Assessing maintenance planning and scheduling using Deep Reinforcement Learning

Maria Grazia Marchesano [a)], Guido Guizzi [a)], Giuseppe Converso [a)] and Liberatina Carmela Santillo [a)]

a. Università degli Studi di Napoli "Federico II", Dipartimento di Ingegneria Chimica, dei Materiali e della Produzione Industriale, P.le Tecchio, 80, 80125- Napoli- Italy

(mariagrazia.marchesano@unina.it, g.guizzi@unina.it, giuseppe.converso@unina.it, santillo@unina.it)

Abstract: Maintenance scheduling is critical in many industries, and recent advances in Deep Reinforcement Learning (DRL) have shown that it can optimise scheduling decisions in complex and dynamic contexts. Traditional methods of maintenance scheduling frequently confront obstacles, making DRL an appealing alternative. This study presents a novel approach for autonomously determining optimal maintenance scheduling decisions in production systems that blends a simulation-based model with a DRL agent. The learning agent makes intelligent judgements based on the chance of failure and machine availability through trial and error. The setup of the DRL setting, particularly the reward function, has a considerable impact on the approach's performance. The proposed hybrid simulation-based and DRL methodology outperforms existing heuristic methods in rigorous evaluation, demonstrating its promise for efficient and effective maintenance planning and scheduling. This work sets the way for better system reliability and productivity in companies that rely on complex systems.

Keywords: Industry 4.0, Maintenance, Deep Reinforcement Learning, PPO.

I. INTRODUCTION

Maintenance scheduling plays a critical role in numerous industries, as it directly impacts the operational efficiency, reliability, and cost-effectiveness of equipment and systems (Rai *et al.*, 2021)(Grassi *et al.*, 2023). Effective maintenance scheduling ensures that maintenance activities are strategically planned and executed at the right time, balancing the need for equipment availability with the necessity of minimizing downtime and disruptions (Mao *et al.*, 2021)(Converso *et al.*, 2023). Deep reinforcement learning (DRL) finds application in the maintenance phase of manufacturing systems (Li *et al.*, 2023) (Marchesano *et al.*, 2021) (Marchesano, Guizzi, *et al.*, 2022). General maintenance activities can be classified into reactive, preventive, and predictive maintenance based on the timing of maintenance (Paz and Leigh, 1994). DRL helps improve productivity, flexibility, and adaptability and reduces human labour in these activities (Waubert de Puiseau, Meyes and Meisen, 2022).

In reactive maintenance, uncertainty such as the type and condition of returned products for repair exists (Nunes, Santos and Rocha, 2023). To address the resulting high volatility in reactive maintenance, scholars (Wurster *et al.*, 2022)(Mao *et al.*, 2022) have used Petri-Net to transform disassembly sequence planning into DRL-solvable Markov Decision Processes (MDPs).

DRL has also been applied in preventive maintenance policies design and optimization to improve production system performance (Su *et al.*, 2022). With the integration of the Internet of Things (IoT), real-time production data can be collected and fed back to the DRL system, allowing for continuous maintenance policy optimization and better decision-making (Usuga Cadavid *et al.*, 2020).

In addition to production systems, DRL-based maintenance of tools plays a significant role in ensuring machining quality and improving the productivity of automatic systems (Valet *et al.*, 2022). For instance, DRL algorithms have been combined with convolutional neural networks (CNNs) and improved actor-critic algorithms for bearing and tool fault recognition while transfer learning-based DRL methods have been integrated into Long Short-Term Memory (LSTM) networks to predict tool wear and remaining useful life (Wang *et al.*, 2020) .

In their literature review, Panzer et al. (Panzer and Bender, 2021) found that DRL was more effective than other maintenance strategies in reducing average maintenance costs for multi-component systems. Compared to a run-to-failure strategy, the DRL algorithm reduced maintenance costs by approximately 20%, 7% for an age-dependent strategy, and 5% for an opportunistic maintenance strategy. The approach considers interdependencies between multiple components with competing failure probabilities and avoids the static and ineffective maintenance limits of conventional methods in large-scale systems. Furthermore, DRL was able to reduce maintenance costs without requiring experience-based or predefined thresholds. Finally, in their paper, Panzer et al. (Panzer, Bender and Gronau, 2022) proposed a list of additional maintenance-related publications of DRL in recent years.

Overall, recent publications on DRL in maintenance-related fields have shown the potential for the approach to be more effective and efficient than conventional methods (Kosanoglu, Atmis and Turan, 2022).

The objective of this study is to present a novel approach that integrates a simulation tool and a DRL algorithm for the purpose of effective scheduling and planning of maintenance events in a production line flow shop. This integrated framework combines the advantages of simulation techniques with the intelligent decision-making capabilities of DRL, aiming to optimize the maintenance process and enhance overall productivity. By leveraging this comprehensive approach (Marchesano, Staiano, *et al.*, 2022), the proposed methodology aims to improve the efficiency and effectiveness of maintenance operations in a production line flow shop environment.

II. Proposed Approach

The proposed methodology introduces an integrated simulation tool and Deep Reinforcement Learning (DRL) algorithm to facilitate efficient scheduling and planning of maintenance events in a production line Flow Shop setting. The integrated simulation tool provides a virtual environment that accurately replicates the production line Flow Shop, allowing for detailed modeling and simulation of machine operations, job flows, and maintenance events (Huang, Chang and Arinez, 2020). By capturing the dynamic nature and intricate complexities of the system, the simulation tool enables the evaluation of diverse maintenance strategies and their impact on overall performance (Akl *et al.*, 2022).

At the heart of the proposed approach lies the DRL algorithm, which serves as the intelligent decision-making component. The DRL algorithm operates through the interaction between an agent and the simulated environment, learning optimal policies through iterative trial and error. By considering factors such as machine failure probabilities, job priorities, and scheduling constraints, the agent aims to maximize cumulative rewards while making informed decisions regarding maintenance scheduling within the production line flow shop.

The integrated simulation tool and DRL algorithm offer several notable advantages. Firstly, the virtual simulation environment allows for the assessment of different maintenance strategies without the need for disrupting actual production operations. This mitigates potential risks and costs associated with real-world experimentation. Secondly, the methodology provides a platform for evaluating the effectiveness of various maintenance scheduling policies under diverse scenarios, facilitating the identification of optimal strategies for enhancing production efficiency and minimizing downtime. Lastly, the integration of simulation and DRL enables the development of adaptive maintenance planning systems that can dynamically adjust to changes in machine failure patterns, job priorities, and production demands.

A. *Proximal Policy Optimization*

The Proximal Policy Optimization (PPO) algorithm, introduced by Schulman et al. (Schulman *et al.*, 2017), is a widely used reinforcement learning (RL) algorithm. It belongs to the Policy Gradient family of methods and follows a two-step process involving data collection and optimization of a "surrogate" objective function using stochastic gradient ascent. Unlike model-based RL algorithms, PPO learns the policy directly without constructing an explicit model of the environment. To ensure learning stability, the algorithm incorporates Trust Region Optimization, which limits the size of policy updates. PPO also employs a Clipping Objective Function to restrict the magnitude of policy updates, preventing instability during training.

The main steps of the PPO algorithm can be summarized as follows:

1. Collect trajectories: The agent interacts with the environment to gather trajectories,

which consist of sequences of states, actions, and rewards.

2. Compute advantages: The value network is utilized to calculate the advantage of each state-action pair. The advantage represents the difference between the expected reward and the value of the current state.
3. Update policy: The policy network is updated using the PPO loss function, which strikes a balance between exploring new actions and exploiting known actions.
4. Update value network: The value network is updated by minimizing the mean squared error between the expected reward and the value of the current state.
5. Repeat: Steps 1 to 4 are iterated until the agent has learned a policy that maximizes the reward signal.

By following these steps, the PPO algorithm iteratively improves the policy by adjusting the parameters of the policy network and the value network based on the collected trajectories and their associated advantages.

III. Problem formulation

The proposed methodology entails employing the AnyLogic simulation software to model the production line, thereby facilitating the simulation of line operations as well as corrective and preventive maintenance events (Figure 1). This software provides the capability to configure essential parameters for the Reinforcement Learning Experiment. Subsequently, the configured model will be exported and implemented in Python, leveraging the ALPyne library. Following the simulation setup, an RL algorithm (PPO) will be developed in Python to train an agent on a single-machine system, aiming to derive an optimal policy. Subsequently, this policy will be evaluated and tested on a Flow-Shop system consisting of 5 machines. The Anylogic environment facilitates the creation of a model specifically designed for addressing the Flow Shop Scheduling Problem (FSSP), enabling experimentation with various configurations of the problem. Consequently, the Anylogic model serves as a valuable framework for investigating and analysing different Flow Shop scenarios.

The hypothesis of the problem are:

- Number of jobs to be processed: 50;
- Number machines: Single machine and 5-machines Flow Shop;
- Processing Times: triangular distribution (min,max,mode) (20,100,50);
- Corrective Maintenance Time: 51 min;
- Preventive Maintenance Time: 17 min;
- Weibull scale parameter α: 450;
- Weibull scale parameter β: 1.5.

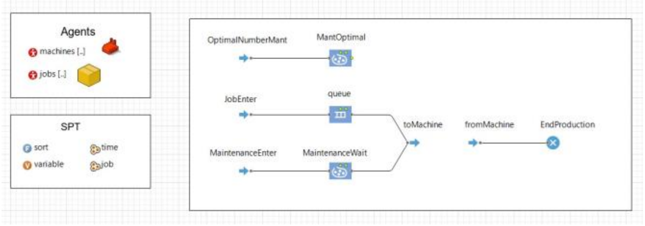


Figure 1 The simulation model.

In these cases, jobs are processed using the Shortest Processing Time (SPT) rule.

In the following paragraph, we will elucidate the formulation of the DRL approach. We will begin by providing an overview of the selected software used for implementing the PPO algorithm. Subsequently, we present the system's characteristics in terms of state observations, available actions, and the reward structure.

IV. DRL Configuration

Regarding the Reinforcement Learning aspect, the problem formulation and setup were implemented within Anylogic through a dedicated section within the software (*RLExperiment*). This section allows for the definition of observation, action, and configuration parameters. The resulting experiment will then be exported and implemented in the ALPyne library, ensuring seamless integration between Anylogic and the subsequent reinforcement learning pipeline.

ALPyne serves as a Python library specifically developed to establish a connection between AnyLogic and Python. Its primary objective is to facilitate the interactive execution of RL models exported from AnyLogic. As AnyLogic does not possess native RL algorithms, ALPyne acts as a vital tool for running RL experiments seamlessly. Consequently, the execution of RL experiments within the AnyLogic environment is not directly supported, necessitating the use of a compatible platform like ALPyne for conducting RL-related tasks. The initial phase of utilizing ALPyne involves configuring the AnyLogic model by providing the necessary information in the designated sections pertaining to Configuration, Observation, Action, and stopping conditions within the Reinforcement Learning experiment module. Additionally, the *takeAction* method should be invoked when the agent requires making decisions during the simulation. Once the model is appropriately configured, it needs to be exported to initiate the training phase. It is important to note that the training phase takes place on a distinct platform independent of AnyLogic. Following the model export, the subsequent step entails setting up the training phase using ALPyne. This process involves importing essential libraries and configuring the observation and action spaces. The libraries that need to be imported include *ALPyneClient*, *ModelRun*, *Observation*, *Action*, and *BaseALPyneEnv*. *ALPyneClient* is responsible for initializing the ALPyne application, while *ModelRun* represents an individual simulation run within the *ALPyneClient* object. *Observation* is a read-only object that represents an observation obtained from the simulator, and Action is an object that represents an action to be sent to the simulator. Additionally, the *BaseALPyneEnv* class is imported to simplify the usage of ALPyne in conjunction with Gym environments.

When integrating ALPyne with Gym environments, it is essential to define the observation and action spaces, taking into account the appropriate data types that correspond to the configurations utilized in the AnyLogic model. Subsequently, the code needs to handle the conversion of Observation objects to align with the data type specified by the observation space. Similarly, the action received from the RL library should be transformed into an Action object. Furthermore, the code should calculate the numerical reward based on the observation recorded after executing an action. In this particular instance, the Observation space is represented as a Box type, which denotes a continuous n-dimensional space. Specifically, it encompasses a 5-dimensional space containing the following variables:

- Failure Probability (FP);
- The percentage of completed jobs at time t;
- The ratio of the mean processing time to the sum of the processing time at time t and the time in corrective maintenance(C_{Rt});
- The ratio of the sum of the processing time at time t and the time in preventive maintenance to the mean processing time (p_{Rt});
- The ratio of the total processing time at time t to the sum of the total processing time, corrective time and preventive time at time t (T_{Rt}).

While, the Action space is of type Discrete and has two possible values: 0 or 1, which correspond to no maintenance or maintenance, respectively.

Finally, the reward function is defined. Specifically, the method calculates the reward for the agent, based on the observation received from the simulation, as:

$reward_t = FP \cdot C_{Rt} \cdot p_{Rt} \cdot T_{Rt}$

The objective of the learning agent is to strike a balance between minimizing the Makespan (the total time required for all jobs to be completed) and the number of maintenance activities, including both corrective and preventive tasks. To maximize the product's performance, which is the primary objective of the learning agent, the individual ratios C_{Rt}, p_{Rt}, and T_{Rt} need to be optimized. Maximizing C_{Rt} involves minimizing the time required for a single corrective maintenance task, which corresponds to the occurrence of failures. Conversely, maximizing p_{Rt} requires maximizing the duration of a single preventive maintenance task. Although this may initially seem contradictory, it is counterbalanced by the maximization of T_{Rt}, which entails minimizing both the total time spent on corrective maintenance and the total time spent on preventive maintenance simultaneously. The significance of this optimization is further influenced by the failure probability (FP).

V. Performance Comparison

In this section, we present the obtained results that validate the policy implemented. The assessment of policy effectiveness was conducted based on the number of preventive maintenance actions, the occurrence of corrective maintenance (i.e., breakdowns), and the overall Makespan. The Makespan refers to the total duration or time required to complete a set of tasks within a given process. It is a widely used performance metric in scheduling, optimization, and production planning to evaluate process or system efficiency and effectiveness.

Additionally, the number of preventive maintenance actions executed by the learning agent was compared with the optimal maintenance number calculated using the empirical formula proposed by Branda et al. (Branda *et al.*, 2021a, 2021b), as discussed in their published work.

Subsequently, the resulting policy was tested on the same single-machine system, yielding notable outcomes in terms of achieving a balanced approach between maintenance actions and breakdown occurrences. Specifically, across 20 test replications with randomized seeds, the simulations exhibited an average of 6.3 maintenance tasks and 9.05 breakdowns, with a total Makespan of approximately 3312.5 minutes. It is worth noting that the average number of maintenance tasks performed by the learning agent was lower than the optimal number determined by the empirical formula (resulting 7 with the hypothesis of this study) proposed by Branda et al. (Branda *et al.*, 2021b).

Figure 2 displays how the maintenance tasks and the breakdowns are distributed over time.

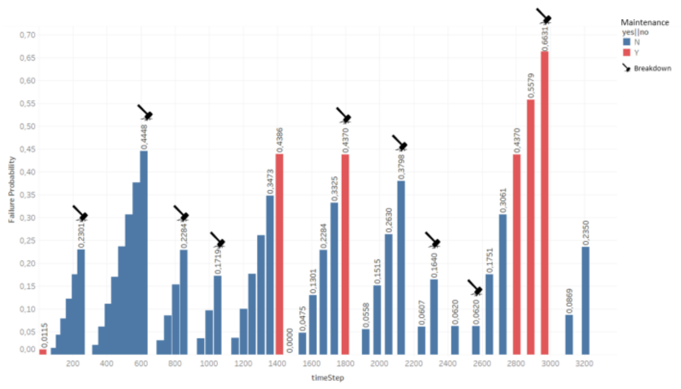
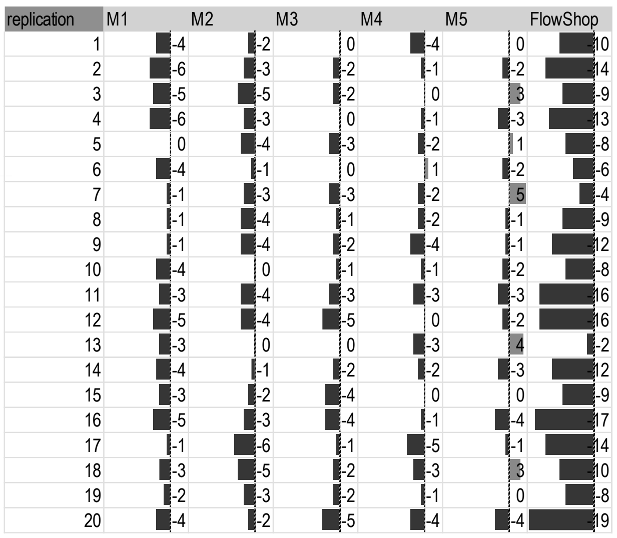


Figure 2. The system behaviour in terms of the number of maintenance tasks and breakdowns depends on the failure probability across the time steps.

The action is executed once a job completes its operation on the machine. In Figure 2, the action is represented by a hammer symbol when a failure occurs, and it is visually highlighted in red when the DRL system determines a preventive maintenance activity. Since the job scheduling in this system follows the shortest processing time (SPT) rule, a higher number of maintenance tasks are frequently performed at a higher failure probability rate. These maintenance tasks tend to accumulate towards the end of the simulation time period. The breakdown events are relatively evenly distributed across the time steps, indicating an area for potential improvement in the system. However, it is worth noting that this maintenance schedule does not significantly affect the Makespan, the total time required to complete all jobs. As a result, this scheduling strategy ensures a balanced distribution of maintenance activities throughout the various process stages, thereby minimizing their impact on production time and ensuring optimal efficiency. Subsequently, the effectiveness of the policy generated based on the best reward was evaluated through simulation. In this evaluation, the policy was tested on a configuration consisting of five identical machines, characterized by identical Weibull parameters.

TABLE I. COMPARISON OF THE RESULTS OF THE MEAN VALUES

System	**Mean Makespan**	**Number of Breakdowns**	**Number of maintenance tasks**
Single Machine	3312,5	9,05	6,3
Flow Shop	3901,15	43,75	24,8



replication	M1	M2	M3	M4	M5	FlowShop
1	-4	-2	0	-4	0	-10
2	-6	-3	-2	-1	-2	-14
3	-5	-5	-2	0	3	-9
4	-6	-3	0	-1	-3	-13
5	0	-4	-3	-2	1	-8
6	-4	-1	0	1	-2	-6
7	-1	-3	-3	-2	5	-4
8	-1	-4	-1	-2	-1	-9
9	-1	-4	-2	-4	-1	-12
10	-4	0	-1	-1	-2	-8
11	-3	-4	-3	-3	-3	-16
12	-5	-4	-5	0	-2	-16
13	-3	0	0	-3	4	-2
14	-4	-1	-2	-2	-3	-12
15	-3	-2	-4	0	0	-9
16	-5	-3	-4	-1	-4	-17
17	-1	-6	-1	-5	-1	-14
18	-3	-5	-2	-3	3	-10
19	-2	-3	-2	-1	0	-8
20	-4	-2	-5	-4	-4	-19

Figure 3. The deviation between the number of maintenance tasks done and the optimal value derived from the paper's formula (Branda *et al.*, 2021b).

As in the previous experiments, a total of 20 replications were conducted to show how the metrics considered varied. Specifically, the optimal number of maintenance actions for each machine and the optimal number in total, considering the whole system, were compared with the actual number of maintenance actions, as well as the number of breakdowns and the Makespan.

The results of all 20 replications consistently showed that the total number of actual maintenance actions was lower than the total optimal number (Figure 3). Furthermore, the total Makespan did not increase significantly compared to the single-machine case (as shown in Table I).

VI. Conclusion

The present study adopts an innovative approach to address the challenge of scheduling maintenance tasks in a Flow Shop environment. Traditional tools commonly used for maintenance scheduling have proven to be inadequate in addressing this complex decision problem, necessitating the exploration of new methodologies.

Deep reinforcement learning (DRL) emerges as a promising avenue for optimizing maintenance planning and scheduling in multi-machine manufacturing systems. This study showcases the obtained findings, with a particular focus on the ability to schedule a reduced number of maintenance operations compared to the ideal solution. These results underscore the effectiveness and potential benefits of the proposed method.

Future investigations will entail a more comprehensive experimental plan aimed at enhancing our understanding and implementation of maintenance planning and scheduling in multi-machine production systems. The research project aims to explore the adaptability and efficacy of the proposed approach in more intricate production contexts, incorporating a multi-product production system and accounting for different levels of stochasticity within the model. Additionally, resource allocation to maintenance activities will be considered, encompassing the identification of manpower, equipment, and other necessary resources. By incorporating resource constraints, the study endeavors to establish a more practical and realistic maintenance planning strategy. Through these additional experiments, the project aims to test and refine the proposed integrated approach for maintenance planning and scheduling in multi-machine production systems. The outcomes of these trials will contribute to advancing our comprehension and application of DRL learning techniques in real-world production scenarios, ultimately enhancing operational efficiency and productivity.

VII. References

[1] Akl, A. M. *et al.* (2022) 'A Joint Optimization of Strategic Workforce Planning and Preventive Maintenance Scheduling: A Simulation–Optimization Approach', *Reliability Engineering and System Safety*, 219(November 2021), p. 108175. doi: 10.1016/j.ress.2021.108175.

[2] Branda, A. *et al.* (2021a) 'Dataset of metaheuristics for the flow shop scheduling problem with maintenance activities integrated', *Data in Brief*, 36, p. 106985. doi: 10.1016/j.dib.2021.106985.

[3] Branda, A. *et al.* (2021b) 'Metaheuristics for the flow shop scheduling problem with maintenance activities integrated', *Computers and Industrial Engineering*, 151(November 2020), p. 106989. doi: 10.1016/j.cie.2020.106989.

[4] Converso, G. *et al.* (2023) 'Predicting Failure Probability in Industry 4.0 Production Systems: A Workload-Based Prognostic Model for Maintenance Planning', *Applied Sciences (Switzerland)*, 13(3). doi: 10.3390/app13031938.

[5] Grassi, A. *et al.* (2023) 'A Genetic-Algorithm-Based Approach for Optimizing Tool Utilization and Makespan in FMS Scheduling', *Journal of Manufacturing and Materials Processing*, 7(2). doi: 10.3390/jmmp7020075.

[6] Huang, J., Chang, Q. and Arinez, J. (2020) 'Deep reinforcement learning based preventive maintenance policy for serial production lines', *Expert Systems with Applications*, 160, p. 113701. doi: 10.1016/j.eswa.2020.113701.

[7] Kosanoglu, F., Atmis, M. and Turan, H. H. (2022) 'A deep reinforcement learning assisted simulated annealing algorithm for a maintenance planning problem', *Annals of Operations Research*. doi: 10.1007/s10479-022-04612-8.

[8] Li, C. *et al.* (2023) 'Deep reinforcement learning in smart manufacturing: A review and prospects', *CIRP Journal of Manufacturing Science and Technology*, 40, pp. 75–101. doi: 10.1016/j.cirpj.2022.11.003.

[9] Mao, J. Y. *et al.* (2022) 'A hash map-based memetic algorithm for the distributed permutation flowshop scheduling problem with preventive maintenance to minimize total flowtime', *Knowledge-Based Systems*, 242, p. 108413. doi: 10.1016/j.knosys.2022.108413.

[10] Mao, J. yang *et al.* (2021) 'An effective multi-start iterated greedy algorithm to minimize makespan for the distributed permutation flowshop scheduling problem with preventive maintenance', *Expert Systems with Applications*, 169(December 2020), p. 114495. doi: 10.1016/j.eswa.2020.114495.

[11] Marchesano, M. G. *et al.* (2021) 'A deep reinforcement learning approach for the throughput control of a flow-shop production system', *IFAC-PapersOnLine*, 54(1), pp. 61–66. doi: 10.1016/j.ifacol.2021.08.006.

[12] Marchesano, M. G., Staiano, L., *et al.* (2022) 'Deep Reinforcement Learning Approach for Maintenance Planning in a Flow-Shop Scheduling Problem', in *Frontiers in Artificial Intelligence and Applications*.

[13] Marchesano, M. G., Guizzi, G., *et al.* (2022) 'Dynamic scheduling of a due date constrained flow shop with Deep Reinforcement Learning', *IFAC-PapersOnLine*, 55(10), pp. 2932–2937. doi: 10.1016/j.ifacol.2022.10.177.

[14] Nunes, P., Santos, J. and Rocha, E. (2023) 'Challenges in predictive maintenance – A review', *CIRP Journal of Manufacturing Science and Technology*, 40, pp. 53–67. doi: 10.1016/j.cirpj.2022.11.004.

[15] Panzer, M. and Bender, B. (2021) 'Deep reinforcement learning in production systems: a systematic literature review', *International Journal of Production Research*. doi: 10.1080/00207543.2021.1973138.

[16] Panzer, M., Bender, B. and Gronau, N. (2022) 'Neural agent-based production planning and control: An architectural review', *Journal of Manufacturing Systems*, 65(November), pp. 743–766. doi: 10.1016/j.jmsy.2022.10.019.

[17] Paz, N. M. and Leigh, W. (1994) 'Maintenance Scheduling: Issues, Results and Research Needs', *International Journal of Operations & Production Management*, 14(8), pp. 47–69. doi: 10.1108/01443579410067135.

[18] Rai, R. *et al.* (2021) 'Machine learning in manufacturing and industry 4.0 applications', *International Journal of Production Research*, 59(16), pp. 4773–4778. doi: 10.1080/00207543.2021.1956675.

[19] Schulman, J. *et al.* (2017) 'Proximal Policy Optimization Algorithms', pp. 1–12. Available at: http://arxiv.org/abs/1707.06347.

[20] Su, J. *et al.* (2022) 'Deep multi-agent reinforcement learning for multi-level preventive maintenance in manufacturing systems[Formula presented]', *Expert Systems with Applications*, 192(November 2021), p. 116323. doi: 10.1016/j.eswa.2021.116323.

[21] Usuga Cadavid, J. P. *et al.* (2020) 'Machine learning

applied in production planning and control: a state-of-the-art in the era of industry 4.0', *Journal of Intelligent Manufacturing*. Springer, pp. 1531–1558. doi: 10.1007/s10845-019-01531-7.

[22] Valet, A. *et al.* (2022) 'Opportunistic maintenance scheduling with deep reinforcement learning', *Journal of Manufacturing Systems*, 64(March), pp. 518–534. doi: 10.1016/j.jmsy.2022.07.016.

[23] Wang, Z. *et al.* (2020) 'Performance degradation assessment of rolling bearing based on convolutional neural network and deep long-short term memory network', *International Journal of Production Research*, 58(13), pp. 3931–3943. doi: 10.1080/00207543.2019.1636325.

[24] Waubert de Puiseau, C., Meyes, R. and Meisen, T. (2022) 'On reliability of reinforcement learning based production scheduling systems: a comparative survey', *Journal of Intelligent Manufacturing*, 33(4), pp. 911–927. doi: 10.1007/s10845-022-01915-2.

[25] Wurster, M. *et al.* (2022) 'Modelling and condition-based control of a flexible and hybrid disassembly system with manual and autonomous workstations using reinforcement learning', *Journal of Intelligent Manufacturing*, 33(2), pp. 575–591. doi: 10.1007/s10845-021-01863-3.