Computers & Industrial Engineering 213 (2026) 111813

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Beyond static schedules: Dynamic maintenance optimization with double deep reinforcement learning

Allan Jonathan da Silva [a,b,*], Luís Domingues Tomé Jardim Tarrataca [c], Leonardo Fagundes de Mello [a], Fabricio Maione Tenório [b], Rodrigo Rodrigues de Freitas [b,e], Felipe do Carmo Amorim [d], Marcio Antelio Neves da Silva [b], Cintia Machado de Oliveira [b,e]

[a] National Laboratory for Scientific Computing - LNCC, Av. Getulio Vargas, 333 - Quitandinha, Petrópolis, 25651-075, RJ, Brazil
[b] Production Engineering Department, Celso Suckow da Fonseca Federal Center for Technological Education of Rio de Janeiro - CEFET/RJ, Rod. Governador Mário Covas, s/n - Santana, Itaguaí, 23812-101, RJ, Brazil
[c] Computing Engineering Department, Celso Suckow da Fonseca Federal Center for Technological Education of Rio de Janeiro - CEFET/RJ, Rua do Imperador, 971 - Centro, Petrópolis, 25620-003, RJ, Brazil
[d] Postgraduate Program in Mechanical Engineering and Materials Technology (PPEMM), Celso Suckow da Fonseca Federal Center for Technology and Education (CEFET/RJ), Av. Maracanã, 229 - Maracanã, Rio de Janeiro, 20271-110, RJ, Brazil
[e] Postgraduate Program in Energy and Society (PPGES), Celso Suckow da Fonseca Federal Center for Technology and Education (CEFET/RJ), Av. Maracanã, 229 - Maracanã, Rio de Janeiro, 20271-110, RJ, Brazil

ARTICLE INFO

Keywords:
Maintenance optimization
Deep reinforcement learning
Reliability engineering
Dynamic scheduling
Artificial intelligence
Reliability-centered maintenance

ABSTRACT

This study presents an adaptive framework for dynamic preventive maintenance optimization based on the Double Deep Q-Network (DDQN) algorithm. The objective is to learn cost-optimal preventive maintenance policies under stochastic and partially observable failure behavior, relying solely on observed failure and maintenance events rather than condition-monitoring data or known degradation models. Equipment hazard function is modeled using non-homogeneous Poisson processes, including power-law and bathtub models, while maintenance actions follow variable restoration levels defined through the proportional age-reduction model. Training is performed on simulated failure trajectories using a standard workstation in under two hours, and the trained agent performs inference nearly instantaneously.

Results demonstrate that the DDQN-based adaptive policy consistently outperforms analytical periodic and static benchmarks, as well as a dynamic genetic algorithm and a standard reinforcement learning implementation, by achieving lower average maintenance costs and reduced variability across a wide range of corrective-to-preventive cost ratios. The method remains robust under perturbed and uncertain hazard conditions, maintaining stable performance without retraining.

These findings highlight the potential of the proposed DDQN approach as a computationally efficient and generalizable tool for reliability-centered maintenance optimization, capable of adapting to stochastic cost structures and cumulative corrective effects while operating effectively in data-limited industrial environments.

1. Introduction

The industrial sector, which is characterized by complex machinery and equipment, requires rigorous maintenance to ensure operational efficiency, cost control, and safety. However, maintenance activities often involve substantial expenses. According to Harudin and Yusof (2014), the U.S. industry spends nearly 80% of its maintenance budget for correcting chronic failures. In the United Kingdom, Kumar and Parida (2008) found that maintenance spending in the manufacturing sector ranges from 12 to 23% of total factory operating costs. These figures highlight the need for effective cost-reduction strategies, making optimal maintenance scheduling crucial for the efficient allocation of resources across industries.

As argued by Swanson (2001), maintenance plays a critical role in extending equipment lifespan and enhancing productivity in industrial environments. The significant impact of maintenance on the operational expenditure of a facility and its overall performance has been extensively documented in the literature (see e.g. Pintelon and Parodi-Herz (2008)). One of the key mechanisms through which these effects are achieved is the maintenance scheduling process, which involves

* Corresponding author at: National Laboratory for Scientific Computing - LNCC, Av. Getulio Vargas, 333 - Quitandinha, Petrópolis, 25651-075, RJ, Brazil.
E-mail address: allanjs@lncc.br (A.J. da Silva).

https://doi.org/10.1016/j.cie.2026.111813
Received 21 February 2025; Received in revised form 6 January 2026; Accepted 8 January 2026
Available online 10 January 2026

systematic planning and resource allocation to optimize equipment availability while minimizing downtime (Duffuaa et al., 1999).

Different forms of maintenance, namely preventive, predictive (condition-based maintenance), and corrective, play distinct roles in maintenance strategies (Tsang, 2002). Preventive maintenance, based on routine and scheduled interventions, serves as a proactive approach to avoid unscheduled breakdowns and reduce costs. Predictive maintenance, in turn, leverages advanced analytical techniques to identify potential failures in advance, reducing repair expenses and downtime (Levitt, 2003). Moreover, efficient resource optimization, including the careful allocation of personnel and assets to maintenance tasks, remains essential for achieving cost-effectiveness (Kister & Hawkins, 2006).

Despite advances in predictive and condition-based maintenance, many industrial settings operate with limited or no access to condition-monitoring data owing to cost, feasibility, or technological constraints. In such contexts, existing data-driven or dynamic maintenance methods that rely heavily on continuous sensor information cannot be directly applied. This limitation highlights a critical challenge in reliability engineering: designing adaptive maintenance scheduling strategies that minimize costs when the decision-making agent has no direct information regarding the equipment condition or its underlying hazard function.

Recent studies in reinforcement learning (Bukhsh et al., 2025; Huang et al., 2020) have shown that RL-based agents can be effectively applied to maintenance optimization. However, these approaches often assume the availability of extensive condition-monitoring data to guide decision-making. As a result, their applicability in data-sparse industrial environments remains limited. Addressing this limitation is essential, since many industrial systems operate without full state observability or comprehensive condition feedback.

In this context, the present study aims to bridge this research gap by proposing a Deep Reinforcement Learning (DRL) framework capable of dynamically scheduling preventive maintenance actions solely based on observed failure events. Specifically, it employs a Double Deep Q-Network (DDQN) architecture that supports adaptive decision-making in the absence of condition-monitoring data. The agent was trained using simulated failure information derived from theoretical hazard models, including the power-law and bathtub profiles. This approach advances the field of reliability-centered maintenance toward more generalizable and realistic applications in which uncertainty and incomplete information are intrinsic.

This study introduces a Deep Reinforcement Learning approach for dynamically scheduling preventive maintenance, contributing to reliability engineering and reliability-centered maintenance by reducing overall maintenance costs.

1.1. Motivation

In industrial production, machines deteriorate at different rates owing to variations in usage, environmental conditions, and inherent system complexity (Jardine et al., 2006). This variability creates significant challenges in defining maintenance schedules that balance operational efficiency, cost, and reliability. Unexpected failures can disrupt production, cause financial losses, and reduce system dependability. Traditional fixed-interval maintenance policies, although simple to implement, are not flexible enough to adapt to stochastic changes in failure behavior, costs, or operating conditions. Consequently, they often lead to excessive maintenance or unforeseen breakdowns. This limitation highlights the need for strategies that are both optimized and dynamically responsive to real-time events.

In many industrial settings, maintenance personnel often operate without access to continuous condition-monitoring data, a situation that remains common across several sectors. Many systems still rely exclusively on preventive or periodic maintenance actions, without real-time assessment of equipment health. For instance, Saraygord Afshari et al. (2022) notes that aircraft engine components, particularly in legacy fleets, follow preventive replacement cycles determined by flight hours rather than sensor-based degradation analysis. In the mining industry, large mechanical assets such as excavator shovels (Javadnejad et al., 2022) and dump trucks (Moniri-Morad & Sattarvand, 2023) are serviced at pre-defined intervals because of harsh operating conditions. Even in infrastructure systems such as railways (Sedghi et al., 2022), condition data are frequently unavailable or unreliable. Consequently, operators rely on historical failure statistics rather than real-time health indicators. These examples support the central premise of our study: it is both realistic and necessary to design adaptive maintenance frameworks that can perform effectively even when condition-monitoring data are not available. By addressing this common yet underexplored scenario, the proposed approach increases the practical relevance of reinforcement learning in industrial applications.

Existing dynamic and reinforcement learning based maintenance frameworks aim to enable decision-making by learning from experience. However, many of these models, including those proposed by Huang et al. (2020), Tanhaeean et al. (2025) and Bukhsh et al. (2025), assume access to condition monitoring data, known failure distributions, risk-free corrective maintenance actions, or explicit information on degradation states. Such assumptions restrict their practical use in industrial environments where direct condition data are unavailable, expensive to collect, or unreliable due to sensor limitations or infrastructure constraints. As a result, these models often fail to generalize to partially observable environments where only failure events can be observed. This limitation defines a key research challenge: developing a reinforcement learning framework capable of learning adaptive maintenance policies without condition monitoring data or predefined failure models.

In this study, we propose a Double Deep Reinforcement Learning (DDRL) framework to dynamically optimize preventive maintenance schedules based exclusively on observable failure and maintenance events. The DDRL structure builds on the Double Q-learning technique introduced by van Hasselt (2010) and later extended as Double Deep Q-Networks by van Hasselt et al. (2015). It is adapted here to a maintenance optimization setting characterized by stochastic failures, uncertain recovery effects, and incomplete information. DDRL is well-suited to this problem because it reduces the overestimation bias associated with standard DQN methods. This bias reduction is essential in maintenance optimization, since inaccurate value estimation may result in premature or suboptimal decisions. In addition, the decoupled evaluation and selection networks improve learning stability in sparse-reward and high-dimensional environments, which are typical in industrial maintenance scenarios where feedback is event driven and infrequent.

A preventive maintenance optimization problem aims to determine the most cost-effective schedule for performing maintenance actions on a system. The objective is to identify the optimal timing of preventive interventions in order to minimize total maintenance cost while ensuring acceptable reliability (Wang, 2002). However, the combinatorial nature of the planning task, particularly under uncertainty, makes it computationally demanding. The complexity results from the exponential growth of possible sequences of maintenance actions and failure occurrences over time, as illustrated below.

The combinatorial space of a perfect preventive maintenance optimization problem is vast and highly intricate. In a 60-day planning window, for instance, each day can represent a potential point for the first failure before the initial preventive maintenance. The complexity increases further because the first preventive maintenance can also occur on any of those 60 days, introducing an additional decision dimension.

The first branching of the search space occurs when the model considers whether a failure happens before the first preventive maintenance. This event creates a decision tree, where each branch represents

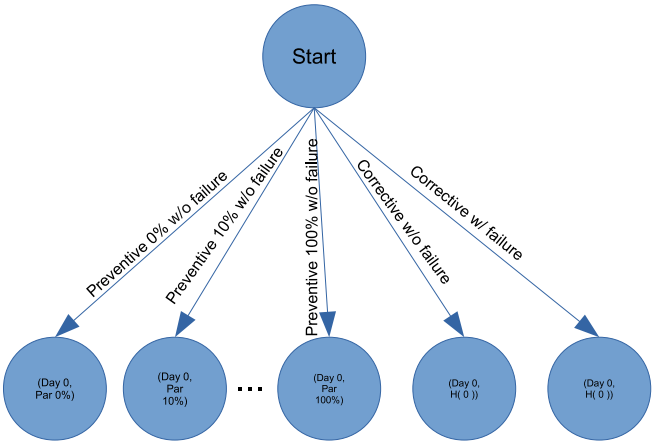

Fig. 1. Tree search example up to level 1.

a distinct system trajectory. Once the first preventive maintenance is executed, the search space bifurcates again, forming new branches that represent different possible days for the second preventive maintenance.

The problem becomes progressively more complex because each preventive maintenance restores the system to a condition in which new failures can arise. Therefore, every node in the decision tree that represents a preventive maintenance event generates several new branches, each corresponding to one or more possible failures within the 60-day horizon. This process repeats after every maintenance action, producing an exponentially expanding decision tree.

Fig. 1 illustrates this behavior. For clarity, only the first level of the search tree is shown. The root node (level 0) generates all possible configurations associated with Day 0 (Level 1). In our application, seven state variables are evaluated at each node, resulting in a branching factor of $b = 7$. Each resulting leaf node then expands into a subtree of identical structures that represent the next day. This recursive process continues until all 60 days have been explored, producing a tree of depth $d = 60$. The total number of computational paths from the root to any leaf node equals 7^{60}, which implies an exponential time complexity. Combinatorial optimization and scheduling problems of this nature are known to be NP-hard. Classic examples include single-machine scheduling with deadlines (Garey & Johnson, 1979), job-shop scheduling (Lenstra & Rinnooy Kan, 1978) and resource-constrained scheduling (Blazewicz et al., 1983).

Furthermore, the variability of the recovery factor, represented by the symbol ρ in this study, introduces an additional layer of complexity. Different ρ values modify system dynamics, affecting both the failure rate and the efficiency of preventive maintenance actions. When recovery factors and cost ratios are modeled as stochastic rather than deterministic variables, the optimization space becomes even more nonlinear and uncertain.

As discussed by Zio (2009), dynamically scheduling maintenance actions in response to evolving failure patterns is considerably more challenging than defining static policies, even in condition-based scenarios. Nevertheless, most existing approaches still rely on extensive condition monitoring data or pre-assumed degradation models. The proposed DDRL-based framework addresses this limitation by using reinforcement learning to derive maintenance strategies directly from stochastic failure event data, without the need for prior knowledge of the hazard function or underlying reliability parameters.

1.2. Problem statement and hypothesis

The core problem examined in this study concerns the limitations of maintenance scheduling methods in handling real-time failure dynamics and system variability under informational constraints. Standard strategies are predetermined and therefore unable to adapt to stochastic and evolving degradation patterns. As a result, they often lead to either premature preventive actions or delayed interventions, both of which increase maintenance costs and reduce system availability.

In practical industrial contexts, systems frequently operate under partial observability, where direct condition monitoring or sensor-based data are unavailable or impractical to obtain owing to economic or technical constraints. Under these circumstances, maintenance decisions must rely solely on indirect information, such as the timing and frequency of observed failures. This study considers a partially observable environment in which the maintenance agent has access only to failure event data, without explicit knowledge of the underlying hazard function, degradation trajectory, or system health state.

Based on these assumptions, the central research question addressed in this work is as follows: How can a reinforcement learning framework learn an optimal preventive maintenance policy that minimizes total maintenance cost when only stochastic failure events and maintenance actions, and not condition monitoring data, are available? To answer this question, the proposed model formulates the maintenance process as a Markov Decision Process (MDP) with a state representation restricted to observable failure-related variables. The objective is to design an adaptive scheduling policy using a deep reinforcement learning algorithm that is capable of dynamically adjusting preventive actions according to the probabilistic patterns inferred from these observable events, thereby achieving greater cost-effectiveness and adaptability than traditional static strategies.

1.3. Contribution

This study advances the field of industrial maintenance optimization by introducing a reinforcement learning framework that addresses the main limitations of the existing methods, particularly their reliance on risk-free maintenance actions, fully observable system states, and predefined degradation models. In contrast to most prior RL-based maintenance frameworks, which assume access to complete condition monitoring data or known hazard parameters, the proposed approach operates under informational constraints and learns optimal maintenance policies directly from observed failure and maintenance events. This formulation explicitly targets the research gaps highlighted by Zhang et al. (2024), who observed that *"the inherent limitation of observation methods and precision in engineering contexts often renders system states partially observable"* and that *"traditional MDP models assume precise knowledge of system dynamics, which is rarely the case in real-world applications"*. By formulating maintenance decision-making as a partially observable problem, this study contributes to the development of more realistic and practical reinforcement learning solutions for industrial environments.

The specific contributions of this study are summarized as follows:

1. *Dynamic Maintenance Strategies*
   This paper introduces an algorithm that enables the agent to autonomously plan maintenance actions in response to stochastic failure events and variable cost ratios. Unlike existing DRL-based models, which learn fixed policies assuming known degradation distributions, risk-free corrective maintenance actions, and fully observable system states, the proposed approach allows the policy to evolve dynamically as new failure information becomes available.

2. *DDRL Framework*
   The adoption of Double Deep Reinforcement Learning in this study is not merely an architectural preference but a methodological necessity for reliable decision-making in uncertain, partially observable environments. Traditional Deep Q-Networks are prone to overestimation bias when value estimates are derived from sparse or noisy rewards, a situation common in maintenance systems, where feedback occurs only after failures or preventive actions. By decoupling the action selection and target evaluation networks, the DDRL framework mitigates this bias, yielding more stable convergence and robust policy learning. Whereas previous studies have applied RL to deterministic systems, this study extends DDRL to inherently stochastic settings characterized by uncertain costs, imperfect maintenance effects, and unobservable degradation states. This methodological extension establishes the DDRL as a bias-correcting and stability-enhancing mechanism for maintenance learning tasks.
3. *Adaptive Learning under Partial Observability*
   A novel adaptive learning framework is introduced in which the agent operates under partial observability, without access to system condition, hazard rate, or failure distribution. The agent is trained with a belief representation of the environment through continuous rewards, relies solely on discrete failure and maintenance events as feedback (observation space), and incrementally constructs an internal policy that generalizes across different hazard patterns (e.g., power-law and bathtub models). This design directly addresses the challenges identified by Zhang et al. (2024), who emphasize the need for decision-making methods capable of handling uncertain parameters and hidden system states. Training the agent under such conditions enables robust adaptation to unseen operating regimes and stochastic cost variations.
4. *Cost-Oriented Formulation*
   This study formulates the optimization objective explicitly as the minimization of the total stochastic maintenance cost. The reward function integrates preventive and corrective costs, as well as recovery factor variability and cumulative failure costs, allowing the agent to learn cost-optimal policies across diverse operational scenarios. This cost-centric formulation enhances the practical relevance of the proposed policy for industries seeking to balance reliability and financial performance.
5. *Exploration of the Unseen Hazard Function*
   This study demonstrates that the DDRL agent can infer optimal preventive maintenance strategies without observing the underlying hazard or degradation function. The agent receives no information about the functional form or parameters of the risk model during inference, relying exclusively on the temporal patterns of failures and maintenance events. This represents a departure from prior DRL and reliability-centered maintenance studies that depend on predefined hazard models. By showing that effective maintenance decisions can be learned from event data alone, the proposed framework contributes both theoretical and practical advancements to reinforcement learning in reliability engineering.

Together, these contributions address a key gap in the literature by introducing a robust, cost-oriented, and dynamically adaptive reinforcement learning framework capable of operating under uncertainty and limited observability. The proposed DDRL approach strengthens both the theoretical understanding and the practical applicability of maintenance optimization in real-world industrial environments where condition monitoring data are scarce or unavailable.

1.4. Article outline

The remainder of this paper is structured as follows. Section 2 reviews the relevant literature on maintenance optimization and reinforcement learning. Section 3 presents the proposed cost model and reliability formulations, defines the problem within a Markov Decision Process framework, and details the implementation of the Double Deep Q-Network algorithm. Section 4 discusses the main results, beginning with the analysis of algorithmic hyperparameters and proceeding to the evaluation of maintenance cost performance for the trained agents. Finally, the last section concludes the paper and outlines directions for future research.

2. Literature review

Optimization of maintenance scheduling has been a central theme in industrial engineering, operations research, and computational science for decades. Classical optimization approaches, ranging from dynamic programming (Bellman, 1958) to genetic algorithms (Berrichi et al., 2009), integer linear programming, and algebraic formulations, have provided the foundation for preventive maintenance modeling and cost optimization. Subsequent studies such as Baek (2007) and Castro et al. (2014) extended these formulations to address reliability, availability, and cost trade-offs. However, these traditional optimization techniques are computationally intensive and require explicit knowledge of system failure models, which limits their adaptability in complex, uncertain, and data-scarce industrial environments.

Recent studies have explored the use of metaheuristic algorithms, particularly GA-based frameworks, to dynamically allocate maintenance activities and reduce downtime under uncertainty. For example, the study presented by Ruiz-Rodríguez et al. (2024) proposed a RL and a GA approach to optimize downtime assignment and workforce scheduling. Although this model successfully minimizes the mean time to repair through efficient labor allocation, its objective function focuses narrowly on downtime reduction rather than the minimization of total stochastic maintenance costs. Moreover, the model does not incorporate stochastic variability in cost structures or recovery factors, nor does it generalize the failure dynamics beyond simplified distributions. Ruiz-Rodríguez et al. (2024) acknowledged the high computational burden of GA optimization and the limitations of scaling such methods to larger problem spaces, which underscores the need for more sample-efficient and adaptive approaches such as deep reinforcement learning (DRL). Our study extends this direction by directly targeting total cost minimization under stochastic costs and failure behavior, rather than only reducing downtime.

The integration of DRL into maintenance optimization has shown promising results in terms of cost reduction and decision adaptability. The framework proposed by Bukhsh et al. (2025) introduced maintenance planning models with online and offline DRL to minimize maintenance costs of the water pipes network over time. Although effective, this approach relies on the assumption that the failure process follows an exponential distribution, with the failure rate and probability known to the agent. Consequently, the agent's decision-making is guided by prior knowledge of the degradation law, which simplifies the state representation but restricts generalization to real-world scenarios where the failure rate follows realistic distributions and is not completely known. Furthermore, the reward structure is based on deterministic or expected costs, rather than stochastic variations, which limits robustness against uncertainty or nonlinearities in maintenance costs or repair outcomes. Such characteristics can also be observed in Tanhaeean et al. (2025). By contrast, our proposed DDRL framework relaxes these assumptions by training under informational constraints, where only failure events, not their underlying rates or hazard parameters, are observable, and by incorporating stochastic cost structures directly into the learning process.

Recent contributions further illustrate both the potential and constraints of MDPs and RL algorithms in maintenance and reliability engineering in general. Yuan et al. (2025) proposed a DRL formulation that jointly optimizes maintenance cost and reliability through a hybrid Gamma-Wiener degradation model and a reshaped, explicitly multi-objective reward, achieving strong performance in safety-critical applications. Zhu et al. (2025) investigated joint condition-based maintenance and spare-parts sourcing under supply uncertainty, formulating the problem as an MDP and using DQN primarily as a scalable alternative to value iteration. Yang et al. (2025) addressed the coordination between dispatching and preventive maintenance in multi-product manufacturing systems via a continuous-time MDP formulation and structural analysis, establishing the existence of an optimal control-limit policy for PM decisions. da Silva et al. (2025) demonstrated the applicability of reinforcement learning in a distinct domain of reliability engineering by framing accelerated life test planning as a sequential parameter-estimation problem rather than a maintenance task. In their framework, a DDQN agent was trained to dynamically configure stress levels, test durations, and sample allocations in order to minimize the statistical uncertainty of a key reliability parameter.

Another relevant strand of research applies DRL to manufacturing systems, emphasizing production efficiency and loss minimization. Huang et al. (2020) developed a Double Deep Q-Network (DDQN) model to determine optimal preventive maintenance intervals in a serial production line with intermediate buffers. The model effectively captures the interaction between production flow and maintenance actions, but assumes that the component lifetimes follow a known Weibull distribution and that the exact virtual age of the system is observable to the agent. As a result, the framework depends on an accurate degradation model and are not extended to cases where the equipment condition is unobservable. Moreover, the reward function proposed by Huang et al. (2020) emphasizes minimizing production loss rather than explicitly minimizing the total maintenance cost under uncertainty. Hence, although it demonstrates the potential of DRL in complex manufacturing contexts, it remains limited to fully observable settings.

It is also important to note that the DDQN architecture of Huang et al. (2020) does not incorporate an adaptive policy capable of responding to stochastic variations in the operational environment. The learned policy in their work was derived under fixed, deterministic conditions, assuming stable production dynamics and a known failure distribution. Consequently, it does not adapt online to unexpected changes or to the emergence of new degradation patterns inferred from data. By contrast, the DDRL framework proposed in this study is designed to operate under intrinsic uncertainty, where failure occurrences, repair effect and cost parameters vary stochastically. The agent continuously interacts with an environment that evolves according to random failures and variable maintenance costs, learning an adaptive policy that dynamically reschedules maintenance actions in response to real-time events. This distinction highlights that, while the study by Huang et al. (2020) leverages the learning capability of DRL to optimize a predefined structure, our approach explicitly extends the paradigm toward responsive and adaptive maintenance decision-making under uncertainty.

Several other studies have similarly employed RL or metaheuristic algorithms to optimize maintenance actions in predictive contexts, such as (Tanhaeean et al., 2025), and including power generation scheduling (El-Sharkh & El-Keib, 2003), wind turbine systems (Zhong et al., 2019), and railway infrastructure (Sresakoolchai & Kaewunruen, 2023). Despite their methodological diversity, most rely on explicit knowledge of degradation laws or condition-monitoring data, such as vibration or temperature signals, to guide decision-making. As noted by Sikorska et al. (2011) and Ren et al. (2019), such reliance on sensor data limits the applicability of AI-based maintenance frameworks in environments where the monitoring infrastructure is incomplete or prohibitively costly. Furthermore, cost structures are frequently simplified to deterministic values, neglecting the stochastic nature of real maintenance operations, where cost ratios and recovery factors vary with time and context.

In light of these limitations, the present study advances the state of the art by addressing the challenge of cost-minimizing, event-driven maintenance dynamic scheduling under informational constraints. Specifically, the proposed DDRL method operates without access to condition-monitoring or hazard data, relying solely on observed failure events and maintenance actions to infer optimal maintenance policies. Unlike the GA-based method in Ruiz-Rodríguez et al. (2024), our approach circumvents the computational overhead of population-based search, yielding an inference that is nearly instantaneous.

3. Methodology

3.1. Reliability fundamentals

Reliability engineering has evolved to become an integral part of ensuring efficiency, availability, maintainability, safety, and cost-effectiveness in diverse sectors ranging from aviation to manufacturing (see e.g. Zio et al., 2019 and Nor et al., 2021). This discipline is responsible for identifying requirements, analyzing, designing, verifying, validating, assuring quality, and maintaining various systems to guarantee their reliability under prescribed conditions and timeframes (Zio et al., 2019). The significance of reliability engineering is highlighted by its irreplaceable role in promoting progress across major industries, and improving the quality of a wide range of products and systems (Zuo, 2021).

Modern multidisciplinary maintenance programs in reliability engineering aim to enhance system reliability by considering the operational characteristics, production necessity, and other vital factors (Patiño-Rodriguez & Carazas, 2019). This field is particularly relevant in the development and production of sophisticated equipment and systems, which must be achieved within shorter timespans and under stringent costs and legal constraints (Birolini, 1996).

The following mathematical definitions of reliability engineering relevant to this study can be found in Meeker and Escobar (2014) and in Elsayed (2021). Let $f(t)$ be the failure probability density function such that

$$f(t) \geq 0, \quad \forall t \geq 0,$$

and

$$\int_0^\infty f(s)ds = 1.$$

The corresponding probability of failure from time 0 to time t is given by:

$$F(t) = \int_0^t f(s)ds, \tag{1}$$

and the survival or reliability function is given by

$$C(t) = 1 - F(t) = \int_t^\infty f(s)ds. \tag{2}$$

The following function

$$h(t) = \lim_{\Delta t \to 0} \frac{C(t) - C(t+\Delta t)}{\Delta t C(t)} = \frac{1}{C(t)}\left[-\frac{d}{dt}C(t)\right] = \frac{f(t)}{C(t)} \tag{3}$$

is known as the hazard function. The hazard function in reliability engineering refers to the instantaneous failure rate of a system at a specific time, given that it has survived up to that time (Esary et al., 1970). This is a fundamental concept used to model the probability of a system failing at a given moment, thereby providing insights into the system's reliability over time.

Suppose that, in failure event, minimal repairs are executed. Thus, according to Elsayed (2021), the expected number of failures in $[0, t]$ is given by:

$$H(t) = \int_0^t h(s)ds. \tag{4}$$

3.2. Reliability-centered maintenance

Following Pham (2003) and da Silva (2023), maintenance encompasses measures taken to manage the deterioration process that could potentially result in system failure. They can be categorized into two types: maintenance and corrective maintenance. Preventive maintenance focuses on preventing system failures by conducting periodic inspections and repairs, whereas corrective maintenance involves restoring the system to its functional state after a failure occurs through appropriate actions. The effectiveness of equipment post-repair is contingent upon the type of repair executed. In this study, we address the scheduling problem in terms of cost minimization.

Consider a scenario in which the equipment is only minimally repaired when it fails, and the impact of imperfect preventive maintenance is modeled using the Proportional Age Reduction (PAR) criterion (Malik, 1979). If the maintenance was perfect, the equipment would be restored to its original state. The PAR approach, used by Pham (2003) and da Silva (2023), assumes that each preventive maintenance action reduces the equipment's age by a proportionate amount to the operating time that has elapsed since the most recent scheduled maintenance.

A Nonhomogeneous Poisson process (Elsayed, 2021) was used to describe the failure patterns in each maintenance cycle. In this process, the age of the equipment in the kth cycle is decreased by a fraction ρ of the most recent scheduled maintenance action τ_{k-1}. Following da Silva (2023), the hazard function (3) at time t is

$$h(t) = h(t - \rho\tau_{k-1}), \quad \tau_{k-1} < t < \tau_k. \tag{5}$$

Shin et al. (1996) proposed the following hypotheses to model minimal repairs interspersed with scheduled imperfect preventive maintenance actions:

1. Suppose that l units are observed until T_i, $i = 1, \ldots, l$.
2. Suppose that each equipment i is subjected to m_i scheduled maintenance actions at $\tau_{i,1} < \tau_{i,m_i} \leq T_i$.
3. The ith equipment experiences $r_{i,k}$ failures during the kth preventive maintenance cycle $(k = 1, \ldots, m_i + 1)$
4. Let $t_{i,k,j}$ be the time of the jth failure of the ith equipment that occurs in the kth maintenance cycle.

The maintenance model adopted in this study considers that preventive actions with possibly varied recovery factors are executed periodically. Let

- c_p be the preventive maintenance cost;
- c_m be the corrective maintenance cost.

Following Pham (2003), the expected maintenance costs for periods $[t_{m+1}, t_{m+2}]$ are given by.

$$V(t_{m+1}, t_{m+2}) = \frac{c_m H(t_{m+1}, t_{m+2}) + c_p}{t_{m+2} - t_{m+1}}, \tag{6}$$

where $H(t_{m+1}, t_{m+2}) = \int_{t_{m+1}}^{t_{m+2}} h(s)ds$. The static maintenance strategy is determined by finding the preventive maintenance intervals $[t_m, t_{m+i}]$; $[t_{m+i}, t_{m+i+1}]$... which minimizes the expected overall maintenance cost V given by Eq. (6).

To model the failure behavior of systems over time, we employed two forms for the hazard function in this study: the power law model given by

$$h(t) = \frac{\beta}{\alpha}\left(\frac{t}{\alpha}\right)^{\beta-1}, \qquad \alpha, \beta > 0, \tag{7}$$

and the bathtub model is given by

$$h(t) = \frac{\beta_1}{\alpha_1}\left(\frac{t}{\alpha_1}\right)^{\beta_1-1} + \frac{\beta_2}{\alpha_2}\left(\frac{t}{\alpha_2}\right)^{\beta_2-1}, \ \alpha_1, \beta_1, \alpha_2, \beta_2 > 0. \tag{8}$$

The parameters α and β control the level and slope of the risk function, respectively. The latter represents the class of non-monotonic phenomena hazard functions, as discussed by Diamoutene et al. (2021). Interestingly noted by Gaonkar et al. (2021) is that, the bathtub function is inappropriate for predicting the hazard rates of electronic components, products, and systems. The authors provided several case studies of electronic equipment where the hazard function is given by the power law model (7), which is derived from the Weibull distribution via Eq. (3).

In reliability engineering, estimating the parameters of hazard functions is crucial for modeling the failure behavior of systems. The maximum likelihood estimation (MLE) method is commonly used to derive these parameters from observed failure data (reliability applications can be found in Khan & King, 2012 and Murthy, 1979). MLE aims to identify the values of parameters that maximize the likelihood of the observed data occurring given a specific probability distribution or model.

Following Pham (2003), the likelihood function for a set of m preventive maintenance actions, r failures of l identical machines, is given by

$$L = \prod_{i=1}^{l}\left\{\prod_{k=1}^{m_i+1}\left[\prod_{j=1}^{r_{i,k}} h(t_{i,j,k} - \rho\tau_{i,k-1})\right] \times \exp\left[-\sum_{k=1}^{m_i+1}\int_{\tau_{i,k-1}}^{\tau_{i,k}} h(x - \rho\tau_{i,k-1}) - dx\right]\right\}. \tag{9}$$

The methodology encompasses the simulation of failure times using bathtub model (8) to emulate the deterioration of the system. This function enables the modeling of failure patterns, especially for repairable systems, as observed in practical scenarios.

The bathtub function models the initial higher failure rates (early failures) followed by a period of lower (random failures) relatively constant failure rates which is subsequently followed by an increasing failure rate. The power law function captures only the increasing failure pattern as systems age (wear-out failures).

In order to simulate corrective maintenance action times, Algorithm 1 was used to generate random failures:

Algorithm 1 Failure Time Generation under PAR and Risk Factor Model

Require: Current time T_k, last failure time F_k, last preventive maintenance time M_k, age reduction factor ρ, risk multiplier θ, maximum risk $\theta_{\max}$, Weibull parameters (α_1, β_1)

Ensure: Real failure time T_{k+1}^{fail} and failure indicator fail $\in \{0, 1\}$

function LocalDeltaH($T_k, M_k, \rho, \beta_1, \alpha_1$)
  Compute effective age interval under PAR:
  $a = \max((T_k - 1) - \rho M_k, 0), \qquad b = \max(T_k - \rho M_k, 0)$
  Compute hazard increment:
  $\Delta H = (b/\alpha_1)^{\beta_1} - (a/\alpha_1)^{\beta_1}$
  return $\max(\Delta H, 0)$
end function

function LocalFailureTime($T_k, F_k, M_k, \rho, \theta, \theta_{\max}, \beta_1, \alpha_1$)
  Compute hazard increment ΔH = LocalDeltaH($T_k, M_k, \rho, \beta_1, \alpha_1$)
  Compute capped risk: $\theta^* = \min(\max(\theta, 10^{-6}), \theta_{\max})$
  Compute instantaneous failure probability:
  $p = 1 - \exp(-\theta^* \Delta H)$
  Draw $u \sim \text{Uniform}(0, 1)$
  if $u < p$ **then**
    $T_{k+1}^{\text{fail}} \leftarrow T_k$ ▷ Failure occurs in the current step
    fail $\leftarrow 1$
  else
    $T_{k+1}^{\text{fail}} \leftarrow F_k$ ▷ No new failure in $(T_k - 1, T_k]$
    fail $\leftarrow 0$
  end if
  return $(T_{k+1}^{\text{fail}}, \text{fail})$
end function

function FailureGenerator($T_0, H, M_0, \rho, \theta, \theta_{\max}, \alpha_1, \beta_1$)
  Initialize $F_0 \leftarrow 0$
  for $k = 1, 2, \ldots, H$ **do**
    $(F_k, \text{fail}) \leftarrow$ LocalFailureTime($T_k, F_{k-1}, M_{k-1}, \rho, \theta, \theta_{\max}, \beta_1, \alpha_1$)
    if fail $= 1$ **then**
      Record failure event at time T_k
    end if
  end for
end function

Elsayed (2021) defined the cost incurred by the system during period t_0, comprising the sum of the corrective maintenance action cost times the number of failures and the preventive maintenance action cost, called the total maintenance cost, as follows:

$$V(t_0) = c_c H(t_0) + c_p, \tag{10}$$

where c_c represents the cost of each corrective maintenance action, $H(t_0)$ denotes the number of failures encountered up to time t, and c_p signifies the preventive maintenance cost incurred per action. Then, the reward function R to be minimized is given by:

$$R(t_i) = c_c \times \text{condition}(t_i) + c_p \times \text{PM action}(t_i), \tag{11}$$

for $i = 1, 2, \ldots, n$, where 'condition' is a binary variable representing the actual status of the equipment, and 'PM action' is a binary variable that describes if a preventive maintenance action was performed or not.

3.3. Markov decision process

We propose the model exhibited in Fig. 2, which integrates the DDQN algorithm with a preventive maintenance framework. By observing state variables, interacting with the environment through maintenance actions, and receiving penalties in the form of maintenance costs, the agent schedules preventive maintenance actions aiming at minimizing the costs.

In order to formalize the reinforcement learning framework, the preventive maintenance optimization problem is cast as a Markov Decision Process (MDP) (see Sutton & Barto, 2018). The MDP is defined by the tuple

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma),$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $\mathcal{P}$ the transition kernel, $\mathcal{R}$ the reward function, and $\gamma \in (0, 1)$ the discount factor.

State space. The state representation was designed to capture all observable aspects of the system's operational history while intentionally excluding condition-based variables, since the proposed framework explicitly addresses partially observable environments where no direct condition-monitoring data are available (see Zhang et al., 2024). At decision epoch $k \in \mathbb{N}$, the system state is represented by the 7-dimensional vector

$$s_k = (F_k,\ T_k,\ A_k,\ f_k^{\text{PM}},\ f_k^{\text{tot}},\ a_{k-1},\ M_k) \in \mathcal{S},$$

where:

- $F_k \in \mathbb{N}$ is the last failure time (days),
- $T_k \in \mathbb{N}$ is the current time,
- $M_k \in \mathbb{N}$ is the time of the last preventive maintenance.
- $A_k \in \mathbb{R}^+$ is the effective age of the equipment after the most recent preventive action, computed under the PAR model as $A_k = \max(T_k - \rho_{k-1} M_k, 0)$,
- $f_k^{\text{PM}} \in \mathbb{N}$ is the number of failures since the last preventive maintenance,
- $f_k^{\text{tot}} \in \mathbb{N}$ is the cumulative number of failures up to epoch k,
- $a_{k-1} \in \mathcal{A} \in [0, 1]$ is the previous preventive maintenance action,

All state variables are normalized in the environment to the unit interval using problem-specific scales (e.g., $T_k/60$, $f_k^{\text{PM}}/2$, $f_k^{\text{tot}}/10$), ensuring numerical stability during training. This representation enables the process to remain Markovian under imperfect maintenance conditions, since the variables $(A_k, f_k^{\text{PM}}, M_k)$ preserve sufficient information about system history to characterize its degradation trajectory. Furthermore, these choices enable the agent to be responsive and act dynamically to the current environmental condition.

It is assumed that following each event occurring at T_k, either a failure or a preventive maintenance action, the equipment resumes operation, and consequently its evaluation, only at T_{k+1}. This assumption renders the proposed approach a discrete-event simulation framework.

Formally, the state space is endowed with the product σ-algebra

$$\Sigma = \mathcal{B}(\mathbb{R}_+)^{\otimes 6} \otimes 2^{\{0, 0.25, 0.50, 0.75, 1\}},$$

ensuring measurability of both transition probabilities and reward mappings.

Following the taxonomy proposed by Powell (2019), the components of s_k are predominantly informational state variables that summarize the observable operational history of the system. Quantities such as the last failure time (F_k), the last preventive maintenance time (M_k), the effective age (A_k), and the accumulated failure counts $(f_k^{\text{PM}}, f_k^{\text{tot}})$ encode sufficient information about past events to infer the unobserved degradation state. The current time T_k represents the only physical variable, whereas previous action a_{k-1} provides a control-related historical marker. Although the belief variables are not explicitly modeled, the agent's value function implicitly infers probabilistic expectations about future failures conditioned on these historical summaries.

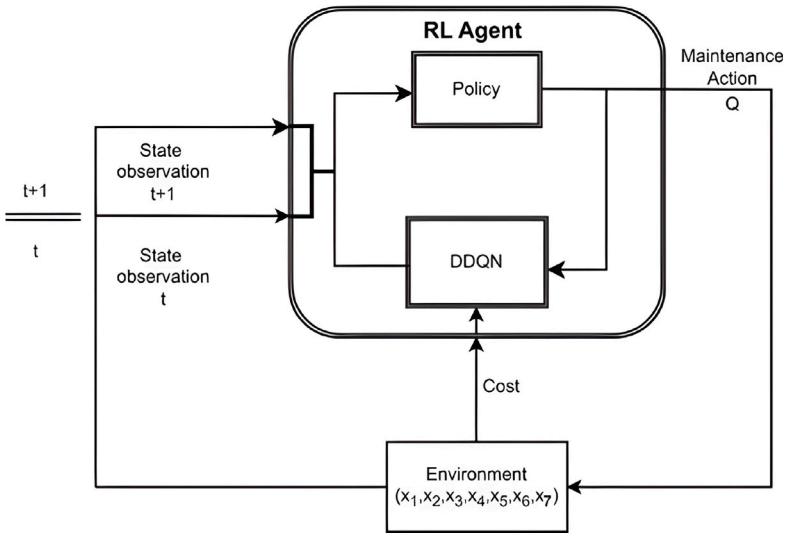

Fig. 2. Maintenance optimization RL model.

Action space. At each epoch the agent selects an action $a_k \in \mathcal{A} = \{0, 1\}$ for the perfect maintenance case, and

$$a_k \in \mathcal{A} = \{0, 0.25, 0.50, 0.75, 1\},$$

for the imperfect maintenance case, where $a_k = 0$ represents no preventive maintenance, and $a_k > 0$ corresponds to a preventive action with proportional age reduction factor $\rho = a_k$. Whenever a failure occurs, corrective maintenance is automatically performed, regardless of the chosen a_k.

Transition kernel. The system evolves stochastically according to the underlying hazard function and the selected action. Given (s_k, a_k), the environment computes the hazard increment ΔH_k between T_k and $T_k + \Delta t$ based on the PAR-adjusted virtual age, and defines the instantaneous failure probability as

$$p_k = 1 - \exp(-\theta_k \, \Delta H_k), \tag{12}$$

where $\theta_k = 1 + \kappa_{\text{risk}} \, \mathbf{1}_{\{F_k > M_k\}}$ represents the additional risk introduced by corrective repairs since the last preventive action. A failure indicator is then sampled as $F_k^{\text{fail}} \sim \text{Bernoulli}(p_k)$.

If $a_k > 0$, a preventive maintenance action is executed, the variable M_{k+1} is updated to $T_k + \Delta t$, and the local failure counter f_{k+1}^{PM} is reset to zero. Otherwise, the system continues to age normally. Whenever $F_k^{\text{fail}} = 1$, a corrective repair is applied (without age reduction), and both counters f^{PM} and f^{tot} are incremented. This defines the transition kernel $\mathcal{P}(s_{k+1} \mid s_k, a_k)$ on (S, Σ).

Reward function. During training, at each epoch, the environment computes a dense-shaped normalized step cost that provides continuous feedback even in the absence of failures. Let c_p and c_c denote the base average normalized preventive and corrective maintenance costs, respectively, and let $\kappa_{\text{cost}} > 0$ be a sensitivity factor penalizing consecutive failures since the last preventive action. The instantaneous cost is defined as follows:

$$C(s_k, a_k) = \frac{c_p \, a_k + c_c \, (1 + \kappa_{\text{cost}} f_k^{\text{PM}}) \, p_k}{\max(c_p, \; c_c \, (1 + \kappa_{\text{cost}} f_k^{\text{PM}}))}, \tag{13}$$

and the reward is given by its negative value,

$$R(s_k, a_k) = -C(s_k, a_k). \tag{14}$$

This formulation ensures a normalized and bounded reward signal, improving learning stability in sparse-failure environments.

It is important to note that, during training, the agent interacts with the environment through the shaped instantaneous cost $C(s_k, a_k)$, which ensures the presence of a dense and continuous reward signal even in failure-free trajectories. However, the agent is evaluated using the true cumulative cost function defined in Eq. (11), which is consistent with the economic formulation of the maintenance problem and the analytical benchmarks. This design choice allows the policy to learn efficiently under a smoother optimization scenario while still being assessed according to the actual operational performance criterion. Furthermore, by relying on the term p_k within the shaped cost during the training, the agent implicitly develops a belief about the current hazard rate through the estimated increment ΔH_k, enabling it to infer and act upon the underlying risk dynamics even when the true degradation process deviates from the nominal model.

Bellman equations. In the present maintenance optimization setting, the Bellman recursion formalizes the trade-off between immediate maintenance costs and the expected future deterioration risk. Let $V^{\pi}(s_k)$ denote the expected discounted return starting from state s_k under a policy π. The corresponding Bellman equation is

$$V^{\pi}(s_k) = \mathbb{E}\big[-C(s_k, \pi(s_k)) + \gamma \, V^{\pi}(s_{k+1}) \mid s_k\big], \quad s_{k+1} \sim \mathcal{P}(\cdot \mid s_k, \pi(s_k)). \tag{15}$$

Here, the first term captures the instantaneous maintenance expenditure, either preventive, proportional to the restoration factor ρ, or corrective, conditional on a stochastic failure, whereas the second term expresses the expected future operational cost under continued degradation. The optimal state–action value function $Q^*(s_k, a_k)$ satisfies

$$Q^*(s_k, a_k) = -C(s_k, a_k) + \gamma \, \mathbb{E}_{s_{k+1} \sim \mathcal{P}(\cdot \mid s_k, a_k)} \left[\min_{a_{k+1} \in \mathcal{A}} Q^*(s_{k+1}, a_{k+1}) \right], \tag{16}$$

where the minimization reflects the cost-minimization formulation of the problem (as opposed to reward maximization). Therefore, the optimal preventive maintenance policy is obtained as

$$\pi^*(s_k) = \arg\min_{a_k \in \mathcal{A}} Q^*(s_k, a_k), \tag{17}$$

which yields the restoration level $\rho \in [0, 1]$ that minimizes the expected long-term maintenance cost.

Unlike conventional MDP formulations, the transition operator $\mathcal{P}$ here implicitly encodes both the degradation law and the stochastic occurrence of failures through the hazard increment ΔH_k, whose dependence on ρ links the preventive decision to future system reliability. This coupling makes the Bellman operator domain-specific: each action not only incurs an immediate cost but also alters the virtual

age distribution that determines the subsequent failure probability. In this sense, the Bellman equation captures the recursive propagation of maintenance effectiveness over time, providing a dynamic balance between short-term expenditure and long-term reliability.

3.4. Double deep reinforcement learning algorithm

In order to find the optimal maintenance policies, the reinforcement learning method applied in this study utilizes the double deep Q-learning algorithm, which is a variation of the Q-learning approach, that can be found in the reference book of Sutton and Barto (2018). Szepesvári (2010) presents some reinforcement learning algorithms.

The Double Deep Q-Network (DDQN) algorithm, an advancement in reinforcement learning introduced by van Hasselt et al. (2015), is essential in addressing the overestimation bias of Q-values and stabilizing training. This is achieved through a novel approach of decoupling action selection from target Q-value estimation by employing two distinct neural networks: policy and target networks.

The efficacy of DDQN extends across various domains, as demonstrated by a variety of applications, such as enhancing driving safety and fuel economy in autonomous vehicles through vehicle-to-infrastructure communication networks (Liu et al., 2020), optimizing timeslot scheduling in network traffic (Ryu et al., 2023), controlling hysteresis phenomena in mode-locked fiber lasers (Kokhanovskiy et al., 2022), improving relay selection and power allocation in secure cognitive radio networks (Huang et al., 2021), optimizing hospital occupancy (Rajendran & Geetha, 2021), financial trading strategies (Brim, 2020), and dynamic planning of transmission networks (Wang et al., 2021).

The key details of DDQN implementation, which can be found in Ravichandiran (2020), include the following:

- *Double Q-Value Estimation:* Two separate Q-value networks are employed, one for selecting the best action and the other for evaluating its value. This decoupling mitigates the overestimation bias inherent to traditional Q-learning and improves robustness when learning from sparse and noisy maintenance cost signals.
- *Target Q-Network Updates:* The target network parameters are updated periodically to stabilize learning. By keeping the target network fixed for several steps and updating it incrementally with the online network parameters, this mechanism reduces oscillations in the target estimates and enhances the convergence stability under stochastic failure dynamics.
- Neural Network Configuration: A feed-forward neural network architecture with fully connected hidden layers, followed by an activation function.
- Exploration-Exploitation Strategy: Implements an $\epsilon-$greedy exploration strategy to balance exploration and exploitation in action selection.
- *Experience Replay:* A replay buffer stores and samples past transitions to break temporal correlations between consecutive maintenance decisions. This improves sample efficiency and ensures more stable convergence by allowing the agent to learn from a representative mix of past experiences rather than from sequentially correlated episodes.

The output of the neural networks exhibits variability based on the specific application context. In scenarios where the agent's decision-making is limited to a binary choice between perfect maintenance or non-maintenance of the equipment, the neural network architecture is configured with two distinct output nodes, as shown in Fig. 3. Conversely, in more complex decision-making situations where the agent is tasked with determining whether maintenance is performed and specifying the corresponding restoration level, the neural network output encompasses Q-values for a range of five discrete values associated with the recovery factor ρ, which were evenly distributed between 0 and 1, as shown in Fig. 4. This dichotomy in the neural network output configuration is influenced by the specific decision-making requirements of a given application: perfect or imperfect maintenance, reflecting the need for adaptability and precision in different operational contexts.

Figs. 3 and 4 provide visual representations of the neural network outputs corresponding to these distinct decision scenarios.

Algorithm 2 Double Deep Q-Network (DDQN) for Preventive Maintenance with 7-State PAR Model

Initialize online network $Q(s,a;\theta)$ and target network $Q(s,a;\theta^-)$ with random weights
Initialize replay buffer D
for episode $= 1, 2, \ldots, E$ **do**
  Initialize environment using `resetFcn`, obtaining:
  $s_0 = (\text{FailTime}_0,\ T_0,\ \text{AgeEff}_0,\ f_{\text{PM},0},\ f_{\text{tot},0},\ a_{-1},\ \text{PMtime}_0)$
  for step $k = 0, 1, \ldots$ until episode horizon H **do**
    With probability ϵ, select random action $a_k \in \mathcal{A}$, otherwise select greedy action $a_k = \arg\max_a Q(s_k, a; \theta)$
    Apply action a_k in environment via `stepFcn`:
    if $a_k > 0$ **then**
      Set preventive maintenance level $\rho = a_k$
      Reset fails_since_PM $\leftarrow 0$
      Update PM_last $\leftarrow T_k + \Delta t$
      Add preventive cost c_p
    end if
    Compute effective risk factor $\theta = 1 + \kappa_{\text{risk}} \cdot \mathbb{1}_{\{\text{fail since last PM}\}}$
    Sample failure occurrence $F_k \sim \text{Bernoulli}(1 - e^{-\theta \Delta H})$
    if $F_k = 1$ **then**
      fails_since_PM $\leftarrow$ fails_since_PM $+ 1$,
      total_fails $\leftarrow$ total_fails $+ 1$
      Apply corrective maintenance (minimal repair)
      Add corrective cost $c_c(1 + \kappa_{\text{cost}} \cdot \text{fails\_since\_PM})$
      Update last failure time FailTime $\leftarrow T_k + \Delta t$
    end if
    Update effective age: AgeEff $= \max((T_k + \Delta t) - \rho \cdot \text{PM\_last}, 0)$
    Normalize new state:
    $s_{k+1} = (\text{FailTime}_n,\ T_n,\ \text{AgeEff}_n,\ f_{\text{PM},n},\ f_{\text{tot},n},\ a_k,\ \text{PMtime}_n)$
    Compute dense-shaped cost
    $$\text{step\_cost} = \frac{c_p a_k + c_c(1 + \kappa_{\text{cost}} f_{\text{PM}}) p_{\text{fail}}}{\max(c_p,\ c_c(1 + \kappa_{\text{cost}} f_{\text{PM}}))}$$
    Reward $r_k = -\text{step\_cost}$
    Store (s_k, a_k, r_k, s_{k+1}) in replay buffer D
    Sample minibatch $\{(s_j, a_j, r_j, s_{j+1})\}$ from D
    for each sample (s_j, a_j, r_j, s_{j+1}) **do**
      $a^* = \arg\max_{a'} Q(s_{j+1}, a'; \theta)$
      $Y_j^{\text{DDQN}} = r_j + \gamma Q(s_{j+1}, a^*; \theta^-)$
      Compute loss $L_j = (Y_j^{\text{DDQN}} - Q(s_j, a_j; \theta))^2$
    end for
    Update $\theta \leftarrow \theta - \eta \nabla_\theta L_j$
    if $k \bmod \tau = 0$ **then**
      Update target network: $\theta^- \leftarrow \theta$
    end if
    $s_k \leftarrow s_{k+1}$
  end for
end for

The simulations of this study were performed using MATLAB (The MathWorks Inc., 2022). The reinforcement learning toolbox provides appropriate functions for training reinforcement learning algorithms and achieving optimal policies. The rlDQNAgent function was used in this study.

3.4.1. Double deep Q-learning update rule

To learn the optimal policy in the proposed MDP, we propose the DDQN Algorithm 2. At each training step, the agent samples a transition (s_k, a_k, r_k, s_{k+1}) from the replay buffer, where s_k is the current state, a_k is the selected preventive maintenance intensity, $r_k = -R(s_k, a_k)$ is the immediate reward (negative of the maintenance cost), and s_{k+1} is the next state sampled according to the kernel $\mathcal{P}(\cdot|s_k, a_k)$, with failure indicator drawn from a Bernoulli distribution with parameter $p^{\text{true}}(s_k, a_k)$.

The DDQN target value is defined as

$$Y_k^{\text{DDQN}} = r_k + \gamma\, Q\Big(s_{k+1}, \arg\max_{a' \in \mathcal{A}} Q(s_{k+1}, a'; \theta)\,;\, \theta^-\Big), \tag{18}$$

where:

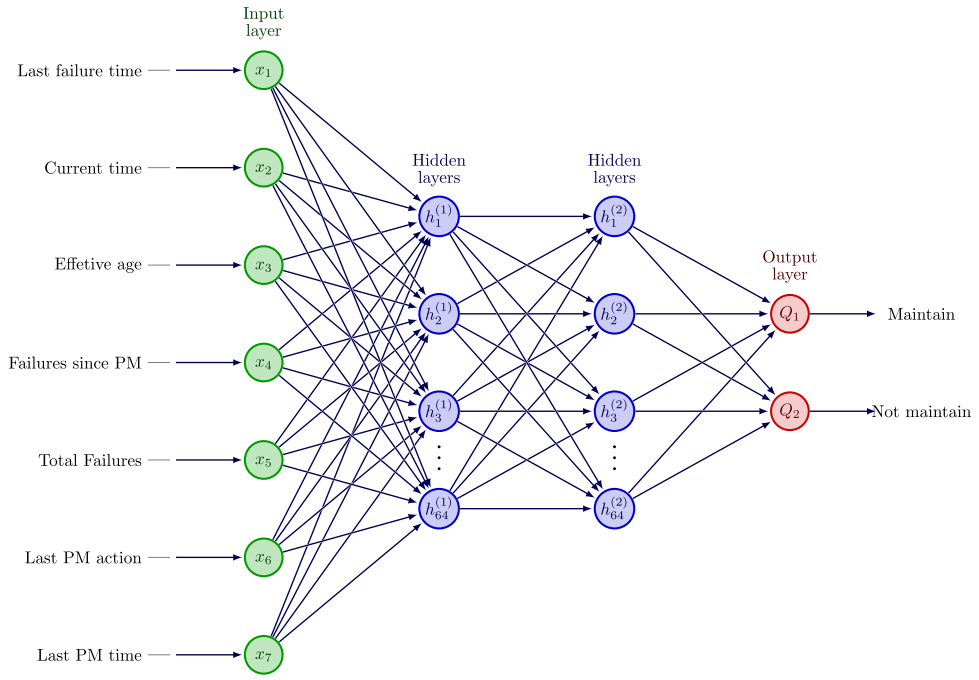

Fig. 3. Neural network architecture for the perfect maintenance case.

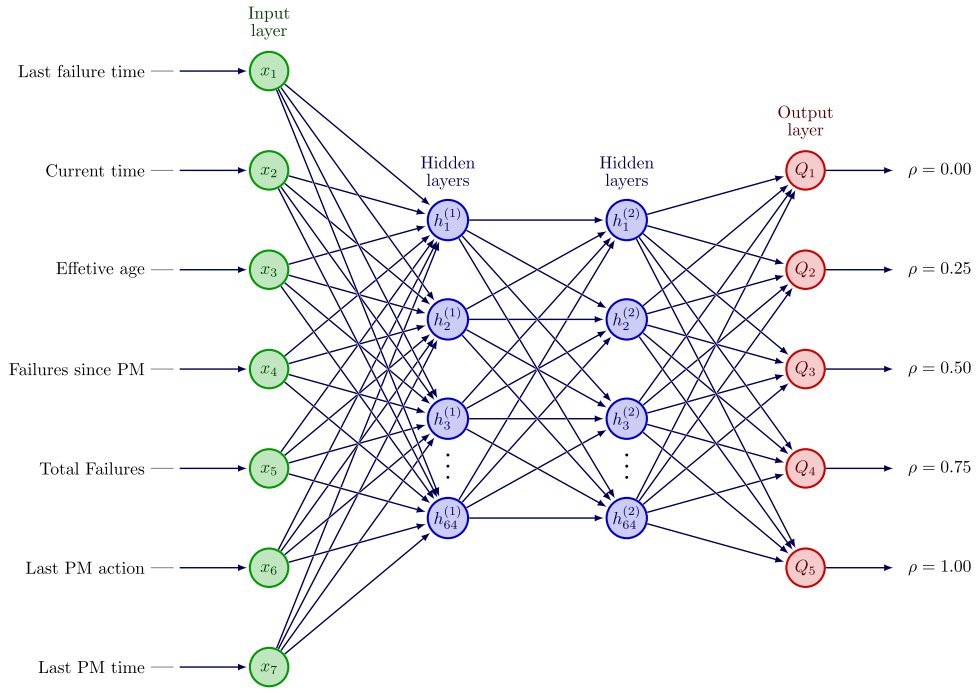

Fig. 4. Neural network architecture for the imperfect maintenance case.

- $Q(\cdot;\theta)$ is the online network with parameters θ,
- $Q(\cdot;\theta^-)$ is the target network with parameters θ^-,
- θ^- is updated by θ every τ steps to stabilize learning.

The corresponding loss function is

$$L_k(\theta) = \mathbb{E}_{(s_k,a_k,r_k,s_{k+1})\sim\mathcal{D}}\left[\left(Y_k^{\text{DDQN}} - Q(s_k,a_k;\theta)\right)^2\right], \tag{19}$$

where D is the experience replay memory distribution. Stochastic gradient descent is used to update θ by minimizing $L_k(\theta)$, whereas θ^- is synchronized with θ at fixed intervals.

This update rule incorporates the stochasticity of the environment: failures are sampled according to $F_k \sim \text{Bernoulli}(p^{\text{true}}(s_k,a_k))$, and corrective costs may follow a random distribution $C_c(N_k) \sim D(c_c,\sigma^2)$. Hence, the DDQN update naturally accounts for both probabilistic failures and random cost realizations in the Bellman target. The decoupling of action selection and evaluation ensures stable convergence despite the noisy and non-stationary reward structure of the maintenance environment.

3.5. Evaluation methodology and cost simulation framework

To ensure a fair comparison between the RL-based agent and the analytical maintenance policies, a unified evaluation framework was implemented. All simulations were conducted over a 60-day horizon ($T = 60$) with a daily decision step, and each policy was evaluated across $N_{\text{epi}} = 1000$ independent Monte Carlo episodes using shared random sequences to guarantee identical sampling of failure events.

Three types of policies were evaluated for each corrective-to-preventive cost ratio $c_m/c_p \in \{8, 4, 2, 1, 0.5, 0.25\}$: (i) the *single-action* policy, in which a single preventive action is scheduled at an optimal day within the horizon given by the minimum of Eq. (6), (ii) the *periodic* policy, with an analytically optimized maintenance interval τ^* by searching optimal maintenance intervals that minimizes Eq. (6); and (iii) the RL-based policy, represented by the greedy deterministic policy derived from the trained DDQN agent. All three policies were tested under identical stochastic realizations of the failure process, generated by the same matrix of random draws. This guarantees that differences in total cost arise solely from the decision policy, not from randomness in the environment.

The underlying base degradation process follows a non-homogeneous Poisson process (NHPP) with a Weibull (power-law) intensity where the shape and scale parameters were set to $\beta_1 = 1.065$ and $\alpha_1 = 4.721$, values consistent with reliability data reported for a die-casting machinery (see de Souza & da Silva, 2024). A similar order of magnitude for the parameters can also be observed in other contexts, for instance in the estimates reported by Saraygord Afshari et al. (2022) and Moniri-Morad and Sattarvand (2023). Preventive maintenance actions reduce the virtual age according to the PAR model with factor ρ, whereas corrective maintenance is minimally restorative. To represent imperfect repairs and the risk accumulation observed in industrial practice, the instantaneous failure probability incorporates a multiplicative post-failure risk factor θ, defined as

$$\theta = 1 + \kappa_{\text{risk}}\, \mathbf{1}_{\{\text{failure since last PM}\}}, \quad \text{with } \theta \le \theta_{\max}. \tag{20}$$

The probability, as shown in 12, of at least one failure occurring within the step $(t, t + \Delta t]$ is then

$$p_{\text{fail}} = 1 - \exp(-\theta\, \Delta H), \tag{21}$$

where ΔH is the cumulative hazard increment under the PAR-adjusted age. The parameter $\kappa_{\text{risk}} = 0.25$ controls the magnitude of the temporary increase in failure rate following a corrective event, and $\theta_{\max} = 3.0$ defines its saturation limit.

The total cost accumulated within each episode is computed by combining preventive and corrective costs while accounting for the sensitivity of corrective maintenance to repeated failures. Whenever a preventive action of magnitude $\rho \in [0, 1]$ is taken, a cost proportional to its intensity is incurred:

$$C_{\text{PM}} = c_p\, \rho. \tag{22}$$

If a failure occurs, the corrective cost increases with the number of consecutive failures since the last preventive action:

$$C_{\text{CM}} = c_m \left(1 + \kappa_{\text{cost}}\, f^{\text{PM}}\right), \tag{23}$$

where f^{PM} is the number of failures accumulated since the previous PM and $\kappa_{\text{cost}} = 0.25$ quantifies the cost amplification effect due to repeated unplanned interventions. This structure reflects both operational risk escalation and the rising indirect costs associated with equipment downtime.

For each policy, the environment evolves deterministically in time, whereas the occurrence of failures follows the sampled probabilities p_{fail}. The total episode cost is then obtained by summing the instantaneous preventive and corrective components across all time steps:

$$C_{\text{total}} = \sum_{t=1}^{T} \left(C_{\text{PM},t} + C_{\text{CM},t}\right). \tag{24}$$

The average and standard deviation of C_{total} over all episodes are reported as performance metrics for each policy and cost regime. Because all methods share the same random seeds and hazard realizations, the comparison isolates the effect of the decision logic rather than stochastic variability.

This evaluation methodology extends classical reliability-based cost simulations by incorporating post-failure risk escalation (κ_{risk}) and cost sensitivity to repeated failures (κ_{cost}), both of which were present during RL training. Therefore, the performance of the trained DDQN agent can be assessed under conditions that faithfully reproduce the stochastic environment and cost dynamics to which it was exposed, allowing a direct and statistically rigorous comparison with periodic and single-action analytical policies.

3.6. Comparison methods

To assess the performance of the RL-based maintenance policy, three alternative optimization strategies were implemented and evaluated under identical stochastic environments. All methods were simulated using the same random failure sequences and cost parameters, ensuring that differences in performance reflected only the decision-making approach.

3.6.1. Static analytical policies.

Two classical maintenance strategies were adopted as analytical baselines. The first is the *periodic* policy, in which preventive maintenance is executed at fixed intervals τ^* that minimize the expected cost per unit time. The optimal interval is obtained by solving

$$\tau^* = \arg\min_{\tau} \frac{c_m H(\tau) + c_p}{\tau}, \tag{25}$$

where $H(\tau) = \int_0^{\tau} h(s)\, ds$ is the cumulative hazard function of the Weibull failure model, and c_p and c_m are the preventive and corrective maintenance costs, respectively. The second is the *single-action* policy, in which exactly one preventive maintenance operation is scheduled within the horizon at the time τ^* that minimizes the total expected cost over the period $[0, T]$.

3.6.2. Dynamic genetic algorithm (GA-S).

As a non-learning, model-based method, a dynamic Genetic Algorithm (GA-S) was implemented. At each decision epoch, the GA re-optimizes a sequence of future preventive actions over the remaining horizon by simulating the stochastic failure process as a black-box environment. Each chromosome encodes a binary sequence of maintenance actions (perform or skip) for the remaining days, and its fitness is evaluated as the expected total cost obtained from Monte Carlo simulations of the same Weibull–PAR process, including the post-failure risk amplification (κ_{risk}) and corrective cost escalation (κ_{cost}). Selection, crossover, and mutation operators evolve the population over several generations, and the best individual's first action is applied to the system before the process repeats for the next day. This re-optimization scheme allows the GA to approximate a dynamic decision process without learning an explicit policy or state-value function.

4. Results and analysis

This section presents the experimental results obtained using the proposed DDQN-based preventive maintenance algorithm presented in Section 3. The analysis was divided into two main parts. First, a comprehensive sensitivity study was conducted to evaluate the influence of key hyperparameters on learning performance and policy stability, including the target network update frequency, replay buffer size, neural network architecture, mini-batch size, learning rate, look-ahead parameter, and discount factor. These experiments aimed to identify configurations that yield robust convergence and consistent value estimation under a stochastic maintenance environment.

In the second part, the trained agent is evaluated under multiple corrective-to-preventive cost ratios, with both perfect and imperfect maintenance actions, illustrating how the learned policies adapt to different economic trade-offs between corrective and preventive maintenance. The performance of the reinforcement learning agent is then compared against benchmark maintenance strategies, including analytical static schedules and periodic policies, to assess its relative efficiency and generalization capability.

The simulations employed reliability parameters consistent with the failure data reported in the literature for die-casting machinery. The power-law (Weibull) parameters used in this study, $\beta_1 = 1.065$ and $\alpha_1 = 4.721$, are of the same order of magnitude commonly found across various types of industrial equipment, such as jet engine parts and mining trucks, ensuring that the experimental results reflect realistic degradation and failure dynamics.

4.1. Analysis of hyperparameters

This subsection presents a systematic evaluation of the influence of key hyperparameters on the performance and stability of the Double Deep Q-Network algorithm. Each parameter was varied independently around the baseline configuration defined in Section 4.2, whereas all others were held constant. The results are discussed in terms of mean and variance of the total cost, convergence behavior or other quantitative characteristics of the learned policies.

4.2. Training configuration and baseline hyperparameters

The reinforcement learning agent was trained using an ϵ-greedy exploration strategy, with the exploration rate (ϵ) linearly decaying from 1.0 to 0.01 over 10,000 timesteps, corresponding to a decay rate of 8×10^{-6} per step and the random seed fixed at rng(3). This schedule ensures a gradual transition from an exploratory to an exploitative behavior, as shown in Fig. 5, allowing the agent to initially sample the environment widely before focusing on the exploitation of the most promising actions. The full training process required approximately two hours on a standard workstation equipped with an Intel Core i5 processor (3.2 GHz) and 8 GB of RAM. Within the same computational setup, the inference process of the trained agent is effectively instantaneous.

The baseline configuration adopted for the analyses was established after extensive sensitivity testing across the main hyperparameters. The set of values found to produce stable and consistent convergence across independent runs is summarized below:

- Target Update Frequency: 1000;
- Look-Ahead Parameter: 3;
- Discount Factor (γ): 0.99;
- Network Architecture: Two hidden layers of 64 neurons each with ReLU, and a linear activation in the output layer;
- Batch Size: 64;
- Learning Rate: 0.00005;
- Replay Buffer Length: 3×10^5;
- L_2 Regularization Factor: 1×10^{-5};

The inclusion of an L_2 regularization term was found to play an important role in stabilizing the learning dynamics. By penalizing large network weights, this term helps control overfitting to transient fluctuations in the value function estimates, promoting smoother convergence and improved generalization when exposed to unseen operational conditions.

Fig. 5 illustrates the evolution of the training process under this baseline configuration. The learning curve shows the progressive reduction and subsequent stabilization of the cumulative cost across episodes, indicating the agent's ability to extract a consistent policy from the simulated experience. The steady convergence trend observed supports the selection of this configuration as the reference setup for the subsequent hyperparameter analyses. In the following subsections, the parameters above will be individually varied to assess its influence on learning efficiency, convergence stability, and overall policy performance.

4.3. Analysis of hyperparameters

4.3.1. Analysis of target network update frequency: DQN vs. DDQN

The first set of experiments was conducted to evaluate the influence of the target network update frequency on the performance of the Double Deep Q-Network (DDQN) compared with the standard Deep Q-Network (DQN). The action space in this experiment was binary, consisting of either performing a perfect preventive maintenance action or taking no action. The hyperparameter configuration followed the base setup described earlier, with the target network update frequency varying in $1, 10, 60, 100, 1000$ and the random seed fixed at rng(3).

The experiments were performed with $\kappa_{\text{risk}} = \kappa_{\text{cost}} = 0.25$, Weibull parameters $\beta_1 = 1.065$ and $\alpha_1 = 4.721$, a time horizon of 60 days, and $1,000$ simulation trials per configuration. The case $c_c/c_p = 1$ was of particular interest, since it represents the most sensitive decision region where the choice between performing preventive maintenance or accepting a corrective cost is least obvious.

For the baseline case ($c_c/c_p = 1$), the DDQN consistently achieved lower mean cost values than the DQN, with average reductions exceeding 20% relative to the standard agent. This improvement remained stable across different values of *Target Update Frequency*, with no clear performance gap among 1, 10, 60, or 100 updates.

4.3.2. Influence of the look-ahead parameter in DDQN learning

A complementary set of experiments was conducted to investigate the influence of the 'number of steps to look-ahead' parameter on the performance and stability of the DDQN agent. This parameter controls the number of future steps considered when computing the target return during the Bellman update. In standard one-step Q-learning, the target is defined as

$$Y_t^{(1)} = r_t + \gamma Q(s_{t+1}, a^*_{t+1}; \theta^-), \tag{26}$$

where $a^*_{t+1} = \arg\max_a Q(s_{t+1}, a; \theta)$ is the greedy action according to the current online network. When the look-ahead parameter is extended to n steps, the target becomes

$$Y_t^{(n)} = r_t + \gamma r_{t+1} + \cdots + \gamma^{n-1} r_{t+n-1} + \gamma^n Q(s_{t+n}, a^*_{t+n}; \theta^-), \tag{27}$$

propagating the influence of near-future rewards into the update.

In the DDQN algorithm described in Algorithm 2, the lines corresponding to the target computation are modified as follows:

$$\forall\ (s_j, a_j, r_j, s_{j+1}) \in \mathcal{D}: \quad Y_j^{\text{DDQN}} \leftarrow \sum_{i=0}^{n-1} \gamma^i r_{j+i} + \gamma^n Q(s_{j+n}, a^*_{j+n}; \theta^-). \tag{28}$$

This adjustment affects only the target generation step, leaving the remaining parts of the training loop unchanged. Conceptually, the parameter n controls the temporal depth of reward propagation and balances bias and variance in value estimation. The parameter was tested for $n \in \{1, 2, 3, 5\}$, keeping all the other hyperparameters fixed. The results show that setting $n = 3$ produced the lowest mean maintenance cost and the smallest standard deviation across $1,000$ simulated episodes, outperforming both shorter and longer look-ahead horizons. With $n = 1$ or $n = 2$, the agent converged faster but exhibited higher cost variability and evidence of premature stabilization of the $Q(s, a)$ values, indicating a bias toward short-term rewards and suboptimal policies. For $n = 5$, convergence became slower and less stable, as longer look-ahead returns incorporated higher variance due to stochastic failure events.

These observations are consistent with the theoretical trade-off between bias and variance in n-step methods. A smaller n leads to faster but biased updates that rely heavily on noisy one-step transitions,

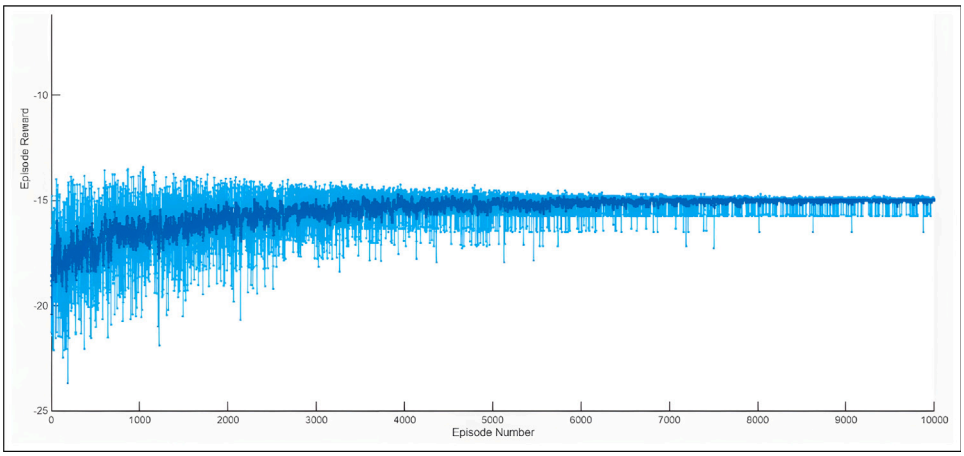

Fig. 5. Episode reward (blue) and 10-episode moving average reward (bold blue) during he training of a DDQN agent for maintenance scheduling.

whereas a larger n reduces bias at the expense of amplifying variance and slowing convergence. In this preventive maintenance context, where each maintenance action influences the risk and cost structure for a few subsequent days, a look-ahead horizon of $n = 3$ provides an effective compromise. It captures the short-term delayed effect of preventive maintenance actions on equipment reliability without excessively propagating stochastic noise from future failures.

4.3.3. Sensitivity to the discount factor

A sensitivity analysis was also conducted to assess the effect of the discount factor γ on the learning performance of the DDQN agent. The discount factor regulates the relative importance assigned to future rewards in the Bellman update, and therefore determines how strongly the agent values long-term outcomes with respect to immediate costs.

The parameter was tested for $\gamma \in \{1.0, 0.999, 0.99, 0.95, 0.9\}$, keeping all other hyperparameters fixed to the baseline configuration, but using with $n = 3$. Across $10,000$ training episodes, no significant differences in performance were observed for $\gamma \geq 0.95$. The resulting mean costs and standard deviations were statistically indistinguishable, indicating that the learned policy and value estimates were robust to small variations in the discount factor within this range. Only the configuration with $\gamma = 0.9$ showed a noticeable degradation in performance, producing higher average maintenance costs.

This outcome is consistent with the expected behavior of sequential decision processes with moderately long horizons. Given that each episode spans 60 decision steps (days) and maintenance actions have delayed but bounded effects, high values of γ near unity appropriately preserve the temporal dependencies between actions and their future costs. When γ is reduced to 0.9, the agent becomes overly myopic, prioritizing immediate cost minimization over preventive decisions that yield benefits several steps ahead. Consequently, the learned policy tends to underperform by deferring maintenance actions too long.

4.3.4. Effect of network size on learning performance

The neural network architecture used by both the online and target critics in the DDQN agent consists of two fully connected hidden layers with ReLU activation functions. To evaluate the influence of network capacity on learning performance, several configurations were tested by varying the number of neurons in each hidden layer while keeping the network depth fixed at two layers. The number of neurons per layer was set to $\{32, 64, 128, 256, 512\}$.

The decision to restrict the analysis to two hidden layers follows from the *universal approximation theorem* (Chen & Chen, 1995), which establishes that a feed-forward neural network with a single hidden layer and a sufficient number of units can approximate any continuous function on a compact domain to arbitrary precision. In practice, deeper networks often increase training complexity and variance without proportional gains in representational power for problems of moderate dimensionality such as the present seven-state MDP. Therefore, exploring different network widths provides a more relevant measure of model capacity than varying the number of layers.

Table 1
Performance of DDQN agents with different numbers of neurons per hidden layer.

Neurons	Mean cost	Std. deviation
32	57.17	5.44
64	25.67	5.94
128	39.53	5.77
256	28.51	6.40
512	27.83	7.97

The ReLU activation function was adopted because of its well-established advantages in stabilizing gradient propagation and accelerating convergence in deep Q-learning frameworks, as widely reported in the reinforcement learning literature, and it was therefore selected as a standard choice without further empirical comparison.

The results of this analysis are summarized in Table 1. Each configuration was trained under identical conditions and evaluated across $1,000$ simulation trials using the same random seed. The mean maintenance cost and its standard deviation were computed during the inference phase of the trained policies.

The results indicate that the configuration with 64 neurons in each hidden layer achieved the lowest mean cost while maintaining one of the smallest standard deviations. Smaller networks (e.g., 32 units) were unable to capture the nonlinear structure of the value function, leading to systematically higher costs. Larger networks (128 units or more) did not improve the average performance and tended to exhibit slightly higher variability, suggesting overparameterization and increased sensitivity to stochastic fluctuations during training.

4.3.5. Effect of mini-batch size on training stability

Another hyperparameter evaluated in this study was the mini-batch size used for stochastic gradient descent during the critic updates. The mini-batch size determines the number of experience tuples randomly sampled from the replay buffer at each training iteration. This parameter directly affects the bias–variance trade-off of the gradient estimates and, consequently, the stability of the learning process.

Three configurations were tested, with batch sizes of $\{64, 128, 256\}$ samples per update, while keeping all other training parameters fixed. Each configuration was trained for the same number of episodes under identical random seeds. The performance of the resulting agents was

Table 2
Performance of DDQN agents with different mini-batch sizes.

Batch size	Mean cost	Std. deviation
64	25.67	5.94
128	45.45	16.58
256	44.42	16.23

Table 3
Performance of DDQN agents with different learning rates.

Learning Rate	Mean cost	Std. deviation
0.05000	45.38	16.72
0.00500	45.38	16.72
0.00050	43.56	18.17
0.00005	25.67	5.93

assessed using the mean and standard deviation of the maintenance cost over $1,000$ inference episodes.

As shown in Table 2, the configuration using a mini-batch of 64 samples achieved the lowest mean cost and the smallest standard deviation among all tested values. Increasing the batch size to 128 or 256 did not improve convergence; instead, both resulted in higher variability and worse average performance. This behavior can be attributed to the reduced stochasticity of the gradient updates when using large batches, which limits exploration in parameter space and slows the adaptation of the Q-network to the nonstationary reward landscape.

Conversely, smaller batches introduce moderate noise in the gradient estimation, which helps the optimization escape local minima and better capture the underlying structure of the value function. In this context, a mini-batch size of 64 offered a favorable balance between learning stability and representational generalization. Therefore, this configuration was adopted as the standard setting for all the subsequent DDQN training experiments.

4.3.6. Influence of the learning rate on convergence behavior

The learning rate is one of the most critical hyperparameters in deep reinforcement learning, as it determines the step size of the gradient descent updates during network training. Excessively large values can lead to unstable oscillations or divergence, whereas values that are too small can result in slow convergence and poor adaptation to nonstationary environments. To assess its effect, four values of the critic learning rate were tested: $\{0.05, 0.005, 0.0005, 0.00005\}$, while keeping all other hyperparameters identical to the baseline configuration.

Each configuration was trained under identical conditions and evaluated using 1000 inference episodes. Table 3 summarizes the resulting mean and standard deviation of the maintenance cost for each learning rate.

The results clearly show that lower learning rates produced substantially better outcomes. For LR $= 0.00005$, the agent achieved the lowest mean cost and the smallest standard deviation, indicating both superior convergence and higher stability of the learned policy. In contrast, larger learning rates (0.05 and 0.005) resulted in nearly identical and significantly worse performance, reflecting the characteristic oscillatory behavior and loss of precision typical of overly aggressive gradient updates. An intermediate value (0.0005) slightly improved the average cost but still exhibited high variance.

These findings are consistent with the sensitivity of DDQN algorithms to the magnitude of parameter updates, particularly in environments with dense but noisy rewards such as the present maintenance optimization problem. A small learning rate ensures smoother adjustments of the Q-network weights, preventing overreaction to stochastic variations in the replayed transitions and promoting gradual, stable convergence. Based on these observations, a learning rate of 5×10^{-5} was adopted as the default configuration in subsequent experiments.

Table 4
Comparison of replay buffer sizes and corresponding performance metrics.

Buffer size	Mean Cost	Standard Deviation
3×10^4	38.26	6.74
3×10^5	25.67	5.94
3×10^6	26.34	6.31

4.3.7. Effect of replay buffer size

To assess the influence of experience replay on training stability, additional experiments were conducted with three replay buffer sizes: 3×10^4, 3×10^5, and 3×10^6 transitions. The results, summarized in Table 4, indicate that the intermediate configuration with 3×10^5 samples produced the lowest mean total cost and the smallest variability, representing the most stable and cost-efficient policy. Empirically, small replay buffers (10^4) tend to limit the diversity of past experiences available for training, causing the agent to overfit to recent transitions and leading to oscillatory learning behavior. Conversely, excessively large buffers (10^6) dilute the relevance of recent experiences, slowing convergence and introducing outdated samples that reduce learning responsiveness. These observations are consistent with established reinforcement learning literature, which emphasizes that experience replay improves convergence by breaking temporal correlations, provided that the stored experiences remain representative of the current policy distribution. Therefore, a buffer size of 3×10^5 was adopted as a practical compromise, ensuring sufficient sample diversity for stable learning while maintaining adequate sensitivity to recent environmental dynamics.

4.3.8. Evolution of the Q-values during training

The convergence properties of value-based Deep Reinforcement Learning algorithms are inherently influenced by the stochastic nature of the training process, the bootstrapped construction of target values, and the use of neural function approximation. In the present study, both the classical Deep Q-Network (DQN) and the Double Deep Q-Network (DDQN) were trained under identical conditions, employing a Target Update Frequency of 1000 steps.

As shown in Fig. 6, the DDQN produced higher converged $Q(s, a)$ values (less negative in the cost-based formulation) and a more efficient policy compared with the DQN, as described in Section 4.3.1, despite exhibiting slightly greater variance during the final training episodes. The DDQN agent clearly differentiates the value of 'not maintain' action from 'maintain' action in the first step of the training process. This difference can be attributed to the distinct bias characteristics of each algorithm. The DQN tends to overestimate its target values due to the correlation between action selection and evaluation within the same estimator. Such coupling introduces a persistent maximization bias that can destabilize the value updates and drive the learning process toward suboptimal local attractors. This effect is evidenced in the DQN results (left panel), where the agent converged prematurely to a lower $Q(s, a)$ plateau and failed to escape from it in the later stages of training.

In contrast, the DDQN mitigates this maximization bias by decoupling action selection from target evaluation, employing two decorrelated estimators that generate more consistent and less correlated target values. This modification results in smoother updates and allows the DDQN agent to explore a broader portion of the value-function space, leading to a policy that achieves a lower expected maintenance cost. Because the target and online networks evolve asynchronously and are periodically synchronized, each target update introduces small discontinuities in the learning dynamics. The resulting oscillations represent bounded fluctuations around a stationary point rather than evidence of divergence.

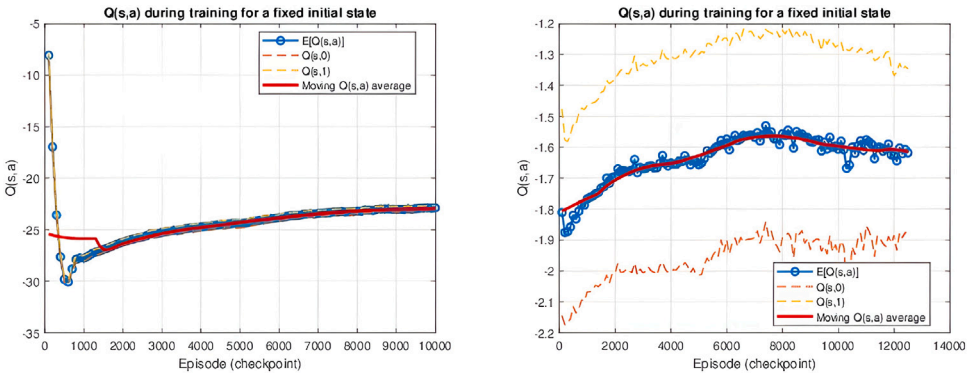

Fig. 6. $Q(s,a)$ estimation during training. (Left) DQN algorithm. (Right) DDQN algorithm with target network update frequency equal to 1000.

4.4. Agent results

This subsection reports the performance of the trained agents under different maintenance cost structures and risk models. The results include comparisons with analytical benchmark policies, with an ordinary RL algorithm as well as with heuristic optimization baselines. Both robustness and generalization are examined by evaluating the agents under perturbed failure dynamics and stochastic maintenance environments, providing insight into the learned policy's adaptability and stability.

4.4.1. Genetic algorithm and standard RL configuration and comparative performance

For benchmarking purposes against the *periodic*, *static*, and *reinforcement learning* (RL) policies, a *genetic algorithm* (GA) was implemented with fixed evolutionary parameters and a dynamic stopping criterion proportional to the RL inference time. The GA employed a population of 10 individuals, up to 50 generations, a crossover probability of $p_c = 0.8$, a mutation probability of $p_m = 0.1$, and 30 internal Monte Carlo simulations ($n_{\text{sim}} = 30$) to estimate the expected cost of each chromosome under the agent's internal (belief) model, which neglected the second Weibull hazard component.

Unlike the RL agent, the GA performs an on-line stochastic search during inference, evolving binary sequences of maintenance actions over the remaining decision horizon. To ensure computational fairness in the comparison, a time-based stopping criterion was introduced: the GA is interrupted when its total inference time for a given episode exceeds k times the average inference time of the RL agent under the same evaluation conditions. The parameter k thus represents the available computational budget, where larger values allow a broader search over the decision space at the cost of higher inference time.

For comparative purposes, an additional Double Deep Q-Network (DDQN) agent adapted from Huang et al. (2020) was also implemented using the reward function defined in Eq. (11). The state space of this implementation was composed of four variables: the elapsed operating time, the time since the last failure, the cumulative age effect resulting from maintenance actions, and the total number of failures observed. This configuration mirrors the structure proposed by Huang et al. (2020), enabling a fair comparison between the standard DDQN formulation and the proposed DDRL framework in terms of cost minimization performance and learning stability.

When evaluating the baseline DDQN under the simplified condition of $\kappa_{\text{risk}} = \kappa_{\text{cost}} = 0$, that is, when corrective maintenance actions produce no cumulative effects on either the risk or the cost structure, the standard model achieved slightly better performance than the proposed DDRL framework. In this static environment, where adaptation offers no clear advantage, the baseline DDQN, due to its reduced state space and non-adaptive policy, obtained an average cost of 13.003 with a standard deviation of 3.1798, whereas the proposed RL approach reached an average cost of 15.608 and a standard deviation of 3.6745. Even under perturbed conditions, the standard DDQN maintained a marginally better performance, with a mean cost of 30.478 and a standard deviation of 3.797, compared to 35.248 and 4.116, respectively, for the proposed DDRL agent. However, once any cumulative effect from corrective maintenance is introduced, through either the risk or cost terms, the standard DDQN's performance deteriorates, similarly to analytical static policies, while the proposed model remains stable and continues to adapt effectively to the changing environment.

Table 5
Comparison of average and standard deviation of episode costs for different decision policies.

Policy	Mean Cost	Std. Deviation
Periodic	30.6645	9.84
Single PM	30.6645	9.84
Proposed RL	25.2350	5.85
Standard RL	45.3750	16.72
GA ($k = 5$)	43.8595	16.21
GA ($k = 10$)	43.4565	15.97
GA ($k = 100$)	30.7145	11.46

The results presented in Table 5 and in Table 6 correspond to the case of practical interest in which the corrective and preventive maintenance costs are equal ($c_m = c_p$), the restoration factor ρ is equal to 1 and $\kappa_{\text{risk}} = \kappa_{\text{cost}} \neq 0$. In Table 5 the agents model's belief matches the real baseline hazard function. Conversely, in Table 6, the real hazard function differs from the baseline belief. The single-term Weibull process is enhanced with a two-component model defined by the parameters $\beta_1 = 1.065 + 0.25$, $\alpha_1 = 4.721 - 1$, $\beta_2 = 0.875$, and $\alpha_2 = 8$. For each method, the tables report the average cost and its standard deviation across 1000 simulated episodes. The periodic and single preventive maintenance policies yield identical results, as expected from their analytical equivalence under constant intervals. The proposed RL agent clearly outperforms both analytical policies, the standard RL agent and the GA algorithm, achieving a lower mean cost and a smaller variance. The GA results, shown for three values of k, reveal the impact of the time budget on solution quality: for $k = 5$ and $k = 10$, the GA produces significantly higher costs and dispersion (approximately 75%), while for $k = 100$ the mean cost approaches that of the RL policy (approximately 20% higher for the first case and 25% higher for the second), although with considerably larger variability (approximately 97% higher for the first case and 171% for the second).

As the value of k increases, the GA is granted more time to explore the decision space, allowing for a more thorough search and a greater likelihood of discovering cost-efficient sequences. This behavior aligns with the exploratory nature of evolutionary algorithms, where additional generations and evaluations expand the coverage of potential action plans. However, this improvement is achieved solely by allocating more computational resources rather than through inherent

Table 6
Comparison of average and standard deviation of episode costs for different decision policies with perturbed environment.

Policy	Mean Cost	Std. Deviation
Periodic	111.06	19.03
Single PM	111.06	19.03
RL	61.96	7.87
Standard RL	186.12	36.55
GA ($k = 5$)	181.59	34.91
GA ($k = 10$)	176.50	35.95
GA ($k = 100$)	77.57	19.76

decision efficiency. Even with $k = 100$, the standard deviation of the GA's results remains more than twice that of the RL agent, indicating persistent inconsistency across independent runs.

4.4.2. Comparative evaluation of the RL policy trained with fixed model parameters

Given the high computational cost of the genetic algorithm (GA), which makes it unsuitable for real-time or large-scale applications, and the limited adaptability of the standard RL, whose policy fails to respond effectively to environments with corrective maintenance effects or stochastic costs, the comparative analysis in this section focuses solely on the proposed RL agent relative to the analytical maintenance policies. Two analytical baselines are considered: a *static* policy and a *periodic* policy, both derived from the power-law failure model used in the RL training.

In the first part of the analysis, the RL agent is evaluated under the same baseline environment used for training, corresponding to a single-term Weibull failure model with fixed parameters. The preventive maintenance cost is normalized to $c_p = 1$, and several values of the corrective-to-preventive cost ratio c_m/c_p are examined. For the representative case $c_m = c_p$, both analytical policies identify the same optimal preventive interval $t^* = 30$, yielding identical performance statistics with a mean episode cost of 31.05 and a standard deviation of 9.87 across 1000 Monte Carlo simulations. The RL policy achieves a lower mean cost of 25.53 and a standard deviation of 5.89.

A two-sample Student's t-test rejects the null hypothesis of equal means with a p-value of 3×10^{-49}, and the Brown–Forsythe test (see Brown & Forsythe, 1974) for equality of variances also rejects the null hypothesis with a p-value of 4×10^{-40}. These results confirm that the RL policy not only achieves lower average cost but also yields more stable outcomes within the baseline model. However, when the same policy and analytical baselines are evaluated under environments consistent with their respective belief models, the RL advantage does not extend across all cost ratios.

To better understand the operational logic of the learned policy, three random trajectories were analyzed under the nominal failure model. These trajectories correspond to distinct Monte Carlo realizations in which the same RL policy interacts with independent sequences of stochastic failures. Despite being generated under identical model parameters, each trajectory exhibits a different temporal pattern of failures and preventive maintenance actions, reflecting the adaptive nature of the learned policy. The corresponding outcomes for the three realizations are summarized as follows:

1. Total cost = 22.50, number of failures = 13, number of PMs = 2;
2. Total cost = 36.25, number of failures = 20, number of PMs = 2;
3. Total cost = 15.00, number of failures = 9, number of PMs = 1.

Table 7 shows the evolution of the actions and failure occurrences over the 60-day horizon for each episode.

From these trajectories, one can observe clear signs of adaptive behavior in the learned policy. Although the complete policy relies on a multidimensional state representation, including variables such as effective age, time since last PM, and cumulative number of failures, the partial information reported in Table 7 is sufficient to illustrate the agent's responsiveness to changing maintenance conditions. The RL agent reacts to local degradation patterns, triggering maintenance only when the recent failure history or inferred operational risk becomes significant.

The following results refer to the perturbed environment, in which the agent was trained under the baseline model but later evaluated in an uncertain setting where the failure rate parameter β varies uniformly within the interval $[\beta, \beta + 0.3]$. This configuration introduces stochastic variability into the hazard function, allowing the assessment of the model's adaptability. The results presented in Table 8 demonstrate that the RL agent maintained competitive performance under these perturbations, confirming its robustness and capacity to generalize beyond the nominal training conditions.

A second evaluation scenario, on the other hand, introduces a deliberate perturbation to the failure model, replacing the single-term Weibull process with a two-component model defined by the parameters $\beta_1 = 1.065 + 0.25$, $\alpha_1 = 4.721 - 1$, $\beta_2 = 0.875$, and $\alpha_2 = 8$. Under this perturbed environment, shown in Fig. 7, the analytical policies continue to rely on the baseline failure model, whereas the RL agent operates without modification. In this case, the RL policy demonstrates a significant relative reduction in both the mean and variance of the total cost across all values of c_m/c_p, as can be seen in Fig. 8, indicating improved robustness when the underlying failure dynamics deviate from those assumed during training.

Table 9 summarizes the results of hypothesis testing between the RL policy and the periodic analytical policy for each cost ratio. In all evaluated cases, the t-tests indicate statistically significant differences in the mean costs between the RL and the analytical policies ($p \ll 0.001$). Similarly, the Brown–Forsythe tests reveal significant differences in the variances.

To evaluate the adaptability of the proposed DDRL framework to variations in the underlying failure dynamics, two perturbation scenarios were tested after training the agent under the baseline power-law hazard function with an increasing failure rate. As shown in Fig. 9, the trained agent was subsequently exposed to failure processes governed by two distinct bathtub-shaped hazard functions. The first perturbation was defined by parameters $\beta_1 = 2.565$, $\alpha_1 = 22.721$, $\beta_2 = 0.6$, and $\alpha_2 = 2.5$, while the second used $\beta_1 = 1.715$, $\alpha_1 = 22.721$, $\beta_2 = 0.9$, and $\alpha_2 = 4$. Despite being trained exclusively under a monotonically increasing risk assumption, the DDRL agent successfully adapted its policy to these non-monotonic failure behaviors. As illustrated in Fig. 10, the model exhibited robustness not only to changes in the slope of the hazard function but also to its overall shape. In both perturbation cases, the reinforcement learning approach achieved a reduction in both the mean and variance of the total maintenance cost compared to the baseline, demonstrating its capacity for generalization and stable decision-making under differing reliability conditions.

These findings suggest that the RL policy adapts to the stochastic variability inherent in the maintenance environment and retains stable performance when evaluated under unseen operating conditions. The absence of degradation in policy effectiveness under model perturbations indicates that the learned strategy generalizes beyond the training distribution and does not exhibit signs of overfitting. In the next subsection, this aspect is further investigated by analyzing an RL agent trained with randomized model parameters, in order to explicitly assess its generalization and robustness across heterogeneous failure dynamics.

4.4.3. Evaluation under imperfect preventive maintenance

In this analysis, the preventive maintenance action is modeled as imperfect, such that each intervention partially restores the system depending on the selected restoration factor $\rho \in [0, 0.25, 0.50, 0.75, 1]$. A value of $\rho = 1$ corresponds to perfect maintenance, fully rejuvenating the system to an "as-good-as-new" state, whereas smaller values of

Table 7
Full trajectories of three RL realizations under the nominal model.

Day	Episode 1			Episode 2			Episode 3		
	PM Action	Fail Flag	Fails Count	PM Action	Fail Flag	Fails Count	PM Action	Fail Flag	Fails Count
1	0	0	0	0	1	1	0	0	0
2	0	0	0	0	0	1	0	0	0
3	0	1	1	0	0	1	0	1	1
4	0	0	1	0	0	1	0	0	1
5	0	0	1	0	0	1	0	0	1
6	0	1	2	0	0	1	0	0	1
7	0	0	2	0	0	1	0	0	1
8	0	1	3	0	0	1	0	1	2
9	0	0	3	0	0	1	0	0	2
10	0	0	3	0	0	1	0	1	3
11	0	1	4	0	1	2	0	0	3
12	0	0	4	0	0	2	0	1	4
13	0	0	4	0	1	3	0	0	4
14	0	0	4	0	0	3	0	0	4
15	0	0	4	0	1	4	0	0	4
16	0	0	4	0	0	4	0	0	4
17	0	0	4	0	0	4	0	0	4
18	0	0	4	0	1	5	0	0	4
19	0	0	4	1	1	6	0	0	4
20	0	0	4	0	0	6	0	0	4
21	0	1	5	0	0	6	0	0	4
22	1	0	5	0	0	6	0	0	4
23	0	0	5	0	1	7	0	0	4
24	0	0	5	0	0	7	0	0	4
25	0	0	5	0	0	7	0	0	4
26	0	0	5	0	1	8	0	0	4
27	0	0	5	0	1	9	0	1	5
28	0	0	5	0	1	10	0	0	5
29	0	0	5	0	0	10	0	0	5
30	0	0	5	0	1	11	0	0	5
31	0	0	5	0	1	12	0	0	5
32	0	0	5	0	0	12	0	0	5
33	0	1	6	1	1	13	0	0	5
34	0	1	7	0	0	13	0	0	5
35	0	0	7	0	0	13	0	0	5
36	0	0	7	0	0	13	0	0	5
37	0	1	8	0	0	13	0	0	5
38	0	0	8	0	0	13	0	0	5
39	0	0	8	0	0	13	0	0	5
40	0	0	8	0	1	14	0	0	5
41	0	0	8	0	0	14	0	1	6
42	0	1	9	0	0	15	0	0	6
43	0	0	9	0	0	15	0	0	6
44	0	0	9	0	1	16	0	0	6
45	0	0	9	0	0	16	0	0	6
46	0	1	10	0	0	17	0	0	6
47	1	0	10	0	0	17	0	0	6
48	0	0	10	0	0	17	0	0	6
49	0	0	10	0	0	17	0	0	6
50	0	0	10	0	0	17	0	0	6
51	0	0	10	0	0	17	0	0	6
52	0	0	10	0	1	18	0	0	6
53	0	1	11	0	0	18	0	0	7
54	0	1	12	0	0	19	0	1	8
55	0	1	13	0	1	20	0	1	9
56	0	0	13	0	0	20	1	0	9
57	0	0	13	0	0	20	0	0	9
58	0	0	13	0	1	20	0	0	9
59	0	0	13	0	0	20	0	0	9
60	0	0	13	0	0	20	0	0	9

ρ represent partial restorations. The PM cost is assumed to increase linearly with ρ, reflecting a proportional trade-off between restoration efficacy and expenditure. The RL agent is thus required to learn not only when to perform maintenance but also how much restoration effort to apply, given its impact on long-term costs.

The first set of experiments evaluates the agent trained under the baseline power-law model used throughout previous sections. The performance of the learned policy was compared against the optimal periodic and single-PM analytical benchmarks. The resulting average and standard deviation of the total costs per episode are summarized in Table 10. In all cases, two-sample t-tests and Brown–Forsythe variance tests indicated statistically significant differences between the RL and the analytical policies ($p \ll 10^{-4}$), confirming that the RL policy consistently achieves lower mean and variance of costs. Fig. 11 illustrates the results.

The results demonstrate that, even when the agent must choose among multiple levels of imperfect restoration, the learned policy consistently yields lower costs and reduced variability compared to the analytical baselines. This indicates that the RL framework successfully captures the nonlinear interaction between restoration intensity, accumulated hazard, and long-term cost.

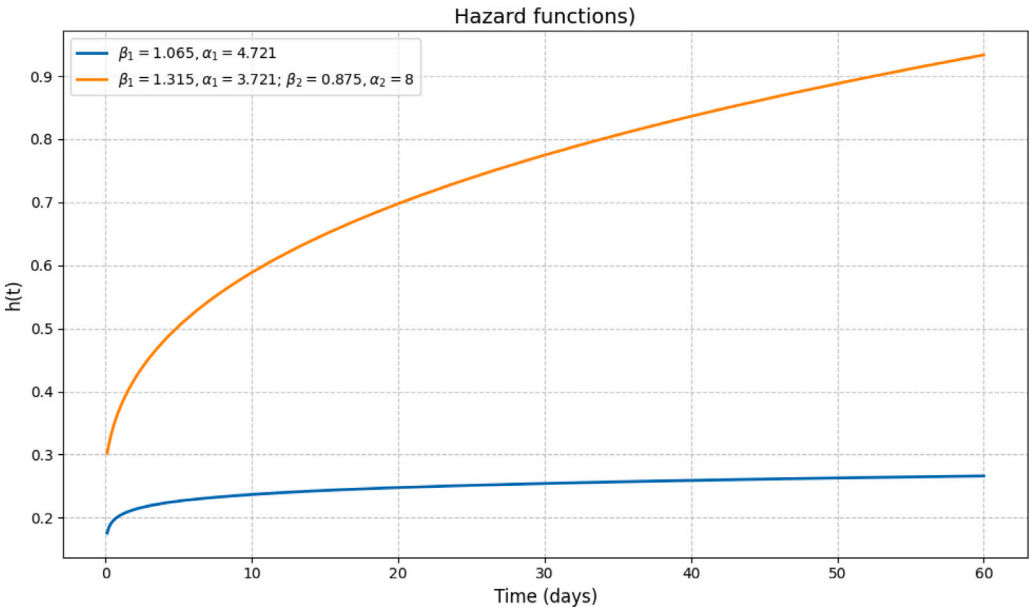

Fig. 7. Original (blue) and perturbed (orange) hazard functions.

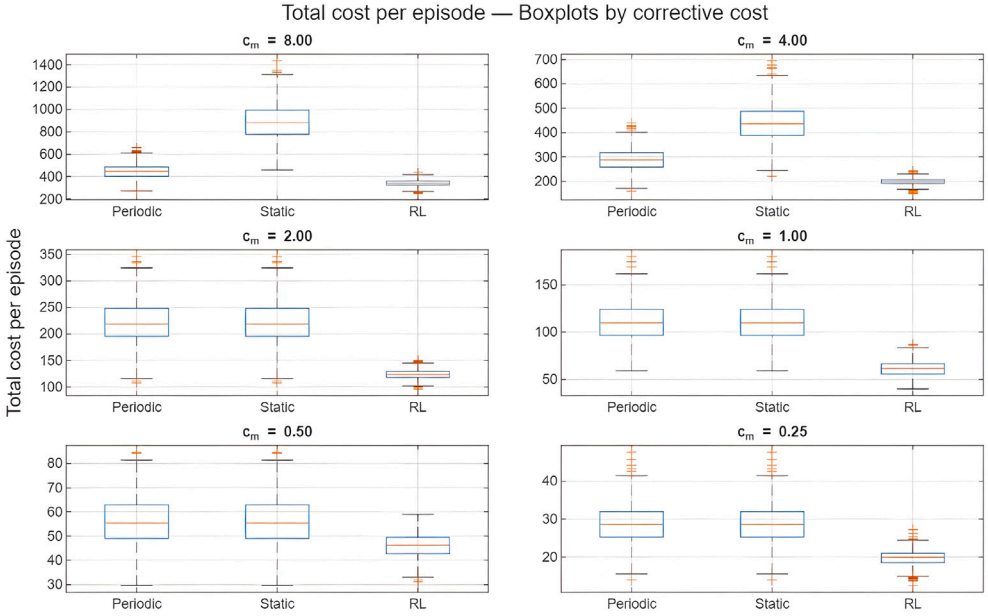

Fig. 8. Total cost per episode comparison among static, periodic and RL policies.

Table 8
Comparison of average total costs under the perturbed environment with stochastic variation of $\beta \in [\beta, \beta + 0.3]$.

c_m	τ_{periodic}	$\bar{V}_{\text{periodic}}$	σ_{periodic}	τ_{single}	$\bar{V}_{\text{single}}$	σ_{single}	$\bar{V}_{\text{RL}}$	σ_{RL}
8	9	237.54	49.04	30	398.22	108.75	230.28	29.93
4	15	143.85	33.79	30	198.58	53.27	144.09	14.57
2	30	100.81	26.32	30	101.33	26.83	94.35	7.84
1	30	51.35	13.82	30	50.76	13.56	35.94	6.54
0.5	30	25.84	6.84	30	25.78	6.89	29.96	4.97
0.25	30	13.57	3.37	30	13.58	3.34	12.38	2.03

Table 9
Comparison between RL and Periodic policies for different values of c_m. Reported are the p-values for the two-sample t-test and the Brown–Forsythe test for equality of variances.

c_m	p-value (t-test)	p-value (Brown–Forsythe)
8.000	0	1.075×10^{-80}
4.000	0	3.376×10^{-130}
2.000	0	1.911×10^{-174}
1.000	0	2.149×10^{-102}
0.500	0	7.935×10^{-89}
0.250	0	4.401×10^{-106}

These results also reveal an additional behavioral insight into the agent's decision-making process under imperfect maintenance conditions. When the corrective maintenance cost (c_m) is lower than the preventive cost, the agent behaves responsively to the environment, strategically waiting for actual failure events before deciding for maintenance interventions, thereby exploiting the lower corrective cost

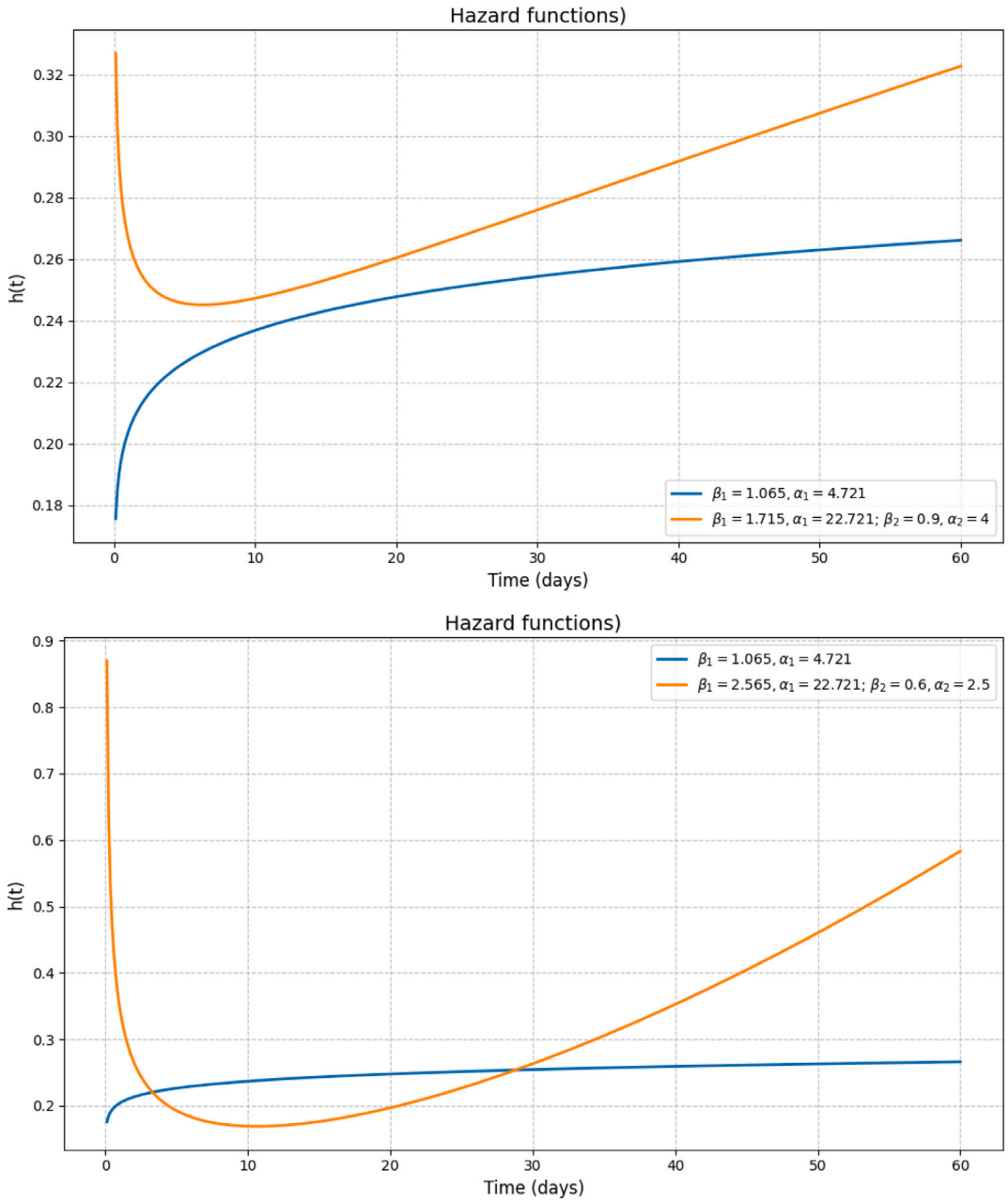

Fig. 9. Different hazard models do evaluate the model's robustness. Original (blue) and perturbed (orange) hazard function.

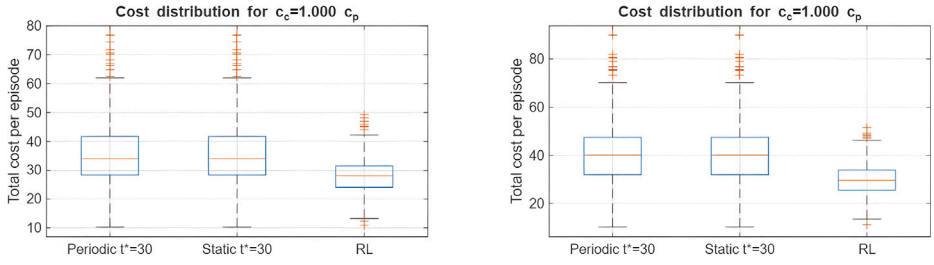

Fig. 10. Inference results of perturbed power-law models: lower risk convex hazard shape (left panel) and higher risk convex hazard shape (right panel).

regime and the post-maintenance age effect (see Table 11 for the $c_m = c_p$ case). Conversely, in scenarios where the corrective cost equals or exceeds the preventive cost, the agent adopts a more proactive stance, executing at least the minimal nonzero preventive action (ρ = 0.25) on a daily basis (see Table 12). The table presents the same simulated sequence of failures subjected to different policies, that is, policies learned for distinct cost ratios. Although the failure occurrences are identical across simulations (see Fail Flag columns), the resulting

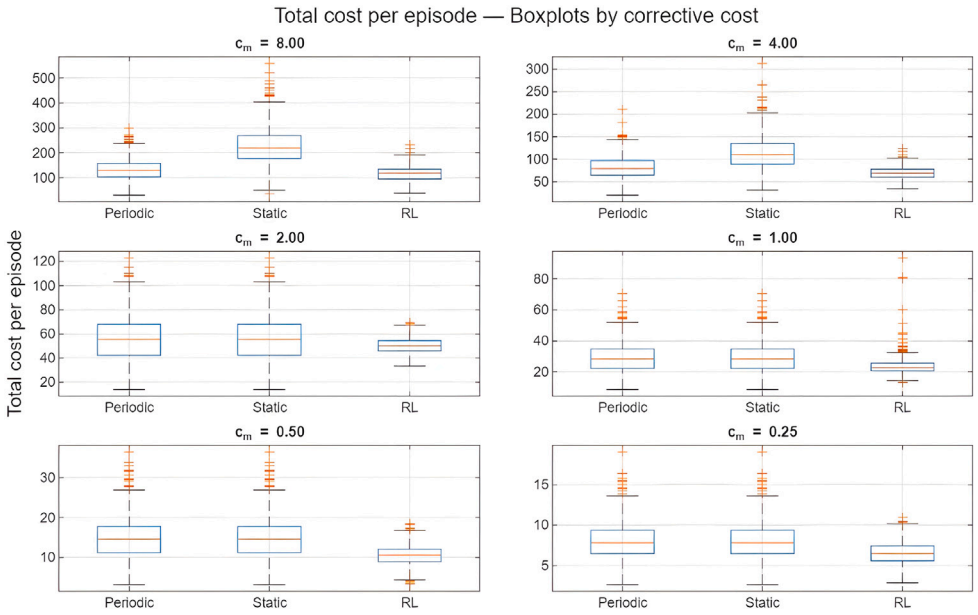

Fig. 11. Total cost per episode comparison among static, periodic and RL with imperfect action policies.

Table 10
Comparison of average total costs under imperfect preventive maintenance ($\rho \in [0, 1]$) for the evaluated model.

c_m	τ_{periodic}	$\bar{V}_{\text{periodic}}$	σ_{periodic}	τ_{single}	$\bar{V}_{\text{single}}$	σ_{single}	$\bar{V}_{\text{RL}}$	σ_{RL}
8	9	132.58	39.26	30	224.47	77.56	117.74	27.72
4	15	81.45	24.73	30	114.43	38.69	70.11	13.76
2	30	56.82	18.94	30	56.82	18.94	50.07	6.60
1	30	28.88	9.52	30	28.88	9.52	23.47	5.33
0.5	30	14.95	4.93	30	14.95	4.93	10.63	2.35
0.25	30	8.05	2.41	30	8.05	2.41	6.49	1.32

effective ages differ due to the varying levels of restoration applied by each agent (see Age Eff. columns). Each policy exhibits a unique pattern of preventive intensity, with all agents performing some level of daily intervention to minimize long-term costs. This behavior illustrates that the DDRL agent dynamically adapts its restoration decisions according to the cost structure, balancing reactivity and proactivity in a manner consistent with cost-optimal decision-making.

A second evaluation was performed using a perturbed environment that follows a two Weibull components with parameters $\alpha_1 = 4.721$, $\alpha_2 = 11$, $\beta_2 = 0.85$, and a stochastic β_1 drawn from a uniform distribution with mean 1.065 and variation of ± 0.05. The agent was evaluated assuming its mean value as the belief model, while the true environment was generated using the random realizations of β_1. The corresponding results are summarized in Table 13.

The results obtained under this configuration show that the RL policy maintains its superior performance even when evaluated in a simplified environment, where the degradation follows the mean value of the parameter β_1 used during training. In this case, the agent had been trained under heterogeneous conditions with randomly sampled β_1 values, yet was evaluated under a single, nominal model that coincides with its belief. The consistently low mean and variance of the resulting costs suggest that the policy learned from a diverse training environment is able to retain its effectiveness when applied to a specific and less variable scenario. This indicates that exposure to a broader range of dynamics during training enhances the agent's ability to form stable and robust decision rules, rather than overfitting to particular degradation trajectories.

Moreover, when the same agent trained under randomized β_1 values was evaluated in environments that also exhibited random β_1 realizations, its performance remained statistically indistinguishable from the case where the evaluation used the nominal (mean) β_1. Both the mean and variance of the total cost distributions showed no significant deviation according to the t-test and Brown–Forsythe results. This consistency indicates that the agent effectively internalized the variability of the degradation dynamics during training, achieving stable performance across both deterministic and stochastic instances of the failure model. Such behavior reinforces the interpretation that the learned policy is not sensitive to small parameter fluctuations and demonstrates an inherent capacity to generalize within the same family of degradation processes.

4.5. Further discussions

We present additional points related to the results that supplement and extend the previously mentioned findings. By highlighting these points, we aim to enrich academic literature and establish a more robust foundation for future research in this field.

Notably, by considering the power-law hazard model with $\beta = 1$, or any other combination of β_i parameters of the bathtub model that reflect a constant failure rate, the RL agent did not predict any preventive maintenance intervals (see Pham (2003) and da Silva (2023)). In a scenario where the risk function remains constant, preventive maintenance is ineffective, as predicted by reliability engineering theory. The RL agent astutely recognized the impracticality of preventive actions in situations where the risk remained unaltered, showing a nuanced understanding of the underlying system dynamics.

Interestingly as in the previous case, the RL agent learned that when the recovery factor is set to zero, no preventive maintenance is necessary. It is important to remember that the RL agent does not have information regarding the post-maintenance effect on the hazard function. The agent observes only the behavior of time between the subsequent failures. This aligns with the logical expectation that in the absence of any recovery from maintenance actions, the cost-effectiveness of preventive measures vanishes, and the agent correctly adapts its strategy accordingly.

An aspect observed during the simulations involves adaptive rescheduling of preventive maintenance when an unexpected failure or unscheduled maintenance occurs. The agent dynamically adjusts the subsequent preventive maintenance schedule based on the occurrence of these events.

Table 11
Full trajectories of three RL realizations under imperfect actions for $c_m = c_p$.

Day	Episode 1			Episode 2			Episode 3		
	PM Action	Fail Flag	Age Eff.	PM Action	Fail Flag	Age Eff.	PM Action	Fail Flag	Age Eff.
1	0.25	0	0.75	0.25	0	0.75	0.25	1	0.75
2	0.25	0	1.3125	0.25	0	1.3125	0.25	0	1.3125
3	0	0	2.3125	0	1	2.3125	0	0	2.3125
4	0	0	3.3125	0	0	3.3125	0	0	3.3125
5	0	1	4.3125	0	0	4.3125	0	0	4.3125
6	0	1	5.3125	0	1	5.3125	0	0	5.3125
7	1	1	0	1	0	0	0	0	6.3125
8	0.25	1	0.75	0.25	1	0.75	0	0	7.3125
9	0.25	1	1.3125	0.25	0	1.3125	0	0	8.3125
10	0	1	2.3125	0	0	2.3125	0	0	9.3125
11	0	0	3.3125	0	1	3.3125	0	1	10.3125
12	0	0	4.3125	0	0	4.3125	1	0	0
13	0	0	5.3125	0	0	5.3125	0.25	0	0.75
14	0	0	6.3125	0	0	6.3125	0.25	0	1.3125
15	0	0	7.3125	0	0	7.3125	0	1	2.3125
16	0	1	8.3125	1	0	0	0	0	3.3125
17	1	1	0	0.25	0	0.75	0	0	4.3125
18	0.25	0	0.75	0.25	0	1.3125	0	1	5.3125
19	0	0	1.75	0	0	2.3125	1	1	0
20	0	0	2.75	0	0	3.3125	0.25	0	0.75
21	0	0	3.75	0	1	4.3125	0.25	0	1.3125
22	0	0	4.75	0	0	5.3125	0	1	2.3125
23	0	0	5.75	1	0	0	0	0	3.3125
24	0	0	6.75	0.25	0	0.75	0	0	4.3125
25	0	1	7.75	0.25	0	1.3125	0	0	5.3125
26	1	0	0	0	0	2.3125	1	1	0
27	0.25	0	0.75	0	0	3.3125	0.25	1	0.75
28	0	0	1.75	0	0	4.3125	0	0	1.75
29	0	0	2.75	0	0	5.3125	0	0	2.75
30	0	0	3.75	0	0	6.3125	0	1	3.75
31	0	0	4.75	0	0	7.3125	0	1	4.75
32	0	0	5.75	1	0	0	1	0	0
33	0	0	6.75	0.25	0	0.75	0.25	1	0.75
34	0	0	7.75	0.25	0	1.3125	0	0	1.75
35	0	1	8.75	0	0	2.3125	0	0	2.75
36	1	0	0	0	0	3.3125	0	0	3.75
37	0.25	0	0.75	0	1	4.3125	0	0	4.75
38	0	0	1.75	1	0	0	0	0	5.75
39	0	0	2.75	0.25	0	0.75	0	0	6.75
40	0	0	3.75	0.25	0	1.3125	0	1	7.75
41	0	0	4.75	0	0	2.3125	1	0	0
42	0	0	5.75	0	1	3.3125	0.25	0	0.75
43	0	0	6.75	1	0	0	0.25	0	1.3125
44	0	0	7.75	0.25	0	0.75	0	0	2.3125
45	0	0	8.75	0.25	0	1.3125	0	0	3.3125
46	1	1	0	0	0	2.3125	0	0	4.3125
47	0.25	0	0.75	0	0	3.3125	0	0	5.3125
48	0.25	0	1.3125	0	1	4.3125	0	1	6.3125
49	0	0	2.3125	0	0	5.3125	0.25	0	5.4844
50	0	0	3.3125	0	0	6.3125	0	0	6.4844
51	0	0	4.3125	0	0	7.3125	0	0	7.4844
52	0	0	5.3125	0	0	8.3125	0	0	8.4844
53	0	1	6.3125	0	1	9.3125	0	0	9.4844
54	0.25	0	5.4844	0.25	0	7.7344	0	0	10.484
55	0	0	6.4844	0	0	8.7344	0	0	11.484
56	0	0	7.4844	0	1	9.7344	0	1	12.484
57	0	0	8.4844	0.25	1	8.0508	0.25	0	10.113
58	0	0	9.4844	0	1	9.0508	0	1	11.113
59	0	0	10.4844	0	0	10.051	0	0	12.113
60	0	0	11.4844	0	0	11.051	0	0	13.113

Dynamic maintenance task scheduling allows for real-time adaptation to changes, improves equipment reliability, and reduces maintenance costs. As noted by Byon and Ding (2010), dynamic maintenance strategies can lead to considerable improvements in reliability and costs compared to static strategies. As industries continue to embrace more data-driven and intelligent systems (see e.g. Wehbi et al. (2026)), the ability to dynamically adapt to changing conditions and optimize maintenance schedules will become increasingly valuable.

A comparison between the baseline configuration adopted in this study and that proposed by van Hasselt et al. (2015) shows a close alignment in key hyperparameters. Both use ReLU activations, a discount factor of $\gamma = 0.99$, and target network updates at fixed intervals, confirming their effectiveness in stabilizing DDQN learning. Despite addressing problems of very different dimensionalities, the configurations remain comparable. The main differences reflect adaptations to the maintenance optimization context: a smaller replay buffer (3×10^5 vs. 1M) improved responsiveness to recent experiences, and a shorter training horizon proved sufficient for convergence in a low-dimensional state space. Moreover, a higher target update frequency (every 1000 steps vs. 10,000) enhanced stability under sparse-reward conditions. Overall, these adjustments preserve the core structure of the original

Table 12
Full trajectories of three RL realizations under different maintenance costs.

Day	$c_m = 2$			$c_m = 4$			$c_m = 8$		
	PM Action	Fail Flag	Age Eff.	PM Action	Fail Flag	Age Eff.	PM Action	Fail Flag	Age Eff.
1	0.25	0	0.75	0.25	0	0.75	0.25	0	0.75
2	0.25	0	1.31	0.25	0	1.31	0.25	0	1.31
3	0.5	0	1.16	0.25	0	1.73	0.25	0	1.73
4	0.5	0	1.08	0.25	0	2.05	0.25	0	2.05
5	0.5	0	1.04	0.25	0	2.29	0.25	0	2.29
6	0.5	0	1.02	0.25	0	2.47	0.25	0	2.47
7	0.5	1	1.01	0.25	1	2.60	0.25	1	2.60
8	0.25	0	1.51	0.5	0	1.80	0.25	0	2.70
9	0.5	1	1.25	0.25	1	2.10	0.25	1	2.77
10	0.5	0	1.13	0.5	0	1.55	0.25	0	2.83
11	0.5	1	1.06	0.25	1	1.91	0.25	1	2.87
12	0.25	0	1.55	0.5	0	1.46	0.25	0	2.91
13	0.5	0	1.27	0.25	0	1.84	0.25	0	2.93
14	0.5	0	1.14	0.25	0	2.13	0.25	0	2.95
15	0.5	0	1.07	0.5	0	1.57	0.25	0	2.96
16	0.5	0	1.03	0.25	0	1.92	0.25	0	2.97
17	0.5	1	1.02	0.25	1	2.19	0.25	1	2.98
18	0.25	0	1.51	0.5	0	1.60	0.25	0	2.98
19	0.5	0	1.26	0.5	0	1.30	0.25	0	2.99
20	0.5	0	1.13	0.25	0	1.72	0.25	0	2.99
21	0.5	0	1.06	0.25	0	2.04	0.25	0	2.99
22	0.25	0	1.55	0.5	0	1.52	0.25	0	2.99
23	0.5	0	1.27	0.25	0	1.89	0.25	0	2.99
24	0.5	1	1.14	0.25	1	2.17	0.25	1	3.00
25	0.5	0	1.07	0.5	0	1.58	0.25	0	3.00
26	0.25	0	1.55	0.5	0	1.29	0.25	0	3.00
27	0.5	0	1.28	0.25	0	1.72	0.25	0	3.00
28	0.5	0	1.14	0.25	0	2.04	0.25	0	3.00
29	0.5	0	1.07	0.5	0	1.52	0.25	0	3.00
30	0.25	0	1.55	0.25	0	1.89	0.25	0	3.00
31	0.5	0	1.28	0.25	0	2.17	0.25	0	3.00
32	0.5	0	1.14	0.5	0	1.58	0.25	0	3.00
33	0.5	0	1.07	0.25	0	1.94	0.25	0	3.00
34	0.25	0	1.55	0.25	0	2.20	0.25	0	3.00
35	0.5	0	1.28	0.5	0	1.60	0.25	0	3.00
36	0.5	1	1.14	0.25	1	1.95	0.25	1	3.00
37	0.25	0	1.60	0.25	0	2.21	0.25	0	3.00
38	0.5	0	1.30	0.25	0	2.41	0.25	0	3.00
39	0.5	0	1.15	0.5	0	1.71	0.25	0	3.00
40	0.25	0	1.61	0.25	0	2.03	0.25	0	3.00
41	0.5	0	1.31	0.25	0	2.27	0.25	0	3.00
42	0.5	0	1.15	0.25	0	2.45	0.25	0	3.00
43	0.25	0	1.62	0.5	0	1.73	0.25	0	3.00
44	0.5	0	1.31	0.25	0	2.05	0.25	0	3.00
45	0.5	0	1.15	0.25	0	2.28	0.25	0	3.00
46	0.25	0	1.62	0.25	0	2.46	0.25	0	3.00
47	0.5	0	1.31	0.5	0	1.73	0.25	0	3.00
48	0.25	0	1.73	0.25	0	2.05	0.25	0	3.00
49	0.75	0	0.68	0.25	0	2.29	0.25	0	3.00
50	0.25	0	1.26	0.5	0	1.64	0.25	0	3.00
51	0.25	0	1.70	0.25	0	1.98	0.25	0	3.00
52	0.75	0	0.67	0.5	0	1.49	0.25	0	3.00
53	0.25	0	1.26	0.25	0	1.87	0.25	0	3.00
54	0.25	1	1.69	0.5	1	1.43	0.25	1	3.00
55	0.75	0	0.67	0.25	0	1.83	0.25	0	3.00
56	0.25	0	1.25	0.25	0	2.12	0.25	0	3.00
57	0.75	0	0.56	0.5	0	1.56	0.25	0	3.00
58	0.5	0	0.78	0.25	0	1.92	0.25	0	3.00
59	0.5	0	0.89	0.25	0	2.19	0.25	0	3.00
60	0.25	0	1.42	0.25	0	2.39	0.25	0	3.00

DDQN setup while optimizing it for faster and more stable learning in this domain.

4.6. Limitations and future research

In this subsection, we critically evaluate the main limitations of the proposed methodology. The discussion aims to clarify the scope of the findings and identify promising directions for future work to overcome these constraints.

1. *Fixed Hazard Functions and Generalization*
   Except for the last case examined, the agents were trained under specific baseline hazard functions and evaluated across variations of power law model. Although the results show robustness under perturbations, a systematic generalization study, both during training and inference, was not conducted. Future work should explore transfer learning and domain randomization techniques to assess the model's ability to generalize across a broader range of failure dynamics and parameter uncertainties.
2. *Fixed Time Horizon*
   All experiments were conducted within a fixed planning horizon, with the objective of minimizing the cumulative cost over this

Table 13
Performance under imperfect maintenance and perturbed failure model with random β_1.

c_m	τ_{periodic}	$\bar{V}_{\text{periodic}}$	σ_{periodic}	τ_{single}	$\bar{V}_{\text{single}}$	σ_{single}	$\bar{V}_{\text{RL}}$	σ_{RL}
8	9	208.28	47.45	30	335.59	98.72	64.27	18.61
4	15	123.61	30.25	30	167.91	47.37	39.48	9.31
2	30	83.55	24.64	30	83.55	24.64	26.98	4.60
1	30	42.40	12.44	30	42.40	12.44	20.92	2.28
0.5	30	21.87	6.23	30	21.87	6.23	12.88	2.08
0.25	30	11.55	3.04	30	11.55	3.04	8.67	1.42

predefined time window. Consequently, it remains uncertain whether the learned policy can generalize to different time horizons or maintain optimal behavior in long-term or continuous operation settings. Future studies should investigate adaptive or open-horizon formulations that allow the agent to optimize over variable planning intervals.

3. *Single-Component Assumption*
The proposed model focuses on a single-component maintenance problem with one failure process. As such, the agents were trained to handle a single hazard function, which limits scalability to systems composed of multiple interdependent components. Extending the framework to multi-action or multi-agent reinforcement learning architectures could enable the coordination of maintenance across multiple assets, where each component's state and cost dynamics influence the global decision-making policy.

5. Conclusions

This study presented a Double Deep Q-Network (DDQN) framework for the dynamic optimization of preventive maintenance policies under stochastic failure conditions. The proposed method learns maintenance decisions directly from failure-event and cost data, without requiring explicit condition monitoring or health indicators. By formulating maintenance as a Markov Decision Process with a structured state representation and dense reward shaping, the approach effectively learns adaptive preventive actions that minimize the total maintenance costs across different cost ratios and maintenance regimes.

Empirical analyses demonstrated that the DDQN-based policy consistently achieved lower expected costs and reduced cost variability when compared with analytical periodic and static benchmarks, even under model perturbations. These results confirm the capacity of the method to adapt to imperfect or uncertain system behavior while maintaining training stability through target-network synchronization, experience replay, and L_2 regularization. Experiments with varying restoration factors ($\rho \in [0, 1]$) further showed that the learned policy generalizes across both perfect and imperfect maintenance settings, preserving the effectiveness when evaluated under perturbed hazard models.

From a practical perspective, the proposed framework highlights the feasibility of applying data-driven maintenance optimization in industrial contexts where health condition data may be unavailable. Integration with existing Computerized Maintenance Management Systems involves coupling the DDQN agent with modules that record event-based maintenance data, taking decisions in real time. Such integration enables the model to operate in closed-loop maintenance decision environments.

Despite promising outcomes, several methodological limitations should be acknowledged. Training and evaluation were conducted in a single-component setting with stationary cost parameters, and the reward function was shaped to provide dense learning feedback rather than the true economic objective. Although this design choice improves learning stability, it introduces an abstraction that may affect the interpretability of the learned policies in practical deployments. Furthermore, the assumption of fully observable state variables, while valid under simulation, may not hold in real settings where failure indicators or maintenance histories are incomplete.

Future research should extend this framework to multi-component systems, partially observable environments, and hybrid reliability models that combine real sensor data with stochastic hazard estimation. Additionally, exploring transfer-learning could enhance adaptation to heterogeneous fleets and evolving failure behaviors. These extensions would further evaluate the generalization capacity of the proposed DDQN architecture in complex industrial maintenance scenarios.

CRediT authorship contribution statement

Allan Jonathan da Silva: Writing – original draft, Writing – review & editing, Validation, Supervision, Software, Methodology, Conceptualization. **Luís Domingues Tomé Jardim Tarrataca:** Writing – review & editing, Software, Supervision, Methodology. **Leonardo Fagundes de Mello:** Writing – review & editing, Software, Methodology. **Fabricio Maione Tenório:** Writing – original draft, Validation, Conceptualization. **Rodrigo Rodrigues de Freitas:** Validation, Conceptualization. **Felipe do Carmo Amorim:** Writing – review & editing, Validation, Conceptualization. **Marcio Antelio Neves da Silva:** Software. **Cintia Machado de Oliveira:** Validation, Conceptualization.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used Paperpal (version 3.104) and ChatGPT in order to translate, check grammar and spelling, and receive feedback on omissions in academic language. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the content of the publication.

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.

Acknowledgements

The research reported in this article did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

However, the Article Processing Charge (APC) for open access publication is covered institutionally under the Read and Publish Agreement between CAPES (Coordination for the Improvement of Higher Education Personnel, Brazil) and Elsevier.

Data availability

No data was used for the research described in the article.

References

Baek, J.-G. (2007). An intelligent condition-based maintenance scheduling model. *International Journal of Quality & Reliability Management*, *24*(3), 312–327. http://dx.doi.org/10.1108/02656710710730898.

Bellman, R. (1958). Dynamic programming and stochastic control processes. *Information and Control*, *1*(3), 228–239. http://dx.doi.org/10.1016/S0019-9958(58)80003-0, URL https://www.sciencedirect.com/science/article/pii/S0019995858800030.

Berrichi, A., Amodeo, L., Yalaoui, F., Châtelet, E., & Mezghiche, M. (2009). Bi-objective optimization algorithms for joint production and maintenance scheduling: application to the parallel machine problem. *Journal of Intelligent Manufacturing*, *20*(4), 389–400. http://dx.doi.org/10.1007/s10845-008-0113-5, URL https://doi.org/10.1007/s10845-008-0113-5.

Birolini, A. (1996). Reliability engineering: Cooperation between university and industry at the ETH zurich. *Quality Engineering*, *8*(4), 659–674. http://dx.doi.org/10.1080/08982119608904677.

Blazewicz, J., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, *5*(1), 11–24.

Brim, A. (2020). Deep reinforcement learning pairs trading with a double deep Q-network. In *2020 10th annual computing and communication workshop and conference* (pp. 0222–0227). http://dx.doi.org/10.1109/CCWC47524.2020.9031159.

Brown, M. B., & Forsythe, A. B. (1974). Robust tests for the equality of variances. *Journal of the American Statistical Association*, *69*(346), 364–367. http://dx.doi.org/10.1080/01621459.1974.10482955, https://www.tandfonline.com/doi/pdf/10.1080/01621459.1974.10482955.

Bukhsh, Z. A., Molegraaf, H., & Jansen, N. (2025). A maintenance planning framework using online and offline deep reinforcement learning. *Neural Computing and Applications*, *37*, http://dx.doi.org/10.1007/s00521-023-08560-7.

Byon, E., & Ding, Y. (2010). Season-dependent condition-based maintenance for a wind turbine using a partially observed Markov decision process. *IEEE Transactions on Power Systems*, *25*(4), 1823–1834. http://dx.doi.org/10.1109/TPWRS.2010.2043269.

Castro, P. M., Grossmann, I. E., Veldhuizen, P., & Esplin, D. (2014). Optimal maintenance scheduling of a gas engine power plant using generalized disjunctive programming. *AIChE Journal*, *60*(6), 2083–2097. http://dx.doi.org/10.1002/aic.14412, https://aiche.onlinelibrary.wiley.com/doi/pdf/10.1002/aic.14412.

Chen, T., & Chen, H. (1995). Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, *6*(4), 911–917. http://dx.doi.org/10.1109/72.392253.

de Souza, F. L. C., & da Silva, A. J. (2024). Statistical learning for maintenance optimization: modeling the hazard function with variable recovery factors. *Revista de GestÃo E Secretariado*, *15*(2), http://dx.doi.org/10.7769/gesec.v15i2.3478.

Diamoutene, A., Noureddine, F., Kamsu-Foguem, B., & Barro, D. (2021). Reliability analysis with proportional hazard model in aeronautics. *International Journal of Aeronautical and Space Sciences*, *22*(5), 1222–1234. http://dx.doi.org/10.1007/s42405-021-00371-1.

Duffuaa, S., Raouf, A., & Campbell, J. (1999). *Planning and Control of Maintenance Systems: Modeling and Analysis*. Wiley.

El-Sharkh, M., & El-Keib, A. (2003). Maintenance scheduling of generation and transmission systems using fuzzy evolutionary programming. *IEEE Transactions on Power Systems*, *18*(2), 862–866. http://dx.doi.org/10.1109/TPWRS.2003.811004.

Elsayed, E. A. (2021). Reliability engineering. In *Wiley series in systems engineering and management*, Wiley.

Esary, J. D., Marshall, A. W., & Proschan, F. (1970). Some reliability applications of the hazard transform. *SIAM Journal on Applied Mathematics*, *18*(4), 849–860, URL http://www.jstor.org/stable/2099436.

Gaonkar, A., Patil, R. B., Kyeong, S., Das, D., & Pecht, M. G. (2021). An assessment of validity of the bathtub model hazard rate trends in electronics. *IEEE Access*, *9*, 10282–10290. http://dx.doi.org/10.1109/ACCESS.2021.3050474.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: a guide to the theory of NP-completeness*. W. H. Freeman & Co..

Harudin, N., & Yusof, S. M. (2014). Speeding maintenance performance through time study. *Applied Mechanics and Materials*, *607*, 860–863.

van Hasselt, H. (2010). Double Q-learning. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, & A. Culotta (Eds.), *Advances in neural information processing systems: Vol. 23*, Curran Associates, Inc..

van Hasselt, H., Guez, A., & Silver, D. (2015). Deep reinforcement learning with double Q-learning. arXiv:1509.06461.

Huang, J., Chang, Q., & Arinez, J. (2020). Deep reinforcement learning based preventive maintenance policy for serial production lines. *Expert Systems with Applications*, *160*, Article 113701. http://dx.doi.org/10.1016/j.eswa.2020.113701.

Huang, C., Chen, G., Gong, Y., & Han, Z. (2021). Joint buffer-aided hybrid-duplex relay selection and power allocation for secure cognitive networks with double deep Q-network. *IEEE Transactions on Cognitive Communications and Networking*, *7*(3), 834–844. http://dx.doi.org/10.1109/TCCN.2021.3063525.

Jardine, A. K., Lin, D., & Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, *20*(7), 1483–1510. http://dx.doi.org/10.1016/j.ymssp.2005.09.012, URL https://www.sciencedirect.com/science/article/pii/S0888327005001512.

Javadnejad, F., Sharifi, M. R., Basiri, M. H., & Ostadi, B. (2022). Optimization model for maintenance planning of loading equipment in open pit mines. *European Journal of Engineering and Technology Research*, *7*(5), 94–101. http://dx.doi.org/10.24018/ejeng.2022.7.5.2907, URL https://www.ej-eng.org/index.php/ejeng/article/view/2907.

Khan, M. S., & King, R. (2012). Modified inverse Weibull distribution. *Journal of Statistics Applications and Probability*, *1*, 115–132. http://dx.doi.org/10.12785/jsap/010204.

Kister, T., & Hawkins, B. (2006). Maintenance planning and scheduling: Streamline your organization for a lean environment. Elsevier Butterworth-Heinemann.

Kokhanovskiy, A., Shevelev, A., Serebrennikov, K., Kuprikov, E., & Turitsyn, S. (2022). A deep reinforcement learning algorithm for smart control of hysteresis phenomena in a mode-locked fiber laser. *Photonics*, *9*(12), http://dx.doi.org/10.3390/photonics9120921, URL https://www.mdpi.com/2304-6732/9/12/921.

Kumar, U., & Parida, A. (2008). Maintenance performance measurement (MPM) system. In K. A. H. Kobbacy, & D. N. P. Murthy (Eds.), *Complex system maintenance handbook* (pp. 459–478). London: Springer London, http://dx.doi.org/10.1007/978-1-84800-011-7.

Lenstra, J. K., & Rinnooy Kan, A. H. G. (1978). Complexity of scheduling under precedence constraints. *Operations Research*, *26*(1), 22–35.

Levitt, J. (2003). *Complete guide to preventive and predictive maintenance*. Industrial Press.

Liu, X., Liu, Y., Chen, Y., Wang, L., & Lu, Z. (2020). Reinforcement learning in V2I communication assisted autonomous driving. In *ICC 2020 - 2020 IEEE international conference on communications* (pp. 1–6). http://dx.doi.org/10.1109/ICC40277.2020.9148831.

Malik, M. A. K. (1979). Reliable preventive maintenance scheduling. *A I I E Transactions*, *11*(3), 221–228. http://dx.doi.org/10.1080/05695557908974463.

Meeker, W., & Escobar, L. (2014). Statistical methods for reliability data. *Wiley series in probability and statistics*, Wiley.

Moniri-Morad, A., & Sattarvand, J. (2023). A comparative study between system reliability evaluation methods: Case study of mining dump trucks. *Journal of Engineering and Applied Science*, *70*(1), 103. http://dx.doi.org/10.1186/s44147-023-00272-y.

Murthy, D. (1979). Parameter estimation with noncontinuous inspection. *IEEE Transactions on Reliability*, *R-28*(2), 148–149. http://dx.doi.org/10.1109/TR.1979.5220530.

Nor, A. K. M., Pedapati, S. R., & Muhammad, M. (2021). Reliability engineering applications in electronic, software, nuclear and aerospace industries: A 20 year review (2000–2020). *Ain Shams Engineering Journal*, *12*(3), 3009–3019. http://dx.doi.org/10.1016/j.asej.2021.02.015, URL https://www.sciencedirect.com/science/article/pii/S2090447921001179.

Patiño-Rodriguez, C. E., & Carazas, F. J. G. (2019). Maintenance and asset life cycle for reliability systems. In L. Kounis (Ed.), *Reliability and maintenance - an overview of cases*. Rijeka: IntechOpen, http://dx.doi.org/10.5772/intechopen.85845, URL https://doi.org/10.5772/intechopen.85845.

Pham, H. (2003). *Handbook of reliability engineering*. Springer London, http://dx.doi.org/10.1007/b97414.

Pintelon, L., & Parodi-Herz, A. (2008). Maintenance: An evolutionary perspective. In K. A. H. Kobbacy, & D. N. P. Murthy (Eds.), *Complex system maintenance handbook* (pp. 21–48). London: Springer London, http://dx.doi.org/10.1007/978-1-84800-011-7_2.

Powell, W. B. (2019). *Reinforcement learning and stochastic optimization: a unified framework for sequential decisions*. Wiley.

Rajendran, P., & Geetha, A. (2021). Optimization of hospital bed occupancy in hospitals using double deep q network (DDQN). In *2021 third international conference on intelligent communication technologies and virtual mobile networks* (pp. 1029–1033). http://dx.doi.org/10.1109/ICICV50876.2021.9388626.

Ravichandiran, S. (2020). Deep reinforcement learning with python: Master classic RL, deep RL, distributional RL, inverse RL, and more with openAI gym and Tensorflow, 2nd edition. *Expert insight*, Packt Publishing.

Ren, S., Zhang, Y., Liu, Y., Sakao, T., Huisingh, D., & Almeida, C. M. (2019). A comprehensive review of big data analytics throughout product lifecycle to support sustainable smart manufacturing: A framework, challenges and future research directions. *Journal of Cleaner Production*, *210*, 1343–1365. http://dx.doi.org/10.1016/j.jclepro.2018.11.025, URL https://www.sciencedirect.com/science/article/pii/S0959652618334255.

Ruiz-Rodríguez, M. L., Kubler, S., Robert, J., & Le Traon, Y. (2024). Dynamic maintenance scheduling approach under uncertainty: Comparison between reinforcement learning, genetic algorithm simheuristic, dispatching rules. *Expert Systems with Applications*, *248*, Article 123404. http://dx.doi.org/10.1016/j.eswa.2024.123404, URL https://www.sciencedirect.com/science/article/pii/S0957417424002690.

Ryu, J., Kwon, J., Ryoo, J.-D., Cheung, T., & Joung, J. (2023). Timeslot scheduling with reinforcement learning using a double deep Q-network. *Electronics*, *12*(4), http://dx.doi.org/10.3390/electronics12041042, URL https://www.mdpi.com/2079-9292/12/4/1042.

Saraygord Afshari, S., Jonnadula, B., Xu, X., Liang, X., & Yang, Z. (2022). Jet engine optimal preventive maintenance scheduling using golden section search and genetic algorithm. *Journal of Prognostics and Health Management*, *2*(1), 45–71. http://dx.doi.org/10.22215/jphm.v2i1.3321, URL https://ojs.library.carleton.ca/index.php/jphm/article/view/3321.

Sedghi, M., Bergquist, B., Vanhatalo, E., & Migdalas, A. (2022). Data-driven maintenance planning and scheduling based on predicted railway track condition. *Quality and Reliability Engineering International*, *38*(7), 3689–3709. http://dx.doi.org/10.1002/qre.3166, https://onlinelibrary.wiley.com/doi/pdf/10.1002/qre.3166.

Shin, I., Lim, T., & Lie, C. (1996). Estimating parameters of intensity function and maintenance effect for repairable unit. *Reliability Engineering & System Safety*, *54*(1), 1–10. http://dx.doi.org/10.1016/S0951-8320(96)00097-X, URL https://www.sciencedirect.com/science/article/pii/S095183209600097X.

Sikorska, J., Hodkiewicz, M., & Ma, L. (2011). Prognostic modelling options for remaining useful life estimation by industry. *Mechanical Systems and Signal Processing*, *25*(5), 1803–1836. http://dx.doi.org/10.1016/j.ymssp.2010.11.018, URL https://www.sciencedirect.com/science/article/pii/S0888327010004218.

da Silva, A. J. (2023). Maintenance policy costs considering imperfect repairs. *Reliability: Theory & Applications*, *18*(1 (72)), 564–574.

da Silva, A. J., Gomes, C. A. A., de Souza, L. R., & Santos, R. S. d. (2025). Optimal accelerated life testing design under constrained resources using double deep Q-learning. *Quality and Reliability Engineering International*, http://dx.doi.org/10.1002/qre.70134.

Sresakoolchai, J., & Kaewunruen, S. (2023). Railway infrastructure maintenance efficiency improvement using deep reinforcement learning integrated with digital twin based on track geometry and component defects. *Scientific Reports*, *13*, http://dx.doi.org/10.1038/s41598-023-29526-8.

Sutton, R., & Barto, A. (2018). Reinforcement learning, second edition: An introduction. In *Adaptive computation and machine learning series*, MIT Press.

Swanson, L. (2001). Linking maintenance strategies to performance. *International Journal of Production Economics*, *70*(3), 237–244. http://dx.doi.org/10.1016/S0925-5273(00)00067-0, URL https://www.sciencedirect.com/science/article/pii/S0925527300000670.

Szepesvári, C. (2010). Algorithms for reinforcement learning. In *Synthesis lectures on artificial intelligence and machine learning*, Morgan & Claypool.

Tanhaeean, M., Ghaderi, S. F., Sheikhalishahi, M., & Khoobani, M. (2025). Reliability-based reinforcement learning driven maintenance policy optimization. *Structure and Infrastructure Engineering*, 1–11. http://dx.doi.org/10.1080/15732479.2025.2451277, https://doi.org/10.1080/15732479.2025.2451277.

The MathWorks Inc. (2022). *MATLAB version: 9.12.0 (R2022a)*. Natick, Massachusetts, United States: The MathWorks Inc., Retrieved from. URL https://www.mathworks.com.

Tsang, A. H. C. (2002). Strategic dimensions of maintenance management. *Journal of Quality in Maintenance Engineering*, *8*(1), 7–39. http://dx.doi.org/10.1108/13552510210420577.

Wang, H. (2002). A survey of maintenance policies of deteriorating systems. *European Journal of Operational Research*, *139*(3), 469–489. http://dx.doi.org/10.1016/S0377-2217(01)00197-7, URL https://www.sciencedirect.com/science/article/pii/S0377221701001977.

Wang, Y., Zhou, X., Zhou, H., Chen, L., Zheng, Z., Zeng, Q., Cai, S., & Wang, Q. (2021). Transmission network dynamic planning based on a double deep-q network with deep ResNet. *IEEE Access*, *9*, 76921–76937. http://dx.doi.org/10.1109/ACCESS.2021.3083266.

Wehbi, L., Bhulai, S., & Slik, J. (2026). Optimizing maintenance of heavy equipment: A data-driven approach. *Computers & Industrial Engineering*, *211*, Article 111645. http://dx.doi.org/10.1016/j.cie.2025.111645, URL https://www.sciencedirect.com/science/article/pii/S0360835225007910.

Yang, Y., Zhang, B., Dauzère-Pérès, S., & Wu, C.-H. (2025). Control-limit policies for coordinated dispatching and preventive maintenance in multi-product manufacturing systems. *European Journal of Operational Research*, http://dx.doi.org/10.1016/j.ejor.2025.12.041, URL https://www.sciencedirect.com/science/article/pii/S0377221725010082.

Yuan, J., Lei, Y., Li, N., Yang, B., Li, X., Chen, Z., & Han, W. (2025). A framework for modeling and optimization of mechanical equipment considering maintenance cost and dynamic reliability via deep reinforcement learning. *Reliability Engineering & System Safety*, *264*, Article 111424. http://dx.doi.org/10.1016/j.ress.2025.111424, URL https://www.sciencedirect.com/science/article/pii/S0951832025006246.

Zhang, Q., Liu, Y., Xiang, Y., & Xiahou, T. (2024). Reinforcement learning in reliability and maintenance optimization: A tutorial. *Reliability Engineering & System Safety*, *251*, Article 110401. http://dx.doi.org/10.1016/j.ress.2024.110401, URL https://www.sciencedirect.com/science/article/pii/S0951832024004733.

Zhong, S., Pantelous, A. A., Goh, M., & Zhou, J. (2019). A reliability-and-cost-based fuzzy approach to optimize preventive maintenance scheduling for offshore wind farms. *Mechanical Systems and Signal Processing*, *124*, 643–663. http://dx.doi.org/10.1016/j.ymssp.2019.02.012, URL https://www.sciencedirect.com/science/article/pii/S0888327019300950.

Zhu, Y., Zheng, M., Su, Z., Xia, T., Lin, J., & Pan, E. (2025). Deep reinforcement learning for joint optimization of maintenance and spare parts ordering considering spare parts supply uncertainty. *Reliability Engineering & System Safety*, *264*, Article 111385. http://dx.doi.org/10.1016/j.ress.2025.111385, URL https://www.sciencedirect.com/science/article/pii/S0951832025005861.

Zio, E. (2009). Reliability engineering: Old problems and new challenges. *Reliability Engineering & System Safety*, *94*(2), 125–141. http://dx.doi.org/10.1016/j.ress.2008.06.002, URL https://www.sciencedirect.com/science/article/pii/S0951832008001749.

Zio, E., Fan, M., Zeng, Z., & Kang, R. (2019). Application of reliability technologies in civil aviation: Lessons learnt and perspectives. *Chinese Journal of Aeronautics*, *32*(1), 143–158. http://dx.doi.org/10.1016/J.CJA.2018.05.014.

Zuo, M. (2021). System reliability and system resilience. *Frontiers of Engineering Management*, *8*(4), 615–619. http://dx.doi.org/10.1007/s42524-021-0176-y.