Management and Production Engineering Review
Volume 16 • Number 3 • September 2025 • pp. 1–15
DOI: 10.24425/mper.2025.156145

Rescheduling Under Disruption: A Product-Driven Framework with Heuristic and Reinforcement Learning Strategies

Patricio SÁEZ[1], JAIME RIVERA[2], PATRICIO SALAS[1], VICTOR PARADA[3]

[1] *Department of Statistics, Universidad de Concepción, Chile*
[2] *Department of Industrial Engineering, Universidad de Concepción, Chile*
[3] *Department of Informatics Engineering, Universidad de Santiago de Chile, Chile*

Received: 14 October 2024
Accepted: 11 July 2025

Abstract
In modern manufacturing, addressing disruptions across multi-stage production requires adaptive and intelligent scheduling. This study evaluates two rescheduling strategies within a product-driven system for the Job Shop Scheduling Problem under disturbances: one based on the Shifting Bottleneck Heuristic (PDS-SBH), and another using a Monte Carlo Reinforcement Learning agent (PDS-RL). Products act as intelligent agents capable of autonomous decisions. A total of 151 simulations were conducted across 14 benchmark instances, with machine-level disruptions modeled as 100%, 200%, and 300% increases in processing times. PDS-SBH achieved average makespan reductions up to 5.2%, serving as a reactive and interpretable baseline. In contrast, PDS-RL consistently outperformed it, achieving reductions of 22.12%, 37.13%, and 53.87%, respectively. These results highlight the superior adaptability of reinforcement learning in uncertain production contexts. The study contributes to the understanding of how combining product-driven architectures with heuristic and learning-based strategies enables the development of intelligent, autonomous, and resilient scheduling systems.

Keywords
Product-driven system, Agent-based models, Manufacturing planning, shifting bottleneck heuristic, intelligent products, Reinforcement learning.

Introduction

Job planning and scheduling in industrial settings encounter significant challenges due to the complex and dynamic nature of operations. These challenges center around the efficient allocation of resources across various processes, prompting researchers and practitioners to devote considerable attention to studying these problems (Rasheed et al., 2019). Specifically, the Job Shop Scheduling Problem (JSSP), a classic combinatorial optimization issue, has garnered focus from numerous studies aiming to enhance efficient job allocation in scenarios characterized by limited resources. However, the occurrence of disturbances, such as changes in resource availability, machinery failures, or workflow interruptions, adds an extra layer of complexity by challenging the efficacy of the initial planning (Meyer et al., 2011). These disturbances necessitate a dynamic rescheduling of jobs due to the potential invalidity of the original schedule under new conditions. While it is feasible in some instances to adjust the operation easily to accommodate the disruption, task rescheduling is often necessary in many cases to minimize the impact of this disruption and recover a solution as close to the original as possible (Zhang et al., 2020).

The JSSP is a fundamental optimization challenge in operations research and computer science, focusing on the optimal assignment of jobs to resources at specific times. Each job consists of a sequence of operations that the system must process on a set of machines in a specified order. Each machine can handle only one operation at a time, and once an operation begins, it must continue uninterrupted until completion. The objective is typically to minimize the time required to complete all jobs (makespan). However, variations may aim to optimize other criteria, such as minimizing tardiness or maximizing resource utilization. The

***Corresponding author**: Patricio Sáez – Department of Statistics Universidad de Concepción, Chile, e-mail: patricsaez@udec.cl*

© 2025 The Author(s). This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/)

JSSP requires sophisticated scheduling algorithms due to its NP-hard nature, making it computationally challenging for even modest-sized problems.

The Job Shop Scheduling Problem with Disturbances (JSSP-D) extends the classical JSSP to account for real-world operational uncertainties and disruptions that can affect the planned schedule. These disturbances can include machine breakdowns, unexpected maintenance, variations in operation times, the arrival of urgent jobs, or the unavailability of resources, which necessitate dynamic adjustments to the existing schedule (Bouazza et al., 2021; Meyer et al., 2011; Wu & Yan, 2023). The objective remains the same as the classic JSSP, such as minimizing the makespan or other performance metrics, but with the added complexity of incorporating flexibility and adaptability into the scheduling process. Solving the JSSP-D involves developing robust and resilient solutions that can quickly respond to changes and re-optimize the schedule in real-time or near-real-time, ensuring minimal impact on overall productivity and efficiency. This problem is inherently more complex than the static JSSP due to the need for continuous adjustment and evaluation of multiple potential future scenarios to maintain optimal or near-optimal performance under uncertainty.

Despite the critical role of rescheduling in Job Shop environments impacted by disturbances, existing research has addressed this issue only to a limited extent. To solve a JSSP, exact methods such as integer programming have been developed (Cui & Lu, 2017). However, the computational burden of these methods increases exponentially with the problem size, rendering the calculation time a significant constraint for the practical application of such algorithms in JSSP-D scenarios (Ku & Beck, 2016). Consequently, the industry often resorts to heuristic procedures, such as the Shifting Bottleneck Heuristic (SBH), which yield satisfactory outcomes within a reasonable timeframe (Bożek & Werner, 2018).

The SBH proposed by Adams et al. (1988) is a highly effective heuristic for addressing the JSSP, emphasizing the detection and step-by-step resolution of bottleneck operations in manufacturing. This heuristic operates by first breaking down the complex job shop environment into individual machine-centered subproblems. It then identifies the current bottleneck, which is the machine or operation with the most significant impact on the overall makespan or another specified performance metric. Once identified, the heuristic optimally schedules jobs for the bottleneck resource, considering the current schedules of other machines. After resolving the bottleneck, the heuristic re-evaluates the production environment, identifies the next bottleneck, shifts its focus, and iteratively optimizes the schedule until it processes all machines. This approach enables a dynamic and focused improvement of the production schedule, addressing the most critical constraints one at a time to enhance operational efficiency.

Although algorithms such as SBH offer some adaptability to address JSSP-D, the field still demands effective strategies that quickly and accurately reorganize jobs in response to unexpected changes. An intelligent system that makes dynamic decisions as difficulties arise during machine operations can effectively tackle this challenge. A promising approach to achieving this is the adoption of a Product-Driven System (PDS), which leverages the inherent intelligence and communicative capabilities of products to facilitate adaptive and responsive scheduling processes.

A PDS is an advanced manufacturing and production approach where the products carry information regarding processing, handling, and movement through the production line. For this reason, the products are known as intelligent products (IP). This approach leverages technologies such as RFID tags, smart sensors, and embedded systems to enable products to communicate directly with production equipment and management systems (Zhou et al., 2021). The goal is to enhance the efficiency, flexibility, and responsiveness of manufacturing operations by enabling products to guide their production processes, reducing manual intervention, and streamlining workflows.

In addition to metaheuristics, Reinforcement Learning (RL) has also been used in recent years to optimize production scheduling (Yang & Xu, 2022). RL is a machine learning mechanism that enables the development of self-learning strategies through interaction with the environment (Chang et al., 2024). An RL agent learns a rescheduling policy after a disruption by using feedback obtained through simulation. This RL-based approach, within a product-driven environment, can help identify the relative strengths of each strategy under different levels of disruption.

This study introduces a model that integrates the PDS architecture, using the SBH to generate the initial solution for the JSSP, under normal conditions to establish a baseline schedule. Then, the system simulates disruptions by increasing processing times across all machines, which deteriorates the original solution and creates a perturbed instance of the problem (JSSP-D). To address this new scenario, the study employs two distinct and independent rescheduling strategies for all remaining operations. The first approach, PDS-SBH, multi-agents re-apply the SBH to reorganize the affected jobs. The second approach introduces a RL strategy, where an agent trained through Monte Carlo simulation learns a rescheduling policy that aims to minimize makespan after disruptions. Both strategies

operate within a product-driven manufacturing environment, were products and machines function as virtual agents with autonomous decision-making capabilities. The study compares the performance of both approaches under varying levels of disruption, evaluating their effectiveness, adaptability, and overall performance in dynamic and uncertain production contexts.

The comparative evaluation between established heuristics and machine learning tools, as conducted in this study, represents a significant methodological contribution. Such analyses help identify the strengths, limitations, and preferred application scenarios for each approach, while laying the groundwork for developing hybrid or complementary strategies aligned with the flexibility, autonomy, and resilience demanded by Industry 5.0. Therefore, this research contributes not only from a technical perspective but also by proposing a clear agenda for future scientific exploration in the field of intelligent production scheduling.

The paper is structured as follows: Section 2 reviews the literature related to rescheduling in job shop environments, SBH, PDS and RL. Section 3 details materials and methods. Section 4 concentrates on the results, whereas Sections 5 and 6 discuss and draw conclusions.

Literature review

Rescheduling is a crucial aspect of modern manufacturing, vital for enhancing operational efficiency in various settings such as Job Shops and Flow Shop Flexible Environment (Gao et al., 2020; Kim & Kim, 2021; Li et al., 2017). This area has been explored from multiple perspectives, focusing on diverse configurations, methodologies, and performance measures to address the dynamic challenges of production scheduling. Researchers have explored various strategies, ranging from mathematical models to heuristic approaches, to optimize production flow, minimize downtime, and enhance overall productivity (Bi et al., 2023; Pérez-Salazar et al., 2019). The study of rescheduling encompasses developing and applying innovative solutions to adapt to changes and maintain a competitive advantage in the manufacturing sector.

The classification of rescheduling problems provides a structured framework for understanding the multifaceted nature of scheduling challenges in manufacturing (Muhuri & Biswas, 2020). This classification encompasses divisions into specific environments, such as Job and Flow Shops, and extends to various manufacturing processes. It further categorizes rescheduling problems based on analysis policies, which dictate how and when rescheduling should occur, and resolution methods, which involve the specific strategies or algorithms employed to achieve optimal rescheduling. This comprehensive classification scheme facilitates the identification of unique characteristics and requirements for different rescheduling scenarios, highlighting the complexity and variability inherent in optimizing production schedules in the face of dynamic and unforeseen changes.

One of the foundational contributions to the study of rescheduling in dynamic manufacturing environments was presented by Yamamoto and Nof (1985). Their work introduced a scheduling and rescheduling procedure within a computerized manufacturing system (CMS), emphasizing the need for real-time schedule revision in response to operational disruptions, such as machine breakdowns. The authors proposed a structured rescheduling framework integrated into a Manufacturing Operating System (MOS), demonstrating through case studies that this approach could reduce total production time by up to 7% compared to fixed sequencing and dispatching rules. This early integration of intelligent control mechanisms and heuristic rescheduling laid the groundwork for subsequent developments in adaptive production planning, making it a seminal reference for modern rescheduling strategies that combine rule-based and autonomous decision-making.

Various rescheduling methods have been explored to optimize manufacturing processes, including integer programming, constraint-based rescheduling, genetic algorithms, and heuristic rescheduling (Bhongade et al., 2023; Gao et al., 2019; Liang et al., 2023; Upasani & Uzsoy, 2008). Each method offers distinct advantages: integer programming provides exact solutions but can be computationally intensive for large-size problems; constraint-based rescheduling allows for flexibility in handling complex constraints; genetic algorithms offer robust solutions across diverse scenarios with their ability to navigate vast search spaces; and heuristic rescheduling provides quick, practical solutions that, while may not always be optimal, are effective for real-time decision-making. Together, these methods form a comprehensive toolkit, enabling tailored approaches to meet the specific rescheduling needs of different manufacturing environments, balancing efficiency and adaptability.

Within the context of the JSSP-D, the research by Mahmoodjanloo et al. (2022) and Miguel A. Salido (2017) significantly advances our understanding of dynamic scheduling. These studies examine innovative approaches to rescheduling, with a primary focus on how adjustments to machine speed can significantly enhance the effectiveness of rescheduling efforts. By examining these strategies within the JSSP-D frame-

work, their work highlights the potential for adaptive scheduling solutions to mitigate disruptions and maintain production efficiency, demonstrating the importance of flexibility and quick responsiveness in modern manufacturing environments.

Integrating artificial intelligence and innovative technologies into production systems is increasingly centered around IPs and PDS. This trend emphasizes the role of IPs and PDS in enhancing the adaptability and agility of rescheduling capabilities within manufacturing (Meyer et al., 2011; Wu et al., 2019). By embedding intelligence directly into products, these systems enable real-time, autonomous decision-making that can dynamically adjust to changes and disruptions in the production process. This approach enhances the efficiency and flexibility of manufacturing operations, significantly reducing the need for manual intervention and paving the way for more resilient and responsive production environments.

Agent-Based Modeling (ABM) for PDS harnesses the power of simulation to create a responsive production environment. By representing various elements of the manufacturing process as agents with distinct levels of intelligence, ABM facilitates the emulation of complex, collective behaviors (Sáez et al., 2023). These agents can dynamically interact, make decisions, and adapt to changing circumstances, significantly enhancing the flexibility and efficiency of production systems. This approach enables granular analysis and optimization of production processes, enhancing operational resilience and responsiveness to unforeseen challenges.

In PDS, task rescheduling is pivotal for effectively managing operational disturbances. Studies by Bhongade et al. (2023) and de Guimarães et al. (Campos et al., 2020) stand out, showcasing the practical application and efficacy of PDS in overcoming rescheduling challenges. These investigations demonstrate how PDS, leveraging real-time data and autonomous decision-making capabilities of intelligent products, can dynamically adjust schedules in response to unexpected events. This approach minimizes downtime and optimizes production flow, illustrating PDS's substantial contribution to enhancing operational resilience and efficiency in manufacturing environments (Pannequin & Thomas, 2012).

The convergence of PDS and SBH into a hybrid model represents a strategic evolution in addressing the complexity of rescheduling in industrial environments, particularly within JSSP-D with disturbances. This hybrid model leverages the dynamic adaptability of PDS, with its intelligent, autonomous products, and the focused efficiency of SBH in pinpointing and alleviating bottlenecks. Such integration offers a promising pathway to agile, practical strategies for dynamic adaptation to unforeseen changes, ensuring optimized scheduling and enhanced production flow despite the inevitable disruptions.

Despite recent advances in production scheduling methods, their direct application in industrial environments remains a considerable challenge. One of the main obstacles lies in the need to apply these techniques in online scheduling contexts, which imposes critical constraints such as the need to generate alternative solutions quickly (Hwangbo et al., 2024). In this context, several studies have begun to explore the use of RL as an effective tool to address rescheduling problems under uncertainty.

Yang & Xu (2022) investigated, for the first time, intelligent scheduling and reconfiguration of Reconfigurable Flow Lines with dynamically arriving jobs using Deep Reinforcement Learning (DRL). The authors proposed a smart manufacturing architecture that integrates real-time scheduling and reconfiguration decisions through the design of state features, actions, and reward structures for specialized agents. Their approach proved capable of minimizing the total tardiness cost, demonstrating DRL's ability to make intelligent decisions based on current system status and order information.

Similarly, Roesch et al. (2019) applied multi-agent reinforcement learning for industrial load management, with a focus on energy-oriented rescheduling. Through a simulation study, they compared their approach to a simulated annealing metaheuristic, showing that the RL-based model achieved reasonable solutions with low computational cost, making it suitable for real-time applications.

Kardos et al., (2020) introduced a Q-learning approach to reduce the average lead time in job-shop production systems. In their model, intelligent product agents choose machines for each production step based on real-time information. Compared to standard dispatching rules, the RL approach significantly reduced average processing time, highlighting its potential in dynamic scheduling scenarios. The application of RL to dynamic scheduling has shown promising results (Zhang et al., 2020); however, its integration into product-driven architectures remains very limited. Moreover, few studies have conducted systematic comparisons between RL-based agents and well-established heuristics such as the SBH under equivalent experimental conditions.

The relevance and contribution of the proposed PDS-RL model lie in the fact that no previous studies have reported the integration of Monte Carlo simulation with RL to solve scheduling problems in Job Shop environments. This combination proves particularly valuable, as it enables the agent to learn

rescheduling policies through simulated feedback, fostering fast, experiential learning that adapts to changing conditions. Also, while existing literature has explored heuristic techniques such as SBH and more conventional uses of RL, it has not sufficiently addressed intelligent manufacturing environments that incorporate autonomous learning, distributed decision-making, and real-time adaptability. In this regard, the proposed model opens a new research avenue by demonstrating how an agent trained under this framework can make informed and context-aware decisions in the face of severe disruptions.

Materials & Methods

Product-Driven Architecture

The proposed model integrates the decision-making autonomy of intelligent products within a PDS framework to address rescheduling challenges in the JSSP-D. Within this architecture, the system differentiates between two types of agents: product agents and machine agents. Product agents act as intelligent, autonomous entities capable of dynamically determining their production paths. Machine agents, in contrast, represent resources with fixed capabilities and availability constraints.

Each agent operates within a virtual environment that mirrors the physical layout of the production system. Product agents transition through four distinct states: "free" (available for processing), "in process" (currently being processed), "rescheduled" (adjusted due to a disruption), and "completed" (production cycle finished). Machine agents adopt the states "available," "in process," and "disturbed," the latter reflecting malfunctions or unplanned downtimes. These defined states enable agents to interact dynamically, allowing for real-time schedule adjustments in response to disruptions and thereby enhancing system resilience.

Figure 1 illustrates the architecture's dual-layer interaction: a virtual plane where intelligent agents operate and a physical plane where actual production occurs. The virtual representation facilitates seamless synchronization and decision-making, empowering product agents to autonomously navigate and adjust within the system in response to real-time conditions. Furthermore, we initially used SBH as a solution to the problem.

Problem Definition and Disturbance Modeling

The model addresses the classic JSSP. In the JSSP, a set of jobs, each with a predefined sequence of operations, must be processed on specific machines. The model's objective is to minimize the makespan, the total time needed to complete all jobs.

To simulate real-world variability, we extend the JSSP to include disruptions, creating the JSSP-D. We model these disruptions as increases in machine processing times, simulate breakdowns or severe slowdowns. We define three levels of disturbance: Dist_{100} (a 100% increase), Dist_{200} (a 200% increase), and Dist_{300} (a 300% increase) in the processing time for each affected machine.

The model first generates a baseline solution using SBH. When the cumulative execution time reaches 50% of an instance's lower bound, a disturbance triggers altering a selected machine's processing time. The model then reschedules the remaining operations using either the SBH or a RL strategy to mitigate the disruption and restore an optimized makespan.

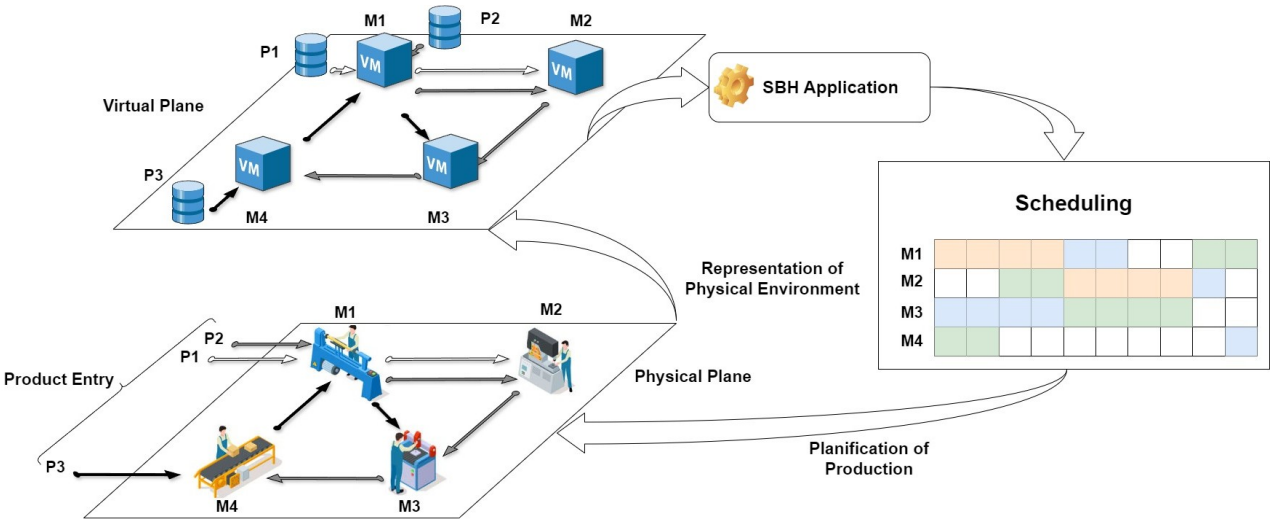

Fig. 1. PDS-SBH application sequence diagram for JSSP programming

The experimental setting utilizes problem instances described in Table 1, characterized by the number of jobs, machines, and their respective operations. For instance, the abz5 instances comprise ten jobs and ten machines, amounting to 100 operations with varying processing times. This specification provides a structured framework for assessing the model's efficiency across various scenarios, enabling a comprehensive evaluation of its performance under diverse operational complexities.

Heuristic-Based Rescheduling Strategy: PDS-SBH

The heuristic-based strategy uses SBH, which is integrated into the product-driven system, to address reprogramming needs after disruptions. Initially, product agents are assigned to machines following a predefined sequence. The model then identifies critical machines whose schedules have a significant impact on the overall makespan. It prioritizes these machines to resolve bottlenecks iteratively.

In the event of a disturbance, product agents dynamically reassess their current positions and interdependencies within the production sequence. They seek alternative processing paths, reprioritize jobs, or adjust sequences to mitigate the impact of the perturbation. SBH is reapplied to the remaining operations to identify and resolve the most significant bottlenecks in the updated context. Figure 2 presents a UML diagram that meticulously outlines the experimental design process, capturing the model's sequence of operations and interactions. It begins with loading the problem instance into the model and calculating the job sequence using SBH. This step involves determining the makespan by applying SBH to devise a sequence that minimizes it, thereby establishing a baseline schedule without disturbances. Each product agent receives details on processing time and operation sequence, leveraging this information to craft a production schedule informed by SBH principles.

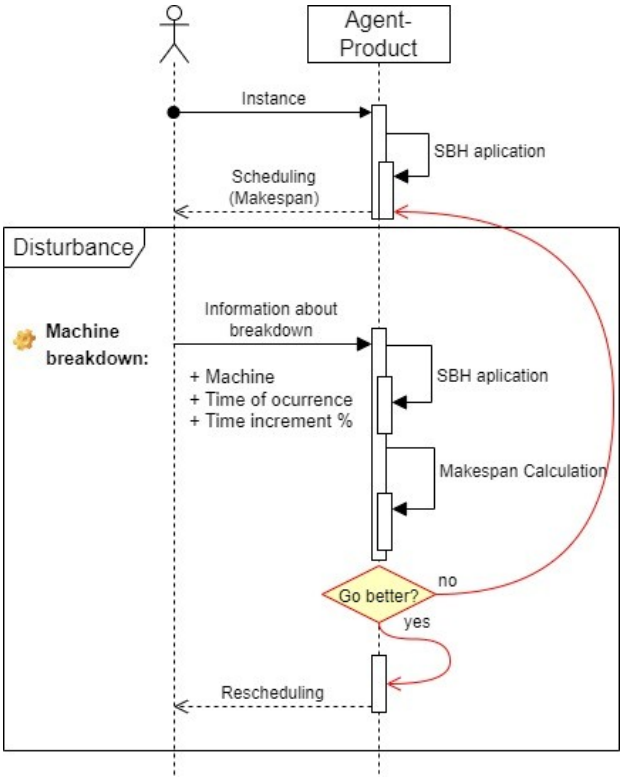

Fig. 2. UML sequence diagram in the model PDS-SBH

The process unfolds in three stages: (1) initial scheduling using SBH, (2) activation of a rescheduling protocol upon detection of a disturbance, and (3) reapplication of SBH to optimize the disrupted schedule. Figure 2 outlines the flow of activities for product agents during a disturbance, demonstrating the model's capacity for localized, agent-based adaptation in line with SBH principles.

Table 1
Problem instances used in experimentation

Instance	Jobs	Machine	Operation	Instance	Jobs	Machines	Operations
Ft06	6	6	36	La21	15	10	150
La01	10	5	50	La29	20	10	200
La06	15	5	75	La38	15	15	225
Abz05	**10**	**10**	**100**	Abz07	20	15	300
Abz06	10	10	100	Abz08	20	15	300
La16	10	10	100	Abz09	20	15	300
La11	20	5	100	Yn04	20	20	400

Learning-Based Strategy: PDS-RL with Monte Carlo

RL, the agent's decisions rely on a policy represented by an action-value function. That function estimates the expected return for each action in each state and guides the agent to choose actions that maximize future outcomes. Combining those return estimates and decision rules gives us the policy.

Learning an optimal policy can be categorized into two main approaches: model-based and model-free. Model-based methods, such as dynamic programming, assume complete knowledge of the environment's transition probabilities and reward function. Model-free methods, exemplified by Monte Carlo, derive policies directly from episodic experience of states, actions, and rewards without constructing an explicit model of the dynamics.

Monte Carlo methods (MC) update action-value estimates based on complete episodes, which enables efficient learning even when the system experiences disturbances. By leveraging complete trajectories, the method recursively updates values through a process known as back-up, solving the Bellman equation incrementally. The agent aims to approximate the optimal action-value function that maximizes cumulative reward. The update rules for terminal and non-terminal states are expressed (1) terminal and (2) non-terminal states as follows (Sutton & Barto, 1998):

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R_i + \gamma R_T - Q(s,a)] \quad (1)$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[R_i + \gamma \max_a R_t - Q\left(s',a'\right)\right] \quad (2)$$

Where $Q(s,a)$ denotes the current state–action value, $Q(s',a')$ the value of the next state–action pair, α the learning rate (extent to which new information overrides old), R_i the immediate reward, R_t the terminal reward, and γ the discount rate (influence of future rewards).

On-policy Monte Carlo control underpins this RL-MC implementation for flexible job shop scheduling. An agent dispatches operations to achieve a minimized final makespan given the specified job shop configuration. The framework begins by defining the core components of the systems – state representation, action set, transition dynamics, and reward scheme – to formalize the scheduling environment.

The state space $\mathcal{S}$ consists of vectors $s = (n_1, \ldots, n_j, b_1, \ldots, b_m)$, where n_j denotes the index of the next pending operation for job j and b_m indicates whether machine m is busy (1) or idle (0). The action space $\mathcal{A}(s)$ comprises all operations whose predecessors have been completed and that are not currently in progress. Upon selecting action an in state s, the transition function $P[s'|sa]$ advances the simulation clock to the next operation completion or machine release event, producing the subsequent state s'. Each episode ends when the agent has dispatched and completed all operations in the job-shop configuration.

Building on the System specification, we design the exploration–exploitation stage as an epsilon-greedy strategy that balances discovery and performance. With probability ε the agent picks a random action, uncovers new state–action pairs, and prevents biased Q-value estimates. With probability $1-\varepsilon$ it selects the action with the highest $Q(s,a)$, and as Q improves and ε decays, the agent exploits its growing knowledge. This method sustains steady learning while avoiding early convergence on suboptimal choices.

After defining how the agent balances exploration and exploitation, we describe how it updates its action-value estimates, shapes rewards, and measures performance. We apply a first-visit Monte Carlo update: we reverse the recorded (s,a,r) trace, set $G=0$ and clear the visited set, then for each tuple compute $G \leftarrow R_{t+1} + \gamma G$ and, if (s',a') lies outside visited, add it, increment $N(s_t,a_t)$ by one, and perform (3):

$$Q(s,a) \leftarrow Q(s,a) + \frac{1}{N_{s,a}}\left[G - Q(s,a)\right] \quad (3)$$

This procedure drives $Q(s,a)$ toward the running average of all observed returns for each state–action pair.

To drive makespan minimization, we structured the reward function and evaluated the final policy under pure exploitation. We assign zero reward at every intermediate step ($R_t = 0$ for $t < T$) and $R_T = -$makespan at episode end, where makespan equals the maximum completion time across all machines. With $\gamma = 1$, each return $G_0, \ldots, G_{T-1} = -$makespan. For evaluation we set $\varepsilon = 0$, dispatch actions by $\arg\max_{a \in \mathcal{A}(s)} Q(s,a)$ run the episode to completion, and record the resulting makespan as the performance metric. Figure 3 presents a UML diagram that meticulously outlines the experimental design process.

This learning process continues iteratively until convergence to a policy that yields minimal makespan under dynamic job shop scheduling with disruptions.

Comparative Schematic of Heuristic- and Learning-Based Rescheduling

To contextualize the comparison, we examine how each rescheduling strategy fits within the broader PDS framework and responds to identical disruptions. By aligning both approaches on the same baseline – and subjecting them to the same perturbation event – this schematic highlights their respective decision-making

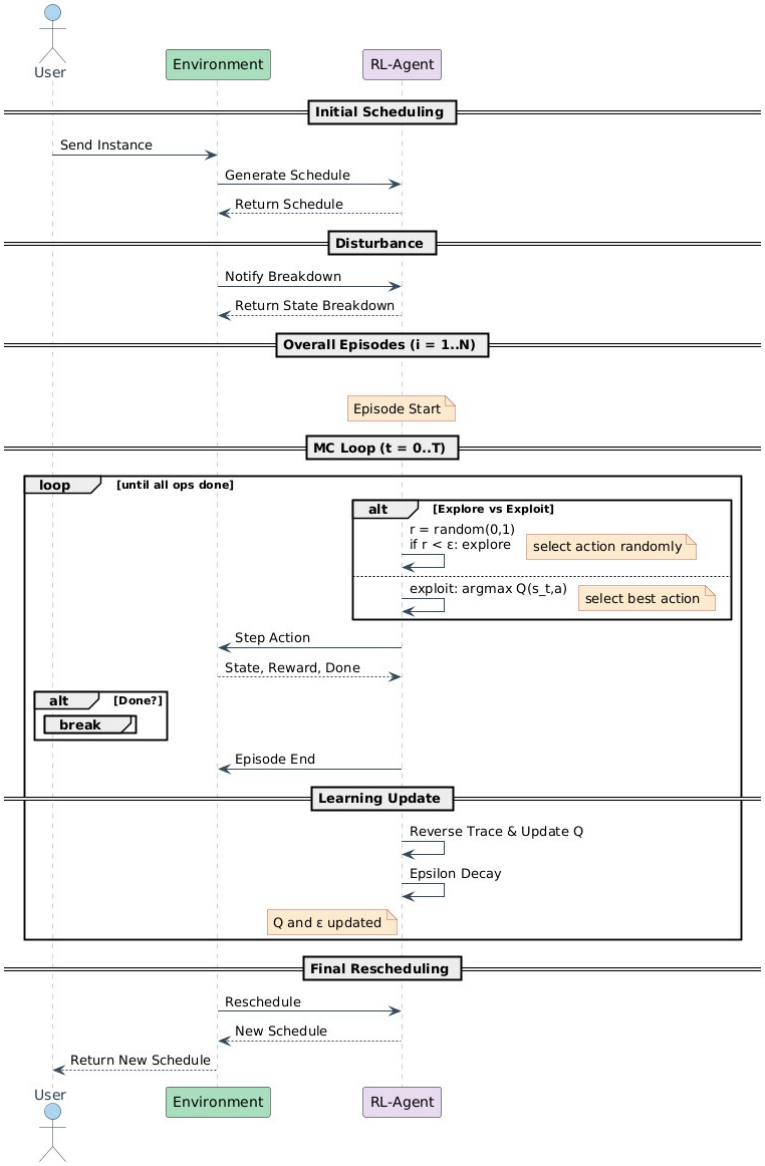

Fig. 3. UML sequence diagram in the model PDS-RL

processes, resource interactions, and adaptation mechanisms. Specifically, it shows how SBH relies on iterative bottleneck resolution using product agents, while RL-MC employs learned action-value estimates to drive policy updates and scheduling adjustments dynamically.

Figure 4 presents a comparative schematic that integrates the two rescheduling strategies explored in this study: the heuristic-based approach (PDS-SBH) and the learning-based approach (PDS-RL with MC). Both strategies begin with the same initial condition, scheduling a Job Shop instance using SBH to obtain a baseline solution. The model introduces a disturbance by increasing the processing time of a selected machine once the execution time reaches a predefined threshold. From this point, the system branches into two distinct rescheduling workflows. In the PDS-SBH path, the model reapplies SBH to the remaining operations, with intelligent product agents interacting to identify and resolve post-disruption bottlenecks. In contrast, the PDS-RL path activates a reinforcement learning agent trained via on-policy MC control, which evaluates the current state of the disrupted environment and makes rescheduling decisions based on a learned action-value function. The diagram highlights the procedural symmetry in the setup and disturbance phases, while also emphasizing the methodological divergence in the rescheduling logic, allowing for a systematic comparison of both strategies under identical conditions.

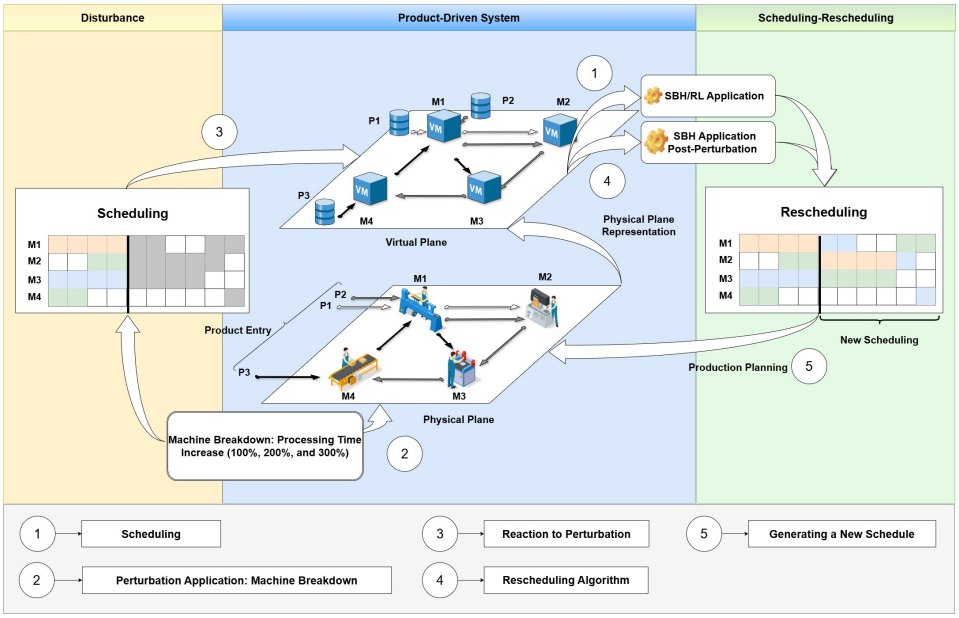

Fig. 4. Flowchart for the post-disturbance intelligent product decision-making process

The model was implemented on version 6.3 of the Netlogo platform using an AMD Ryzen 5 3550H, a 2.1 GHz processor (8 cores), and 12 GB RAM

Results

This section presents the results obtained from simulations conducted across various scenarios, totaling 151 experimental runs, corresponding to one simulation per machine involved in each problem instance. We compare the performance of systems under stable conditions with that of systems affected by individual machine failures. We simulate these failures through the Dist_{100}, Dist_{200}, and Dist_{300} scenarios, using both rescheduling strategies: PDS-SBH and PDS-RL.

The initial solution generated for the JSSP using the SBH consistently yields results close to the optimum, validating its use as a reliable baseline solution before any disturbance occurs. Table 2 presents the results of instances evaluated under undisturbed conditions, including the lower bound reported in the literature, the makespan obtained with SBH, and the percentage gap between the two. These gaps vary significantly across instances, ranging from 4.17% (La01) to 48.69% (La38). In some cases, significant gaps suggest that SBH struggles to approximate the optimal solution, particularly in more complex problems or those with a high number of operations; in others, small gaps indicate stronger performance.

Overall, the results suggest that the effectiveness of SBH is highly dependent on the structure and size of each instance, and that complementary strategies or specific adjustments may be necessary to handle the more complex cases. Despite these variations, the gap values confirm that SBH is a viable and efficient approach for generating initial rescheduling plans. We can then apply reactive strategies to these plans in the event of disruptions.

Table 2
Lower bound, makespan, and gap concerning the optimal result

Instance	LB	SBH	Gap [%]
Ft06	55	61	9.83
La01	666	695	4.17
La06	926	1060	12.64
Abz05	1234	1592	22.49
Abz06	943	1120	15.80
La16	717	1240	42.18
La11	1222	1433	14.72
La21	935	1398	33.12
La29	1105	1635	32.42
La38	943	1838	48.69
Abz07	656	870	24.60
Abz08	648	960	32.50
Abz09	678	1005	32.54
Yn04	929	1323	29.78

Performance of PDS-SBH under Disturbances

Table 3 presents the results for the JSSP-D scenario using the PDS-SBH strategy, where $M_{\mathrm{D-SBH}}$ denotes the makespan achieved after applying the SBH as a rescheduling mechanism. The table presents the average percentage reduction in makespan achieved through the mitigation of disruptions. The results exhibit a clear trend: the magnitude of improvement increases with the severity of the disturbance, rising from 4.26% under Dist_{100} to 4.77% under Dist_{200}, and reaching 5.20% under Dist_{300}.

In addition to mean values, Table 3 includes the standard deviation, along with the maximum and minimum observed improvements (A value of 0.0 shows no improvement). These statistical measures provide a comprehensive overview of the PDS-SBH model's effectiveness and consistency across varying levels of disruption.

Performance of PDS-RL under Disturbances

Table 4 presents the performance of the PDS-RL strategy under the three levels of disturbance: Dist_{100}, Dist_{200}, and Dist_{300}. The metric $M_{\mathrm{D-RL}}$ [%] represents the relative reduction in makespan achieved by the RL-based rescheduling agent compared to the deteriorated solution. The results show a clear upward trend in performance as the level of disruption increases. On average, PDS-RL achieved reductions of 22.12% for Dist_{100}, 37.13% for Dist_{200}, and 53.87% for Dist_{300}.

This progressive improvement suggests that the reinforcement learning agent can effectively utilize the additional flexibility introduced by longer processing times to generate more efficient rescheduling policies. However, the standard deviation also increases with disturbance severity (from 0.110 to 0.274), indicating a higher variability in performance outcomes across instances. Notably, the maximum reduction reaches 133.46% under Dist300, highlighting the potential of PDS-RL to outperform the initially deteriorated plan significantly. At the same time, the presence of minimum values of 0.0% in all scenarios reveals that in some instances, the RL agent failed to improve the baseline, highlighting how the RL Monte Carlo method's emphasis on system-wide policy updates can introduce variability in learning depending on the instance, despite evidence of substantial improvements.

Table 4 provides a comprehensive breakdown of the rescheduling performance for each problem instance under disturbances Dist_{100}, Dist_{200}, and Dist_{300}, comparing the percentage reduction in makespan achieved by the PDS-SBH and PDS-RL strategies (denoted as δ_{SBH} and δ_{RL}, respectively). For each disturbance level, the table reports the improvement relative to the deteriorated makespan following the disruption (M_{D}), offering a direct view of each method's effectiveness across a range of scenarios.

The results reveal clear performance differences between the two approaches. Across most instances, PDS-RL consistently outperforms PDS-SBH, particularly as

Table 3
Makespan reduction by introducing disturbances in PDS-SBH

	Dist_{100}	Dist_{200}	Dist_{300}
Performance	$M_{\mathrm{D\text{-}SBH}}$ (%)	$M_{\mathrm{D\text{-}SBH}}$ (%)	$M_{\mathrm{D\text{-}SBH}}$ (%)
Average	4.260	4.77	5.200
Standard deviation	0.052	0.057	0.066
Maximum value	18.97	26.27	36.06
Minimum value	0.000	0.000	0.000

Table 4
Makespan reduction by introducing disturbances in PDS-RL

	Dist_{100}	**Dist_{200}**	**Dist_{300}**
Performance	$M_{\mathrm{D-RL}}$ (%)	$M_{\mathrm{D-RL}}$ (%)	$M_{\mathrm{D-RL}}$ (%)
Average	22.12	37.13	53.870
Standard Deviation	0.110	0.186	0.2740
Maximum Value	54.31	90.83	133.46
Minimum Value	0.000	0.000	0.000

the severity of the disturbance increases. For example, in instance Ft06, while PDS-SBH achieves no improvement under Dist_{100} ($\delta_{\text{SBH}} = 0.00\%$), PDS-RL achieves a significant 15.03% reduction. This trend continues with higher disturbance levels, where PDS-RL reaches 31.69% improvement under Dist_{200} and 46.99% under Dist_{300}, far exceeding the marginal gains achieved by SBH (0.82% and 1.09%, respectively).

A similar pattern can be observed in instance Abz06, where PDS-SBH produces negligible improvements ($\delta_{\text{SBH}} = 0.00\%$, 1.29%, and 2.16% across increasing disturbance levels), while PDS-RL delivers robust performance ($\delta_{\text{RL}} = 23.12\%$, 39.79%, and 58.77%, respectively). In total, PDS-RL achieves higher improvements in all 14 instances across all three disturbance levels, often by a wide margin.

It is notable that for Dist_{300}, PDS-RL achieves a 70% improvement in instance La21, whereas PDS-SBH reaches only 6.74% for the same case. These substantial differences suggest that the RL agent leverages the increased flexibility provided by longer processing times to explore and exploit superior rescheduling strategies. Conversely, the rule-based nature of SBH appears to be limited in adapting to the more complex solution space introduced by higher levels of disruption. In contrast, PDS-SBH achieves its best result in instance La38 under Dist_{100}, where we observe an 11.47% reduction.

Table 5 highlights the superior adaptability and learning capabilities of the PDS-RL strategy, particularly under medium and high disturbance scenarios. While PDS-SBH provides modest and consistent improvements, its performance remains limited in highly constrained instances or under severe disruptions. These findings highlight the significance of reinforcement learning agents in dynamic manufacturing environments, where responsiveness and generalization are essential for operational resilience.

To assess whether the observed differences in performance between the PDS-SBH and the PDS-RL rescheduling strategies are statistically significant, we performed a paired t-test for each disturbance level (Dist_{100}, Dist_{200}, and Dist_{300}) across the 14 benchmark instances. The results confirm that the improvements achieved by PDS-RL are statistically superior in all cases. Specifically, the mean difference in makespan reduction was 18.07% for Dist_{100} ($t = 13.18$, $p < 0.001$), 33.37% for Dist_{200} ($t = 17.92$, $p < 0.001$), and 49.85% for Dist_{300} ($t = 20.97$, $p < 0.001$). These results indicate that the learning-based strategy provides significantly greater reductions in makespan compared to the heuristic approach, and the significance of this difference increases as the severity of the disturbance increases. This result reinforces the conclusion that PDS-RL not only outperforms PDS-SBH consistently

Table 5
Instance-Level Breakdown of Deterioration Reduction Results

Instance	**Dist_{100}**		**Dist_{200}**		**Dist_{300}**	
	δ_{SHB}(%)(SD)	**δ_{RL}(%)(SD)**	**δ_{SHB}(%)(SD)**	**δ_{RL}(%)(SD)**	**δ_{SHB}(%)(SD)**	**δ_{RL}(%)(SD)**
Ft06	0.00 (0.00)	15.03 (0.15)	0.82 (0.02)	31.69 (0.31)	1,09 (0.03)	46.99 (0.46)
La01	2.65 (0.04)	14.24 (0.12)	3.22 (0.05)	36.20 (0.25)	3,22 (0.05)	55.02 (0.39)
La06	6.89 (0.07)	24.68 (0.14)	4.23 (0.05)	39.68 (0.19)	4,36 (0.06)	55.74 (0.30)
Abz05	10.27 (0.05)	28.58 (0.11)	9.21 (0.06)	37.47 (0.23)	11,01 (0.11)	50.06 (0.33)
Abz06	0.00 (0.00)	23.12 (0.09)	1.29 (0.03)	39.79 (0.19)	2,16 (0.06)	58.77 (0.27)
La16	4.39 (0.04)	22.91 (0.08)	6.55 (0.06)	34.60 (0.17)	4,21 (0.05)	47.77 (0.26)
La11	6.57 (0.06)	23.67 (0.15)	5.04 (0.06)	41.69 (0.22)	5,44 (0.06)	60.73 (0.26)
La21	2.52 (0.03)	30.36 (0.07)	5.55 (0.06)	49.62 (0.14)	6,74 (0.09)	70.41 (0.22)
La29	3.94 (0.03)	26.94 (0.11)	3.52 (0.04)	44.14 (0.21)	3,61 (0.05)	62.98 (0.33)
La38	11.47 (0.04)	31.52 (0.07)	8.93 (0.06)	40.84 (0.15)	7,56 (0.07)	52.50 (0.23)
Abz07	2.16 (0.03)	16.97 (0.07)	2.53 (0.04)	38.08 (0.15)	3,98 (0.06)	61.67 (0.21)
Abz08	1.16 (0.03)	24.88 (0.10)	2.81 (0.04)	44.34 (0.18)	4,26 (0.06)	60.88 (0.29)
Abz09	7.27 (0.05)	18.22 (0.07)	9.31 (0.08)	31.14 (0.14)	7,93 (0.08)	47.81 (0.21)

but also does so in a statistically robust manner, particularly under highly dynamic production scenarios.

These results underscore the importance of conducting comparative evaluations between established heuristics and machine learning-based rescheduling strategies. By systematically analyzing the performance of PDS-SBH and PDS-RL under identical conditions and varying disturbance levels, this study not only reveals their respective strengths and limitations but also clarifies the context in which each method is most effective. The superior adaptability of PDS-RL in highly dynamic scenarios suggests that reinforcement learning agents can play a pivotal role in the next generation of intelligent manufacturing systems. Conversely, the consistent, rule-based performance of PDS-SBH continues to offer value in settings where transparency and reliability are prioritized. These findings provide a solid empirical foundation for developing hybrid or complementary approaches, advancing the broader research agenda of intelligent, resilient, and autonomous production scheduling aligned with the principles of Industry 5.0. The following section discusses the broader implications of these findings and outlines avenues for future research.

Discussion

This study introduced and evaluated two alternative rescheduling strategies within product-driven architecture: one based on PDS-SBH, and the PDS-RL. The goal was to assess their effectiveness in mitigating the impacts of machine-level disturbances within JSSP-D.

The results across 151 simulations on 14 benchmark instances reveal a clear contrast between the two strategies. While the PDS-SBH model achieved modest makespan reductions, averaging 4.26%, 4.77%, and 5.20% under Dist_{100}, Dist_{200}, and Dist_{300}, respectively, it demonstrated limited adaptability, particularly in instances with constrained critical paths. In contrast, PDS-RL exhibited a significantly higher capacity to reduce deterioration, achieving average improvements of 22.12%, 37.13%, and 53.87% for the same disturbance levels. These results validate the RL agent's ability to exploit the increased flexibility introduced by longer processing times, enabling the exploration of more effective scheduling trajectories.

Instance-level analysis confirms this trend: PDS-RL consistently outperformed PDS-SBH across all 14 instances and all levels of disturbance. For example, in instance La21 under Dist_{300}, PDS-RL achieved a 70.41% improvement, while PDS-SBH only reached 6.74%. Moreover, the best performance for PDS-SBH, 11.47% in instance La38 under Dist_{100}, was an exception, suggesting that the SBH heuristic may occasionally exceed its baseline but lacks the generalization ability of the RL-based approach.

The increasing standard deviation observed in both strategies as the disturbance level rises further illustrates the challenges of adapting to greater complexity. However, while PDS-SBH's improvements plateau, PDS-RL continues to scale in performance, confirming its robustness in highly dynamic environments.

These findings are particularly relevant considering that lower bounds (LB) commonly used in job shop scheduling originate from computationally expensive optimization methods, which are often impractical for real-time rescheduling. In real-world contexts, where initial static plans rapidly become obsolete due to unexpected disruptions, heuristic or learning-based approaches offer practical and scalable alternatives. PDS-SBH addresses this need by combining agent-based autonomy with rule-based bottleneck resolution. However, the superior performance of PDS-RL underlines the potential of reinforcement learning to serve as a more adaptive and generalizable solution for dynamic scheduling.

Beyond operational results, this study contributes to the literature by, to the best of our knowledge, being the first to apply Monte Carlo Reinforcement Learning in the context of job shop rescheduling. The PDS-RL agent learned effective rescheduling policies via simulation-based feedback, requiring no prior expert rules. This characteristic makes it especially promising for Industry 5.0 environments, where autonomous systems must continually learn and adapt to changing conditions.

Conclusions

This study proposes and evaluates two alternative rescheduling strategies within a product-driven manufacturing architecture to address the Job Shop Scheduling Problem under disturbances (JSSP-D): one based on the PDS-SBH and the other employing a PDS-RL. We conducted a comprehensive experimental analysis using 151 simulations across 14 benchmark instances, considering disturbances of varying severity (Dist_{100}, Dist_{200}, Dist_{300}).

The results demonstrate that both strategies can mitigate the negative impacts of machine-level disruptions, with notable differences in performance. The PDS-SBH model, which combines intelligent product agents with rule-based bottleneck resolution, achieved moderate improvements in makespan reduction (up to 8.62% on average under severe disturbances). While effective as a reactive and interpretable heuristic, its

performance remained constrained in highly complex instances with critical path limitations.

In contrast, the PDS-RL approach delivered significantly superior results, achieving average makespan reductions of 22.12%, 37.13%, and 53.87% for Dist_{100}, Dist_{200}, and Dist_{300}, respectively. These findings underscore the RL agent's ability to leverage flexibility and adapt dynamically to the severity of disruptions. In several instances, the PDS-RL model achieved over 70% improvement compared to the degraded schedule, far exceeding the best performance of PDS-SBH. Moreover, it did so without relying on domain-specific rules; instead, it learned effective policies from experiential feedback through simulation.

This work is the first to implement Monte Carlo Reinforcement Learning for job shop rescheduling. The integration of simulation-based learning within a product-driven system represents a novel contribution to the field of dynamic scheduling, opening new possibilities for fully autonomous and adaptive production systems.

This study opens several avenues for future research aimed at enhancing the robustness and applicability of intelligent rescheduling systems. Expanding the experimental design to include diverse types of disturbances, such as machine unavailability, urgent job insertions, or supply delays, will help evaluate system adaptability under a broader range of operational disruptions. Additionally, future work should incorporate multi-objective performance criteria, including total tardiness, work-in-progress (WIP), energy consumption, and system nervousness, to enable a more comprehensive and realistic assessment of scheduling strategies. Exploring hybrid approaches that combine reinforcement learning with traditional heuristics, as well as investigating decentralized architectures for distributed learning and decision-making, may yield scalable, interpretable, and more resilient solutions. These directions will help align rescheduling strategies with the complex demands of Industry 5.0, where agility, autonomy, and sustainability are key priorities.

While PDS-SBH delivers a reliable and interpretable approach for reactive rescheduling under uncertainty, PDS-RL drives unprecedented adaptability, scalability, and performance – especially in complex and dynamic contexts – underscoring the pivotal role of RL in intelligent manufacturing and its alignment with Industry 5.0's demands for flexibility, autonomy, and resilience.

Acknowledgments

V. Parada gratefully acknowledge financial support from ANID PIA/PUENTE AFB220003 and DICYT-USACH 061919PD.

References

Adams, J., Balas, E., & Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *The Intitute of Management Sciences*, *34*(2), 391–400. DOI: 0025-1909/88/3403/0391

Bhongade, A.S., Khodke, P.M., Rehman, A.U., Nikam, M.D., Patil, P.D., & Suryavanshi, P. (2023). Managing Disruptions in a Flow-Shop Manufacturing System. *Mathematics*, *11*(7). DOI: 10.3390/math11071731

Bi, M., Kovalenko, I., Tilbury, D.M., & Barton, K. (2023). Dynamic distributed decision-making for resilient resource reallocation in disrupted manufacturing systems. *International Journal of Production Research*. DOI: 10.1080/00207543.2023.2200567

Bouazza, W., Sallez, Y., & Trentesaux, D. (2021). Dynamic scheduling of manufacturing systems: a product-driven approach using hyper-heuristics. *International Journal of Computer Integrated Manufacturing*, *34*(6), 641–665. DOI: 10.1080/0951192X.2021.1925969

Bożek, A., & Werner, F. (2018). Flexible job shop scheduling with lot streaming and sublot size optimisation. *International Journal of Production Research*, *56*(19), 6391–6411. DOI: 10.1080/00207543.2017.1346322

Campos, J.T. de G.A.A., Blumelova, J., Lepikson, H.A., & Freires, F.G.M. (2020). Agent-based dynamic scheduling model for product-driven production. *Brazilian Journal of Operations & Production Management*, *17*(4), 1–10. DOI: 10.14488/bjopm.2020.044

Chang, X., Jia, X., & Ren, J. (2024). A reinforcement learning enhanced memetic algorithm for multi-objective flexible job shop scheduling toward Industry 5.0. *International Journal of Production Research*, *63*(1), 119–147. DOI: 10.1080/00207543.2024.2357740

Cui, W.W., & Lu, Z. (2017). Minimizing the makespan on a single machine with flexible maintenances and jobs' release dates. *Computers and Operations Research*, *80*, 11–22. DOI: 10.1016/j.cor.2016.11.008

Gao, K., Yang, F., Li, J., Sang, H., & Luo, J. (2020). Improved Jaya Algorithm for Flexible Job Shop Rescheduling Problem. *IEEE Access*, *8*, 86915–86922. DOI: 10.1109/ACCESS.2020.2992478

Gao, K., Yang, F., Zhou, M., Pan, Q., & Suganthan, P.N. (2019). Flexible job-shop rescheduling for new job insertion by using discrete Jaya algorithm. *IEEE Transactions on Cybernetics*, *49*(5), 1944–1955. DOI: 10.1109/TCYB.2018.2817240

Hwangbo, S., Liu, J.J., Ryu, J.H., Lee, H.J., & Na, J. (2024). Production rescheduling via explorative reinforcement learning while considering nervousness. *Computers and Chemical Engineering*, *186*(March), 108700. DOI: 10.1016/j.compchemeng.2024.108700

Kardos, C., Laflamme, C., Gallina, V., & Sihn, W. (2020). Dynamic scheduling in a job-shop production system with reinforcement learning. *Procedia CIRP*, *97*(March), 104–109. DOI: 10.1016/j.procir.2020.05.210

Kim, Y.I., & Kim, H.J. (2021). Rescheduling of unrelated parallel machines with job-dependent setup times under forecasted machine breakdown. *International Journal of Production Research*, *59*(17), 5236–5258. DOI: 10.1080/00207543.2020.1775910

Ku, W.Y., & Beck, J.C. (2016). Mixed Integer Programming models for job shop scheduling: A computational analysis. *Computers and Operations Research*, *73*, 165–173. DOI: 10.1016/j.cor.2016.04.006

Li, H., Gajpal, Y., & R. Bector, C. (2017). A survey of due-date related single-machine with two-agent scheduling problem. *Journal of Industrial & Management Optimization*, *13*(5), 1–19. DOI: 10.3934/jimo.2019005

Liang, Z., Zhong, P., Zhang, C., Yang, W., Xiong, W., Yang, S., & Meng, J. (2023). A genetic algorithm-based approach for flexible job shop rescheduling problem with machine failure interference. *Eksploatacja i Niezawodnosc*, *25*(4), 0–3. DOI: 10.17531/ein/171784

Mahmoodjanloo, M., Tavakkoli-Moghaddama, R., Baboli, A., & Bozorgi-Amiri, A. (2022). Distributed job-shop rescheduling problem considering reconfigurability of machines: a self-adaptive hybrid equilibrium optimiser. *International Journal of Production Research*, *60*(16), 4973–4994. DOI: 10.1080/00207543.2021.1946193

Meyer, G.G., Hans Wortmann, J.C., & Szirbik, N.B. (2011). Production monitoring and control with intelligent products. *International Journal of Production Research*, *49*(5), 1303–1317. DOI: 10.1080/00207543.2010.518742

Muhuri, P.K., & Biswas, S.K. (2020). Bayesian optimization algorithm for multi-objective scheduling of time and precedence constrained tasks in heterogeneous multiprocessor systems. *Applied Soft Computing Journal*, *92*, 106274. DOI: 10.1016/j.asoc.2020.106274

Pannequin, R., & Thomas, A. (2012). Another interpretation of stigmergy for product-driven systems architecture. *Journal of Intelligent Manufacturing*, *23*(6), 2587–2599. DOI: 10.1007/s10845-011-0588-3

Pérez-Salazar, M. del R., Aguilar-Lasserre, A.A., Cedillo-Campo, M.G., Posada-Gómez, R., del Moral-Argumedo, M.J., & Hernández-González, J.C. (2019). An agent-based model driven decision support system for reactive aggregate production scheduling in the Green Coffee Supply Chain. *Applied Sciences (Switzerland)*, *9*(22). DOI: 10.3390/app9224903

Rasheed, M.B., Javaid, N., Arshad Malik, M.S., Asif, M., Hanif, M.K., & Chaudary, M.H. (2019). Intelligent Multi-Agent Based Multilayered Control System for Opportunistic Load Scheduling in Smart Buildings. *IEEE Access*, *7*, 23990–24006. DOI: 10.1109/ACCESS.2019.2900049

Roesch, M., Linder, C., Bruckdorfer, C., Hohmann, A., & Reinhart, G. (2019). Industrial load management using multi-agent reinforcement learning for rescheduling. *Proceedings - 2019 2nd International Conference on Artificial Intelligence for Industries, AI4I 2019*, 99–102. DOI: 10.1109/AI4I46381.2019.00033

Sáez, P., Herrera, C., Booth, C., Belmokhtar-Berraf, S., & Parada, V. (2023). A product-driven system with an evolutionary algorithm to increase flexibility in planning a job shop. *PLoS ONE*, *18*(2 February), 1–12. DOI: 10.1371/journal.pone.0281807

Salido, M.A., Escamilla, J., Barber, F., & Giret, A. (2017). Rescheduling in job-shop problems for sustainable manufacturing systems. *Journal of Cleaner Production*, *162*, S121–S132. DOI: 10.1016/j.jclepro.2016.11.002

Sutton, R., & Barto, A. (1998). Reinforcement learning: An Introduction. In *Cambridge: MIT press.*

Upasani, A., & Uzsoy, R. (2008). Integrating a decomposition procedure with problem reduction for factory scheduling with disruptions: A simulation study. *International Journal of Production Research*, *46*(21), 5883–5905. DOI: 10.1080/00207540601156215

Wu, C.-C., Chen, J.-Y., Lin, W.-C., Lai, K., Bai, D., & Lai, S.-Y. (2019). A two-stage three-machine assembly scheduling flowshop problem with both two-agent and learning phenomenon. *Computers and Industrial Engineering*, *130*, 485–499. DOI: 10.1016/j.cie.2019.02.047

Wu, X., & Yan, X. (2023). A spatial pyramid pooling-based deep reinforcement learning model for dynamic job-shop scheduling problem. *Computers and Operations Research*, *160*(September 2022), 106401. DOI: 10.1016/j.cor.2023.106401

Yamamoto, M., & Nof, S.Y. (1985). Scheduling rescheduling in the manufacturing operating system environment. *International Journal of Production Research*, *23*(4), 705–722. DOI: 10.1080/00207548508904739

Yang, S., & Xu, Z. (2022). Intelligent scheduling and reconfiguration via deep reinforcement learning in smart manufacturing. *International Journal of Production Research*, *60*(16), 4936–4953. DOI: 10.1080/00207543.2021.1943037

Zhang, S., Xiang, L., Bowen, Z., & Shouyang, W. (2020). Multi-objective optimisation in flexible assembly job shop scheduling using a distributed ant colony system. *European Journal of Operational Research*, *283*(2), 441–460. DOI: 10.1016/j.ejor.2019.11.016

Zhou, T., Tang, D., Zhu, H., & Zhang, Z. (2021). Multi-agent reinforcement learning for online scheduling in smart factories. *Robotics and Computer-Integrated Manufacturing*, *72*(June), 102202. DOI: 10.1016/j.rcim.2021.102202