Proceedings of the 2025 Winter Simulation Conference
E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.

REINFORCEMENT LEARNING IN PRODUCTION PLANNING AND CONTROL: A REVIEW ON STATE, ACTION AND REWARD DESIGN IN ORDER RELEASE AND PRODUCTION SCHEDULING

Patrick Farwick,[1] and Christian Schwede[1,2]

[1]University of Applied Science and Arts, Bielefeld, NRW, Germany
[2]Fraunhofer Institute of Software and Systems Engineering, Dortmund, Germany

ABSTRACT

Production Planning and Control (PPC) faces increasing complexity due to volatile demand, high product variety, and dynamic shop floor conditions. Reinforcement Learning (RL) offers adaptive decision-making capabilities to address these challenges. RL often relies on simulation environments for the intensive training, allowing for short run times during execution. This paper reviews existing literature to examine how RL agents are modeled in terms of state space, action space, and reward function, focusing on order release and related production scheduling tasks. The findings reveal considerable variation in modeling approaches and a lack of theoretical guidance, particularly in reward design and feature selection.

1 INTRODUCTION

The increasing integration of global markets and accelerating economic processes have intensified competitive pressures on manufacturing companies. These manifest in fluctuating demand, high product variety, and persistent cost constraints (Bauernhansl, 2023). Consequently, manufacturers are confronted with rising expectations regarding flexibility, responsiveness, and delivery performance. All these factors are directly impacted by the effectiveness of Production Planning and Control (PPC) (Schuh and Stich, 2012). Within PPC, order release represents a critical control lever, as it regulates the flow of jobs into the production system and thereby influences key performance indicators such as throughput, lead time, and work-in-progress (WIP) levels (Bauernhansl, 2020; Lödding, 2016).

Traditional order release mechanisms, often rule-based and designed for static environments, struggle to cope with the increasing volatility, customization, and reduced time-to-market requirements in modern manufacturing. While traditional PPC approaches have been extensively studied and successfully applied in many settings, their static and predefined nature offers limited flexibility. (Bauernhansl, 2020; Wiendahl et al., 2015) In particular, they fall short in adapting to real-time data and dynamic shop floor conditions, leaving potential for improvement in responsiveness and decision quality. (Lödding, 2016) Recent advances in Reinforcement Learning (RL) offer promising capabilities to address these challenges. RL algorithms are well-suited for environments characterized by sequential decisions, delayed effects, and non-linear dynamics, all of which are typical in production settings. These developments enable autonomous agents to learn context-dependent control policies that dynamically balance trade-offs between competing objectives such as throughput, lead time, and resource utilization. This makes RL a compelling approach for rethinking traditional order release strategies in modern PPC systems. (Steinbacher et al., 2024) In this context, simulation environments and digital twins play a crucial role. They enable RL agents to learn and test control policies in dynamic and complex production systems without risking real-world setbacks. By simulating alternative control strategies in a virtual simulation, decision-making quality, and adaptability can be investigated. While previous literature, such as (Panzer and Bender, 2021), have provided structured overviews of DRL in PPC systems, this work primarily focuses on algorithmic classification and application areas. Yet even though there already has been some work on RL for order release, there is no systematic

review on the field. This paper presents a review that is grounded in the observation that the effectiveness of RL in production and environments depends not only on the choice of algorithm but fundamentally on the quality of the underlying design elements: the state space, the action space, and the reward structure. While significant attention in recent literature is directed toward algorithmic improvements, these can unfold their full potential only if the basic elements are well modeled. In this regard, modeling state, action, and reward can be seen as foundational to successful learning. (Altenmüller et al., 2020)

This review therefore aims to shift attention back to these core modeling components, which often receive insufficient attention despite their critical role in shaping learning outcomes. Addressing the intersection of order release and RL within PPC, it highlights the limitations of traditional release strategies, the challenges posed by non-linear production dynamics, and discusses how RL can offer adaptive, performance-oriented control policies (Steinbacher et al., 2024). To the best of the authors' knowledge, no dedicated literature review has been conducted so far. The objective is to synthesize current research, identify conceptual and methodological gaps, and outline directions for future work in this emerging field.

2 RELATED WORK

This section outlines the relevance of order release within PPC, introduces RL as a decision-making tool for production control and motivates its application based on recent methodological advances.

2.1 Production System Characteristics

Manufacturing companies operate under diverse conditions and requirements, leading to the use of different production system designs. Each system reflects specific operational and market constraints optimized for different strategies: flexibility, efficiency or adaptability. The choice of production type has a direct impact on planning complexity, scheduling requirements, and order release strategies (Schuh and Stich, 2012). Flow Shops are production systems where all jobs follow the same predefined sequence of operations through a series of workstations. They are typically designed for higher production volumes with standardized workflows, leading to more predictable processing times and efficient material flow (Bauernhansl, 2020, 2023; Lödding, 2016; Schuh and Stich, 2012; Wiendahl et al., 2015). Job Shops are production systems where products follow unique and variable routing through different workstations, typically characterized by high flexibility and low production volumes. Machines and workstations are grouped by function, allowing custom manufacturing but leading to complex scheduling (Bauernhansl, 2020, 2023; Lödding, 2016; Schuh and Stich, 2012; Wiendahl et al., 2015). Matrix-structured Production Systems (MSPS) are hybrid manufacturing approaches that combine elements of both job shops and flow shops, allowing for flexible routing of jobs while maintaining structured workflows. Workstations are dynamically assigned based on real-time production requirements, enabling adaptive manufacturing in response to changing demand and resource availability (Greschke, 2020; Kirchberger et al., 2022; Renna, 2023). Another distinction in production lies in how customer demand is handled. In make-to-stock (MTS) systems, products are manufactured based on demand forecasts and held in inventory before customer orders are received. This approach is typical for mass production environments where short delivery times and cost efficiency are prioritized. In contrast, make-to-order (MTO) systems initiate production only after a customer order is placed. While this results in longer lead times, it enables higher product customization and reduces inventory costs. As product variety increases and demand becomes more volatile, MTO systems are gaining importance in modern manufacturing environments. (Bauernhansl, 2020; Lödding, 2016; Schuh and Stich, 2012; Wiendahl et al., 2015)

2.2 Production Planning and Control

PPC systems are critical for orchestrating the movement of materials and coordinating manufacturing processes. Their primary role is to balance production efficiency with demand fluctuations, ensuring that resources are utilized effectively while maintaining reliable and predictable lead times. These systems are commonly structured hierarchically. At the planning level, production activities are organized, defining

broad targets and constraints. At the scheduling level, operational execution is managed in detail. Bridging these levels, the order release process dictates the timing and selection of work orders entering production. Order release mechanisms, governed by predefined strategies or dynamic policies, regulate WIP levels, and ensure that production flow aligns with performance objectives such as minimizing delays or maximizing throughput. (Gómez Paredes et al., 2020; Hofmann, 2021; Lödding, 2016; Thürer et al., 2015; Wiendahl et al., 2015)

During production control, orders must be scheduled onto available machines in accordance with capacity constraints, processing requirements, and delivery deadlines. This task is also referred to as production scheduling and determines the sequence of operations on the shop floor. Key performance criteria that should be optimized in production systems include lead time, adherence to delivery dates, and capacity utilization. PPC contributes to achieving these objectives through its functions order release and production scheduling. (Bauernhansl, 2020; Gómez Paredes et al., 2020; Hofmann, 2021; Lödding, 2016; Thürer et al., 2015; Wiendahl et al., 2015) In this context, lead time is the time from customer order placement to delivery. Adherence to delivery dates is the extent to which orders are completed and delivered on their promised due dates. Capacity utilization is the degree to which available production resources are used relative to their maximum capacity (Lödding, 2016).

2.3 Traditional Order Release Methods

The most basic form of order release is the immediate release, in which jobs are introduced into the production system without any form of restriction or optimization. While this push-based strategy is straightforward to implement and requires minimal coordination effort, it may lead to excessive WIP and poor system performance. (Lödding, 2016; Thürer et al., 2015)

The inventory-regulating order release is used primarily in MTS systems, where production is triggered when inventory levels fall below predefined thresholds. However, this approach is not applicable in MTO environments, where production only begins after a specific customer order is received. (Lödding, 2016) Traditional order release methods vary according to production objectives and system constraints. A common approach is time-scheduled order release, where release is based on predefined dates derived from backward planning. Backward Infinite Loading (BIL) calculates the release date from the due date using a fixed planned lead time plus a time buffer, resulting in the latest possible release that still meets the delivery deadline. Orders are typically released periodically, such as at the beginning of a shift or workday. This approach primarily aims to ensure adherence to delivery dates. (Liu et al., 2023; Lödding, 2016)

In contrast, load-limited order release, also referred to as Workload Control (WLC), adjusts the release of orders based on actual shop floor capacity and current workload. The goal is to stabilize the production flow and reduce lead times by preventing system overloads. A well-known implementation of this principle is Constant Work-in-Progress (ConWIP), a pull-based mechanism that maintains a constant WIP level by releasing a new job only after another order has been completed. In this context, order release is event-driven and continuous rather than periodic. The WIP level can be defined in terms of the number of jobs or the total lead time within the system. (Haeussler et al., 2022; Hendry et al., 2012; Lödding, 2016; Thürer et al., 2015)

These systems introduce the concept of an order pool, in which unreleased jobs wait until they are released. When working with ConWIP, the order pool is sorted by predefined dispatching or priority rule: When a job is completed, the next job is selected from the pool. (Gómez Paredes et al., 2020; Haeussler et al., 2022; Thürer et al., 2015) Common rules are:

- FIFO (First In, First Out): Jobs are processed in the order they arrive.
- EDD (Earliest Due Date): Jobs with the earliest due dates are prioritized.
- SPT (Shortest Processing Time): Jobs with the shortest processing times are released first.
- LPT (Longest Processing Time): Jobs with the longest processing times are prioritized.

These traditional approaches are widely used in practice and form the basis for many rule-based production control systems. (Bauernhansl, 2020; Gómez Paredes et al., 2020; Liu et al., 2023; Lödding, 2016; Thürer et al., 2015; Wiendahl et al., 2015)

2.4 Advanced Order Release Methods

Beyond traditional rule-based approaches, various advanced methods have been developed to improve order release decisions by incorporating optimization and learning techniques. One such class includes metaheuristic approaches like Genetic Algorithms (GA), which evolve sequences of order releases over multiple generations using operators such as crossover and mutation. These methods aim to explore large solution spaces and can approximate near-optimal policies for complex scheduling problems. Another common approach is Integer Programming (IP) or Mixed-Integer Linear Programming (MILP), where order release and scheduling decisions are formulated as optimization problems subject to constraints on capacity, due dates, and resource availability. Similarly, Constraint Programming (CP) offers a framework for explicitly modeling logical and temporal constraints, making it well-suited for highly complex and combinatorial scheduling environments. (Cunha et al., 2020; Hofmann, 2021; Steinbacher et al., 2023)

Most traditional and advanced order release methods face several limitations in dynamic and complex production environments. Key performance metrics such as throughput, lead time, and tardiness exhibit non-linear behavior in response to changes in parameters like WIP levels, machine capacity, or order sequencing. However, many conventional approaches rely on linear assumptions that fail to capture these dynamics particularly under high load or system congestion. (Hofmann, 2021) These limitations highlight the need for more adaptive and data-driven methods such as RL. This method has recently emerged as a promising alternative for deriving adaptive order release policies. Through interaction with a simulated or real environment, RL agents learn control strategies via trial and error, guided by performance-based rewards. These approaches are particularly relevant for objectives such as reducing lead times and maximizing machine utilization in dynamic and uncertain production settings. (Cunha et al., 2020)

2.5 Reinforcement Learning

Reinforcement Learning (RL) is a subfield of machine learning concerned with learning optimal decision-making policies through interaction with the (often simulated) environment. In contrast to supervised learning, where labeled input-output pairs guide the learning process or unsupervised learning, which identifies patterns without explicit feedback, RL relies on evaluative feedback in the form of rewards to guide behavior over time. In this regard, RL gained significant attention due to its ability to handle sequential decision-making problems, where actions influence not just immediate rewards, but also long-term outcomes. (Russell and Norvig, 2021; Sutton and Barto, 2018)

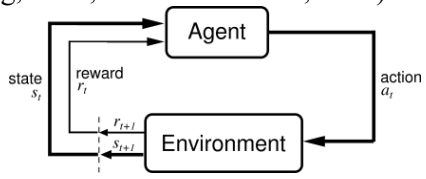

Figure 1: The agent-environment interaction in reinforcement learning. (Sutton and Barto, 2018)

In the RL setting, an agent interacts with its environment by observing the current state, selecting an action, and receiving a reward based on the resulting outcome. Over repeated interactions, the agent learns a policy that aims to maximize the cumulative reward. At its core, RL is grounded in the Markov Decision Process (MDP) framework, which provides a mathematical foundation for modeling decision-making in environments with stochastic transitions. A MDP consists of a set of states that represent the environment, a set of actions available to the agent, and transition probabilities that define the likelihood of moving from one state to another based on the chosen action. The reward function evaluates the effectiveness of an action, guiding the learning process toward favorable long-term outcomes. These components form the foundation for learning an optimal policy in dynamic environments. (Russell and Norvig, 2021; Sutton and Barto, 2018)

RL methods are commonly classified as model-free or model-based, and as value-based or policy-based. Model-free approaches learn directly from interaction, while model-based methods incorporate

predictions about environmental dynamics. Value-based algorithms estimate expected returns of actions in a state to derive policies, whereas policy-based methods optimize policies directly. Among the most widely used algorithms is Q-learning, which estimates the value of state-action pairs. Its deep learning variant, Deep Q-Learning (DQL), uses neural networks to approximate the Q-function, enabling it to scale to larger or continuous state spaces. Further advancements include Double DQN (reducing overestimation bias), Dueling DQL (separating value and policy learning), and Actor-Critic methods (combining value and policy learning). In complex environments involving multiple decision-makers or decentralized control, multi-agent Reinforcement Learning (MARL) has gained increasing attention, enabling collaborative or competitive learning among several agents. (Sutton and Barto, 2018)

3 LITERATURE RESEARCH

This literature review investigates how state, action, and reward spaces are modeled in RL applications for order release within PPC. Initial attempts to restrict the review to RL-based order release alone revealed only a limited body of work. As it became evident that only a limited number of studies explicitly address this topic (Janke et al.; Samsonov et al., 2021; Schneckenreither et al., 2022; Schneckenreither and Haeussler, 2019; Schuh et al., 2022; Schuh et al., 2023), the scope was extended to include RL approaches to production scheduling, particularly when they feature job queues or order pools, which are conceptually and structurally similar to order release systems (Altenmüller et al., 2020; Chang et al., 2022; Da Col and Teppan, 2022; Dittrich and Fohlmeister, 2020; Gui et al., 2023; Kardos et al., 2021; Lei et al., 2022; Liu et al., 2020; Ragazzini et al., 2021; Samsonov et al., 2021; Song et al., 2023; Steinbacher et al., 2024; Wang and Liao, 2024; Zhang et al., 2022). These settings allow comparable insights into the design of agentic modeling, especially in contexts involving order selection or dispatching. The collection focused on studies that explicitly model an order pool, a queue, or a job list from which selections are made. Included papers describe the underlying state-action-reward structure of RL implementations with a preference for flexible production systems like job shops or MSPS. Studies not detailing the state, action, and reward design used were excluded. Simulation tools and experimental setups were only considered insofar as they influenced modeling choices for agent design. In total, 20 relevant papers were identified, among which 18 were addressing job shop setups, one additionally addressed a flow shop setup, and only one paper focused on a MSPS.

3.1 State Space Representation

The reviewed literature shows a broad variety of approaches to modeling the state space in RL applications for PPC. While certain elements such as machine status and order-related information are widely included, their level of granularity, representation, and encoding differ significantly across studies, reflecting varying assumptions about complexity, generalizability, and problem objective.

A central component in nearly all papers is the inclusion of machine status information, which provides the agent with insight into resource availability, workload distribution or idle times. Some studies incorporate detailed, machine-specific features such as machine occupancy (availability), remaining processing time, the processing time of queued jobs or the queue length at individual machines (Altenmüller et al., 2020; Janke et al.; Lei et al., 2022; Liu et al., 2020; Schuh et al., 2022; Schuh et al., 2023; Song et al., 2023; Stricker et al., 2018; Zhang et al., 2022). Others like (Chang et al., 2022; Gui et al., 2023; Heger and Voss, 2020; Samsonov et al., 2021; Schneckenreither et al., 2022; Schneckenreither and Haeussler, 2019; Stricker et al., 2018; Wang and Liao, 2024) apply a more aggregated perspective using system-level indicators such as the average machine utilization rate, estimated or actual earliness and tardiness rates or mean workload across machines. These aggregated representations emphasize performance-relevant system dynamics rather than individual machine status. This diversity is also reflected in the overall granularity of state information, ranging from as few as four (Chang et al., 2022) up to 210 (Altenmüller et al., 2020) input features. Given this variation, it is reasonable to discuss whether more detailed state vectors necessarily lead to better learning outcomes. It seems plausible that a more comprehensive state

representation enables better-informed decisions and supports more effective learning. However, it could also be argued that excessive detail introduces noise or redundancy, potentially obscuring relevant patterns. Correlated or uninformative features may cause the agent to overfit to illegitimate relationships, thereby reducing policy robustness. In such cases, the model may attribute importance to features that are statistically associated with outcomes but lack a causal connection to the underlying system dynamic. (Chang et al., 2022) In (Heger and Voss, 2020; Zhang et al., 2022) machine information across similar resource types is combined to reduce dimensionality. Grouping machines by functional equivalence is frequently used to strike a balance between complexity and expressiveness, as it reduces the number of state variables without disregarding relevant structural characteristics.

In addition to machine states, order-related features are a second essential category. These include due dates, processing times, routing information, and product types. In systems operating under WLC, the order pool is often explicitly modeled, since the agent is typically required to select the next job to release. Higher product variety leads to a larger number of possible job configurations, which directly increases the dimensionality of the state space. Consequently, the overall size of the state vector is closely linked to both the structure of the production system (e.g., number of machines) and the diversity of the product catalogue. (Altenmüller et al., 2020; Heger and Voss, 2020; Janke et al.; Kardos et al., 2021; Lei et al., 2022; Liu et al., 2020; Schuh et al., 2023; Song et al., 2023; Steinbacher et al., 2024; Stricker et al., 2018)

Another point of divergence is the encoding of state features, which varies depending on algorithmic requirements and the nature of the input data. When using tabular Q-learning, discretization of continuous variables is necessary to create a finite state space. This can be achieved via binning but may reduce precision and lead to information loss. In contrast deep RL methods can retain continuous variables, which allows for more nuanced learning but increases model complexity. (Zhang et al., 2022) Categorical features are typically represented using one-hot encoding. For example, in WLC environments, the currently idle machine waiting for a job can be encoded as a one-hot vector or product types may be represented similarly (Altenmüller et al., 2020; Kardos et al., 2021; Steinbacher et al., 2024). Binary encodings are often used to indicate the status of machines (e.g., idle or busy), while continuous values may include system-level KPIs such as the ratio of late jobs, average tardiness, or the proportion of busy machines. (Altenmüller et al., 2020; Liu et al., 2020; Song et al., 2023; Zhang et al., 2022)

Some studies also incorporate knowledge used in the reward calculation directly into the state representation. This design choice has been made by several authors and could be intended to improve the learnability of the state-action-reward relationship. By embedding performance-relevant variables into the state (e.g., earliness/ tardiness ratios or mean WIP), the agent might associate specific states and actions with observed rewards more easily. This alignment may support faster convergence and more stable policy development, especially when rewards are sparse or delayed. (Chang et al., 2022; Heger and Voss, 2020; Schneckenreither et al., 2021; Schneckenreither et al., 2022; Schuh et al., 2023)

Despite these practical insights, there is no consistent theoretical framework guiding the design of state representations in the reviewed literature. Decisions regarding the number, type, and encoding of features are often problem-specific, heuristic or chosen by experts. As a result, a systematic understanding on how the state space representation influences the learning performance remains underdeveloped. Future research could contribute by investigating design principles for state representations, including dimensionality reduction techniques, relevance assessment of features, and the impact of encoding choices on policy generalization.

3.2 Action Space Representation

The way the action space is defined in RL applications for PPC varies considerably across studies and is closely linked to the underlying problem formulation and control objective. Fundamentally, the action space specifies the set of decisions available to the agent at each decision point and thus determines the expressiveness and feasibility of the learned policy. A central requirement for many implementations is that the action space must remain consistent in size across all decisions which is typically achieved by fixing

the number of possible actions either by limiting the size of the order pool or by abstracting actions into higher-level strategies. (Altenmüller et al., 2020)

A key aspect in RL-based order release and scheduling is the agent triggering mechanism, which defines when a decision is made. Two dominant patterns can be observed in the literature: episodic decision-making, where actions are taken at fixed intervals (e.g., at the beginning of a shift or planning period) (Samsonov et al., 2021; Schneckenreither et al., 2022; Schneckenreither and Haeussler, 2019) and event-driven (continuous) triggering, where the agent is activated in response to specific events such as a machine becoming idle or a job being completed (Altenmüller et al., 2020; Chang et al., 2022; Dittrich and Fohlmeister, 2020; Gui et al., 2023; Heger and Voss, 2020; Janke et al.; Kardos et al., 2021; Lei et al., 2022; Liu et al., 2020; Ragazzini et al., 2021; Schuh et al., 2022; Schuh et al., 2023; Song et al., 2023; Steinbacher et al., 2024; Stricker et al., 2018; Wang and Liao, 2024; Zhang et al., 2022). The choice of timing has implications for both system responsiveness and computational effort.

Several works include specialized actions to increase flexibility and maintain feasibility. A commonly used addition is the no-op action, which allows the agent to explicitly decide not to act, as is especially useful in scenarios where premature order release may lead to congestion (Altenmüller et al., 2020; Janke et al.; Ragazzini et al., 2021; Samsonov et al., 2021; Schneckenreither et al., 2022; Schneckenreither and Haeussler, 2019; Schuh et al., 2023; Stricker et al., 2018; Zhang et al., 2022). Another important aspect is the handling of invalid or infeasible actions that would violate constraints. These can be releasing an order that cannot be processed on the selected machine or choosing an index in the order pool that does not correspond to a valid job. These are typically penalized with negative rewards and re-triggering of the decision process, enforcing adherence to feasibility constraints during training and execution. (Altenmüller et al., 2020; Zhang et al., 2022)

The actual form of the action depends on the specific objective and structure of the control problem. Across the literature, several distinct modeling approaches can be identified: One intuitive strategy is to model the agent, so it is selecting the next order to release from a sorted order pool. Here the agent chooses from a fixed-length list of candidate jobs which is the common method in WLC settings. A challenge is that the number of actual orders in the pool varies dynamically depending on order reception. When the order reception is higher than the slots in the pool a pre-sorting must be done, for example by due date. When too few orders arrive, this approach requires careful handling of index validity and often includes placeholder actions to pad the action vector. The action space is modeled such that it chooses an index which is later mapped to the order pool. (Altenmüller et al., 2020; Janke et al.; Schuh et al., 2022; Schuh et al., 2023)

Another straightforward alternative is to model the decision as choosing between predefined dispatching rules (e.g., FIFO, EDD, SPT), rather than directly optimizing the underlying control problem. This strategy allows the system to dynamically adapt its job selection based on the current environment's state. Its advantage lies in the interpretable action space and reduced variance in learning. The next order is chosen from the queue according to the selected dispatching strategy. These studies (Chang et al., 2022; Heger and Voss, 2020; Liu et al., 2020; Wang and Liao, 2024) highlight the benefit of dynamically switching between dispatching strategies to respond to changing production situations and order placement, however focus on simplified MDP settings.

Another action modeling approach chosen by (Ragazzini et al., 2021) is to dynamically modify the WIP limit. The agent decides whether to increase, decrease or retain the current release threshold. This enables indirect control over the release flow and adapts system load over time. The actual order selection then follows a predefined dispatching rule (FIFO). While this method is easy to implement and interpretable, its decision scope is more limited.

A fundamentally different perspective is adopted in time-scheduled order release systems such as those based on the work of (Schneckenreither et al., 2022; Schneckenreither and Haeussler, 2019). Here, the agent does not directly decide which job to release but rather learns dynamic lead times or release offsets. The agents increase or decrease the lead time for an order by one time unit, which indirectly influences when the order is scheduled for release. The actual release is then triggered by a time-scheduled mechanism. The

release date is first calculated by subtracting the due date with the planned lead time and if relevant add a buffer (BIL). Once the release date approaches the current time the order is released in an episodic fashion (beginning of day or shift). Since the release logic is decoupled from the learning agent additional implementation is needed to manage the release mechanism. The validation/ the feasibility of calculated release dates as well as buffers needs to be done too. Invalid actions in this setting might include setting the release date too late to still meet the due date. (Schneckenreither et al., 2021; Schneckenreither et al., 2022; Schneckenreither and Haeussler, 2019) A variation of this approach done by (Samsonov et al., 2021) using predicted processing times which functions as a relative indicator. After the agent predicted the best relative duration for the next order, the duration is mapped to a specific order in the pool. The order which is closest to the agent's time is chosen. Although both (Samsonov et al., 2021) and (Schneckenreither et al., 2022; Schneckenreither and Haeussler, 2019) use a continuous action space, the main difference is that a fixed lead time/ processing time approach is used in this variation. This approach still showed promising results. When extending the order release problem to a comparable production scheduling context, RL can also be used to assign incoming production orders directly to machines. In this case each machine maintains its own queue. In (Dittrich and Fohlmeister, 2020; Kardos et al., 2021; Steinbacher et al., 2024; Stricker et al., 2018; Zhang et al., 2022) the agent selects the next machine on which the job should be processed. A special case is presented in (Chang et al., 2022; Gui et al., 2023; Lei et al., 2022; Song et al., 2023), where a composite action is introduced. Here the agent makes two decisions simultaneously. When a machine becomes idle, it must decide both which jobs to process from the local queue and to which machine the currently finished production order should be assigned. These combined actions enable more flexible policies and eliminate the need for a decentralized multi-agent architecture. (Gui et al., 2023)

In summary, the reviewed literature reveals a range of action modeling strategies, each with different levels of abstraction, complexity, and responsiveness. The main differences lie in whether the agent selects concrete orders or abstract policies, whether control is exerted directly or via intermediate parameters (e.g., lead time or WIP limit) and the level of integration between order release and scheduling decisions. All approaches share the need for a well-structured and fixed-length action space to ensure compatibility with learning algorithms and enable scalable implementation. While the reviewed literature demonstrates that a variety of action space definitions can lead to effective RL policies, the choice of action representation should not be made arbitrarily. Instead, it should reflect the characteristics of the production environment, the control objectives and the constraints imposed by existing systems. One decision criterion is based on the already established control mechanisms. For systems using time-scheduled release logic (e.g., BIL), RL agents that adjust lead times or release offsets offer a seamless integration point without altering the release mechanism itself. In contrast, systems organized around WLC, and order pools are well-suited for RL agents that either select jobs directly or choose dispatching strategies, leveraging the existing logic for order sorting and prioritization.

A robust and simple implementation can be provided with selecting among dispatching rules. This approach also improves transparency and interpretability, as the selected strategies correspond to well-understood heuristics. Being in the context of systems with critical bottlenecks, complex shop layouts (complex job shop or MSPS), or which are disruption-prone (e.g., due to frequent machine failures, missing materials, or last-minute order changes) may justify using more sophisticated adaptive action structures. In any case the action space representation should align with the existing infrastructure and be tailored to the environment at hand, striking a balance between expressiveness, learnability, and practical integration.

3.3 Reward Modeling

The design of the reward function is a central component in RL and directly influences the effectiveness, convergence, and interpretability of the learned policy. In the context of PPC, the reward must reflect operational objectives such as throughput maximization, lead time reduction, adherence to delivery dates, WIP stabilization, or cost minimization. Across the reviewed literature, a wide range of reward formulations were observed, often tailored to the structure of the production environment and the nature of the control task.

A basic distinction can be made between sparse (Dittrich and Fohlmeister, 2020; Samsonov et al., 2021; Schneckenreither and Haeussler, 2019) and dense (Altenmüller et al., 2020; Chang et al., 2022; Gui et al., 2023; Heger and Voss, 2020; Janke et al.; Kardos et al., 2021; Lei et al., 2022; Liu et al., 2020; Ragazzini et al., 2021; Schneckenreither et al., 2022; Schuh et al., 2022; Schuh et al., 2023; Song et al., 2023; Steinbacher et al., 2023; Steinbacher et al., 2024; Stricker et al., 2018; Wang and Liao, 2024; Zhang et al., 2022) reward structures. Sparse rewards, typically assigned only at the end of an episode, are conceptually straightforward and align well with long-term performance metrics (e.g., total tardiness at the end of a shift). Their simplicity, however, presents challenges for learning: the agent receives limited feedback, which can slow down convergence and complicate credit assignment. Nevertheless, sparse rewards may reduce the risk of overfitting to short-term objectives and are suitable for episodic decision settings. In contrast, dense rewards are provided at each decision step. This allows for immediate feedback and accelerates the learning process, particularly in complex environments where exploration is costly. However, dense rewards may introduce unintended incentives: for instance, optimizing short-term machine utilization could conflict with long-term delivery performance. Thus, the appropriateness of sparse vs. dense rewards often depends on the optimization. In (Altenmüller et al., 2020) it is suggested that a transition from dense to sparse rewards may yield robust policies. The idea of having dense rewards in early phases to guide learning and sparse in later stages to align with global objectives remains an open area for research.

Additionally, difference rewards, which evaluate the change in system performance between consecutive steps, rather than the absolute value, are sometimes used. These require less domain-specific tuning and help the agent identify the marginal impact of its actions, making them particularly useful when system behavior is highly dynamic or stochastic. (Chang et al., 2022; Gui et al., 2023; Heger and Voss, 2020; Lei et al., 2022; Song et al., 2023; Wang and Liao, 2024)

In multi-agent settings, a distinction is made between local (Steinbacher et al., 2024; Zhang et al., 2022) and global (Dittrich and Fohlmeister, 2020; Liu et al., 2020; Schneckenreither et al., 2022) rewards. While global rewards promote system-wide optimization, they often suffer from high variance and delayed attribution, making learning unstable. Local rewards provide the agent with feedback on its own surroundings and contribution. Although only a few of the reviewed studies employed MARL, the choice between global and local reward assignment remains an important design consideration in decentralized systems. (Dittrich and Fohlmeister, 2020; Liu et al., 2020; Schneckenreither et al., 2022; Steinbacher et al., 2024; Zhang et al., 2022)

A notable strength of RL is its capacity to optimize multi-objective reward functions, enabling agents to simultaneously balance multiple KPIs, such as minimizing tardiness while maintaining ConWIP (Altenmüller et al., 2020; Ragazzini et al., 2021; Schuh et al., 2022; Steinbacher et al., 2024; Stricker et al., 2018; Zhang et al., 2022). Moreover, RL offers the flexibility to incorporate system constraints directly into the reward formulation. Time-related constraints, WIP limits, or infeasible release dates can be penalized to guide the agent toward valid and efficient policies without requiring hard-coded restrictions (Altenmüller et al., 2020; Steinbacher et al., 2024; Zhang et al., 2022). Nevertheless, the design of multi-objective reward functions remains an open research challenge (Schuh et al., 2022).

To ensure comparability across different system states, rewards are frequently normalized (Chang et al., 2022; Heger and Voss, 2020; Ragazzini et al., 2021; Samsonov et al., 2021; Schneckenreither and Haeussler, 2019; Wang and Liao, 2024; Zhang et al., 2022). This improves training stability and avoids scale imbalances, particularly when multiple reward components or heterogeneous production scenarios are involved (Ragazzini et al., 2021; Zhang et al., 2022).

Despite its flexibility, reward shaping remains one of the most critical and delicate elements in RL design. Poorly designed reward functions may lead to unintended agent behavior such as optimizing easily achievable sub-goals while ignoring system-level performance. In (Steinbacher et al., 2024) the agents learned to consistently prioritize one of two product types, as this led to more frequent short-term rewards at the expense of balanced system utilization. Similarly, in (Altenmüller et al., 2020), a locally rewarded agent focused on maximizing throughput by completing as many early jobs as possible. This led to high overall rewards while systematically neglecting later ones, resulting in extreme delays for a few orders.

These examples highlight the importance of aligning reward components with overarching performance objectives.

4 CONCLUSION AND RESEARCH AGENDA

This review sets out to examine how existing RL studies model the foundational components, the state space, action space, and reward function of learning agents within the context of PPC. While the initial focus lay on RL-based approaches to order release, the limited availability of relevant studies led to a scope broadening to include RL applications in production scheduling. Considered papers included order pools or job queues with a dispatching mechanism, as they exhibit structural and functional similarities to order release systems.

Across literature, modeling approaches vary widely. Regarding the state space, machine-related and order-specific features are almost always included, yet the level of detail and aggregation varies significantly, often made heuristically without theoretical justification or systematic comparison.
Similarly, action spaces range from job selection and dispatching rule switching to lead time prediction and WIP limit adjustment, reflecting different production logics. These design choices must align with the underlying system: time-scheduled vs. WLC-based environments.
Reward modeling, though flexible, remains a critical challenge. The use of sparse, dense, or difference-based rewards, along with global or local signals, requires careful alignment with system objectives. Improper reward shaping may result in undesired behaviors. Moreover, multi-objective reward functions represent a notable strength of RL and appear to offer considerable potential for more widespread and effective use in future applications.

RL is broadly recognized as a promising extension or alternative to traditional PPC mechanisms, particularly in complex and volatile environments. Yet, order release remains noticeably underrepresented compared to scheduling problems. Despite their increasing industrial relevance, MSPS have similarly rarely been considered, particularly in conjunction with order release mechanisms. Moreover, key modeling decisions, regarding state, action, and reward representation are often based on heuristics, rather than systematic frameworks. In addition, many studies rely on overly simplified simulations, which limit transferability and practical relevance of the findings. This suggests that there is further untapped potential for RL-based order release strategies to enhance performance indicators such as lead time reduction and adherence to delivery dates.

Building upon the identified research gaps, future research should focus on RL-based order release in MSPS, embedded within realistic simulation models or digital twins. Such environments should capture some degree of disruption, dynamic conditions, and resource constraints, ensuring a more accurate assessment of control strategies. While much of the existing literature emphasizes algorithmic comparisons, future research should prioritize the design of the agent's interaction model, specifically the structure of state and action spaces. Special attention should be given to reward modeling, which reflects operational trade-offs and has a critical impact on policy behavior. Multi-agent environments may further provide valuable opportunities to examine decentralized decision-making and the interplay between local and global objectives, especially in the case of the so far underrepresented MSPS.

REFERENCES

Altenmüller, T., Stüker, T., Waschneck, B., Kuhnle, A. and Lanza, G. (2020) 'Reinforcement learning for an intelligent and autonomous production control of complex job-shops under time constraints', *Production Engineering*.

Bauernhansl, T. (ed) (2020) *Management in der Produktion*, Berlin, Heidelberg, Springer Vieweg.

Bauernhansl, T. (ed) (2023) *Handbuch Industrie 4.0*, 3rd edn, Berlin, Heidelberg, Springer Vieweg.

Chang, J., Yu, D., Hu, Y., He, W. and Yu, H. (2022) 'Deep Reinforcement Learning for Dynamic Flexible Job Shop Scheduling with Random Job Arrival', *Processes*.

Cunha, B., Madureira, A. M., Fonseca, B. and Coelho, D. (2020) 'Deep Reinforcement Learning as a Job Shop Scheduling Solver: A Literature Review'.

Da Col, G. and Teppan, E. C. (2022) 'Industrial-size job shop scheduling with constraint programming', *Operations Research Perspectives*.

Dittrich, M.-A. and Fohlmeister, S. (2020) 'Cooperative multi-agent system for production control using reinforcement learning', *CIRP Annals*.

Gómez Paredes, F. J., Godinho Filho, M., Thürer, M., Fernandes, N. O. and Jabbour, C. J. C. (2020) 'Factors for choosing production control systems in make-to-order shops: a systematic literature review', *Journal of Intelligent Manufacturing*.

Greschke, P. (2020) *Matrix-Produktion: Konzept einer taktunabhängigen Fließfertigung*, Norderstedt, BoD – Books on Demand.

Gui, Y., Tang, D., Zhu, H., Zhang, Y. and Zhang, Z. (2023) 'Dynamic scheduling for flexible job shop using a deep reinforcement learning approach', *Computers & Industrial Engineering*.

Haeussler, S., Neuner, P. and Thürer, M. (2022) 'Balancing earliness and tardiness within workload control order release: an assessment by simulation', *Flexible Services and Manufacturing Journal*.

Heger, J. and Voss, T. (eds) (2020) *Dynamically Changing Sequencing Rules with Reinforcement Learning in a Job Shop System with stochastic influences*, IEEE.

Hendry, L., Huang, Y. and Stevenson, M. (2012) 'Workload control: Successful implementation taking a contingency-based view of production planning and control', *International Journal of Operations & Production Management*.

Hofmann, C. (ed) (2021) *Vorausschauende und reaktive Mehrzieloptimierung für die Produktionssteuerung einer Matrixproduktion* (Dissertation), Karlsruher Institut für Technologie, Shaker Verlag.

Janke, T., Riesener, M., Schmitz, S., Fulterer, J. and Eisbein, H. 'Evaluation Concept for Order Release Based on Simulation and Reinforcement Learning Against Changes to the Production Configuration'.

Kardos, C., Laflamme, C., Gallina, V. and Sihn, W. (2021) 'Dynamic scheduling in a job-shop production system with reinforcement learning', *Procedia CIRP*.

Kirchberger, M., Heeger, M., Altay, A., Liebrecht, C., Overbeck, L., Kandler, M., Lanza, G., Voigt, C. and Franke, J. (2022) 'Simulationsgestütztes Vorgehensmodell zur Realisierung einer Matrixfertigung', *Zeitschrift für wirtschaftlichen Fabrikbetrieb*.

Lei, K., Guo, P., Zhao, W., Wang, Y., Qian, L., Meng, X. and Tang, L. (2022) 'A multi-action deep reinforcement learning framework for flexible Job-shop scheduling problem', *Expert Systems with Applications*.

Liu, C.-L., Chang, C.-C. and Tseng, C.-J. (2020) 'Actor-Critic Deep Reinforcement Learning for Solving Job Shop Scheduling Problems', *IEEE Access*.

Liu, J., Land, M. J., Bokhorst, J. A. C. and Chen, Q. (2023) 'Improving coordination in assembly job shops: redesigning order release and dispatching', *Flexible Services and Manufacturing Journal*.

Lödding, H. (2016) *Verfahren der Fertigungssteuerung*, Berlin, Heidelberg, Springer Berlin Heidelberg.

Panzer, M. and Bender, B. (2021) 'Deep reinforcement learning in production systems: a systematic literature review', *International Journal of Production Research*.

Ragazzini, L., Negri, E. and Macchi, M. (2021) 'A Digital Twin-based Predictive Strategy for Workload Control', *IFAC-PapersOnLine*.

Renna, P. (2023) 'Performance analysis of matrix-structured manufacturing systems compared to dedicated flow and reconfigurable manufacturing lines', *Production Engineering*.

Russell, S. J. and Norvig, P. (2021) *Artificial intelligence: A modern approach*, Hoboken, Pearson.

Samsonov, V., Kemmerling, M., Paegert, M., Lütticke, D., Sauermann, F., Gützlaff, A., Schuh, G. and Meisen, T. (2021) 'Manufacturing Control in Job Shop Environments with Reinforcement Learning'.

Schneckenreither, M. and Haeussler, S. (2019) 'Reinforcement Learning Methods for Operations Research Applications: The Order Release Problem'.

Schneckenreither, M., Haeussler, S. and Gerhold, C. (2021) 'Order release planning with predictive lead times: a machine learning approach', *International Journal of Production Research*.

Schneckenreither, M., Haeussler, S. and Peiró, J. (2022) 'Average reward adjusted deep reinforcement learning for order release planning in manufacturing', *Knowledge-Based Systems*.

Schuh, G., Gützlaff, A., Schmidhuber, M., Fulterer, J. and Janke, T. (2022) 'Towards an Automated Application for Order Release', *Procedia CIRP*.

Schuh, G., Schmitz, S., Maetschke, J., Janke, T. and Eisbein, H. (2023) 'Application of a Reinforcement Learning-based Automated Order Release in Production'.

Schuh, G. and Stich, V. (2012) *Produktionsplanung und -steuerung 1*, Berlin, Heidelberg, Springer Berlin Heidelberg.

Song, W., Chen, X., Li, Q. and Cao, Z. (2023) 'Flexible Job-Shop Scheduling via Graph Neural Network and Deep Reinforcement Learning', *IEEE Transactions on Industrial Informatics*.

Steinbacher, L., Wegmann, T. and Freitag, M. (2024) 'Production control with Reinforcement Learning for a matrix-structured production system', *International Journal of Production Research*.

Steinbacher, L. M., Rippel, D., Schulze, P., Rohde, A.-K. and Freitag, M. (2023) 'Quality-based scheduling for a flexible job shop', *Journal of Manufacturing Systems*.

Stricker, N., Kuhnle, A., Sturm, R. and Friess, S. (2018) 'Reinforcement learning for adaptive order dispatching in the semiconductor industry', *CIRP Annals*.

Sutton, R. S. and Barto, A. G. (2018) *Reinforcement learning: An introduction*, Cambridge Massachusetts, The MIT Press.

Thürer, M., Land, M. J., Stevenson, M., Fredendall, L. D. and Godinho Filho, M. (2015) 'Concerning Workload Control and Order Release: The Pre‐Shop Pool Sequencing Decision', *Production and Operations Management*.

Wang, Z. and Liao, W. (2024) 'Smart scheduling of dynamic job shop based on discrete event simulation and deep reinforcement learning', *Journal of Intelligent Manufacturing*.

Wiendahl, H.-P., Reichardt, J. and Nyhuis, P. (2015) *Handbook Factory Planning and Design*, Berlin, Heidelberg, Springer Berlin Heidelberg.

Zhang, Y., Zhu, H., Tang, D., Zhou, T. and Gui, Y. (2022) 'Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems', *Robotics and Computer-Integrated Manufacturing*.

AUTHOR BIOGRAPHIES

PATRICK FARWICK is a Master Student enrolled in the Data Science major at the University of Applied Science Bielefeld. He is currently working on order release within Production Planning and Control and Reinforcement Learning. His research interests include Digital Twins, Simulation and Matrix-structured Production Systems. His email address is patrick.farwick1@hsbi.de.

CHRISTIAN SCHWEDE is Professor for Big Data Analysis at the University of Applied Science and Arts Bielefeld. He is a member of the Center for Applied Data Science and a Board member of the Institute for Data Science Solutions. Additionally, he is a senior scientist at Fraunhofer ISST in Dortmund responsible for AI in Logistics. His research area is the application of data- and model-driven methods to optimize logistics and production. He is specialized in automotive logistics and agent-based decentralized production systems. His email address is christian.schwede@hsbi.de.