
OPTIMIZING JOB SHOP SCHEDULING IN THE FURNITURE INDUSTRY: A REINFORCEMENT LEARNING APPROACH CONSIDERING MACHINE SETUP, BATCH VARIABILITY, AND INTRALOGISTICS

Malte Schneevogt, M.Sc.

(Lead Author)

Faculty of Wood Technology and Construction
Rosenheim Technical University of Applied Sciences
Hochschulstr. 1, 83024 Rosenheim
malte.schneevogt@stud.th-rosenheim.de

Dipl.-Ing (FH) Karsten Binninger, M.Sc.

(Supervising Scientist)

Center for Research, Development and Transfer
Rosenheim Technical University of Applied Sciences
Hochschulstr. 1, 83024 Rosenheim
karsten.binninger@th-rosenheim.de

Prof. Dr.-Ing. Noah Klarmann

(Supervising Professor)

Faculty of Management and Engineering
Rosenheim Technical University of Applied Sciences
Hochschulstr. 1, 83024 Rosenheim
noah.klarmann@th-rosenheim.de

September 19, 2024

ABSTRACT

This paper explores the potential application of Deep Reinforcement Learning (DRL) in the furniture industry. In order to offer a broad product portfolio, most furniture manufacturers are organized as a job shop, which ultimately results in the Job Shop Scheduling Problem (JSSP). The JSSP is addressed with a focus on extending traditional models to better represent the complexities of real-world production environments. Existing approaches to JSSPs frequently fail to consider critical factors such as machine setup times, varying batch sizes, intralogistics, buffer capacities, or deadlines, which are essential in industrial settings. In order to overcome these limitations, a concept for a model is proposed that incorporates these elements, providing a higher level of information detail to enhance scheduling accuracy and efficiency. The concept introduces the integration of DRL for production planning, which is particularly suited to batch production industries such as the furniture industry. The model extends traditional approaches to JSSPs by including job volumes, buffer management, transportation times, and machine setup times. This enables more precise forecasting and analysis of production flows and processes, accommodating the variability and complexity inherent in real-world manufacturing processes. In a training environment the Reinforcement Learning (RL) agent learns to optimize scheduling decisions. The agent operates within a discrete action space, making decisions based on detailed observations of machine states, job volumes, and buffer statuses. A reward function, specifically tailored to the specific industrial applications, guides the agent's decision-making process, thereby promoting efficient scheduling and meeting production deadlines. Two integration strategies for implementing the RL agent are discussed: episodic planning, which is suitable for low-automation environments, and continuous planning, which is ideal for highly automated plants. While episodic planning can be employed as a standalone solution, the continuous planning approach necessitates the integration of the agent with Enterprise Resource Planning (ERP) and Manufacturing Execution Systems (MES). This integration enables real-time adjustments to production schedules based on dynamic changes.

Keywords Job Shop Scheduling · Production Scheduling · Reinforcement Learning · Markov Decision Process

1 Introduction

The optimization of production planning is a crucial aspect of industrial manufacturing processes. It increases the overall production efficiency, ensures timely delivery, and reduces costs by utilizing resources effectively. In recent years, Reinforcement Learning (RL) has emerged as a powerful tool in solving complex optimization problems across various domains. Its ability to learn optimal policies through interaction with environments has revolutionised diverse domains, showcasing its versatility and effectiveness. In robotics, RL has been utilized for autonomous navigation, manipulation, and control tasks [1][8], in gaming, it demonstrates superhuman performance in complex games, such as Go [19] or Dota 2 [10]. RL offers several advantages over traditional optimization methods, including its adaptability to dynamic environments, its ability to handle large state and action spaces, and its capability to learn from experience without requiring explicit problem knowledge. These advantages make it an ideal tool for industrial process optimization.

Due to its high level of complexity, the furniture industry requires a particularly diligent and optimized production planning. Its products are composed of numerous individual components, each of which undergoes separate manufacturing processes. A furniture article can be made from hundreds of components, each with an individual path through the production. Some components are simply purchased parts that do not require any further processing in the factory, while other parts involve complex manufacturing processes among many different machines. These processes need to be coordinated and integrated during production, particularly if various components or articles are produced on the same production lines, resulting in the batch production of products or components. In order to produce a wide range of products, most furniture shops are designed as job shops. This setup offers a high degree of flexibility for batch production, but inherently leads to the Job Shop Scheduling Program (JSSP), a Combinatorial Optimization Problem (COP) known from process optimization.

In a JSSP a set of n jobs $J = \{J_0, J_1, J_2, \dots, J_n\}$ is to be processed on m machines $M = \{M_1, M_2, \dots, M_m\}$. Each machine can only process one operation at a time. Each job is assigned with a specific machine sequence that must be followed during the production of the particular product. For each machine, the operations have specific processing times d_{ij} where $i \in (1, m)$ and $j \in (1, n)$. The total number of operations O is $n \times m$. The number of possible schedules in a JSSP is growing exponentially by $(n!)^m$. Common scheduling heuristics such as First-Come-First-Serve or Earliest-Due-Date-First become ineffective as the number of jobs and machines increases, and computation times become unfeasible due to the exponential growth of combinatorial possibilities. Algorithms such as Genetic Algorithms [14], Simulated Annealing [23], or Taboo Search [21] struggle to effectively solve the JSSP due to its NP-hard nature and the presence of complex dependencies among operations and resources [15].

Deep Reinforcement Learning (DRL) was successfully applied to the JSSP on several occasions, finding competitive solutions for JSSP benchmark problems [26, 6, 9, 27, 25, 17]. DRL is a machine learning technique that combines deep learning and RL to enable agents to learn and make decisions in complex environments by using neural networks to process and act on high-dimensional data. The agent takes actions based on the observations and its policy, receives rewards as feedback, and adjusts its neural network parameters in order to maximize cumulated future rewards, the so-called return (cf. Figure 1). This process enables the agent to learn optimal behaviours for complex tasks in high-dimensional spaces.

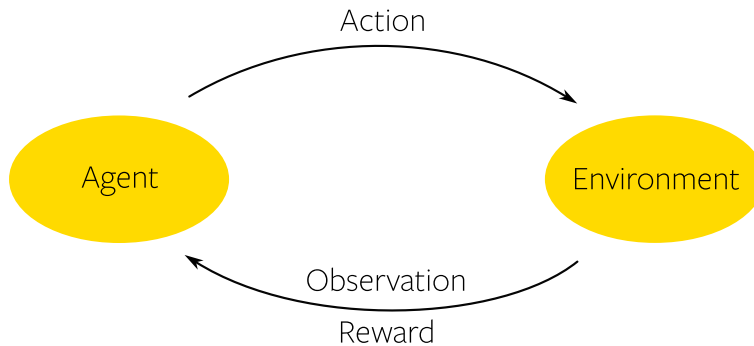


Figure 1: Reinforcement Learning Circuit [20]

Advances in deep learning techniques, particularly the development of deep neural networks, have significantly improved the ability of RL algorithms to handle high-dimensional and continuous state spaces. Due to their ability to approximate complex value functions and policies, DRL algorithms, such as Deep Q-Networks (DQN) and Proximal Policy optimization (PPO), have demonstrated remarkable performance in challenging tasks. The availability of large-scale computing resources, coupled with distributed computing frameworks such as TensorFlow or PyTorch,

has enabled researchers to efficiently train RL agents even on complex environments. This scalability has facilitated the exploration of RL in real-world applications, where complex decision-making processes are prevalent. The JSSP is a sequential decision-making process, and thus it can be modelled as a Markov Decision Process (MDP). This mathematical framework is commonly used in Artificial Intelligence (AI) and operations research for decision-making in uncertain environments where probabilistic state transitions can be influenced by the agent's actions. While a generic JSSP provides a good theoretical model for scheduling in general, real-world production environments often have additional complexities that are not captured in the JSSP model. This paper presents a framework for a DRL model that addresses the complexity of a real-world production environment and demonstrates how the proposed model can be applied in the furniture industry.

2 State of the Art

As early as 1995, Zhang und Dietterich [28] explored the application of RL techniques to the JSSP by further developing the results of the scheduling algorithms by Deale et al. [3], who used a simulated annealing approach for job shop scheduling. The system learns to make informed decisions that lead to more efficient schedules and thereby shows the potential of RL to solve JSSPs.

In 2000, Aydin and Öztemel [1] presented an approach that is composed of a simulated job shop environment and a RL agent that selects the most appropriate priority rule from a set of available rules to assign jobs to machines. The aim of their work is to provide a flexible and adaptive approach to job shop scheduling, capable of handling a dynamic manufacturing environment, coming one step closer to a fully automated, intelligent manufacturing system.

Gabel and Riedmiller [4] introduced a novel approach to solve JSSPs in 2012, by using distributed policy search RL. This multi-agent approach treats the JSSP as a series of sequential decision-making tasks, where each RL-agent operates autonomously and uses a probabilistic dispatching policy for decision making. These dispatching policies are represented by a small set of real-valued parameters that are continuously adapted and refined, to enhance the overall performance of the scheduling process. Although the computation time was reduced, the achieved solutions were not better than those of conventional solvers.

In 2008, Pezzella et al. [14] presented a genetic algorithm that solves the Flexible Job Shop Scheduling Problem (FJSP). Unlike in a JSSP, the operations of a FJSP can be assigned to one of several machines, instead of only one specific machine. Due to the additional decision layer of selecting machines for each operation, the complexity of a FJSP is even higher than in a JSSP. The presented algorithm outperformed existing models and traditional dispatching rules. Their approach utilizes DRL to tackle the problem more effectively by including innovative approaches for representation learning and policy learning. They demonstrated that genetic algorithms are effective in solving FJSPs.

Foundational research in the field of DRL for continuous control tasks was presented by Lillicrap et al. [8] in 2015. This research expands the capabilities of deep learning beyond the discrete domain to tasks where actions can take any value within a continuous range. They employ an actor-critic architecture where the actor generates actions, and the critic evaluates them based on a learned value function. Also, they use a replay buffer to store and reuse past experiences to mimic successful techniques.

Shahrabi et al. [18] address the complex problem of dynamic job shop scheduling, where job arrivals are unpredictable and machine breakdowns occur randomly. The paper presents a Q-factor algorithm that optimizes scheduling decisions dynamically. The authors employ a variable neighbourhood search to explore the solution space and identify the most effective scheduling method. By using RL, the authors determine the optimal parameters for rescheduling processes in response to changes in the environment. This approach is designed to address real-world challenges in manufacturing. The results demonstrate the significance of their method for a dynamic job shop environment, as the optimal strategies are also updated dynamically.

By employing Google DeepMind's DQN agent algorithm present a successful application of RL to production scheduling. Their framework consists of multiple DQN agents that cooperate and learn to achieve the defined objectives, resulting in self-organized, decentralized manufacturing systems that are capable of adapting to a dynamic manufacturing environment. After a short training phase the system presents scheduling solutions that are on par with solutions based on expert knowledge. Even though the approach cannot beat heuristics, this research represents a significant step towards the application of AI to real-world industrial processes and intelligent production systems.

Further developing both Gabel and Riedmiller's [4] and Lillicrap et al.'s approach [8], in 2020 Liu et al. [9] utilized an actor-critic DRL-architecture to approach the JSSP as a sequential decision-making problem. The model consists of an actor network and a critic network, including convolution layers and fully connected layers. The actor network learns how to act in a dynamic environment, while the critic network evaluates these actions. This approach is effective in managing unexpected events such as machine breakdowns, additional orders or material shortages that may interrupt the

production process. It offers more robust and efficient scheduling solutions while still producing comparative solutions for smaller benchmark problems, outperforming traditional dispatching rules and executing almost as fast as simple dispatching rules. With increasing sizes of the instances, the performance eventually declined.

Han and Yang [6] deal with increased complexities and uncertainties of JSSPs by proposing a duelling double DQN with priority replay. This DRL-framework combines the advantages of real-time response and flexibility of a deep convolutional neural network and RL to dynamically respond to complex and changing production environments. Scheduling is seen as a sequential decision-making problem, where the scheduling states at each time step are expressed as multi-channel images. The action space is a combination of easy to execute heuristic rules. The duelling double DQN is used to optimize learning through continuous interactions with the environment, resulting in a more accurate and stable learning performance in environments with high-dimensional action spaces and complex state evaluations. Experimental results show that this method achieves optimal solutions for small-scale problems and outperforms traditional heuristics for larger problems comparable to genetic algorithms but wasn't tested with a new dataset.

Zhang et al. [27] developed a Graph Neural Network (GNN) capable of solving problems regardless of their size. Like Han and Yang [6], they utilized a disjunctive graph to represent the state space of the JSSP. The GNN interprets the disjunctive graph representation, allowing the system to generalize solutions across different scales of problems. The conducted experiments reveal that this approach enables the agent to learn high-quality Priority Dispatching Rule (PDR)s from basic features, outperforming existing PDRs and performing well on much larger instances that were unseen in the training phase, although the generalized results fell short of being optimal.

Also, Park et al. [13] use the combination of RL and a GNN to solve the JSSP by formulating the scheduling process as a sequential decision-making problem. They employ a GNN for representation learning to encode the spatial structure of the JSSP into node features that are used to determine the optimal scheduling actions. The training of these components is conducted using PPO. Experiments demonstrate that the GNN approach can outperform traditional dispatching rules and other RL-based approaches on various benchmark JSSPs. Furthermore, the framework can learn transferable scheduling policies that apply to new JSSP scenarios (in terms of size and parameters) without additional training, highlighting its adaptability and efficiency.

An overview of how to model the JSSP as a sequential decision process and how a DRL architecture can be applied to the JSSP is given by Wang et al. [25]. Just like Park et al. [13] they employ a PPO algorithm to reduce the complexity. The performance of the proposed model is compared with heuristic rules and meta-heuristic algorithms. It is shown that this approach not only produces comparative results but is also able to realize adaptive scheduling and shows a certain generalization capability when faced with unseen situations.

The approach of Oren et al. [11] doesn't specifically tackle the JSSP itself, but more generally NP-hard COPs [5] that are found in job shop scheduling. The method is designed to work both online and offline, which is crucial for adapting to a dynamic production environment. A DQN is used offline to learn optimal policies through a simulation environment. The graph representation of the states enables the system to handle varying problem sizes and configurations. These policies are applied online to optimize the decisions in real time. The combined approach utilizes available time for deliberations, effectively reducing the gap to dedicated COP solvers.

Tassel et al. [22] propose a new DRL algorithm that is specifically tailored for job shop scheduling tasks, using recent advances of DRL to handle the growing complexity of JSSPs. The authors developed a compact state space representation, along with a simple dense reward function that is closely related to the sparse make-span minimization objective of COP methods. Benchmark instances provided by Taillard [21] show that their method finds solutions 11% better make-span than the best dispatching rule on Taillard's instances, 10% better than Han and Yang [6] and around 18% better than Zhang et al. [27].

Zhao and Zhang [29] investigate the application of DRL in dynamic job-shop production control. A dynamic job shop is a type of job shop where the scheduling environment is not static but changes over time. This dynamic nature can stem from several factors, including the arrival of new jobs, machine breakdowns, variable processing times, or changes in job priorities. As intelligent manufacturing becomes more and more important in industrial production systems, they address the lack of an evaluation mechanism that can accurately measure the control efficiencies of different scheduling plans. The authors create a multi-objective optimization model for a production control system, and subsequently introduce DRL to it. This is followed by a proposal of a dynamic job shop production control method that is also based on DRL and the explanation of the collaboration strategy for multiple subsystems. Experiments proved that their approach is effective.

A comprehensive literature review on the applications of DRL in production systems is presented by Panzer and Bender [12] in 2022. They discuss the challenges of modern production environments, such as the high level of complexity and the demand for high throughput, all while maintaining adaptability and robustness in case of variations in the process or unforeseen events. They highlight the increasing use of DRL to optimize production systems and the significant

contributions and developments in the field that are improving the efficiency and flexibility of production processes. 89% of the benchmarked implementations increase the scheduling performance, reached lower total tardiness, higher profits, or other problem-specific objectives. In the field of production scheduling, 67% of the reviewed papers applied value-based algorithms.

With the growing interest in using RL methods for production scheduling, it becomes increasingly challenging or even impossible to reproduce existing studies with the same degree of accuracy. To make the research more widely applicable and to exploit its strengths for industrial applications, Rinciog and Meyer [16] propose to standardize the approaches. They propose a framework for applying RL in this context by modelling production scheduling as a MDP. The standardization is done in three steps: The standardization of the description of production setups used in RL studies is based on an established nomenclature. This is followed by the classification of RL design decisions from existing publications. Finally, recommendations for a validation scheme that focuses on reproducibility and sufficient benchmarking are proposed.

A novel algorithm for improving the generalization capabilities and solution effectiveness of a DRL agent that solves JSSPs is proposed by Vivekanandan et al. [24]. The authors introduce a new method called Order Swapping Mechanism to achieve better generalized learning. By using a set of known benchmark instances [21] they compare their results with the work of other groups [6][27][22] that used the same benchmark instances and demonstrate that this approach outperforms previous methods. The results demonstrate that the agent does not outperform the approach of Tassel et al. [22], yet it does provide a size-dependent generalization. It outperforms the PDR based DRL approach of Zhang et al. [27] and performs similarly to other state-of-the-art DRL algorithms.

Serrano-Ruiz et al. [17] present a method for scheduling in a quasi-realistic job shop environment. They create a digital twin of the job shop model as a MDP and use DRL for optimization. Their approach uses a deterministic framework for formulation and implementation and is validated by comparison with known heuristic priority rules. Experiments show that the model not only captures the benefits of heuristic rules but also leads to a more balanced performance across various indicators, outperforming traditional heuristic methods.

The analyzed papers demonstrate that the JSSP can be effectively solved by using various DRL approaches. However, most of the papers are based on simplified models with little relevance to the reality of production. In this reality, a large number of factors play a role, the quantification of which is sometimes difficult and can have a significant influence on the effectiveness or success of the modelling. The decisive factors and their influence on production planning are described in more detail in Section 3.

3 Methodology

3.1 Necessity for an Extended Approach

Despite the existence of various RL approaches proposed by different researchers with the intention of solving generic JSSPs in industry-relevant problem sizes, it remains a challenge to translate the complexity of a real-world production environment into generic JSSPs. The following complexities are identified throughout this work:

1. Generic JSSPs typically involve machines that process various jobs without any specific machine setup. It is therefore necessary to reduce the time required for machine setup to a minimum, as this can otherwise result in a significant loss of production time.
2. Jobs are typically defined by a machine sequence and a fixed processing time. In the case of varying batch sizes, the processing times may be approximately linearly dependent on the batch size. Previous approaches did not consider varying batch sizes or processing times. With an enhanced generalization capability, contemporary DRL agents should be capable of accommodating varying processing times.
3. The field of intralogistics is not included in the scope of a generic JSSP. The transportation times between different machines or production facilities can be considerable, often taking several minutes, and therefore play an important role in the scheduling process.
4. In a real-world production environment, a variety of storage spaces can be found to buffer inconsistencies in production processes or to increase production flexibility. The dimensions of these buffering zones were not considered in previous approaches. An overfilled buffer zone may impede the entire production process and thus necessitates consideration in the scheduling process.
5. Deadlines are frequently absent from models, even though they are an essential component of ensuring the timely delivery of products.

These simplifications make the JSSP easier to model mathematically, but real-world manufacturing systems often involve complex scheduling challenges that are not captured by these simplifications. In order to create a training environment that closely resembles the complex processes of a real production environment, it is necessary to develop an extended approach with a higher level of information detail.

3.2 Research Hypothesis and Objective

Based on a comprehensive analysis of an existing furniture factory, a concept for the implementation of RL in production planning is proposed. In order to formulate a general concept that is valid for a broad industry, the following constraints are given:

- The production is set up as a Job Shop
- The machines produce products in batches
- The batches are manufactured on a recurring basis, but the production is frequently switched to enable the production of a wider range of products
- The overall range of products doesn't change drastically after training, as this would require a re-training of the agent. Minor changes like "color changes" would not disrupt the production process.

3.3 The Model

Generic JSSP-models are typically described by n jobs $J = \{J_1, J_2, \dots, J_n\}$, where each job has m operations O ($J_i = \{O_{i1}, O_{i2}, \dots, O_{im}\}$) to be processed on m machines $M = \{M_1, M_2, \dots, M_m\}$. Each operation has its designated processing machine and time d_{ij} . This model is extended by the introduction of the following elements:

Job Volumes The volumes of the jobs before and after each operation are quantified and mapped. With knowledge of the volumes of each job at any given point in the production process, the required storage spaces can be estimated with greater precision. This information is used to forecast and analyze the utilization of the buffers.

Buffers In a production system, storage areas are distributed across the shop floor and are used to hold materials, unfinished, or finished goods between different stages of production or between production and shipping. The primary purpose of these areas is to absorb variability in the production process, including fluctuations in demand, supply disruptions, machine breakdowns, or other unforeseen circumstances that may impact the production schedule. These storage areas are defined as buffers and are used to store the jobs before being processed at a machine. Each buffer is characterised by its capacity, which may be expressed in various units, including storage volume, pallet storage capacity, or any other meaningful unit. Buffers serve to absorb variability in the production process, enabling a more flexible production, preventing bottlenecks and thereby smoothing out the production flow.

Quantity Factor δ The processing time of each operation is subject to significant variation due to fluctuations in batch sizes, which are quantified using the quantity factor δ . The quantity factor is employed to determine the duration for which a machine is occupied in processing a component, particularly when batch sizes exhibit considerable variability. The total processing time of an operation is calculated as follows:

$$\text{total processing time of an operation} = \delta \cdot d_{ij} \quad \text{with } d_{ij} \text{ as the processing time of one single element of an operation} \quad (1)$$

This methodology enables the calculation of the processing time of a job on a machine based on the actual batch size of a job.

Transportation Times t In order to describe the intralogistic processes, transportation times are introduced. These figures represent the time required to transport a job from one machine to the next. In the majority of cases, the transportation times between two points are symmetrical, as the required transportation time is independent of the direction of travel. Table 1 illustrates an example of transportation times.

Machine Setup Times s A machine that is involved in the production of various jobs may require different setups in order to perform the specific operations. These setup times are considered, when the production requires a switch from one setup to another. Table 2 illustrates an example of symmetric setup times, where the individual setup times are

Table 1: Example for transportation times (mock-up data)

Transportation Time	to Machine 1	to Machine 2	to Machine 3
from Machine 1	0	10	15
from Machine 2	10	0	15
from Machine 3	15	15	0

Table 2: Example for symmetric machine setup times (mock-up data)

Machine 1	to neutral Setup	to Setup 1	to Setup 2	to Setup 3
from neutral Setup	0	4	4	4
from Setup 1	4	0	8	8
from Setup 2	4	8	0	8
from Setup 3	4	8	8	0

mostly independent of the previous setup. The neutral setup represents a neutral state of the machine, for example a saw without an installed saw blade.

In certain instances, the process of setting up a machine may take longer than the process of dismantling it. This phenomenon is indicated in Table 3, which represents an example of asymmetric setup times. The individual setup times are significantly influenced by the previous setup. The dismantling of a machine to a neutral setup is shorter than setting up any other setup. However, switching directly between setups reveals synergies that shorten the overall setup time.

Table 3: Example for asymmetric machine setup times (mock-up data)

Machine 2	to neutral Setup	to Setup 1	to Setup 2	to Setup 3
from neutral Setup	0	9	7	8
from Setup 1	5	0	10	13
from Setup 2	5	8	0	6
from Setup 3	5	8	7	0

In automated CNC machines, different jobs may have different machine setups, which can be achieved without the need for a physical setup change. Instead, a virtual setup change can be implemented with a simple mouse click. Consequently, the actual setup times do not need to be calculated. Table 4 illustrates an example of this phenomenon, demonstrating the transition between Setup 1 and Setup 2.

Overall Processing Time T_{total} With the introduction of the quantity factor δ , the transportation time t and the setup time s , the overall processing time of an operation can be calculated as follows:

$$T_{total} = \delta \cdot d + t + s \quad (2)$$

Deadlines The deadline is the latest point in time at which the production of a job must be completed. The deadline can refer to a shipping date or an upcoming maintenance schedule for a machine or a complete production line. Failure to meet the deadline can result in several consequences, making it an important factor to consider in production planning.

This extended approach allows for the acquisition of more detailed information. An example of the resulting operations is illustrated in Table 5. Taking these factors into account, a training environment can be created that bridges the gap between reality and model.

3.3.1 Environment Outline

The job shop environment can be implemented using toolkits such as OpenAI Gym and libraries such as Stable-Baselines3, which are specifically designed for developing and comparing reinforcement learning algorithms. In the training environment, an agent learns to solve the problem and optimize its policy parameters by interacting with the environment through actions.

Table 4: Example for asymmetric machine setup times, partially without physical setup change (mock-up data)

Machine 3	to neutral Setup	to Setup 1	to Setup 2	to Setup 3
from neutral Setup	0	4	7	5
from Setup 1	2	0	0	7
from Setup 2	2	0	0	6
from Setup 3	2	6	8	0

Table 5: List of operations of the exemplary JSSP (mock-up data)

Operation name	Machine	Machine Setup	Machining time	Quantity	Volume	Deadline
O_{11}	M1	s_1	0,04	400	30	-
O_{12}	M2	s_1	0,08	400	20	-
O_{13}	M3	s_1	0,06	400	15	120
O_{21}	M1	s_2	0,12	100	10	-
O_{22}	M3	s_2	0,14	100	8	-
O_{23}	M2	s_2	0,13	100	5	110
O_{31}	M1	s_3	0,06	400	20	-
O_{32}	M2	s_3	0,04	400	15	-
O_{33}	M3	s_3	0,06	400	10	100

The following example JSSP is considered:

Three jobs $J = \{J_1, J_2, J_3\}$ are to be produced on three machines $M = \{M_1, M_2, M_3\}$ with a machine sequence of

$$J_1 = \{M1, M2, M3\}$$

$$J_2 = \{M1, M3, M2\}$$

$$J_3 = \{M1, M2, M3\}$$

The batch sizes of the jobs J are

$$J_1 = 400 \text{ pcs}$$

$$J_2 = 100 \text{ pcs}$$

$$J_3 = 400 \text{ pcs}$$

The required machine setups for the operations are

$$O_{11} = s_1 \quad O_{12} = s_1 \quad O_{13} = s_1$$

$$O_{21} = s_2 \quad O_{22} = s_2 \quad O_{23} = s_2$$

$$O_{31} = s_3 \quad O_{32} = s_3 \quad O_{33} = s_3$$

and the machining times for a single one element of each job are

$$d_{11} = 0.04 \text{ min} \quad d_{12} = 0.08 \text{ min} \quad d_{13} = 0.0625 \text{ min}$$

$$d_{21} = 0.12 \text{ min} \quad d_{22} = 0.14 \text{ min} \quad d_{23} = 0.13 \text{ min}$$

$$d_{31} = 0.0625 \text{ min} \quad d_{32} = 0.04 \text{ min} \quad d_{33} = 0.0625 \text{ min}$$

With this information and the assumption that Job 3 is already located at Machine 1, the overall processing time of operation O_{31} can be calculated as follows:

$$\begin{aligned}
T_{total O_{31}} &= \delta \cdot d + t + s \\
&= 400 \cdot 0.0625 \text{ min} + 0 \text{ min} + 4 \text{ min} \\
&= 29 \text{ min}
\end{aligned} \tag{3}$$

The volumes of the jobs change with every operation

$$V_{11} = 30 \text{ m}^3 \quad V_{12} = 20 \text{ m}^3 \quad V_{13} = 15 \text{ m}^3$$

$$V_{21} = 10 \text{ m}^3 \quad V_{22} = 8 \text{ m}^3 \quad V_{23} = 5 \text{ m}^3$$

$$V_{31} = 20 \text{ m}^3 \quad V_{32} = 15 \text{ m}^3 \quad V_{33} = 10 \text{ m}^3$$

Table 5 shows a corresponding list of operations.

A buffer is located in front of every machine. In order to be processed on a machine, a job needs to be stored in the respective buffer first. The buffer capacities are

$$B1 = 60 \text{ m}^3$$

$$B2 = 43 \text{ m}^3$$

$$B3 = 30 \text{ m}^3$$

With this information, the production flow can be mapped as shown in Figure 2.

To ensure the agent’s training closely resembles reality, the training environment is designed with specific constraints:[15][2][24]

1. Machines are limited to processing one job at a time
2. The processing of an operation cannot be interrupted
3. There are no precedence constraints between operations of different jobs
4. Each job has a fixed machine sequence
5. Each operation requires a specific machine setup
6. Machines need to be set up accordingly before processing a job.

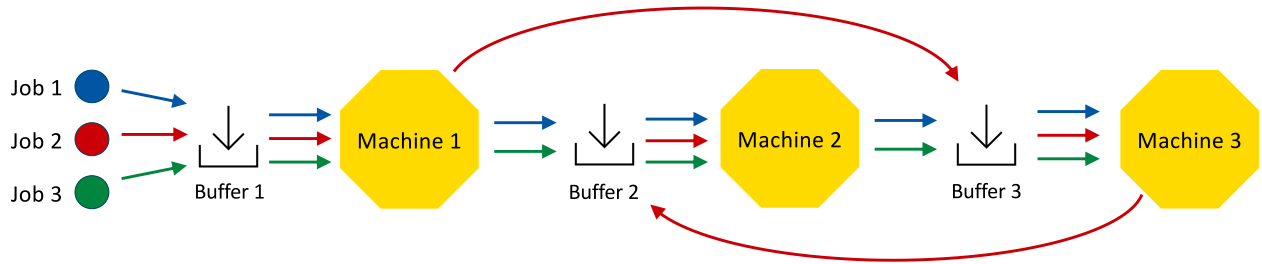


Figure 2: Production mapping of exemplary JSSP

Time Step Transition [24] In accordance with these constraints, the environment treats operations as atomic, implying that they cannot be interrupted, even if they consist of multiple individual elements (based on their batch size). Once a machine has been assigned to a job, it becomes occupied and is no longer available for assignment to other jobs. In the provided example, jobs J_1 , J_2 , and J_3 all have Machine 1 as their first machine in their machine sequence. Once the agent has selected Machine 1 for one of the assigned jobs, no further actions are necessary, and the agent can proceed to the next time step. In order to ensure optimal agent training, time steps and machine assignments are determined based on the eligibility of the operations. At each time step, the agent identifies all eligible operations O_{ij} , based on the predefined job order of the problem scenario. The eligibility criteria for an operation depend on the job order and the current status of the machines. This approach facilitates targeted learning by presenting the agent with relevant and actionable decision points, thereby simplifying the problem-solving process within the computational framework.

Due to its sequential nature, the process of assigning a machine to a job can be modelled as a MDP.

3.3.2 Action Space

The environment is controlled by a single discrete action space, through which the agent selects the appropriate jobs for processing on a given machine at each time step. The agent’s choices are limited to the subset of available jobs and machines, ensuring that the agent’s decisions are both relevant and feasible in the context of the current operational parameters and machine availability. The actions taken by the agent modify the environment and change its state.

3.3.3 Observations

The decision to assign a machine to a job is based on a set of observations that is provided to the agent. Each observation represents the current state of the environment. It is updated at each time step and holds the following information:

(1) machine info

A matrix of dimensions $3 \times m$, which contains information about the currently processed jobs, the progress of the operations, and the current machine setup. Given that m represents the number of machines in the environment, each column of the matrix represents a machine. The first line contains information about the currently processed jobs, the second line contains information about the remaining operation time and the third line contains information about the current machine setup.

$$(1)t_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Machine 1 Machine 2 Machine 3

(2) job info

A matrix of dimensions $2 \times n$, containing information about the current job volumes and the remaining time until the deadline is reached. With n representing the number of jobs in the environment, each column represents a job. The first line contains information about the current job volume, while the second line contains information about the remaining time until the deadline is reached.

$$(2)t_0 = \begin{bmatrix} 30 & 10 & 20 \\ 120 & 110 & 100 \end{bmatrix}$$

Job 1 Job2 Job 3

(3) buffer info

A b -dimensional vector represents the capacity status of the buffers in the environment with b representing the number of buffers in the environment. This vector can reflect the utilization of the buffers in units or in percent, depending on what is most appropriate for the model.

$$(3)t_0 = [\begin{matrix} 60 \\ 0 \\ 0 \end{matrix}]$$

Buffer 1 Buffer 2 Buffer 3

These three observations provide an overview of the status of the entire environment. Additional information, such as machine availability, can be extracted from these observations. If a machine has no job assigned to it, it is considered to be idling and available.

3.3.4 Transition Probability Function

Transition probabilities represent the likelihood of transitioning from one state to another after taking a given action. The transition probabilities are estimated based on the experiences of past actions and resulting observations in the training environment. They are then continuously adapted and honed in the training phase. This function is capable of capturing the dynamics of the system under different observations.

3.3.5 Reward

The reward function is employed to quantify the system performance and guide the decision-making process. It must be tailored closely to the goals of the specific industrial application, as it is highly sensitive. The reward corresponds to the scheduling goal and defines the specialization of the agent. In order to maintain a high level of learning efficiency, it is necessary for the agent to be rewarded in a densely manner. This implies that the agent is rewarded for actions that are specific to the JSSP, such as achieving a shorter overall processing time. Additionally, the agent may be rewarded for actions that are specific to its industrial application. One method of enhancing the learning process is to provide the agent with a negative reward for unwanted actions, such as overfilling a buffer or failing to meet a job's deadline. Identifying the optimal reward function is an iterative process that must be conducted on an individual basis.

Discount Factor γ As with the reward function, the discount factor must be set according to the specific application in question. The discount factor between 0 and 1 determines the relative importance of future rewards in the decision-making process. A discount factor of 0 indicates that the agent is short-sighted and only considers immediate rewards, while a discount factor close to 1 implies that the agent values future rewards to a similar extent as immediate rewards.

In order to identify the optimal policy, the application of PPO balances the trade-off between policy improvement and stability.

4 Integration in the Furniture Industry

In a general JSSP, each job is associated with a unique set of tasks that must be completed in a specific order. This concept can also be applied to the production of furniture, where articles are composed of several components that must be processed in a specific sequence on various machines. Consequently, each article corresponds to a group of jobs, with each component considered being a separate job. In order to determine this set of jobs, it is necessary to break down each article into its various components and determine their respective machining sequences. The identification of production information, such as the individual machining sequence, volume alterations, and processing duration for each component at every production step, is typically obtained from the so-called Bill of Materials (BOM), which is a commonly utilized document in the industry. The level of detail provided by the BOM determines the comprehensiveness of the production information. Obtaining such information is indispensable for the development of optimized production planning. Given the intricate nature of furniture production systems, the implementation of DRL-supported production planning necessitates a comprehensive analysis of the factory in question. Consequently, the presented solutions thereby may not be readily transferable to all furniture production facilities, as each facility will require a solution that is precisely tailored to the factory and its individual scheduling goals. This chapter presents an example of how a RL-agent can be employed for production scheduling in the furniture industry, where components are manufactured in batches in a job shop environment. For that purpose, the elements of the extended model that is presented in Chapter 3 may be represented as follows:

Jobs A piece of furniture is typically constructed from multiple components, which are then assembled or packaged at a designated station in order to create the final product. In the context of the job shop analogy, each component represents a distinct job with its own machine sequence and processing time for each operation. Therefore, the furniture articles are broken down into individual components. For each component, the processing sequence on the respective machines, the processing time, and the machine setups are mapped. If no further processing occurs within the factory, purchased parts may be excluded from this analysis. Each job is then defined by its machine sequence, machine setup, total processing time (cf. 3.3), deadline, and its volume, which undergoes change throughout the production process.

Job volumes It is common in furniture production, particularly for solid wood furniture, to undergo significant volume changes throughout the production process. The volume of the raw material at the beginning of the process may be up to five times greater than that of the end product. In order to calculate the space requirements of each job at any point during production, it is necessary to record the volume change in each process step.

Buffers Each buffer on the shop floor is assigned to a machine and located in front of it in the process flow. Buffers are treated similarly to machines, but they can store several jobs simultaneously and have no processing time. The capacity of these buffers is specified in terms of the number of storage spaces for pallets or by volume.

Machines Another aspect of the environment is the machinery used for production. Each machine is mapped in detail, including its features, processing times, possible setups, and setup times required for each operation.

Quantity Factor δ The quantity factor δ is employed to modify batch sizes, which have a significant impact on processing time and, consequently, the make span of a job. Training the agent with varying batch sizes enhances its generalization capability, enabling it to handle varying job sizes. It has been observed that the production of furniture does not always occur in consistent batch sizes. The quantity factor helps determining the duration of a machine's utilization in the processing of a component.

Transportation Times The transportation times between machines and buffers in the production are mapped. This includes a full analysis and documentation of the transportation times between the machines on the shop floor. With this information, the overall completion time of an order can be determined.

Deadlines Each job is described with a deadline that specifies the latest point in time by which its processing must be completed.

Environment The production environment is mirrored as closely as possible by the training environment, which comprises jobs, machines, and buffers. The agent is trained to set up machines, assign jobs to machines, and prevent buffers from overflowing, all while maintaining the correct machine order for each job and meeting the production deadlines.

Action Space The agent is controlling the environment in a single discrete action space. At each time step suitable machines and buffers are determined for the available jobs.

Observations The observations provided to the agent are (1) machine info, (2) job info, and (3) buffer info (cf. 3.3.3).


Reward The reward function guides the decision-making process and thus must be tailored to the specific optimization goal. It needs to include rewards for correctly setting up a machine for an according job, assigning machines to jobs in the correct machine sequence of a job and storing jobs in buffers before assigning them to a machine. Additionally, the reward function contains rewards for the specific optimization goals, such as low buffer levels, reducing make span times, or achieving other, industry-specific objectives.

4.1 Integration Strategy

The integration of an RL agent must be compatible with existing production and planning systems and be able to be seamlessly integrated without disrupting ongoing production. Once the training environment has been established and the agent has been trained in accordance with the desired optimization goals, it can then be utilized to support production planning. Two principal concepts may be distinguished with regard to the integration: episodic and continuous production planning. The decision to utilize either episodic or continuous planning depends on various factors, including the nature of the production process, the degree of variability in orders, and the scheduling system's required flexibility.

4.1.1 Episodic Planning

In companies with low levels of automation and networking, episodic planning represents an appropriate approach. It may be employed when orders are processed in batches or when the production line is configured to manufacture specific types of furniture for a defined period before transitioning to a different type. The integration of an episodic planning system is less complex than in a continuous planning system (cf. 4.1.2), as it does not require interfaces with existing production systems. The planning process is divided into distinct episodes, each with a clear beginning and end. These episodes may be defined as the production of a week or the production of an individual order from a specific client. Each episode consists of a limited number of steps and actions, making it easier to predict and evaluate outcomes. The orders are entered into a dashboard, which includes information on quantities and deadlines. Alternatively, a specific time period may be selected for scheduling. Figure 3 illustrates an exemplary dashboard, which displays orders, quantities, and deadlines. The system may offer the users a range of optimization goals to choose from, or alternatively, suggests different schedules based on various models. This enables production planners to respond effectively to the dynamic changes of the production environment.

Article Number	Description	Quantity	Deadline	...
01 234	Article 01	3000	May 15	...
56789	Article 02	750	May 15	...
98765	Article 03	1000	May 15	...
43210	Article 04	750	May 13	...
...
				

Optimise for:

☒ Speed
 ☐ Efficiency
 ☐ Empty Buffers

Start Scheduling

Figure 3: Exemplary layout of a production planning dashboard

Once the orders have been entered into the dashboard, the agent starts with the scheduling process. With the JSSP example provided in chapter 3.3.1, the initial observations at t_0 are as follows:

$$(1)t_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$(2)t_0 = \begin{bmatrix} 30 & 10 & 20 \\ 120 & 110 & 100 \end{bmatrix}$$

$$(3)t_0 = [60 \ 0 \ 0]$$

At the initial time step t_0 , all machines are in neutral setups, idling, and thus available for use. All three jobs require Machine 1 as their first machine in the sequence. The agent assigns Job 3 to Machine 1, sets up the machine for the job, and creates a new time step t_{29} , for when the processing of Job 3 will be finished (cf. Formula (3)). Machine 2 is set up to take over Job 3 at the new time step t_{29} , that will eventually be finished at time step t_{46} , as illustrated in Figure 4.

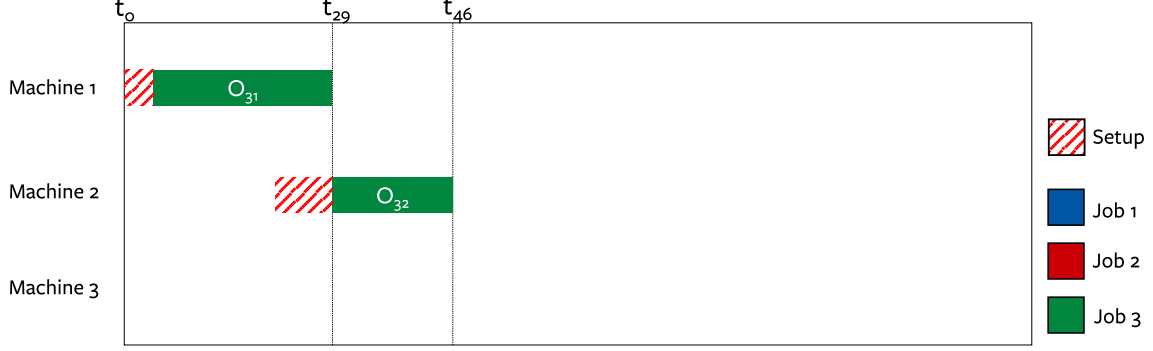


Figure 4: Scheduling with a trained agent at t_0

Since there are no other viable operation at t_0 , the agent skips to the next time step t_{29} , where it is provided with a new, updated set of observations:

$$(1)t_{29} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ s_3 & s_3 & 0 \end{bmatrix}$$

$$(2)t_{29} = \begin{bmatrix} 30 & 10 & 15 \\ 91 & 81 & 71 \end{bmatrix}$$

$$(3)t_{29} = [40 \ 15 \ 0]$$

Based on this new set of observations, the agent decides at t_{29} to assign Job 1 to Machine 1, set it up accordingly, and create a new time step t_{53} for the end of processing Job 1 on Machine 1. Furthermore, Machine 2 is assigned with Job 3, while Machine 3 is set up to process Job 3 after it is done on Machine 2 at t_{46} . This is shown in Figure 5.

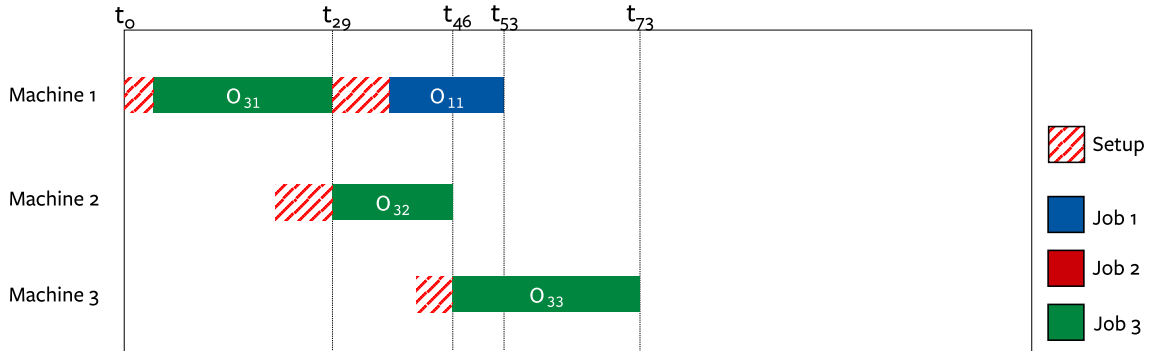


Figure 5: Scheduling with a trained agent at t_{29}

These time step transitions are repeated until no operations are left for processing and the production is completed. The finished schedule is shown in Figure 6. After completion, the system resets, allowing for evaluation and optimization of each episode independently.

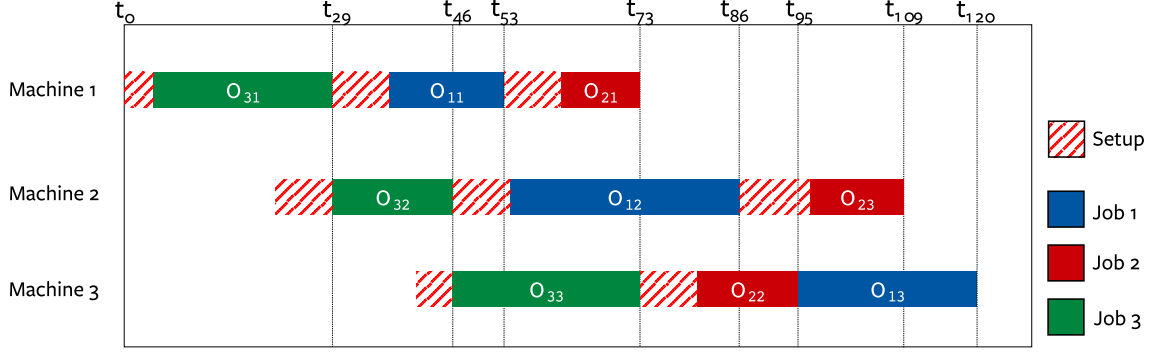


Figure 6: Completed scheduling

Limitations of Episodic Planning In a real-world production environment, a number of factors may influence the production planning process, including unforeseen events such as machine breakdowns, delivery delays, or changes in customer requirements. As the episodic planning process is conducted in advance, it is not possible to incorporate unforeseen events into the scheduling process. The introduction of a new product to the portfolio necessitates the re-training of the agent. Even an agent with high generalization capabilities is unable to predict the correct order of operations for the components of a new furniture article. A trained agent is constrained to the specific set of jobs for which it was trained.

4.1.2 Continuous Planning

In a continuous planning approach, the agent is fully integrated into the production system and plays an active role in the scheduling process. This approach is suitable for highly networked and automated plants where production is subject to significant variations in orders, requiring production schedules to be adjusted on the fly to accommodate new orders, changes in design, or material availability. This integration comes at the cost of a significantly higher level of complexity: The agent is situated between the Enterprise Resource Planning (ERP)-system and the Manufacturing Execution System (MES) (see Figure 7). It processes real-time production data from both systems and adjusts the

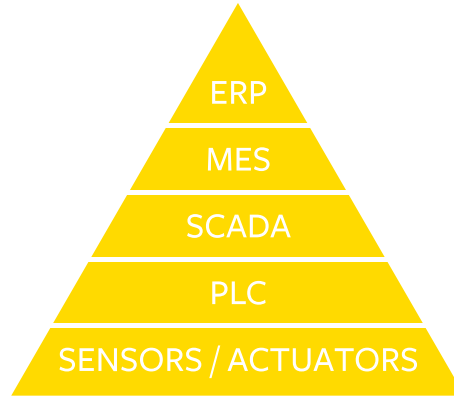


Figure 7: Automation Pyramid

production accordingly. The ERP system provides information regarding orders, material stocks, and delivery changes, while MES provides machine data such as processing times, operation progression, machine breakdowns, or transport times of transport systems. Significant alterations to this data will trigger an event, for which the agent will recalculate the scheduling, analogous to a time step in episodic planning. The scheduling results are written back into the systems, where production planners can supervise and adjust the proposed scheduling, if necessary. Unlike the episodic approach, which only provides various scheduling suggestions for the planned episodes, the production planning is actively influenced by the agent.

Limitations of Continuous Planning optimized production planning frequently requires the simultaneous consideration of multiple objectives, including a minimized lead time, maximized machine utilization and minimized inventory

costs. These objectives may, however, be in conflict with one another. Despite the agent's full integration into the production systems, the choice of the optimization goal remains the prerogative of an experienced production planner, who sets and adjusts the goals according to current needs. The interface communication between the agent, the ERP system and the MES requires a highly detailed customization, tailored to the specific needs and local conditions on site. As with the episodic approach, the continuous approach requires a retraining of the agent, when a new furniture article is entered into the system.

4.2 Implementation Strategy

The following procedure is proposed for the successful implementation of a DRL-supported production planning system:

1. **preparation and planning:** A detailed examination of the interfaces between the systems points out if specific adapters need to be developed. Furthermore, the data flow required to exchange relevant information between the agent and the systems must be defined. This may include production schedules, machine statuses, order details, and other system-specific information.
2. **goal definition:** Measurable results are achieved by the precise definition of the agent's goals and objectives. These goals may be the optimization of the throughput, the reduction of production costs or other specific goals.
3. **definition of the state space:** This includes defining the relevant variables and factors that influence the state of the production system as well as defining the data flow, i.e. how the required information is received, transmitted, and processed by the agent. It is essential that the observations include all information relevant for the agent's decision-making process.
4. **definition of the action space:** A precise definition of the actions that the agent can perform to influence the system's state. Examples of actions include the adjustment of machine setups, the assignment of machines and jobs or a simple no-op (no operation).
5. **definition of the rewards and penalties:** In RL, the agent is trained through the provision of feedback in the form of rewards and penalties. By clearly defining the reward function, the specialization of the agent can be specified. The reward definition should represent the goals defined at the beginning of this procedure. This may be an iterative process in the training phase. An example could be a reward for an increased throughput or a penalty for a buffer overflow.
6. **agent training:** In the training phase the agent interacts with the training environment that was previously defined in the procedure. The agent uses RL algorithms to learn to make decisions that optimize its received reward.
7. **test environment:** A test environment may be developed to simulate the production environment and test the agent under controlled conditions, thereby reducing the risk of production downtime. The development and optimization of the agent is conducted in cooperation with experienced production planners of the manufacturer. Conducting a pilot project in parallel with the existing systems allows for the evaluation of the agent's decisions and its impact on production processes.
8. **deployment and scaling:** Once the agent has been trained it can be deployed in the production system. This process is conducted in a systematic and sequential manner, starting with less critical processes in order to minimize potential risks. A feedback loop is established for the continuous evaluation of the agents' performance and adaptation to the changing environment, based on real production data.
9. **employee training and change management:** Staff is trained on the use of the new system, to check the proposed schedules, and override the decisions made by the agent. The implementation of change management strategies ensures the effective utilization of the new system by the employees and strengthens their acceptance of these new technologies.
10. **maintenance and continuous improvement:** Regular reviews and adjustments of the agent ensure an optimal performance and adaptation to the constantly changing production conditions.

5 Conclusion and Outlook

This paper presents a concept for DRL-supported production planning that can be adapted for optimized job shop scheduling. In Chapter 3, the elements of the DRL model are presented, including factors that aim to bridge the gap between real-world production environments and production models. The consideration of fluctuating job volumes

throughout the production process ensures more precise estimations of the required space conditions within the buffer zones across the shop floor. This prevents overfilled buffer areas, which clog the production flow as well as the accumulation of unnecessary materials and capital lockup. The introduction of the quantity factor δ determines the processing times of each operation, depending on its batch size, while transport times are added to represent the production duration more accurately. Jobs are specified with a deadline to simulate a delivery date, while machines are described with machine setups, that represent their current state in a more realistic manner. Furthermore, changes in the setup are also taken into account for the processing duration. The introduction of these framework conditions enables the generic JSSP model to be extended in a manner that more accurately reflects the high complexity of a real-world production environment. This extended approach allows the construction of a training environment, in which a RL agent can learn to optimally set up machines, assign jobs to machines, and move jobs between machines and buffers in accordance with specific optimization goals.

This approach is applied to the industrial furniture production in Chapter 4, tackling industry-specific challenges including the translation from a real-world furniture article to a job in the model. Two implementation concepts are presented for the integration of an RL agent into production planning systems: episodic and continuous planning systems. The production planning process for an episodic planning approach is illustrated using an example JSSP, highlighting the functionality of a trained agent at different time steps in the planning process. While episodic planning can be integrated as a low-tech, standalone solution, a continuous planning agent is fully integrated into the existing production systems. This enables real-time scheduling and allows for the prompt reaction to machine breakdowns as soon as the error message appears in the MES. In the event of predicted material shortages, as indicated by the data from the ERP system, the production of specific articles may be postponed in accordance with the anticipated shortage. However, the interface communication between the agent and the production system, as well as the implementation of the agent into the system, require a complex, customized solution.

A challenge remains in precisely defining a reward function that considers the proposed elements and the industry specific optimization goals. It remains unclear how to best prevent buffer overfill, particularly given that buffer levels are only checked at the time steps in episodic planning, rather than in between time steps. It remains to be investigated whether an overfilled buffer causes a deadlock of the system when a job cannot be moved to the next buffer because it is full. The consideration of the batch size for the processing duration with the introduced quantity factor δ can be extended to the calculation of the required transportation times. Larger batch sizes may include more pallets to be moved from one place to another, which would result in longer transportation times. A further challenge arises when the production system is not organised as a generic job shop but as a dynamic job shop: When the same furniture components can be produced by different machines on several routes and thus alternative routes through production are possible. This significantly increases the level of complexity, as the number of possible combinations grows at an even faster rate than in a generic job shop. The following step of this process is the realisation of the concept described above in order to examine, how an agent would deal with the increased level of complexity and increasing problem sizes. It is also important to note that the scheduling agents described above have been designed to support human production planners, rather than creating artificial copies of them.[7]

Abbreviations

AI	Artificial Intelligence
BOM	Bill of Materials
COP	Combinatorial Optimization Problem
DQN	Deep Q-Networks
DRL	Deep Reinforcement Learning
ERP	Enterprise Resource Planning
FJSP	Flexible Job Shop Scheduling Problem
GNN	Graph Neural Network
JSSP	Job Shop Scheduling Program
MDP	Markov Decision Process
MES	Manufacturing Execution System
PDR	Priority Dispatching Rule
PPO	Proximal Policy optimization
RL	Reinforcement Learning