Robotics and Computer–Integrated Manufacturing 78 (2022) 102406

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim

Full length article

Multi-agent deep reinforcement learning based Predictive Maintenance on parallel machines

Marcelo Luis Ruiz Rodríguez [a,*], Sylvain Kubler [a], Andrea de Giorgio [a], Maxime Cordy [a], Jérémy Robert [b], Yves Le Traon [a]

[a] SnT, University of Luxembourg, 6 Rue Richard Coudenhove-Kalergi, L-1359 Luxembourg, Luxembourg
[b] Cebi Luxembourg S.A., 30 rue J.F. Kennedy, L-7327 Steinsel, Luxembourg

ARTICLE INFO

Keywords:
Predictive Maintenance
Scheduling
Reinforcement learning
Multi-agent systems
Industry 4.0

ABSTRACT

In the context of Industry 4.0, companies understand the advantages of performing Predictive Maintenance (PdM). However, when moving towards PdM, several considerations must be carefully examined. First, they need to have a sufficient number of production machines and relative fault data to generate maintenance predictions. Second, they need to adopt the right maintenance approach, which, ideally, should self-adapt to the machinery, priorities of the organization, technician skills, but also to be able to deal with uncertainty. Reinforcement learning (RL) is envisioned as a key technique in this regard due to its inherent ability to learn by interacting through trials and errors, but very few RL-based maintenance frameworks have been proposed so far in the literature, or are limited in several respects. This paper proposes a new multi-agent approach that learns a maintenance policy performed by technicians, under the uncertainty of multiple machine failures. This approach comprises RL agents that partially observe the state of each machine to coordinate the decision-making in maintenance scheduling, resulting in the dynamic assignment of maintenance tasks to technicians (with different skills) over a set of machines. Experimental evaluation shows that our RL-based maintenance policy outperforms traditional maintenance policies (incl., corrective and preventive ones) in terms of failure prevention and downtime, improving by ≈ 75% the overall performance.

1. Introduction

The industry has benefited from technologies such as cyber–physical systems, internet of things, big data analytics, or still Artificial Intelligence (AI) [1], with the overall aim of increasing profitability, production, capacity, quality, employee safety, and decreasing costs [2]. Such technologies are widely used when it comes to maintenance operations, as it can represent in some cases up to 60% of the total turnover [3,4].

An increasing number of Predictive Maintenance (PdM) systems are emerging in all sectors of the industry [5]. Some market studies report that PdM can reduce time required to plan maintenance by 50%, increase equipment uptime by 20% and reduce costs by 10% [6]. A PdM system can be seen as a three stage-process, as depicted in Fig. 1:

1. *Remote monitoring:* sensors and cameras are deployed on production lines to capture real-time events and communicate these data to on site or cloud systems;
2. *Failure Prediction:* analytics tools with predictive (ML) capabilities analyze the collected data to determine when a system (component, equipment, process) is likely to fail;
3. *Task Scheduling:* based on the computed predictions, and expert knowledge, decision support systems are designed, which often include the optimization of maintenance task scheduling and operator assignment.

Each of these stages is the subject of significant research efforts, for example to address interoperability issues in stage (1) [7], to make AI (Machine Learning — ML) models more robust, accurate and transparent for predicting failures in stage (2) [8,9], or still to improve state-of-the-art maintenance scheduling strategies [10] and better model expert knowledge in the decision process [11,12].

As emphasized in Fig. 1, although stage (1) (remote monitoring) is a prerequisite for PdM, the main goal of any PdM framework is to predict and optimally schedule the maintenance interventions/tasks.

[*] Corresponding author.

E-mail addresses: marcelo.ruiz@uni.lu (M.L. Ruiz Rodríguez), sylvain.kubler@uni.lu (S. Kubler), andrea@degiorgio.info (A. de Giorgio), maxime.cordy@uni.lu (M. Cordy), jeremy.robert@cebi.com (J. Robert), yves.letraon@uni.lu (Y. Le Traon).

[1] Some articles also refer to "prescriptive" maintenance when it is included in a final decision support system.

https://doi.org/10.1016/j.rcim.2022.102406
Received 23 June 2022; Accepted 26 June 2022
Available online 8 July 2022

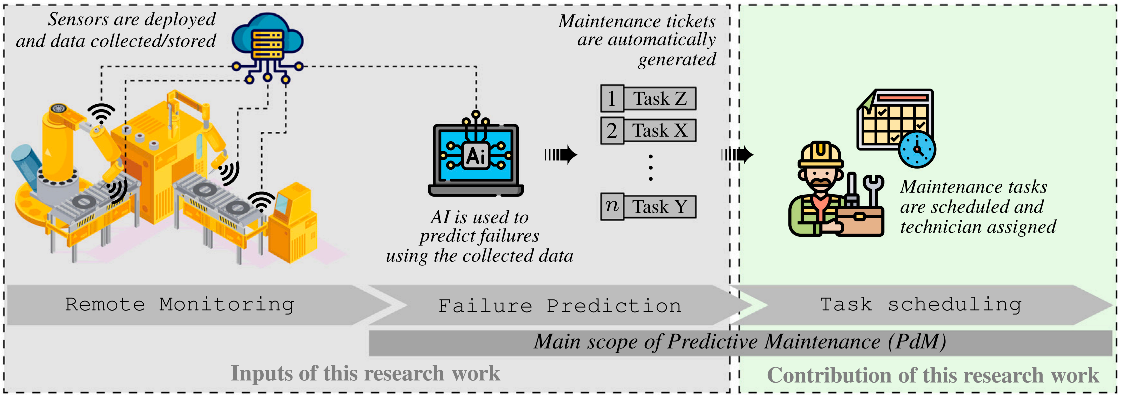

Sensors are deployed and data collected/stored
Maintenance tickets are automatically generated
1 Task Z
2 Task X
n Task Y
Ai
AI is used to predict failures using the collected data
Maintenance tasks are scheduled and technician assigned
Remote Monitoring
Failure Prediction
Task scheduling
Main scope of Predictive Maintenance (PdM)
Inputs of this research work
Contribution of this research work

Fig. 1. Key stages to move towards Predictive Maintenance (PdM).

In this respect, maintenance can be corrective (applied after failure occurrence) or preventive (applied before failure occurrence) [13]. The former is known as corrective maintenance (CM), also known as run-to-failure, while the latter can be divided into two policies known as Preventive and Predictive[1] maintenance. The most simple policy is CM, which seeks to maximize the lifetime of the components, but with the disadvantage that it can quickly becomes costly due to production interruptions caused by the long equipment downtime. Preventive Maintenance (PM), also known as time-based maintenance, is a more advanced and effective strategy, as it anticipates failures and avoids equipment breakdowns that could be costly to repair. The disadvantage of this type of maintenance is that the equipment receives maintenance that is not yet necessary, reducing the lifespan of some components and resulting in additional costs. The key to overcome the trade-off between opting for CM and/or PM is to determine the optimal maintenance timing such that the overall profit of the manufacturing system is optimized. This has led to the so-called PdM policy, also referred to as condition-based maintenance [14], which uses statistical or AI models to calculate the degradation of the equipment or the remaining useful life (RUL) of a component [15,16]. This makes possible the prediction of when and what part of the system is likely to fail in order to optimally perform maintenance decision making (incl., maintenance task scheduling, technician assignment, *etc.*).

In today's literature, several studies have proposed PdM frameworks using machine learning (ML) techniques [17]. While both supervised and unsupervised learning techniques have already been widely used in the manufacturing industry, accounting for 90–95% of all applications according to [18], especially for PdM [19], reinforcement learning (RL) has been much less studied [20]. This is unfortunate because RL provides many interesting features [21], such as learning by interacting with the environment, measuring the utility of actions that yield long-term benefits, optimizing complex sequential decisions under uncertainty, adding that RL takes advantage of the multi-agent approach, which allows for multi-objective optimization [22]. To overcome this gap in research, the present article proposes a novel maintenance policy based on RL, which aims at reducing unexpected failures and maintaining a high uptime. As emphasized in Fig. 1, our research does not progress the state-of-the-art in failure prediction (i.e., in determining failure distribution), but in dynamic maintenance task scheduling (failure distributions being used as inputs of our strategy), where the originality compared to state-of-the-art RL-based maintenance policy models lies in the fact that, to the best of our knowledge, it is the first multi-agent RL model taking into account different types of failures, along with varying maintenance times (depending on technician skills).

Section 2 discusses the current state-of-the-affairs pertaining to (dynamic) maintenance task scheduling, with a specific focus on RL-based maintenance strategies. Section 3 presents the RL model underpinning the proposed maintenance policy. Section 4 presents the experimental setup and evaluation of our system in two manufacturing scenarios. Discussion and conclusion are given in Sections 5 and 6 respectively. Note that all acronyms used in this paper are summarized in Table 1.

Table 1
Acronyms used in the present article.

Notation	Description
AC	Ant Colony
AI	Artificial Intelligence
ANSGA-III	Adaptive-reference-point-based Nondominated Sorting Genetic Algorithm
BB	Branch and Bound
BD	Benders Decomposition
BH	Black Hole
BIP	Binary Integer Programming
BOMP	Bi-Objective Mathematical Programming
CEA	Coevolutionary algorithms
CM	Corrective Maintenance
GA	Genetic Algorithm
GP	Goal Programming
GrA	Greedy Algorithm
IP	Integer Programming
LPT	Longest Processing Time
MA	Memetic Algorithm
MC	Monte Carlo
MDP	Markov Decision Process
MG	Markov Game
MILP	Mixed Integer Linear Programming
MINLP	Mixed Integer Non-Linear Programming
MIP	Mixed Integer Programming
ML	Machine Learning
MP	Multiparametric Programming
MVO	Multiverse Optimizer Algorithm
NSGA-II	Non-dominated Sorting Genetic Algorithm II
OTA	Online Task Allocation
PdM	Predictive Maintenance
PM	Preventive Maintenance
RA	Random Maintenance
RHA	Rolling Horizon Algorithm
RL	Reinforcement Learning
RUL	Remaining Useful Life
SA	Simulated Annealing
TS	Tabu Search
VNS	Variable Neighborhood Search
XAI	eXplainable Artificial Intelligence

2. Maintenance policies and strategies

Several maintenance policies and strategies have been explored over the past few years. Section 2.1 provides a quick overview of the trends followed by the scientific community in this respect, while Section 2.2 discusses more in-depth the policies/strategies that have been proposed and designed based on the RL theory.

2.1. State-of-affairs and trends

A dynamic maintenance task scheduling problem, like the standard scheduling problem, consists of assigning a set of maintenance tasks to a set of technicians, while minimizing the overall system downtime and cost, and taking into account several specific constraints such as safety and process constraints, technician skills, job duration, *etc.* [10,23,24]. This problem is known to be NP-hard [25,26]. Over the past decades, maintenance policies and strategies have been explored using different types of theories and methods. In order to understand the trend in theories/techniques used in that respect, a review of the literature has been carried out using the following search query on the Web of Science database:

```
TS = ("maintenance") AND
TS = ("scheduling" OR "schedule")
AND
TS = ("manufacturing")
```

Around 350 journal papers were collected (conference articles being excluded from our analysis). After screening titles, abstracts the content of the articles (checking their relevance and quality), 56 articles were identified and reviewed, which have been classified in Table 2 based on the year of publication and the type of method(s) used for optimization. These methods are classified into three main classes: *(i) Metaheuristics:* BH, VNS, GA, SA, NSGA-II, ANSGA-III, MA, MVO, CEA, GrA, TS, AC, LPT; *(ii) Mathematical Programming:* IP, MIP, MILP, MINLP, BIP, BD, RHA, MP, BOMP, BB, OTA, GP; and *(iii) Machine Learning (ML):* NN, RL. (all abbreviations being given in Table 1).

Table 2 brings to light that most of the state-of-the-art maintenance scheduling strategies (≈85% of the reviewed literature) are designed based on metaheuristics or mathematical programming techniques; GA being the most widely used technique. While these techniques are very powerful for solving NP-hard problems at a given point in time, they usually assume deterministic environments/problems (i.e., that all parameters and information are known *a priori* such as the number of tasks to be performed, their duration, and level of criticality, the operator availability, *etc.*) [81]. While this assumption may hold in some specific industrial processes, it does not in most real-life cases when the process involves uncertainties regarding the constraints, input values, and objective functions (e.g., due to sudden machine failures, reassignment of personnel, change of machine priority, component shortage). As metaheuristics or mathematical programming are not designed, by nature, to deal with stochastic processes, some studies investigate optimization–simulation frameworks – *also known as "Simheuristic"* – that combines metaheuristics with simulation (Monte Carlo, discrete-event, agent-based, *etc.*) to solve the stochastic problem [82–84]. However, such an approach is not optimal, suffering from some limitations as been discussed e.g. in [85]. In contrast, RL is based on the Markov Decision Process (MDP) theory, which is designed to deal with stochastic processes. Table 2 report some research studies that have adopted this approach, although they are still limited in number. Nonetheless, considering the arguments discussed above, this research work adopts an RL-based maintenance approach.

The next Section discusses existing RL-based maintenance strategies, along with their *pros* and *cons*.

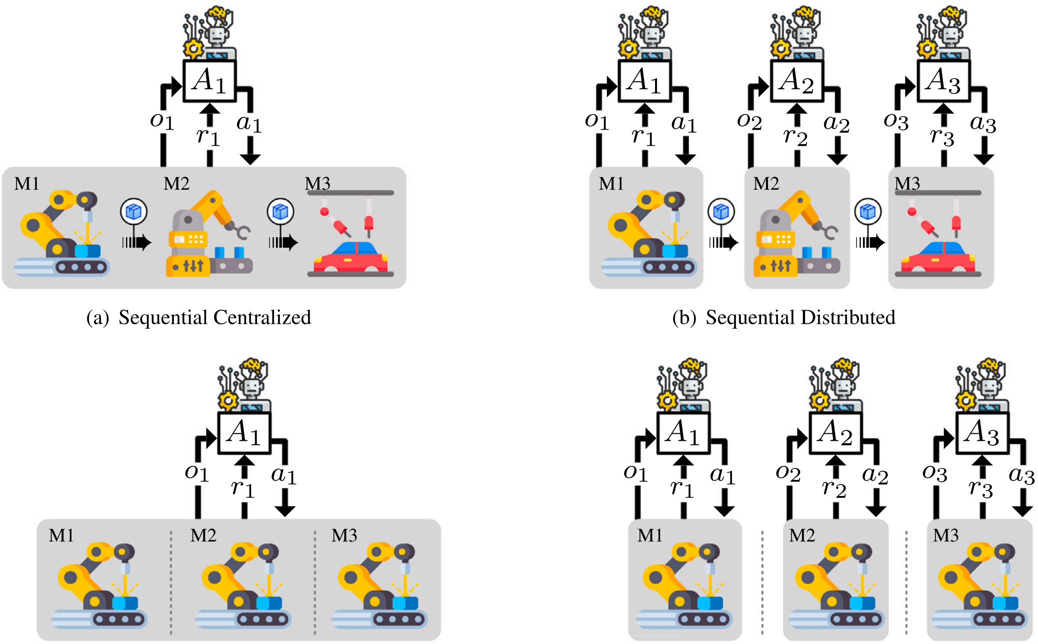
Table 2
Trend in methods used for maintenance task scheduling.

	Year	Metaheur.	Math Prog.	ML
[27]	2022	GA		
[28]	2021	BH, VNS		
[29]	2021	GA, MC		
[30]	2021	GA		
[31]	2021	GA		
[32]	2021	GrA		
[33]	2021	GA		
[34]	2021	CEA		
[35]	2021	GrA		
[36]	2021	GA		
[37]	2021		BD	
[38]	2021		RHA	
[12]	2021		MP	NN
[39]	2021			RL
[40]	2021			RL
[41]	2020	GA		
[42]	2020	MVO		
[43]	2020	TS		
[44]	2020	NSGA-II		
[45]	2020	ANSGA-III		
[46]	2020	MA	MIP	
[47]	2020	GA	MIP	
[48]	2020		BOMP	
[49]	2020		BB	
[50]	2020			RL
[51]	2020			RL
[52]	2020			NN
[53]	2019	GA		
[54]	2019	OTA		
[55]	2019	AC		
[56]	2019	NSGA-II		
[57]	2019		BIP	
[58]	2019		MILP	
[59]	2019		MINLP	
[60]	2019			RL
[61]	2019			RL
[62]	2019			RL
[10]	2018	GA		
[63]	2018	SA, GA		
[64]	2018	SA		
[65]	2017	NSGA-II		
[66]	2017	MA		
[67]	2017		MILP	
[68]	2017			RL
[69]	2016	NSGA-II		
[70]	2016	LPT	MILP	
[71]	2015	NSGA-II		
[72]	2015	SA, GA		
[73]	2015	TS		
[74]	2014	GA		
[75]	2014		MILP	
[76]	2013	SA, GA		
[77]	2013		IP	
[78]	2013		GP	
[79]	2013		MIP, RHA	
[80]	2012	GA		
Total		**34**	**16**	**10**

2.2. RL-based maintenance

As reviewed in the previous Section (see Table 2), a few studies – *8 in total* – have proposed RL-based maintenance strategies. Those strategies usually address a *sequential* or *parallel* manufacturing process. The former (sequential) refers to processes where multiple machines are connected and dependent on each other, as depicted in Figs. 2(a) and 2(b) (machine M3 requiring, as inputs, goods manufactured by M2, which itself requires goods manufactured by M1), while the latter (*parallel*) refers to processes where machines are independent of one another (i.e., goods being manufactured by machines in parallel), as depicted in Figs. 2(c) and 2(d). For each type of manufacturing process, two types of agent architectures can be applied for decision-making, namely *centralized* (decisions regarding the different components/processes are taken by a single logical agent) or *distributed* (decisions are distributed among multiple logic agents), thus leading to the four architectural design possibilities depicted in Fig. 2. In the following, the 8 articles adopting an RL-based maintenance approach are further discussed, first focusing on studies adopting a centralized approach, and then on studies adopting the distributed one.

Xu and Cao [62] propose a centralized approach to schedule the maintenance tasks of a machine tool, the objective being to improve



(a) Sequential Centralized

(b) Sequential Distributed

(c) Parallel Centralized

(d) Parallel Distributed

Fig. 2. Possible agent architectural designs when using Reinforcement Learning (RL) in industrial processes.

the energy efficiency of the production process. This policy can be applied to different states of machine degradation while taking into account productivity, product quality, and energy consumption. A control policy for manufacturing facilities linked to a finished products buffer is proposed by Xanthopoulos et al. [68], where the agent policy allows to alternate between production and maintenance actions. This approach was compared to other production policies, such as Kanbas and (s,S), along with maintenance policies such as condition-based and periodic-based. Experiments have shown that the RL-based policy succeeds in reducing the cost function, providing high-level of service while keeping inventory as low as possible. An extension of this work was proposed by Paraschos et al. [51] to include quality data that is related to the system degradation level. In addition, the authors include a recycling policy so that agents can additionally perform the action of recycling second-class goods. This system is guided to maximize profits for the different products, and to minimize maintenance and production costs. Their work was tested in different scenarios with variable production times and deterioration frequencies. Results showed that the proposed policy achieves higher profits than other policies, and is effective in managing inventory levels and preventing equipment deterioration. Huang et al. [61] have been working on the problem of PM for a serial production line with multi-stage machines, this work having been extended in [50] to find the optimal time to execute preventive actions. The authors have compared their work with respect to different maintenance policies such as CM, PM, opportunistic, and different RL algorithms, whose results show that the RL policy manages to keep maintenance and production costs below the other policies. Wang et al. [39] propose the optimization of production scheduling and maintenance of multiple factories in which collaborative maintenance policy is performed. In their model, both machine deterioration and unexpected failures are taken into account with the aim of maximizing the total profit, where collaborative maintenance is ensured through the use of blockchain technology.

Regarding studies adopting the distributed scheme, let us cite Kuhnle et al. [60] who propose an approach to determine the best window of opportunity to perform maintenance in a stochastic production environment. Their work is based on the use of multiple independent agents that learn the policy based on information from the production system buffer and time-to-failures. The RL policy outperforms CM and PM policies with respect to completed jobs and evidenced how the agents learn to execute maintenance closer to failure times with a low buffer volume. Su et al. [40] extended Huang et al. work [50,61] of single agents to multi agents for PM purposes on a serial production line with multi-stage machines. Their main contribution is to demonstrate convergence towards better policies and to solve scalability issues by the joint action space of the single agent approaches. Although the above-discussed studies are interesting in many respects, they still suffer from two main limitations: (l1) they do not take into account the availability of technicians; (l2) they are limited to a single type of failure.

Table 3
Overview of agent architecture designs adopted by state-of-the-art studies.

	Centralized agent	Distributed agents
Sequential	[50,61]	[40]
Parallel	[39,51,62,68]	[60]

Table 3 provides an overview of what agent architecture designs have been adopted by the above-reviewed studies, showing that most of them adopt the centralized scheme. This can be partly explained by the fact that a distributed agent framework brings additional challenges/complexity such as agent heterogeneity, communication, the definition of collective goals (cooperation), scalability, non-stationarity, and so forth [86,87]. However, we believe that this approach is more realistic and can become more powerful if well addressed and managed. This is one of the main reasons why we propose and investigate a new multi-agent deep RL-based maintenance policy in this paper, with the objective to overcome the limitations (l1), (l2) previously discussed.

3. Multi-agent deep RL-based maintenance policy

This Section presents the mathematical formalization underlying our RL system. Section 3.1 introduces the environment considering a set of identical-parallel machines subject to multiple failures. Section 3.2 presents the RL-Framework explaining in detail the interaction that the agents have with the environment. All notations used in this work are summarized in Table 4.

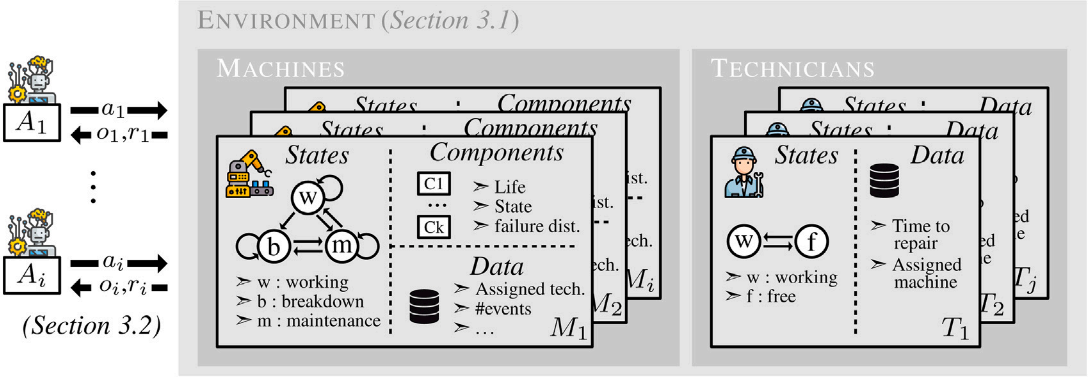


Fig. 3. Environment proposed for multi-agent deep RL-based PdM.

Table 4
Notation used in this paper.

Notation	Description
M	Set of machines
C_m	Set of components of $m \in M$
f_c	Failure time of $c \in C_m$
T	Set of technicians
r_{tc}	Repairing time by technician $t \in T$ given a component $c \in C_m$
$W(x;\alpha,\beta)$	2-parameter Weibull distribution
z_m	State of $m \in M$
g_t	State of $t \in T$
w_c	State of $c \in C_m$
l_c	Lifespan of component $c \in C_m$
e_m	Remaining maintenance time of m
r_m	Reward obtained by agent in charge of m
$\mathbf{o_m}$	Local observation obtained by agent in charge of m
$\mathbf{a}_m$	Action taken by the agent in charge of m
η	Reward penalty due to breakdown prediction
p	Total profit

3.1. System description

Let us consider a factory where the production is performed by M identical machines, as shown in Fig. 3. Each machine $m \in M$ is integrated by C_m components, which can fail at $f_c \mid c \in C_m$ timesteps. The machine m can receive maintenance by any available technician $t \in T$ to restore or repair any $c \in C_m$. Each t has a repairing time r for each type of component c. The repairing time r by technician t given a component c is defined as $r_{tc} \in \mathbb{Z}^{+}$.

We can represent three states for each machine that evolve in discrete-time steps. The first is *working*, which indicates that the machine is running normally and has not shown any type of failure. The second is *breakdown*, in which one or more c have failed and the machine cannot return to the *working* state until c has/have been repaired. To model the failure of a component, a 2-parameter Weibull distribution is used, as commonly adopted in the literature to describe equipment failures [88], where the probability density function is defined as $W(x;\alpha,\beta) = \frac{\beta x^{\beta-1}}{\alpha^{\beta}} e^{-(\frac{x}{\alpha})^{\beta}}$, with α the scale parameter and β the shape. The last state is *maintenance*, which indicates that a technician t has been assigned to repair a c in r_{tc} time steps. If the machine does not present any other type of failure, then it will switch back to the *working* state. Ideally, the machine should move from the *working* state to the *maintenance* state without having to go through the *breakdown* state. Finally, any available technician can be assigned when the machine is in the *working* or *breakdown* state.

3.2. Multi-agent models

In order to obtain optimal or near-optimal maintenance policies, an RL approach is employed. According to this, multiple decision-making agents are placed in an environment whose dynamics are initially unknown [21]. This environment can be formally described using an extension of the MDP called Markov Games (MG), also known as Stochastic Games. MDP is a mathematical formulation of a problem in which an agent (decision-maker) selects actions sequentially to transit through different states guided by rewards. A MDP can be expressed as a 5-tuple $\langle S, A, \mathcal{P}, R, \gamma \rangle$, which consists of a state space S indicating all possible states the agent can be in; an action space A indicating all possible actions the agent can take; a transition function $\mathcal{P} : S \times A \rightarrow S$ indicating the probability of transitioning from any state $s \in S$ to state $s' \in S$ given that the agent took action $a \in A$; a reward function $R : S \times A \times S \rightarrow \mathbb{R}$ that returns an immediate reward given by the transition from (s,a) to s'; and a discount factor $\gamma \in [0,1]$ indicating how myopic the agent is, a $\gamma = 0$ indicating that the agent only cares about immediate reward and in the case of $\gamma \rightarrow 1$ the agent gives more weight to future state information.

MG is a generalization of MDP to work with multiple agents. MG for N agents can be defined by the 6-tuple $\langle N, S, \{A_i\}_{i \in N}, \mathcal{P}, \{R_i\}_{i \in N}, \gamma \rangle$, which consists of a state space S indicating the state space observed by all the agents; an action space A_i indicating all possible actions the agent i can take; a transition function $\mathcal{P} : S \times A \rightarrow S$, given that $A = A_1 \times \cdots \times A_i$, indicating the probability of transitioning from any state $s \in S$ to any state $s' \in S$ for any joint action $a \in A$; a reward function $R_i : S \times A \times S \rightarrow \mathbb{R}$ that returns an immediate reward received by agent i for a transition from (s,a) to s'; and a discount factor $\gamma \in [0,1]$.

The proposed predictive maintenance policies (RL policies) performed two tasks: (i) keep the machines in a *working* state as long as possible; (ii) prevent any failure that could lead to a *breakdown* state by triggering a maintenance action. Using the MG framework, agents determine their actions based on the observations given by the system. Therefore, it is required to design local observations that provide the basis for the agent's actions and provide useful information for training. To perform the above described tasks, agents observe a representation of the system state that includes the state of all the machines and technicians. However, each agent is limited to observing the lifetime of the components of its own machine and any remaining time of maintenance. This limits the decision-making of the agents (since they are partially unaware of the complete state of the other machines), which contributes to decrease the complexity of their decision-making.

The system state observed by an agent a_m responsible for the machine m is represented by the vector given in (1), where $z_i \in \{0,1,2\}$ gives information whether machine $i \in M$ is in a *working*, *breakdown* or *maintenance* state respectively, $g_j \in \{0,1\}$ whether a technician $j \in T$ is working on a particular machine (0) or free (1), w_c represents the state of component $c \in C_m$, l_c the lifespan of component $c \in C_m$, and e_m the remaining maintenance time of machine m.

$$\mathbf{o}_m = (z_1, \ldots, z_{|M|}, g_1, \ldots, g_{|T|}, w_1, \ldots, w_{|C_m|}, l_1, \ldots, l_{|C_m|}, e_m) \quad (1)$$

According to the observation, each agent must select whether to perform a maintenance action by selecting an available technician and the component to be maintained or to delay the maintenance. The action of agent $\mathbf{a}_m$ responsible for machine m is given in (2) where d means that no technician will be allocated to m.

$$\mathbf{a}_m \in T \times C_m \cup \{\mathbf{d}\} \tag{2}$$

To guide the agents in learning the policy, a reward function is defined for benefiting the agent to keep the machine in the *working* state and for predicting the best time to trigger maintenance actions. This function is defined in (3), where $\mathbf{o}_m$ is the observation of the current state of machine m, $\mathbf{a}_m$ the action taken, and $\mathbf{o}'_m$ the next observation obtained from machine m. η is defined in (4), with $f_c \sim W(x;\alpha,\beta)$ the time steps in which the selected component c by the maintenance action will fail, and l_c the current time steps of the component.

$$r_m(\mathbf{o}_m, \mathbf{a}_m, \mathbf{o}'_m) = \begin{cases} 1 & \text{if } \mathbf{o}'_m = \mathit{working} \\ 1-\eta & \text{if } \mathbf{o}'_m = \mathit{maintenance} \\ 0 & \text{if } \mathbf{o}'_m = \mathit{breakdown} \\ -2 & \text{if } \mathbf{a} = \mathit{invalid} \end{cases} \tag{3}$$

$$\eta = \begin{cases} \frac{f_c - l_c}{f_c + l_c} & \text{if } f_c > l_c \\ 1 & \mathit{otherwise} \end{cases} \tag{4}$$

Overall, the agents learn how to keep machines in the *working* state as long as possible due to the fact that this condition allows them to obtain the highest reward, and, in an indirect way, to identify the best technicians to perform the maintenance in the shortest amount of time steps to move out of the *breakdown* and *maintenance* state as quickly as possible.

4. Evaluation

For evaluation purposes, the proposed RL-based maintenance policy is going to be compared with traditional ones such as CM and PM. Section 4.1 describes such policies. Section 4.2 presents the experimental set-up. Section 4.3 presents the training phase of the multi-agent system. Section 4.4 presents and discusses the obtained results.

4.1. Maintenance policy benchmarking

To evaluate the proposed RL-based maintenance policy, three maintenance policies are implemented: corrective (CM), preventive (PM), and a random one (RA), in a similar way as proposed in [60]. Each of these policies is further described in the following.

CM is the simplest policy. It is based on waiting for any of the components c in machine m to fail before performing the maintenance action. Once the machine goes to the *breakdown* state, it checks if there is a technician t available. If any, technician t is assigned to m, otherwise, m makes the t query at each time step until it can perform the assignment. The pseudocode of this policy, which has been implemented in our environment, is detailed in Algorithm 1.

Algorithm 1 CM policy

```
$m \leftarrow$ current machine
$C_m \leftarrow$ set of Components
if state($m$) = breakdown then
    for $t$ in $T$ do
        if state($t$) = available then
            for $c$ in $C$ do
                if state($c$) = break then
                    do maintenance to $c$ by $t$
                    break
```

The RA policy is part of the preventive maintenance policy because it tries to perform maintenance actions before the failure occurs. For this, an ϵ can be defined, which determining the probability of requesting a technician t to perform maintenance actions to any of the components $c \in C_m$, even if the maintenance is not ‘necessary’ (i.e., not critical). In case there is no technician available, the operation is repeated for the next time step. The pseudocode of this policy is detailed in Algorithm 2.

Table 5
Parameters of the studied scenarios.

	$\lvert M \rvert$	$\lvert T \rvert$	$\lvert C_m \rvert$	α_c	β_c	r_{tC}
Scenario 1	3	2	2	$\langle 6,8 \rangle$	$\langle 5,5 \rangle$	$\langle (8,2),(2,8) \rangle$
Scenario 2	5	3	2	$\langle 6,8 \rangle$	$\langle 5,5 \rangle$	$\langle (8,2),(2,8),(8,8) \rangle$

Algorithm 2 RA policy

```
Parameters $\epsilon \in (0,1]$
$m \leftarrow$ current machine
$C_m \leftarrow$ set of Components
$r \leftarrow$ random number between 0 and 1
$c_r \leftarrow$ random $c$ from $C$
if $r < \epsilon$ then
    for $t$ in $T$ do
        if state($t$) = available then
            do maintenance to $c_r$ by $t$
            break
```

The PM policy performs maintenance actions periodically. A time period $\mathcal{T}$ should be set for each $c \in C$. This will indicate the times at which maintenance will be requested to be performed on the components $c \in C$. In case that c failed before the maintenance action is performed, the PM policy can request a technician t to perform a corrective action to the particular component that failed. The pseudocode of this policy is detailed in Algorithm 3.

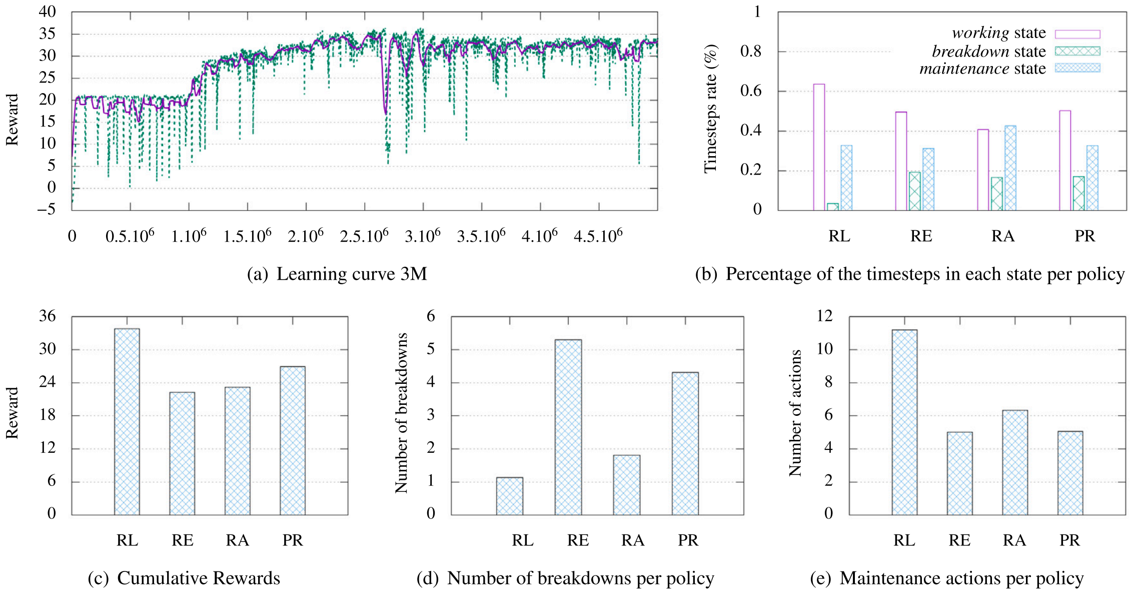
Algorithm 3 PM policy

```
Parameters $\mathcal{T} = \{\tau_1, \ldots, \tau_c\}$ period of maintenance for each $c$ in $C_m$
$i \leftarrow$ current timestep
$m \leftarrow$ current machine
$C_m \leftarrow$ set of Components
if state($m$) = working then
    for $\tau$ in $\mathcal{T}_c$ do
        if i = $\tau$ then
            for $t$ in $T$ do
                if state($t$) = available then
                    do maintenance to $c$ by $t$
                    break
else
    do apply the CM Policy
    break
```

The next Sections detail the experimental environment and setup in which the three above maintenance policies and ours (RL) have been implemented for evaluation and comparison purposes.

4.2. Experimental setup

To test our system, a variable number of agents and technicians are defined, and two types of component failures are modeled based on the $W(x;\alpha,\beta)$ distribution. A different α for each type of failure is used, which analogously represents the time to failure, and a single $\beta \geq 2$ is set to model wear-out failures [89]. Three technicians are defined, where only the first two are used in the first scenario (as reported in Table 5). The repair time of the technicians is set so that one technician is highly experienced in repairing one type of failure (short repairing



(a) Learning curve 3M

(b) Percentage of the timesteps in each state per policy

(c) Cumulative Rewards

(d) Number of breakdowns per policy

(e) Maintenance actions per policy

Fig. 4. Training and evaluation of the 3M-Scenario.

time) and inexperienced in repairing the other (long repairing time), the opposite applying for the second technician. The third technician is inexperienced in repairing both types of failures. To evaluate the proposed policy with respect to the policies described in Section 4.1 (CM, PM, RA), the horizon of the episode is limited to 1.5 times the maximum 90th percentile of the failure distributions. In this way, we can evaluate the occurrence of both failure distributions in a short period of time. The configuration used for each scenario (3 machines and 5 machines) is given in Table 5.

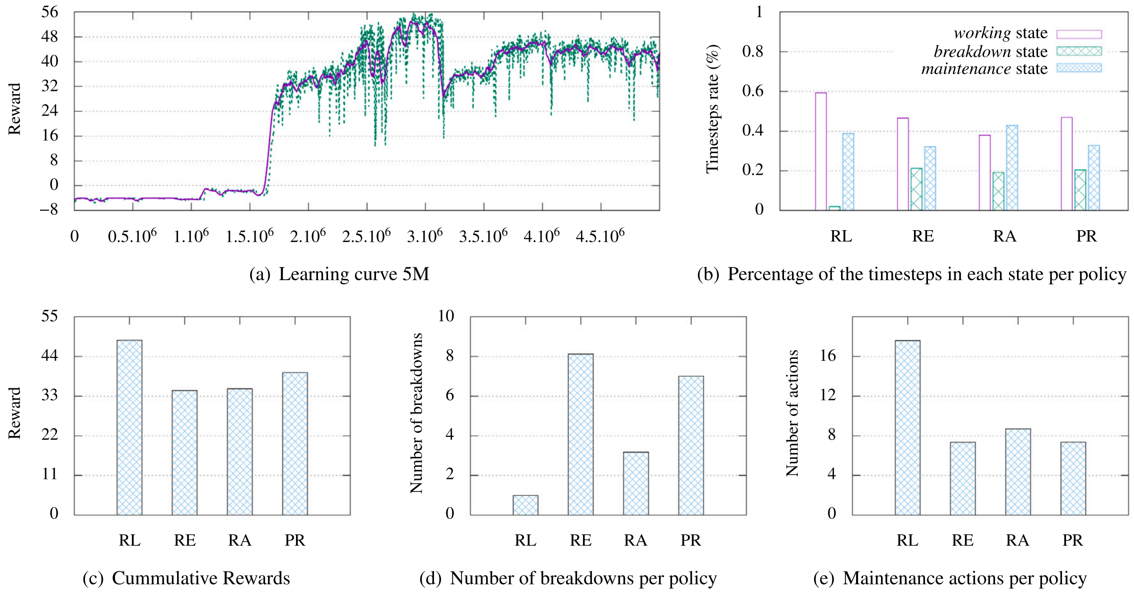
4.3. Training of RL agents

To evaluate our system, the agents are trained using the Proximal Policy Optimization algorithm [90] based on the RLlib library implementation. This is part of the policy gradient methods and ensures that the policy update stays close to the previous policy by optimizing a clipped surrogate objective. In addition, we use Curiosity [91] as an exploration strategy to address the lack of diversity of the reward (sparsity) due to the fact that some actions may not provide an immediate reward (e.g., delay action). Curiosity is a type of intrinsic motivation for exploration that encourages the agent to experience novel states. Deepak Pathak et al. [91] define curiosity as the error in an agent's ability to predict the consequence of its own actions in a visual feature space learned by a self-supervised inverse dynamics model. The agents are trained with completely independent policies for 5 millions steps. A fully connected 2-layer of 16 × 16 is used, a batch size of 2000, and a learning rate of 1e−3, while keeping the remaining hyperparameters by default. Figs. 4(a) and 5(a) show the training curves (dashed green curves) of the 3M and 5M-scenarios respectively, and their average (purple) over the 50k steps, where agents manage to learn the policy despite the non-stationarity effect of the learning of multiple agents.

4.4. Maintenance policy comparison analysis

As detailed in Section 4.1, policies learned by our multi-agent system are evaluated with respect to the CM, PM, and RA policies. All the actions of these policies are valid, so unlike RL agents, these agents know what each agent chooses when forming the joint action to be executed in the environment. This prevents, for example, multiple agents to choose the same available technician, which RL agents do not know *a priori* when performing the joint action. In the RA policy, as shown in Algorithm 2, an $\epsilon = .35$ is set, which prevents the agent from executing a maintenance action too quickly. For the CM policy, there is no parameter to be adjusted/set in Algorithm 1 since the agents wait until the machines switch to the *breakdown* state for requesting a maintenance action. In the PM policy, as shown in Algorithm 3, the maintenance action is triggered based on the failure distributions. In this respect, the maintenance request times are given by $\mathcal{T} = \{\mathbb{E}[f_1], \dots, \mathbb{E}[f_{|C_m|}]\}$, where $\mathbb{E}[f_c]$ represents the expected value of the distribution of failures $f_c \sim W(x;\alpha,\beta)$ for component $c \in C_m$. In case there is a failure before requesting maintenance, then the CM policy is performed. To evaluate the different maintenance policies, four metrics are defined: percentage of time that the machines were in working state; performance based on the total reward obtained for each policy; total number of breakdowns per episode; total number of maintenance actions per episode. In our experiments, results were obtained by averaging the results over 1000 episodes.

Results obtained from our experiments are given in Figs. 4 and 5 for the 3M and 5M scenarios. Looking at metric *percentage of time in working state* (see Figs. 4(b) and 5(b)), results show that the RL policy outperforms the three other policies (CM, RA, PM), as RL allows machines to stay ≈60% of the time in *working* state, against 40% to 50% for CM, RA and PM. Besides, RL spends ≈20% less time in breakdown state than CM, RA, PM, while spending a similar amount of time in the maintenance one. This brings evidence that the RL agents not only succeed in predicting when to execute the maintenance actions to avoid a failure, but also in improving agent coordination (e.g., avoiding them to pick the same technician at the same time). Looking at metric *total reward*, as can be seen in Figs. 4(c) and 5(c) for the 3M and 5M scenarios, the cumulative reward in the RL policy outperforms the other policies between 20% to 35%. This suggests that, because agents are guided by the reward to keep the machine in a working state, and they request maintenance actions as close as possible to the failures, it is a policy that reduces unexpected failures while reducing the collision of maintenance requests. The latter is due to the fact that, if several agents request at the same time the same technician, this is considered as an invalid action, which is penalized and causes a restart of the environment, thus preventing the maximization of rewards. Looking at *total number of breakdowns* (see Figs. 4(d) and 5(d)), the policy learned by the RL agents – *being guided to obtain a reward for approaching the time step in which there will be a breakdown* – succeeds in reducing the number of breakdowns. One explanation lies in the fact that the other policies do not have adequate knowledge of when to trigger the maintenance due to the lack of coordination between agents. This could



(a) Learning curve 5M

(b) Percentage of the timesteps in each state per policy

(c) Cummulative Rewards

(d) Number of breakdowns per policy

(e) Maintenance actions per policy

Fig. 5. Training and evaluation of the 5M-scenario.

be different for the case of the PM policy if $|T| \geq |M|$, since, on average, it would be executing the maintenance action taking advantage of the completely useful life of the components. However this scenario is unrealistic considering a manufacturing scenario. Instead, if the value of ϵ in the RA policy is increased, this would decrease the number of failures but at the cost of an increased number of maintenance actions.

Figs. 4(e) and 5(e) give insight into the *number of maintenance actions* performed by each policy. Interestingly, the RL policy generates a higher number of actions. However, this needs to be put into perspective considering how this contributes to reducing the overall system downtime, for which RL outperforms the three other policies (as previously analyzed with Figs. 4(b) and 5(b)). One reason that might explain this result is that scheduling a higher number of maintenance actions can prevent machines from switching to the failure state, but also avoid any queue for the availability of technicians to trigger the maintenance action.

4.5. *Financial implication*

The different policies in both scenarios were evaluated from an economical aspect considering three key variables obtained from Section 4.4. Since the profit of the company is obtained by their productivity, we evaluate the total profit with respect to the number of time steps that the machines are in the *working* state. However, giving that maintenance actions and breakdowns also represent a cost to the total profit, these two parameters (number of breakdowns and maintenance actions) are also taken into account when calculating the total profit.

Based on this premise, the total profit is calculated using (5), where p represents the total profit, p_{max} the maximum profit, a the total percentage of time steps in the *working* state, b the number of breakdowns, c the number of maintenance actions, and $\alpha \in [0,1]$ and $\beta \in [0,1]$ a ratio of the maximum profit that each breakdown and each maintenance action costs respectively. The maximum possible profit, per episode, is the scenario where the machines are all the time in the *working* state.

$$p = p_{max}(a - (\alpha.b + \beta.c)) \tag{5}$$

Figs. 6(a) and 6(b) illustrate that the top-1 policies with different values of α and β represent between 1% and 10% of the p_{max} for breakdown events and maintenance actions. These are extreme values, but they allow us to identify the frontiers where one policy outperforms another from a total profit perspective. Dark colors (also denoted by "+" in the legend) represent a profit gain, while light colors (denoted by "−") represent a loss. For different values of α and β, the RL, RA, and PM policies perform better. However, unlike RL and PM, the RA policy never leads to a positive profit, which is unrealistic. As can be seen from the results, if the cost per number of maintenance actions increases, the preventive policy becomes a better choice than the RL policy. On the other hand, if β is kept low, the RL policy is not affected by an increase of α, which is due to the prevention of breakdowns.

5. Discussion

Although the present research work opens up a new direction to maintenance task scheduling in uncertain environments and with dynamic assignment of technicians with different skills, some limitations or improvement opportunities should be raised.

First, one of the great advantages of multi-agent systems is the distribution of a complex task into multiple simpler tasks among different agents to avoid the exponential growth of the joint action space [92,93]. However, multi-agent RL systems present different challenges such as agent heterogeneity, communication, the definition of collective goals (cooperation), scalability, design of compact representations of the true state of the environment, and the main problem of the non-stationarity [86,87]. Many approaches propose to adopt a centralized-learning approach (*cf.*, Fig. 2) to address some of these challenges [93,94]. Nevertheless, decentralized-learning has proven to be as effective in addressing this problem [95]. In the current work, we use a fully decentralized learning paradigm of RL that may prove to be computationally expensive due to the large search space on a day-to-day basis, the dynamic nature of the manufacturing floor, and the interactions between agents that may become complex to be managed [96]. To overcome this, new approaches should be investigated. Combining RL with metaheuristics for the exploration of RL agents is a possible direction [97]. Besides, we consider that it is also important to evaluate the design of the agents' observations with respect to the type of learning to be used.

Second, usually for the assignment of technicians to perform a maintenance task, a planner is in charge of obtaining all the requirements, such as the number of technicians needed, the craftsmanship skills, the working hours, the duration of the job, *etc.* [98]. However, when defining policies obtained by RL agents, these policies can exhibit (very) complex behaviors, which can cause distrust when planners,

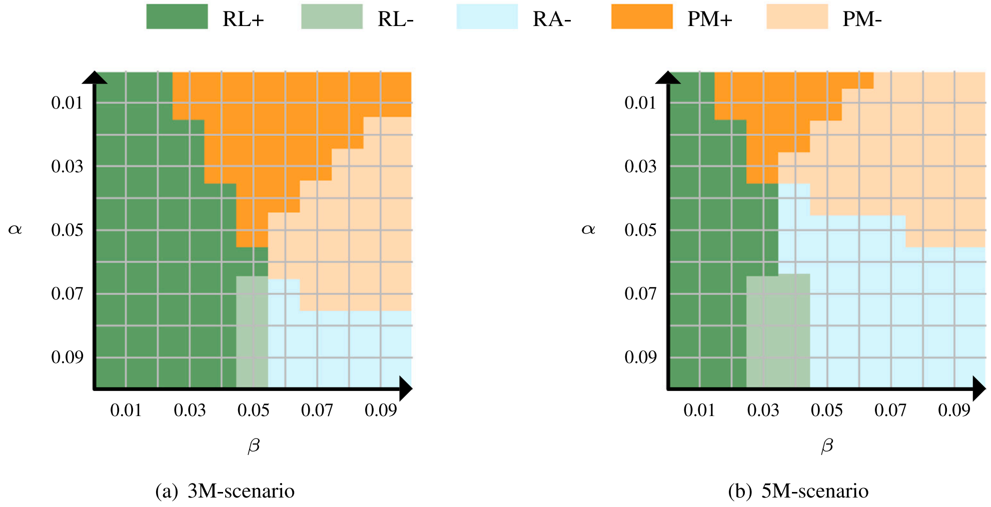


Fig. 6. Top-profit policies for different α and β, which respectively represent the cost (percentage of the maximum profit) of a breakdown and maintenance action. Dark colors (denoted by "+" in the legend) represent a profit gain, while light colors (denoted by "−" in the legend) represent a loss. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

schedulers, or even technicians are part of the decision making process [99]. It is therefore very important in future PdM frameworks to focus "explaining" the decisions resulting from such frameworks, which is also known as XAI (eXplainable AI) in the literature [96,100,101].

Finally, in the approach presented in this paper, the maintenance policy learned by the RL agents is based on equipment degradation, and on the skills of the technicians and their availability. However, as evidenced in Section 2, in order to optimize the learned policy, it is important to explore the development of a joint policy with production activities to look for the best time windows of opportunity [60] to perform the maintenance actions, but also to take into account the quality data (i.e., monitoring how the quality of a given process degrades over time) [33,51].

6. Conclusion

This research work examines a novel stochastic system that experiences multiple types of failures on a set of machines working in parallel (i.e., that are independent from one another). The problem consists in determining the best time to assign a maintenance task to a given technician (depending on her/his availability and skills) on a particular machine in order to avoid system failures and maintain a high operational uptime. As evidenced through the review of literature carried out in this paper, this problem has not yet been properly addressed in the literature, or at least the identified state-of-the-art studies are limited in several respects (e.g., they consider a single type of failure; they do not take into account technician resources in the problem formalization, or still they assume that technicians are always available).

To solve this problem, a multi-agent deep RL system is proposed, where each agent is responsible to monitor a single machine and trigger the (appropriate/optimal) maintenance action(s). The efficiency of the system is experimentally evaluated and compared with three other policies (reactive, preventive and random). Results of the proposed scenario show that the RL policy outperforms the others, reducing up to 80% the breakdown time, and up to 75% the failure prevention. To achieve these results, the RL policy performs more maintenance actions. Although an initial economical evaluation to understand when the RL policy might become too expensive from a profit perspective depending on the cost of spare parts to fix a breakdown, the labor cost of a technician to intervene on a machine, and so forth, such an analysis requires further attention and validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This research was funded in whole, or in part, by the Luxembourg National Research Fund (FNR), grant reference UPTIME4.0 and project reference 16756339. For the purpose of open access, the author has applied a Creative Commons Attribution 4.0 International (CC BY 4.0) license to any Author Accepted Manuscript version arising from this submission.

References

[1] A. Shrivastava, K. Murali Krishna, M. Lal Rinawa, M. Soni, G. Ramkumar, S. Jaiswal, Inclusion of IoT, ML, and blockchain technologies in next generation Industry 4.0 environment, Mater. Today Proc. (2021) http://dx.doi.org/10.1016/j.matpr.2021.07.273.

[2] M. Calış Duman, B. Akdemir, A study to determine the effects of industry 4.0 technology components on organizational performance, Technol. Forecast. Soc. Change 167 (2021) 120615, http://dx.doi.org/10.1016/j.techfore.2021.120615.

[3] K. Komonen, A cost model of industrial maintenance for profitability analysis and benchmarking, Int. J. Prod. Econ. 79 (1) (2002) 15–31, http://dx.doi.org/10.1016/S0925-5273(00)00187-0.

[4] T. Zonta, C.A. da Costa, R. da Rosa Righi, M.J. de Lima, E.S. da Trindade, G.P. Li, Predictive maintenance in the Industry 4.0: A systematic literature review, Comput. Ind. Eng. 150 (2020) 106889, http://dx.doi.org/10.1016/j.cie.2020.106889.

[5] W. Luo, T. Hu, Y. Ye, C. Zhang, Y. Wei, A hybrid predictive maintenance approach for CNC machine tool driven by Digital Twin, Robot. Comput. Integr. Manuf. 65 (2020) 101974, http://dx.doi.org/10.1016/j.rcim.2020.101974.

[6] C. Coleman, S. Damodaran, M. Chandramouli, E. Deuel, Predictive maintenance and the digital supply network, 2022, https://www2.deloitte.com/us/en/insights/focus/industry-4-0/using-predictive-technologies-for-asset-maintenance.html.

[7] F. Tao, Q. Qi, A. Liu, A. Kusiak, Data-driven smart manufacturing, J. Manuf. Syst. 48 (2018) 157–169, http://dx.doi.org/10.1016/J.JMSY.2018.01.006.

[8] Q. Cao, C. Zanni-Merk, A. Samet, C. Reich, F.B. de Beuvron, A. Beckmann, C. Giannetti, KSPMI: A knowledge-based system for predictive maintenance in Industry 4.0, Robot. Comput.-Integr. Manuf. 74 (2022) 102281, http://dx.doi.org/10.1016/j.rcim.2021.102281.

[9] X. Yang, Y. Ran, G. Zhang, H. Wang, Z. Mu, S. Zhi, A digital twin-driven hybrid approach for the prediction of performance degradation in transmission unit of CNC machine tool, Robot. Comput.-Integr. Manuf. 73 (2022) 102230, http://dx.doi.org/10.1016/j.rcim.2021.102230.

[10] Q. Liu, M. Dong, F.F. Chen, Single-machine-based joint optimization of predictive maintenance planning and production scheduling, Robot. Comput.-Integr. Manuf. 55 (2018) 173–182, http://dx.doi.org/10.1016/j.rcim.2018.09.007.

[11] A. Bousdekis, K. Lepenioti, D. Apostolou, G. Mentzas, Decision making in predictive maintenance: Literature review and research agenda for industry 4.0, IFAC-PapersOnLine 52 (2019) 607–612, http://dx.doi.org/10.1016/j.ifacol.2019.11.226.

[12] C.A. Gordon, E.N. Pistikopoulos, Data-driven prescriptive maintenance toward fault-tolerant multiparametric control, AIChE J. (2021) e17489, http://dx.doi.org/10.1002/aic.17489.

[13] S. Wan, D. Li, J. Gao, J. Li, A knowledge based machine tool maintenance planning system using case-based reasoning techniques, Robot. Comput.-Integr. Manuf. 58 (2019) 80–96, http://dx.doi.org/10.1016/j.rcim.2019.01.012.

[14] N. Sakib, T. Wuest, Challenges and opportunities of condition-based predictive maintenance: A review, Procedia CIRP 78 (2018) 267–272, http://dx.doi.org/10.1016/J.PROCIR.2018.08.318.

[15] Y. Wang, L. Zheng, Y. Wang, Event-driven tool condition monitoring methodology considering tool life prediction based on industrial internet, J. Manuf. Syst. 58 (2021) 205–222, http://dx.doi.org/10.1016/J.JMSY.2020.11.019.

[16] H. Sun, J. Zhang, R. Mo, X. Zhang, In-process tool condition forecasting based on a deep learning method, Robot. Comput.-Integr. Manuf. 64 (2020) 101924, http://dx.doi.org/10.1016/j.rcim.2019.101924.

[17] S. Schwendemann, Z. Amjad, A. Sikora, A survey of machine-learning techniques for condition monitoring and predictive maintenance of bearings in grinding machines, Comput. Ind. 125 (2021) 103380, http://dx.doi.org/10.1016/J.COMPIND.2020.103380.

[18] A. Dogan, D. Birant, Machine learning and data mining in manufacturing, Expert Syst. Appl. 166 (2021) 114060, http://dx.doi.org/10.1016/J.ESWA.2020.114060.

[19] K. Lepenioti, M. Pertselakis, A. Bousdekis, A. Louca, F. Lampathaki, D. Apostolou, G. Mentzas, S. Anastasiou, Machine Learning for Predictive and Prescriptive Analytics of Operational Data in Smart Manufacturing, in: Lecture Notes in Business Information Processing, vol. 382 LNBIP, Springer, Cham, 2020, pp. 5–16, http://dx.doi.org/10.1007/978-3-030-49165-9_1.

[20] A. de Giorgio, A. Maffei, M. Onori, L. Wang, Towards online reinforced learning of assembly sequence planning with interactive guidance systems for industry 4.0 adaptive manufacturing, J. Manuf. Syst. 60 (2021) 22–34, http://dx.doi.org/10.1016/J.JMSY.2021.05.001.

[21] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.

[22] Y.G. Kim, S. Lee, J. Son, H. Bae, B. Do Chung, Multi-agent system and reinforcement learning approach for distributed intelligence in a flexible smart manufacturing system, J. Manuf. Syst. 57 (2020) 440–450, http://dx.doi.org/10.1016/J.JMSY.2020.11.004.

[23] Y. Li, W. Gu, M. Yuan, Y. Tang, Real-time data-driven dynamic scheduling for flexible job shop with insufficient transportation resources using hybrid deep Q network, Robot. Comput.-Integr. Manuf. 74 (2022) 102283, http://dx.doi.org/10.1016/j.rcim.2021.102283.

[24] N. Aissani, B. Beldjilali, D. Trentesaux, Dynamic scheduling of maintenance tasks in the petroleum industry: A reinforcement approach, Eng. Appl. Artif. Intell. 22 (7) (2009) 1089–1103, http://dx.doi.org/10.1016/j.engappai.2009.01.014.

[25] C.-Y. Lee, Machine scheduling with an availability constraint, J. Global Optim. 9 (3) (1996) 395–416, http://dx.doi.org/10.1007/BF00121681.

[26] M.H. Fazel Zarandi, A.A. Sadat Asl, S. Sotudian, O. Castillo, A state of the art review of intelligent scheduling, Artif. Intell. Rev. 53 (1) (2020) 501–593, http://dx.doi.org/10.1007/s10462-018-9667-6.

[27] W. Qin, Z. Zhuang, Y. Liu, J. Xu, Sustainable service oriented equipment maintenance management of steel enterprises using a two-stage optimization approach, Robot. Comput.-Integr. Manuf. 75 (2022) 102311, http://dx.doi.org/10.1016/j.rcim.2021.102311.

[28] S. Lu, J. Pei, X. Liu, P.M. Pardalos, A hybrid DBH-VNS for high-end equipment production scheduling with machine failures and preventive maintenance activities, J. Comput. Appl. Math. 384 (2021) 113195, http://dx.doi.org/10.1016/j.cam.2020.113195.

[29] M. Hoffman, E. Song, M.P. Brundage, S. Kumara, Online improvement of condition-based maintenance policy via Monte Carlo tree search, IEEE Trans. Autom. Sci. Eng. (2021) http://dx.doi.org/10.1109/TASE.2021.3088603.

[30] M. Ghaleb, S. Taghipour, H. Zolfagharinia, Real-time integrated production-scheduling and maintenance-planning in a flexible job shop with machine deterioration and condition-based maintenance, J. Manuf. Syst. 61 (2021) 423–449, http://dx.doi.org/10.1016/j.jmsy.2021.09.018.

[31] D. Liu, Z. Xu, F. Li, A three-stage decomposition algorithm for decentralized multi-project scheduling under uncertainty, Comput. Ind. Eng. 160 (2021) 107553, http://dx.doi.org/10.1016/j.cie.2021.107553.

[32] T. Yamada, M. Nagano, H. Miyata, Minimization of total tardiness in no-wait flowshop production systems with preventive maintenance, Int. J. Ind. Eng. Comput. 12 (4) (2021) 415–426, http://dx.doi.org/10.5267/j.ijiec.2021.5.002.

[33] S.M. Hadian, H. Farughi, H. Rasay, Joint planning of maintenance, buffer stock and quality control for unreliable, imperfect manufacturing systems, Comput. Ind. Eng. 157 (2021) 107304, http://dx.doi.org/10.1016/j.cie.2021.107304.

[34] Y. Liu, Q. Zhang, Z. Ouyang, H.-Z. Huang, Integrated production planning and preventive maintenance scheduling for synchronized parallel machines, Reliab. Eng. Syst. Saf. 215 (2021) 107869, http://dx.doi.org/10.1016/j.ress.2021.107869.

[35] X. Zhou, M. Zhu, W. Yu, Maintenance scheduling for flexible multistage manufacturing systems with uncertain demands, Int. J. Prod. Res. 59 (19) (2021) 5831–5843, http://dx.doi.org/10.1080/00207543.2020.1791998.

[36] Q. Yang, X. Meng, H. Zhao, C. Cao, Y. Liu, D. Huisingh, Sustainable operations-oriented painting process optimisation in automobile maintenance service, J. Cleaner Prod. 324 (2021) 129191, http://dx.doi.org/10.1016/j.jclepro.2021.129191.

[37] P. Rokhforoz, O. Fink, Distributed joint dynamic maintenance and production scheduling in manufacturing systems: Framework based on model predictive control and benders decomposition, J. Manuf. Syst. 59 (2021) 596–606, http://dx.doi.org/10.1016/j.jmsy.2021.04.010.

[38] O. Wu, G. Dalle Ave, I. Harjunkoski, L. Imsland, A rolling horizon approach for scheduling of multiproduct batch production and maintenance using generalized disjunctive programming models, Comput. Chem. Eng. 148 (2021) 107268, http://dx.doi.org/10.1016/j.compchemeng.2021.107268.

[39] H. Wang, Q. Yan, J. Wang, Blockchain-secured multi-factory production with collaborative maintenance using Q learning-based optimisation approach, Int. J. Prod. Res. (2021) http://dx.doi.org/10.1080/00207543.2021.2002968.

[40] J. Su, J. Huang, S. Adams, Q. Chang, P.A. Beling, Deep multi-agent reinforcement learning for multi-level preventive maintenance in manufacturing systems, Expert Syst. Appl. 192 (2021) 116323, http://dx.doi.org/10.1016/j.eswa.2021.116323.

[41] J. Zheng, H. Yang, Q. Wu, Z. Wang, A two-stage integrating optimization of production scheduling, maintenance and quality, Proc. Inst. Mech. Eng. B 234 (11) (2020) 1448–1459, http://dx.doi.org/10.1177/0954405420921733.

[42] J. Dong, C. Ye, Research on two-stage joint optimization problem of green manufacturing and maintenance for semiconductor wafer, Math. Probl. Eng. 2020 (2020) http://dx.doi.org/10.1155/2020/3974024.

[43] M. Celen, D. Djurdjanovic, Integrated maintenance and operations decision making with imperfect degradation state observations, J. Manuf. Syst. 55 (2020) 302–316, http://dx.doi.org/10.1016/j.jmsy.2020.03.010.

[44] S. Mi, Y. Feng, H. Zheng, Z. Li, Y. Gao, J. Tan, Integrated intelligent green scheduling of predictive maintenance for complex equipment based on information services, IEEE Access 8 (2020) 45797–45812, http://dx.doi.org/10.1109/ACCESS.2020.2977667.

[45] X. Chen, Y. An, Z. Zhang, Y. Li, An approximate nondominated sorting genetic algorithm to integrate optimization of production scheduling and accurate maintenance based on reliability intervals, J. Manuf. Syst. 54 (2020) 227–241, http://dx.doi.org/10.1016/j.jmsy.2019.12.004.

[46] J. Wang, H. Du, J. Xing, F. Qiao, Y. Ma, Novel energy-and maintenance-aware collaborative scheduling for a hybrid flow shop based on dual memetic algorithms, IEEE Robot. Autom. Lett. 5 (4) (2020) 5613–5620, http://dx.doi.org/10.1109/LRA.2020.3005626.

[47] M. Ghaleb, S. Taghipour, M. Sharifi, H. Zolfagharinia, Integrated production and maintenance scheduling for a single degrading machine with deterioration-based failures, Comput. Ind. Eng. 143 (2020) 106432, http://dx.doi.org/10.1016/j.cie.2020.106432.

[48] M. Alimian, V. Ghezavati, R. Tavakkoli-Moghaddam, New integration of preventive maintenance and production planning with cell formation and group scheduling for dynamic cellular manufacturing systems, J. Manuf. Syst. 56 (2020) 341–358, http://dx.doi.org/10.1016/j.jmsy.2020.06.011.

[49] S. Assia, I. El Abbassi, A. El Barkany, M. Darcherif, A. El Biyaali, Green scheduling of jobs and flexible periods of maintenance in a two-machine flowshop to minimize makespan, a measure of service level and total energy consumption, Adv. Oper. Res. 2020 (2020) http://dx.doi.org/10.1155/2020/9732563.

[50] J. Huang, Q. Chang, J. Arinez, Deep reinforcement learning based preventive maintenance policy for serial production lines, Expert Syst. Appl. 160 (2020) 113701, http://dx.doi.org/10.1016/j.eswa.2020.113701.

[51] P.D. Paraschos, G.K. Koulinas, D.E. Koulouriotis, Reinforcement learning for combined production-maintenance and quality control of a manufacturing system with deterioration failures, J. Manuf. Syst. 56 (2020) 470–483, http://dx.doi.org/10.1016/j.jmsy.2020.07.004.

[52] E. Ruschel, E.A.P. Santos, E.F.R. Loures, Establishment of maintenance inspection intervals: an application of process mining techniques in manufacturing, J. Intell. Manuf. 31 (1) (2020) 53–72, http://dx.doi.org/10.1007/s10845-018-1434-7.

[53] H. Feng, C. Tan, T. Xia, E. Pan, L. Xi, Joint optimization of preventive maintenance and flexible flowshop sequence-dependent group scheduling considering multiple setups, Eng. Optim. 51 (9) (2019) 1529–1546, http://dx.doi.org/10.1080/0305215X.2018.1540696.

[54] T. Yu, C. Zhu, Q. Chang, J. Wang, Imperfect corrective maintenance scheduling for energy efficient manufacturing systems through online task allocation method, J. Manuf. Syst. 53 (2019) 282–290, http://dx.doi.org/10.1016/j.jmsy.2019.11.002.

[55] L.V. Tran, B.H. Huynh, H. Akhtar, Ant colony optimization algorithm for maintenance, repair and overhaul scheduling optimization in the context of industrie 4.0, Appl. Sci. 9 (22) (2019) 4815, http://dx.doi.org/10.3390/app9224815.

[56] S.S. Amelian, S.M. Sajadi, M. Navabakhsh, M. Esmaelian, Multi-objective optimization for stochastic failure-prone job shop scheduling problem via hybrid of NSGA-II and simulation method, Expert Syst. (2019) e12455, http://dx.doi.org/10.1111/exsy.12455.

[57] T.-P. Chung, Z. Xue, T. Wu, S.C. Shih, Minimising total completion time on single-machine scheduling with new integrated maintenance activities, Int. J. Prod. Res. 57 (3) (2019) 918–930, http://dx.doi.org/10.1080/00207543.2018.1496294.

[58] S. Chansombat, P. Pongcharoen, C. Hicks, A mixed-integer linear programming model for integrated production and preventive maintenance scheduling in the capital goods industry, Int. J. Prod. Res. 57 (1) (2019) 61–82, http://dx.doi.org/10.1080/00207543.2018.1459923.

[59] A. Farahani, H. Tohidi, A. Shoja, An integrated optimization of quality control chart parameters and preventive maintenance using Markov chain, Adv. Prod. Eng. Manage. 14 (1) (2019) http://dx.doi.org/10.14743/apem2019.1.307.

[60] A. Kuhnle, J. Jakubik, G. Lanza, Reinforcement learning for opportunistic maintenance optimization, Prod. Eng. 13 (1) (2019) 33–41, http://dx.doi.org/10.1007/s11740-018-0855-7.

[61] J. Huang, Q. Chang, N. Chakraborty, Machine preventive replacement policy for serial production lines based on reinforcement learning, in: IEEE International Conference on Automation Science and Engineering, Vol. 2019-Augus, IEEE, 2019, pp. 523–528, http://dx.doi.org/10.1109/COASE.2019.8843338.

[62] W. Xu, L. Cao, Optimal maintenance control of machine tools for energy efficient manufacturing, Int. J. Adv. Manuf. Technol. 104 (9) (2019) 3303–3311, http://dx.doi.org/10.1007/s00170-018-2233-1.

[63] H. Feng, L. Xi, L. Xiao, T. Xia, E. Pan, Imperfect preventive maintenance optimization for flexible flowshop manufacturing cells considering sequence-dependent group scheduling, Reliab. Eng. Syst. Saf. 176 (2018) 218–229, http://dx.doi.org/10.1016/j.ress.2018.04.004.

[64] J. Pang, H. Zhou, Y.-C. Tsai, F.-D. Chou, A scatter simulated annealing algorithm for the bi-objective scheduling problem for the wet station of semiconductor manufacturing, Comput. Ind. Eng. 123 (2018) 54–66, http://dx.doi.org/10.1016/j.cie.2018.06.017.

[65] W. Liao, M. Chen, X. Yang, Joint optimization of preventive maintenance and production scheduling for parallel machines system, J. Intell. Fuzzy Systems 32 (1) (2017) 913–923, http://dx.doi.org/10.3233/JIFS-161385.

[66] K. Upasani, M. Bakshi, V. Pandhare, B.K. Lad, Distributed maintenance planning in manufacturing industries, Comput. Ind. Eng. 108 (2017) 1–14, http://dx.doi.org/10.1016/j.cie.2017.03.027.

[67] M. Biondi, G. Sand, L. Harjunkoski, Optimization of multipurpose process plant operations: A multi-time-scale maintenance and production scheduling approach, Comput. Chem. Eng. 99 (2017) 325–339, http://dx.doi.org/10.1016/j.compchemeng.2017.01.007.

[68] A.S. Xanthopoulos, A. Kiatipis, D.E. Koulouriotis, S. Stieger, Reinforcement learning-based and parametric production-maintenance control policies for a deteriorating manufacturing system, IEEE Access 6 (2017) 576–588, http://dx.doi.org/10.1109/ACCESS.2017.2771827.

[69] D.-J. Wang, F. Liu, J.-J. Wang, Y.-Z. Wang, Integrated rescheduling and preventive maintenance for arrival of new jobs through evolutionary multi-objective optimization, Soft Comput. 20 (4) (2016) 1635–1652, http://dx.doi.org/10.1007/s00500-015-1615-7.

[70] O. Souissi, R. Benmansour, A. Artiba, An accelerated MIP model for the single machine scheduling with preventive maintenance, IFAC-PapersOnLine 49 (12) (2016) 1945–1949, http://dx.doi.org/10.1016/j.ifacol.2016.07.915.

[71] S. Wang, M. Liu, Multi-objective optimization of parallel machine scheduling integrated with multi-resources preventive maintenance planning, J. Manuf. Syst. 37 (2015) 182–192, http://dx.doi.org/10.1016/j.jmsy.2015.07.002.

[72] P.P. Tambe, M.S. Kulkarni, A superimposition based approach for maintenance and quality plan optimization with production schedule, availability, repair time and detection time constraints for a single machine, J. Manuf. Syst. 37 (2015) 17–32, http://dx.doi.org/10.1016/j.jmsy.2015.09.009.

[73] M. Celen, D. Djurdjanovic, Integrated maintenance decision-making and product sequencing in flexible manufacturing systems, J. Manuf. Sci. Eng. 137 (4) (2015) http://dx.doi.org/10.1115/1.4030301.

[74] C.S. Wong, F.T.S. Chan, S.H. Chung, Decision-making on multi-mould maintenance in production scheduling, Int. J. Prod. Res. 52 (19) (2014) 5640–5655, http://dx.doi.org/10.1080/00207543.2014.900200.

[75] S. Liu, A. Yahia, L.G. Papageorgiou, Optimal production and maintenance planning of biopharmaceutical manufacturing under performance decay, Ind. Eng. Chem. Res. 53 (44) (2014) 17075–17091, http://dx.doi.org/10.1021/ie5008807.

[76] P.P. Tambe, S. Mohite, M.S. Kulkarni, Optimisation of opportunistic maintenance of a multi-component system considering the effect of failures on quality and production schedule: A case study, Int. J. Adv. Manuf. Technol. 69 (5) (2013) 1743–1756, http://dx.doi.org/10.1007/s00170-013-5122-7.

[77] S. Lee, J. Ni, Joint decision making for maintenance and production scheduling of production systems, Int. J. Adv. Manuf. Technol. 66 (5) (2013) 1135–1146, http://dx.doi.org/10.1007/s00170-012-4395-6.

[78] K.S. Moghaddam, Multi-objective preventive maintenance and replacement scheduling in a manufacturing system using goal programming, Int. J. Prod. Econ. 146 (2) (2013) 704–716, http://dx.doi.org/10.1016/j.ijpe.2013.08.027.

[79] R. Ramezanian, M. Saidi-Mehrabad, P. Fattahi, MIP formulation and heuristics for multi-stage capacitated lot-sizing and scheduling problem with availability constraints, J. Manuf. Syst. 32 (2) (2013) 392–401, http://dx.doi.org/10.1016/j.jmsy.2013.01.002.

[80] C.S. Wong, F.T.S. Chan, S.H. Chung, A genetic algorithm approach for production scheduling with mould maintenance consideration, Int. J. Prod. Res. 50 (20) (2012) 5683–5697, http://dx.doi.org/10.1080/00207543.2011.613868.

[81] B. Cunha, A.M. Madureira, B. Fonseca, D. Coelho, Deep reinforcement learning as a job shop scheduling solver: A literature review, in: International Conference on Hybrid Intelligent Systems, 2018, pp. 350–359, http://dx.doi.org/10.1007/978-3-030-14347-3_34.

[82] A.A. Juan, J. Faulin, S.E. Grasman, M. Rabe, G. Figueira, A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems, Oper. Res. Perspect. 2 (2015) 62–72, http://dx.doi.org/10.1016/j.orp.2015.03.001.

[83] E.M. González-Neira, A.M. Urrego-Torres, A.M. Cruz-Riveros, C. Henao-García, J.R. Montoya-Torres, L.P. Molina-Sánchez, J.-F. Jimenez, Robust solutions in multi-objective stochastic permutation flow shop problem, Comput. Ind. Eng. 137 (2019) 106026, http://dx.doi.org/10.1016/j.cie.2019.106026.

[84] N.N.A. Halim, S. Shariff, S.M. Zahari, Single-machine integrated production preventive maintenance scheduling: A simheuristic approach, Matematika 36 (2020) http://dx.doi.org/10.11113/matematika.v36.n2.1168.

[85] M. Chica, A.A. Juan Pérez, O. Cordon, D. Kelton, Why simheuristics? Benefits, limitations, and best practices when combining metaheuristics with simulation, Stat. Oper. Res. Trans. 44 (2) (2017) 311–334, http://dx.doi.org/10.2139/ssrn.2919208.

[86] F.L. Da Silva, A.H. Reali Costa, A survey on transfer learning for multiagent reinforcement learning systems, J. Artificial Intelligence Res. 64 (2019) 645–703, http://dx.doi.org/10.1613/jair.1.11396.

[87] T.T. Nguyen, N.D. Nguyen, S. Nahavandi, Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications, IEEE Trans. Cybern. 50 (9) (2020) 3826–3839, http://dx.doi.org/10.1109/TCYB.2020.2977374.

[88] A.J. Hallinan, A review of the Weibull distribution, J. Qual. Technol. 25 (2) (1993) 85–93, http://dx.doi.org/10.1080/00224065.1993.11979431.

[89] I.T. Dedopoulos, Y. Smeers, An age reduction approach for finite horizon optimization of preventive maintenance for single units subject to random failures, Comput. Ind. Eng. 34 (3) (1998) 643–654, http://dx.doi.org/10.1016/S0360-8352(97)00281-7.

[90] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, 2017, http://arxiv.org/abs/1707.06347.

[91] D. Pathak, P. Agrawal, A.A. Efros, T. Darrell, Curiosity-driven exploration by self-supervised prediction, in: D. Precup, Y.W. Teh (Eds.), Proc 34th Int Conf Mach Learn, Vol. 70, PMLR, 2017, pp. 2778–2787.

[92] T. Zhou, D. Tang, H. Zhu, Z. Zhang, Multi-agent reinforcement learning for online scheduling in smart factories, Robot. Comput.-Integr. Manuf. 72 (2021) 102202, http://dx.doi.org/10.1016/j.rcim.2021.102202.

[93] J.N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, S. Whiteson, Counterfactual multi-agent policy gradients, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018.

[94] T. Rashid, M. Samvelyan, C.S. De Witt, G. Farquhar, J. Foerster, S. Whiteson, QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning, in: 35th International Conference on Machine Learning, ICML 2018, Vol. 10, 2018, pp. 6846–6859.

[95] C.S. de Witt, T. Gupta, D. Makoviichuk, V. Makoviychuk, P.H. Torr, M. Sun, S. Whiteson, Is independent learning all you need in the starcraft multi-agent challenge? 2020, https://arxiv.org/abs/2011.09533.

[96] D. Mukherjee, K. Gupta, L.H. Chang, H. Najjaran, A survey of robot learning strategies for human-robot collaboration in industrial settings, Robot. Comput.-Integr. Manuf. 73 (2022) 102231, http://dx.doi.org/10.1016/j.rcim.2021.102231.

[97] M.M. Drugan, Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms, Swarm Evol. Comput. 44 (2019) 228–246, http://dx.doi.org/10.1016/J.SWEVO.2018.03.011.

[98] R.D. Palmer, Maintenance Planning and Scheduling Handbook, McGraw-Hill New York, 2019.

[99] H.C. Siu, J. Peña, E. Chen, Y. Zhou, V. Lopez, K. Palko, K. Chang, R. Allen, Evaluation of human-AI teams for learned and rule-based agents in Hanabi, Adv. Neural Inf. Process. Syst. 34 (2021).

[100] P. Madumal, T. Miller, L. Sonenberg, F. Vetere, Explainable reinforcement learning through a causal lens, in: AAAI 2020 - 34th AAAI Conference on Artificial Intelligence, 2020, pp. 2493–2500, http://dx.doi.org/10.1609/aaai.v34i03.5631.

[101] Z. Juozapaitis, A. Koul, A. Fern, M. Erwig, F. Doshi-Velez, Explainable reinforcement learning via reward decomposition, in: Proceedings of the IJCAI 2019 Workshop on Explainable Artificial Intelligence, 2019, pp. 47–53.