit-assignment difficulty in scheduling problem and knowledge-based reward-shaping techniques to encourage cooperation among agents thus improving the system-level performance.
2023	*Evolution strategies-based optimized graph reinforcement learning for solving dynamic job shop scheduling problem*	*Chupeng Su, Cong Zhang, Dan Xia and Baoan Han, Chuang Wang, Gang Chen, Longhan Xie*	Machine breakdown and stochastic processing time	*DJSSP*	This paper develop a robust and adaptive framework for solving dynamic job-shop scheduling problems under machine breakdowns and stochastic processing times. By integrating GNN-based state representations with deep reinforcement learning and evolution strategies, the method aims to achieve stable and high-quality scheduling decisions in uncertain	Sigle-Agent	The model uses **Evolution Strategies (ES)**: ES optimizes model parameters by adding Gaussian noise to generate multiple perturbed versions, evaluating their performance via a fitness function, and updating parameters based on the aggregated feedback.	The action space is defined as the set of all available operations that can be scheduled at a given time. The policy network, trained via the graph embeddings learned from the GIN, directly selects which operation to process next. This approach avoids predefined dispatching rules and allows the agent to make context-aware scheduling decisions, dynamically adapting to changes in the system.	The state space is modeled as a graph representation. Each node represents an operation and contains attributes such as processing time, machine availability, and remaining workload. Edges encode precedence and machine constraints. This structure allows the Graph Isomorphism Network (GIN) to learn both local dependencies (within a job or machine) and global production dynamics.	fitness function derived from the makespan minimization objective, trained using Evolution Strategies (ES). The fitness is inversely related to the makespan, meaning shorter schedules yield higher fitness. To ensure fairness and generalization across different problem scales, the fitness is normalized by considering total processing time and instance complexity.	Single-objective	Makespan	By representing scheduling states as graphs and using Graph Isomorphism Networks (GIN) for embedding, the model effectively captures dynamic and stochastic features. The trained GRL agent demonstrates superior performance over both traditional and DRL-based methods in static and dynamic environments, offering high solution quality, robustness to uncertainties, and near-linear computational efficiency.	Its focus is limited to single-objective optimization and specific types of dynamic disturbances.	Extend the GRL and ES framework to handle multi-objective optimization and more complex scheduling variants	It introduces a graph-based state representation that captures both dynamic and stochastic features of the scheduling environment, enabling the model to handle real-world uncertainties. A Graph Isomorphism Network (GIN) is used to learn rich embeddings of operations and machines, which guide the policy in selecting optimal scheduling actions. The model is trained using Evolution Strategies (ES), providing a stable, parallelizable, and reward-free optimization process.
2023	*Dynamic production scheduling towards self-organizing mass personalization: A multi-agent dueling deep reinforcement learning approach*	*Zhaojun Qin, Dazzle Johnson, Yuqian Lu*	Machine breakdown and stochastic processing time	*DFJSSP*	Enable stable and adaptive job shop scheduling to minimize makespan under dynamic and uncertain manufacturing conditions using multi-agent reinforcement learning approach.	Multi-Agent	**Multi-Agent Deep Duelling Reinforcement Learning (MARL)** to learn scheduling policies for each agent/machine via static training-dynamic execution approach. The **Duelling DQN** is used as the architecture for the multi-agent system, where each agent has a Q-network and a target network. Also, an **ε-greedy policy** is used as cooperation mechanism to balance the exploration of new schedules and the exploitation of existing best schedules.	Each agent's action space consists of a vector mapping available operations of all jobs. At each decision step, an agent chooses one feasible operation or remains idle if none are available. Infeasible actions are masked by setting their Q-values to zero, ensuring only valid scheduling choices are made. The ε-greedy policy governs action selection: agents primarily exploit the best-known actions but occasionally explore alternatives. When multiple agents compete for the same operation, the cooperation mechanism resolves conflicts based on Q-values, ensuring coordinated decision-making across machines.	The state space of each agent represents the current status of all jobs and machines at a given scheduling step. It is structured as a matrix containing job-related such as whether a job has an allocable operation, progress of the current operation, number of finished operations, completion progress by total processing time, and machine-related features like machine utilization rate, number of allocable operations,operation completion progress,processing time progress.	The reward function combines both immediate and terminal components to guide agents toward minimizing the makespan. The immediate reward encourages high machine utilization, promoting efficient use of resources. The terminal reward penalizes total idle time once all jobs are completed, reflecting the overall efficiency of the scheduling sequence. By distributing the terminal reward equally across all actions, the framework ensures each decision contributes fairly to the final performance. This design balances short-term utilization with the long-term goal of reducing makespan, fostering cooperative behavior among agents.	Single-objective	Makespan	The method enables distributed decision-making and cooperative optimization in complex manufacturing systems. The scheduling policies are trained on static instances and then effectively transferred to dynamic environments without retraining. The framework integrates observation, action and reward representations, along with an agent cooperation mechanism, allowing agents to balance individual exploration and collective optimality. The approach consistently outperformed five heuristic baselines, demonstrating robustness and adaptability under diverse dynamic conditions.	The trained policies must be re-generated whenever the production setup changes significantly.	Formulate multi-objective versions of the DFJSP, incorporating additional production factors like setup and transport times and improving policy generalization to handle structural changes in job shop configurations.	The authors introduce a static-training–dynamic-execution strategy, where agents learn scheduling policies from static instances and effectively transfer them to dynamic environments with stochastic processing times and machine breakdowns without retraining. A carefully designed observation, action, and reward structure, combined with a cooperation mechanism based on an ε-greedy policy, enables agents to make coordinated scheduling decisions that balance machine utilization and makespan minimization. Using a dueling DQN architecture, the model efficiently distinguishes valuable states and learns stable policies.
2023	*Combining Reinforcement Learning Algorithms with Graph Neural Networks to Solve Dynamic Job Shop Scheduling Problems*	*Zhong Yang, Li Bi, Xiaogang Jiao*	Random arrival jobs	*DJSSP*	This paper aims to improve dynamic job shop scheduling by reducing reliance on handcrafted state representations through the integration of graph neural networks and deep reinforcement learning.	Sigle-Agent	The proposed scheduling algorithm integrates a **Graph Neural Network (GNN) with a Double Deep Q-Network (DDQN)**. The GNN first encodes the job shop's complex relational structure into a graph-based state representation. This state is then fed into the DDQN, which decides which operation to schedule next by balancing exploration and exploitation through ε-greedy action selection. The DDQN uses two Q-networks to reduce overestimation bias during value updates: the online network selects the optimal action, while the target one evaluates it.	The action space in the proposed framework is defined as **6 dispatching rules**. To improve exploration and generalization, an ε-greedy strategy is employed: the agent begins training with a high level of randomness ($\varepsilon = 0.6$) to encourage diverse exploration, which gradually decreases to 0.01 as training progresses, allowing the model to exploit learned optimal policies.	The state representationt is modeled as a graph structure. Each node carries a feature vector containing scheduling information such as job and operation indices, machine assignment, arrival time, due date, current start time, process state and job urgency. The GNN aggregates information from each node and its neighbors using graph convolutional layers, enabling the model to automatically learn complex dependencies between jobs and machines without relying on handcrafted scheduling heuristics. A normalization step (Laplacian transformation) ensures stable information propagation and prevents feature amplification. The flattened output of the GNN forms the state input to the reinforcement learning module, allowing the agent to perceive a compact yet expressive representation of the entire shop floor in real time.	The reward is computed based on the total completion time of the scheduling sequence, specifically comparing the current completion time to the best completion time achieved so far. A logarithmic term is used to smooth large time differences and stabilize learning, while a small constant ensures numerical safety. The model also employs an elite retention strategy, rewarding solutions that finish within 105% of the best known completion time and penalizing less efficient schedules. This reward design encourages the agent to progressively minimize total makespan while maintaining stable learning dynamics.	Single-objective	Makespan	By designing tailored state, action, and reward structures, the approach effectively captures the dynamic characteristics of job shops. Experimental comparisons confirm that it enhances optimization performance and adaptability, outperforming traditional rule-based and DRL methods.	The model does not account for other real-world perturbations such as machine breakdowns, or varying operational constraints.	Incorporate additional uncertainty factors into the scheduling environment and develop multi-objective optimization capabilities.	The key contributions include the graph-based state representation, which captures structural and dynamic relationships in the job shop without relying on handcrafted features and the integration of GNN with DDQN, improving learning efficiency and convergence over traditional DRL methods. The model also defines a rule-based action space and a globally designed reward function that effectively guides learning toward minimizing makespan. Experimental results demonstrate that this hybrid approach outperforms conventional heuristic and DRL baselines, highlighting its robustness and superior optimization performance under dynamic job arrivals.
2022	*Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems*	*Yi Zhang, Haihua Zhu, Dunbing Tang, Tong Zhou, Yong Gui*	Stochastic job insertions and unpredictable machine failures	*DJSSP*	This paper presents a multi-agent manufacturing system based on deep reinforcement learning (DRL), which integrates the self-organization mechanism and self-learning strategy.	Multi-Agent	The paper applies a **Proximal Policy Optimization (PPO)** reinforcement learning algorithm.	The discrete action space includes two types: standby and machine allocation. The standby action indicates that all machines are currently busy, prompting the system to wait until a task is completed. The machine allocation actions assign the current job to a specific machine for processing. This formulation allows the model to dynamically balance task dispatching and waiting, ensuring that the system avoids overloading machines and maintains smooth workflow execution.	The state space captures the complete condition of the workshop at a given time. It includes three operation attributes for each job (process type, required processing time and job's urgency). Machine states are also represented by three elements: machine type, buffer occupancy and total occupation time, reflecting their workload and availability.	The reward function evaluates each scheduling decision based on a composite reward–penalty mechanism. Valid actions yield a composite reward that combines three optimization factors: ahead-of-schedule time, promoting early task completion; workload balance, encouraging even machine utilization; and production profit, rewarding economically efficient decisions. Invalid actions, such as assigning infeasible tasks to machines, incur a fixed large penalty. The overall learning objective is to maximize the cumulative reward, guiding the AI scheduler toward efficient, balanced, and profitable scheduling strategies.	Multi-objective	Earliness, Workload balance, production profit	Each machine operates as an autonomous agent capable of perception, decision-making, and coordination. The scheduler continuously optimizes scheduling policies based on real-time data and past experience, enabling the system to efficiently handle dynamic events such as machine failures and order insertions. Experiments demonstrate that this approach outperforms traditional methods like GP, DQN, and rule-based scheduling in responsiveness, learning efficiency, and adaptability.	The state space and action space are oversimplified, lacking certain operational factors that could enhance decision accuracy.	Expand the scale and diversity of machines and job types and enrich the state space with additional operational attributes such as order priority.	It develops an AI scheduler embedded in each machine agent, capable of real-time decision-making and self-optimization through continuous learning. The approach achieves superior performance in responsiveness, workload balance, and adaptability to disturbances compared to traditional GP, DQN, and rule-based methods, demonstrating a significant advancement in intelligent, autonomous production scheduling.

Bibliography

[1] Mario Rupp, Max Schneckenburger, Markus Merkel, Rainer Börret, and David K. Harrison. «Industry 4.0: A Technological-Oriented Definition Based on Bibliometric Analysis and Literature Review». In: *Journal of Open Innovation: Technology, Market, and Complexity* 7.1 (2021), p. 68. ISSN: 2199-8531. DOI: https://doi.org/10.3390/joitmc7010068. URL: https://www.sciencedirect.com/science/article/pii/S219985312200837X (cit. on pp. 4, 5).

[2] Minghai Yuan, Qi Yu, Lizhi Zhang, Songwei Lu, Zichen Li, and Fengque Pei. «Deep reinforcement learning based proximal policy optimization algorithm for dynamic job shop scheduling». In: *Computers Operations Research* 183 (2025), p. 107149. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2025.107149. URL: https://www.sciencedirect.com/science/article/pii/S0305054825001777 (cit. on pp. 8, 11, 16, 21, 29).

[3] Hegen Xiong, Shuangyuan Shi, Danni Ren, and Jinjin Hu. «A survey of job shop scheduling problem: The types and models». In: *Computers & Operations Research* 142 (2022), p. 105731. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2022.105731. URL: https://www.sciencedirect.com/science/article/pii/S0305054822000338 (cit. on p. 9).

[4] Athanasios Spanos, Sotiris Gayialis, and Ilias Tatsiopoulos. «An Overview of Classical and Modern Algorithms for the Job Shop Scheduling Problem». In: June 2007 (cit. on p. 9).

[5] Amer Alnuaimi and Tasnim Albaldawi. «An overview of machine learning classification techniques». In: *BIO Web of Conferences* 97 (Apr. 2024), p. 00133. DOI: 10.1051/bioconf/20249700133 (cit. on pp. 13, 14).

[6] Olivia Hunter, Frances Perry, Mina Salehi, Hubert Bandurski, Alan Hubbard, Chad Ball, and Sannia Hameed. «Science fiction or clinical reality: a review of the applications of artificial intelligence along the continuum of trauma care». In: *World Journal of Emergency Surgery* 18 (Mar. 2023). DOI: 10.1186/s13017-022-00469-1 (cit. on p. 14).

[7] Xinquan Wu, Xuefeng Yan, Donghai Guan, and Mingqiang Wei. «A deep reinforcement learning model for dynamic job-shop scheduling problem with uncertain processing time». In: *Engineering Applications of Artificial Intelligence* 131 (2024), p. 107790. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2023.107790. URL: https://www.sciencedirect.com/science/article/pii/S0952197623019747 (cit. on pp. 15, 21, 38, 39).

[8] Kai Zhou, Shuai Yang, Jingtao Zhang, Xiaojun Long, and Zhen Wang. «Research on Intelligent Scheduling and Monitoring Method of Workshop Logistics System». In: *Journal of Physics: Conference Series* 2033 (Sept. 2021), p. 012172. DOI: 10.1088/1742-6596/2033/1/012172 (cit. on p. 15).

[9] Xinquan Wu and Xuefeng Yan. «A spatial pyramid pooling-based deep reinforcement learning model for dynamic job-shop scheduling problem». In: *Computers Operations Research* 160 (2023), p. 106401. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2023.106401. URL: https://www.sciencedirect.com/science/article/pii/S0305054823002654 (cit. on pp. 21, 44, 45).

[10] Wei Chen, Zequn Zhang, Dunbing Tang, Changchun Liu, Yong Gui, Qingwei Nie, and Zhen Zhao. «Probing an LSTM-PPO-Based reinforcement learning algorithm to solve dynamic job shop scheduling problem». In: *Computers Industrial Engineering* 197 (2024), p. 110633. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2024.110633. URL: https://www.sciencedirect.com/science/article/pii/S0360835224007551 (cit. on pp. 21, 33, 34).

[11] Haoyang Yu, Wenbin Gu, Na Tang, and Zhenyang Guo. «A deep reinforcement learning approach for dynamic job-shop scheduling problem considering time variable and new job arrivals». In: *Computers Operations Research* 185 (2026), p. 107263. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2025.107263. URL: https://www.sciencedirect.com/science/article/pii/S0305054825002928 (cit. on pp. 21, 27).

[12] Chien-Liang Liu, Po-Hao Weng, and Chun-Jan Tseng. «Harnessing heterogeneous graph neural networks for Dynamic Job-Shop Scheduling Problem solutions». In: *Computers Industrial Engineering* 203 (2025), p. 111060. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2025.111060. URL: https://www.sciencedirect.com/science/article/pii/S0360835225002062 (cit. on pp. 22, 32).

[13] Chien-Liang Liu, Chun-Jan Tseng, and Po-Hao Weng. «Dynamic Job-Shop Scheduling via Graph Attention Networks and Deep Reinforcement Learning». In: *IEEE Transactions on Industrial Informatics* PP (June 2024), pp. 1–11. DOI: 10.1109/TII.2024.3371489 (cit. on pp. 22, 37).

[14] Zhenyu Liu, Haoyang Mao, Guodong Sa, Hui Liu, and Jianrong Tan. «Dynamic job-shop scheduling using graph reinforcement learning with auxiliary strategy». In: *Journal of Manufacturing Systems* 73 (2024), pp. 1–18. ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2024.01.002. URL: https://www.sciencedirect.com/science/article/pii/S0278612524000025 (cit. on pp. 22, 40, 42).

[15] Chupeng Su, Cong Zhang, Dan Xia, Baoan Han, Chuang Wang, Gang Chen, and Longhan Xie. «Evolution strategies-based optimized graph reinforcement learning for solving dynamic job shop scheduling problem». In: *Applied Soft Computing* 145 (2023), p. 110596. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2023.110596. URL: https://www.sciencedirect.com/science/article/pii/S1568494623006142 (cit. on pp. 22, 48).

[16] Zhong Yang, Li Bi, and Xiaogang Jiao. «Combining Reinforcement Learning Algorithms with Graph Neural Networks to Solve Dynamic Job Shop Scheduling Problems». In: *Processes* 11.5 (2023). ISSN: 2227-9717. DOI: 10.3390/pr11051571. URL: https://www.mdpi.com/2227-9717/11/5/1571 (cit. on pp. 22, 51).

[17] Renke Liu, Rajesh Piplani, and Carlos Toro. «A deep multi-agent reinforcement learning approach to solve dynamic job shop scheduling problem». In: *Computers Operations Research* 159 (2023), p. 106294. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2023.106294. URL: https://www.sciencedirect.com/science/article/pii/S0305054823001582 (cit. on pp. 22, 46).

[18] Zhaojun Qin, Dazzle Johnson, and Yuqian Lu. «Dynamic production scheduling towards self-organizing mass personalization: A multi-agent dueling deep reinforcement learning approach». In: *Journal of Manufacturing Systems* 68 (2023), pp. 242–257. ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2023.03.003. URL: https://www.sciencedirect.com/science/article/pii/S0278612523000468 (cit. on pp. 22, 50).

[19] Yi Zhang, Haihua Zhu, Dunbing Tang, Tong Zhou, and Yong Gui. «Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems». In: *Robotics and Computer-Integrated Manufacturing* 78 (2022), p. 102412. ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2022.102412. URL: https://www.sciencedirect.com/science/article/pii/S0736584522000977 (cit. on pp. 22, 54, 55).

[20] Zhiyuan Sun, Wenmin Han, Longlong Gao, Chenchen Zhu, and Qiongshuai Lyu. «Dynamic job shop scheduling under multiple order disturbances using deep reinforcement learning». In: *Science Progress* 108 (Apr. 2025). DOI: 10.1177/00368504251334281 (cit. on pp. 22, 30, 31).

[21] Wenquan Zhang, Zhaoxian Peng, Fei Zhao, Bo Feng, and Xuesong Mei. «A novel deep reinforcement learning framework based on digital twins for dynamic job shop scheduling problems». In: *Expert Systems with Applications* 296 (2026), p. 128708. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2025.128708. URL: https://www.sciencedirect.com/science/article/pii/S0957417425023267 (cit. on pp. 22, 25).

[22] Ziqing Wang and Wenzhu Liao. «Smart scheduling of dynamic job shop based on discrete event simulation and deep reinforcement learning». In: *Journal of Intelligent Manufacturing* 35 (June 2023), pp. 1–18. DOI: 10.1007/s10845-023-02161-w (cit. on pp. 22, 35).

[23] Liyuan Song, Yuanyuan Li, and Jiacheng Xu. «Dynamic Job-Shop Scheduling Based on Transformer and Deep Reinforcement Learning». In: *Processes* 11 (Dec. 2023), p. 3434. DOI: 10.3390/pr11123434 (cit. on pp. 22, 43).