Application of Reinforcement Learning for Condition-based Maintenance of Multi-Unit Systems

Mehrnaz Salmani

A Thesis

in

The Department

of

Concordia Institute for Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Quality Systems Engineering) at

Concordia University

Montréal, Québec, Canada

September 2023

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Mehrnaz Salmani**

Entitled: **Application of Reinforcement Learning for Condition-based Maintenance of Multi-Unit Systems**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Quality Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Mohsen Ghafouri

_____ External Examiner
Dr. Arash Mohammadi

_____ Examiner
Dr. Mohsen Ghafouri

_____ Supervisor
Dr. Farnoosh Naderkhani

Approved by _____
Chun Wang, Chair
Department of Concordia Institute for Information Systems Engineering

_____ 2023

_____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Application of Reinforcement Learning for Condition-based Maintenance of Multi-Unit Systems

Mehrnaz Salmani

Maintenance is a pivotal aspect of manufacturing systems, particularly those operating on a large scale. With the advent of data-driven methods and machine learning technologies, new avenues have opened for optimizing maintenance policies. In light of this, this thesis introduces advanced methodologies in Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) specifically tailored for large-scale parallel manufacturing systems. We conducted two major studies to advance the field: In the first study, an RL-based algorithm is proposed, moving beyond the traditional focus on system degradation levels to instead concentrate on the count of failed or unhealthy units. This shift allows for a more dynamic and nuanced approach to maintenance. Through Q-learning, our algorithm demonstrated significant superiority over conventional methods such as value iteration, particularly when applied to a system with four parallel units. In the second study, we delve into Deep Reinforcement Learning, developing a framework designed for multi-unit systems experiencing stochastic degradation and unforeseen failures. Unlike traditional methods, our DRL approach incorporates a more intricate reward function that considers a wide array of factors ranging from production costs to maintenance crew deployment. Notably, this study was rigorously tested on a system comprised of 30 parallel units, making it particularly relevant for real-world, large-scale applications. Our research significantly broadens the applicability of machine learning methodologies in maintenance scheduling, demonstrating both robustness and adaptability. These contributions not only validate the efficacy of data-driven approaches in real-world settings but also lay the groundwork for future research in this crucial domain.

Acknowledgments

First and foremost, I extend my deepest gratitude to my advisor, Dr. Farnoosh Naderkhani, for her unwavering guidance, invaluable counsel, and endless patience throughout this challenging academic endeavour. Her extensive knowledge and expertise have significantly enriched my graduate experience, and it has been an honour to work under her mentorship.

I would also like to thank my committee members, Dr. Mohammadi and Dr. Ghafouri, for their insightful comments and encouragement, which incentivized me to widen my research from various perspectives. Also, I extend my deepest appreciation to Dr. Fariba Azizi, Dr. Hasan Rasay and Dr. Abdollah Safari for their unwavering dedication, expertise, and invaluable contributions throughout our joint research endeavour."

I want to thank my family, who, despite the miles that separate us, have been a constant source of inspiration and encouragement, a gift for which simple words can hardly express my gratitude. Furthermore, I would like to express my appreciation to my friends and dear ones here in Montreal, who have become like a second family to me.

Last but certainly not least, I want to reference a quote from a person of wisdom:

"I wanna thank me,

I wanna thank me for believing in me,

I wanna thank me for doing all this hard work,

I wanna thank me for having no days off,

I wanna thank me for never quitting."

Contents

List of Figures

List of Tables

List of Abbreviations

AI Artificial Intelligence

ANN Artificial Neural Networks

CBM Condition-based Maintenance

CNN Convolutional Neural Networks

DDM Data-Driven Maintenance

DDQN Double Deep Q-Networks

DL Deep Learning

DQN Deep Q-Networks

DRL Deep Reinforcement Learning

DT Decision Tree

IoT Internet of Things

KNN K-Nearest Neighbours

LSTM Long Short-Term Memory

MBM Model-Based Maintenance

MDP Markov Decision Process

MDSS Maintenance Decision Support System

MDSS Maintenance Decision Support System

ML Machine Learning

PPO Proximal Policy Optimization

RL Reinforcement Learning

RNN Recurrent Neural Networks

RTF Run-to-Failure

RUL Remaining Useful Life

SPC Statistical Process Control

SVM Support Vector Machines

Chapter 1

Introduction

In the evolving landscape of modern manufacturing, the role of machine reliability and uptime is paramount. An era where industrial progress is synonymous with a nation's development demands robust machinery that can function seamlessly. The ripple effect of even a minor machine downtime can cascade into significant production setbacks, and in industries where margins are tight, such disruptions can have severe economic implications. Hence, the emphasis on maintenance and, more importantly, predictive and preventive maintenance has never been more profound (Rasay, Azizi, Salmani, & Naderkhani, 2023).

Traditionally seen as a necessary overhead, maintenance has now transformed into a strategic component of manufacturing operations. The goal is no longer just about fixing machines but ensuring they are available and efficient throughout their life cycle. High machine availability ensures that production lines remain active, meeting the demands of an ever-competitive market. As industries grow, so does the complexity and sophistication of machinery. The intricate designs and operations of these machines necessitate a more proactive approach to maintenance—one that can anticipate and prevent failures before they occur [(Duffuaa, Raouf, & Campbell, 1999), (Amari, McLaughlin, & Pham, 2006), (Ambani, Li, & Ni, 2009)].

This proactive strategy is embodied in the concept of Condition-Based Maintenance (CBM). Unlike routine maintenance, which relies on scheduled checks irrespective of machine conditions, CBM focuses on real-time data and machinery conditions(Tsang, Jardine, & Kolodny, 1999). CBM can signal when maintenance is required, optimizing machine availability and maintenance costs.

It's the precision of CBM that makes it the preferred choice for industries aiming for operational excellence [(Jardine, Lin, & Banjevic, 2006), (Panagiotidou & Tagaras, 2010)].

Historically, the transition to CBM was steered by statistical and mathematical models. Probabilistic models, reliability theories, and stochastic processes became the bedrock of CBM methodologies. Based on the historical and operational data of machinery, these models can provide insights into wear and tear, estimating the probability of failure over time. While effective, these methods often require extensive data and can sometimes be too rigid or generalized to adapt to the unique dynamics of each machinery. [(Barriento & Achcar, 2019), (Song, Zhang, Jiang, & Zhu, 2018), (Rasay, Naderkhani, & Azizi, 2022)].

With the advent of computational power and data science, the paradigm shifted towards Machine Learning (ML) approaches for CBM. With their ability to process vast datasets and detect complex patterns, ML algorithms brought a new level of precision to CBM. Decision trees, clustering algorithms, and regression models could predict failures with significant accuracy. Further, these algorithms constantly learn from new data, refining their predictions and making CBM more dynamic and adaptable. [(Cortes & Vapnik, 1995), (Widodo & Yang, 2007), (Bordoloi & Tiwari, 2014), (Banerjee & Abraham, 2018)].

As data became more abundant, especially with the Internet of Things (IoT) sensors, the dimensionality and complexity of the data increased. Deep Learning (DL), a subset of ML, harnesses neural networks to process this high-dimensional data. These networks, inspired by the human brain, can detect intricate patterns in large datasets, further refining the predictions and recommendations for maintenance. DL in CBM has opened doors to even more advanced diagnostics, predicting not just when but why a machine might fail [(Zhang & Si, 2020), (Hamer, Waterson, & Jun, 2021), (Álvarez et al., 2019)].

Emerging as one of the most dynamic approaches to CBM is Reinforcement Learning (RL), underpinned by concepts of Markov Decision Processes (MDPs) and further extended by Deep Q-Learning. RL's heart lies in interacting with environments, learning through feedback, and constant adaptation. MDPs, with their emphasis on states, actions, and rewards, offer the mathematical foundation on which RL thrives. Each decision epoch within an MDP provides a structured snapshot, allowing RL agents to determine optimal maintenance strategies by maximizing cumulative rewards

over time. Deep Q-Learning, a DL adaptation of Q-learning, takes this a notch higher, handling vast, complex datasets and honing maintenance decisions with unparalleled precision.[(Ahadi & Sullivan, 2019), (Paraschos, Koulinas, & Koulouriotis, 2020),(Y. Chen, Liu, & Xiahou, 2021),(Valet et al., 2022), (J. Chen & Wang, 2023)].

1.1 Thesis Motivation

In its myriad forms, power has always been the driving force of civilization. Today, its significance is more pronounced than ever, underpinning virtually every facet of our technologically driven lives. Power is a ubiquitous necessity, from lighting our homes to fueling industries and supporting digital infrastructures. As the global community becomes increasingly aware of environmental concerns, the focus is shifting towards cleaner and more sustainable energy solutions. This transition underscores the pivotal role of power grids, specifically wind turbines, in our quest for sustainable energy.

Wind turbines symbolize the vanguard of renewable energy sources, harmonizing technological innovation with environmental sustainability. These majestic structures dotting horizons worldwide aren't merely technological wonders; they are hefty investments, reflecting both their financial cost and the value they bring regarding clean energy. Given their complex designs and multi-faceted components, ensuring their optimal operation becomes paramount (Hart et al., 2020).

A closer inspection of a wind turbine reveals intricate systems working in tandem. Consider the gearbox shown in Fig.1.1(Feng, Qiu, Crabtree, Long, & Tavner, 2013), an integral component of these turbines. This crucial mechanism is a veritable orchestra of numerous components working cohesively to transmute kinetic wind energy into usable electrical power. The sheer complexity of the gearbox, combined with its substantial cost, makes its maintenance an area of prime concern. Neglecting or mismanaging the upkeep of such a crucial component can lead to catastrophic failures, bringing the turbine and the power supply to a grinding halt.

In the ever-increasing complexity of multi-component systems, particularly in critical infrastructures like power plants, transportation networks, and manufacturing facilities, the necessity for a refined, intelligent, and forward-thinking maintenance strategy is becoming unequivocally urgent.

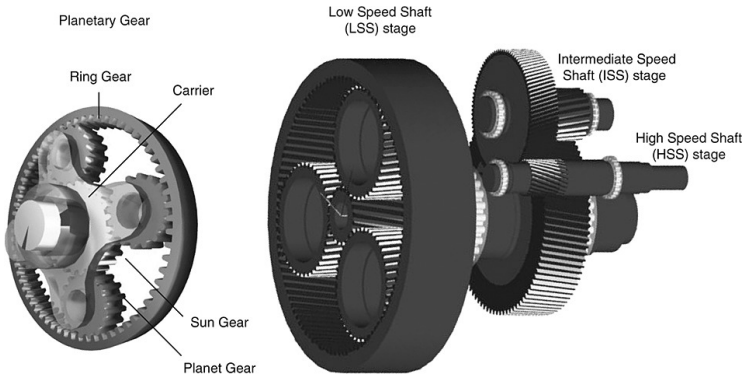

Figure 1.1: Typical gearbox of a wind turbine

Traditional reactive or scheduled maintenance methods are no longer sufficient to meet the stringent requirements for operational efficiency, safety, and longevity. This intensifies the call for sophisticated approaches that are capable of identifying the need for intervention and predictive in nature, allowing for the optimization of maintenance schedules and resources. In light of this imperative, the fields of Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) present themselves as groundbreaking avenues for innovation, as indicated by recent studies (Xiang, Yang, Hu, Su, & Wang, 2022).

RL and DRL are advanced machine learning paradigms that excel in decision-making tasks under uncertainty, learning from interaction with an environment to achieve a particular goal. These techniques can potentially dramatically transform how we approach multi-component system maintenance. They allow for the construction of adaptive and predictive maintenance algorithms that can anticipate potential points of failure and proactively take corrective measures. This isn't merely about identifying when a component is likely to fail but also involves optimizing the sequence and timing of maintenance activities to maximize the operational lifespan of the entire system.

By integrating the computational prowess of RL and DRL into maintenance management systems, it becomes conceivable to generate a mechanism that not only anticipates potential breakdowns but also systematically arranges repair and replacement schedules. This optimization leads to a notable extension in the operational lifespan of the equipment. When applied to systems that are critical for societal well-being, such as power grids or water treatment facilities, the benefits are manifold.

Firstly, such a strategy safeguards the substantial financial investments poured into this complex machinery. When components last longer and operate more efficiently, it results in decreased overheads related to parts replacement and system downtime. Secondly, it significantly enhances the reliability of these systems. A more reliable power grid, for example, ensures a consistent and uninterrupted supply of electricity, which is fundamental for both residential and industrial consumers. This has cascading benefits on productivity, quality of life, and even emergency services, which rely on a stable power supply for critical operations.

In the forthcoming sections, this research will journey into the heart of RL and DRL, exploring their potential to revolutionize maintenance decision-making processes, particularly for intricate multi-component systems. With its myriad components and undeniable importance in today's energy landscape, the wind turbine will serve as our guiding example throughout this exploration.

1.2 Thesis contributions

Maintenance is a necessary component of the manufacturing and industrial sectors that guarantees the machinery functions safely and benefits these sectors by decreasing machine downtime, extending the life of assets, and improving worker safety. As a result, the major purpose of my thesis is to present an algorithm that reduces long-term costs by optimizing the maintenance scheduling for parallel units inside a manufacturing system. The main contributions of my thesis research are summarized as follows.

(i) Dynamic Maintenance for a Large Scale Identical Parallel Manufacturing Systems Using Reinforcement Learning : In this study, an RL approach for dynamic maintenance of a multi-component parallel system subject to stochastic degradation and random failures has been proposed. In brief, the contributions of this paper are as follows:

- One of the significant contributions of this research is introducing an RL-based model geared towards CBM for multi-component systems susceptible to random failures. Instead of using traditional methods, this RL-based model allows for identifying optimal maintenance policies that adapt to the system's real-time conditions. This enhances the effectiveness of maintenance actions and enables the system to be more resilient against unforeseen failures.

- Another pivotal contribution of this thesis is shifting the focus from merely tracking system degradation levels to account for the number of failed or unhealthy units, turning the issue into a dynamic maintenance problem. This new approach allows for more adaptive, targeted maintenance interventions. It enriches the existing analytical landscape by accommodating the specific needs of individual units and opens the door for more advanced, adaptive optimization algorithms in maintenance scheduling.

(ii) Deep Reinforcement Learning for Dynamic Maintenance Policies of Large-Scale Parallel Manufacturing Systems: In this study, a Deep Reinforcement Learning algorithm (DRL) is proposed for a multi-unit system that provides dynamic maintenance scheduling recommendations. The Double Deep Q Networks (DDQN) algorithm is proposed to solve the problem, making the proposed RL solution more practical and effective in terms of time and cost savings than traditional MDP approaches. In summary, the contributions of this paper are as follows:

- A significant advancement in this research is implementing a DDQN model for maintenance scheduling in multi-component systems experiencing degradation. This DDQN model goes beyond traditional methods to identify more dynamic and effective maintenance policies that adapt according to the system's changing health.

- Another key contribution is introducing a Birth/Birth-Death process that continuously updates the health status of the system's units. This dynamic updating enables real-time monitoring and allows for the adaptation of the maintenance strategy based on the most current health information of the units, making the system more responsive and efficient. Observed allows for real-time monitoring and adaptation of the system maintenance strategy based on the current health information of the units.

- The research also innovatively considers both the system's positive and negative operational impacts. Unlike many studies focusing solely on minimizing costs or maximizing production, this work considers the dual aspects of excessive output and unsatisfied demand. This balanced perspective offers a more holistic approach to optimizing system operations.

- Lastly, the efficacy of the proposed algorithm was rigorously tested on a large-scale system

consisting of 30 parallel units. This represents a significant stride forward compared to much of the existing literature, which often limits its scope to simpler systems with only one or two units. The ability to manage and derive insights from such a large and complex system underscores the robustness and broader applicability of the methods proposed in this study.

1.3 Thesis organization

The rest of the thesis is organized as follows:

- Chapter 2 provides an exhaustive literature review concerning the central topic of the thesis. It surveys the most relevant research in the field and sets the stage for in-depth discussions and novel contributions that follow in subsequent chapters.
- Chapter 3 introduces an RL algorithm specifically designed for the dynamic maintenance of identical parallel units. This marks a noteworthy effort to extend the utility of RL methods in maintenance optimization for parallel systems.
- Chapter 4 puts forth a DRL algorithm that employs Deep Q Networks (DQN) to tackle dynamic maintenance issues. This approach symbolizes a more complex and adaptive methodology for maintenance optimization, allowing for more nuanced solutions to common issues.
- Chapter 5.2 serves as the concluding segment of the thesis. In addition to summarizing key findings and contributions, it delves into prospective future research avenues, thereby highlighting the potential for further advancements in this growing field of study.

Chapter 2

Literature Review

Complex manufacturing systems must function at maximum capacity in today's highly interconnected and competitive global market. With rising competition across various industries and manufacturing sectors, industrial researchers, engineers, and production managers are actively working on creating and implementing promising and inventive technologies to enhance the reliability of their systems. Proper maintenance actions play a crucial role in improving the performance and reliability of the systems. Leveraging advanced maintenance technologies such as condition monitoring, predictive analytics, and remote diagnostics can provide real-time insights into the system's health.

CBM, as the state-of-the-art maintenance program, helps improve the reliability of the systems by proactively monitoring the real-time condition of systems and taking timely maintenance actions. By detecting potential issues early on, CBM enhances system reliability, minimizes downtime, and reduces the risk of unexpected and costly failures. CBM outperforms traditional maintenance models such as age-based, block replacement, and Run-to-Failure (RTF) models (de Jonge, Teunter, & Tinga, 2017). Generally speaking, traditional maintenance models may be performed either too regularly, which would result in excessive costs and downtime, or insufficiently frequently, which would increase the chance of unexpected breakdowns. By implementing CBM, the average cost is considerably lowered by eliminating unnecessary maintenance operations (Salmani, Azizi, Rasay, & Naderkhani, 2023).

Nowadays, due to technological advancements and the accessibility of CM data, CBM approaches are being supplemented or substituted by advanced data-driven CBM methods that offer real-time information on equipment conditions, facilitating precise and efficient maintenance interventions through the application of ML and Artificial Intelligence (AI) algorithms. ML algorithms and AI techniques analyze large volumes of equipment data and identify patterns, anomalies, or potential failures. These methods can improve fault detection accuracy, enable predictive maintenance, and optimize maintenance strategies based on equipment performance and historical data. Below, the first CBM program is reviewed in Section 2.1. Then, the application of ML in CBM is reviewed in Section 2.2.

2.1 Condition-based Maintenance

CBM has emerged as a powerful approach in modern industrial practices to optimize maintenance activities, improve equipment reliability, and reduce costs. Unlike traditional maintenance methods that rely on fixed schedules or reactive approaches, CBM utilizes real-time data, advanced sensors, and predictive analytics to monitor equipment conditions and determine maintenance needs. By focusing on the actual health and performance of equipment, CBM enables proactive interventions, allowing organizations to address potential issues before they escalate into costly failures (Azar, Hajiakhondi-Meybodi, & Naderkhani, 2022). By integrating technology and data-driven insights, CBM has transformed maintenance practices, empowering organizations to achieve higher reliability, productivity, and cost-effectiveness in today's competitive business landscape.

The CBM program encompasses three primary stages, illustrated in Figure 2.1: (i) Data acquisition, (ii) Data processing, and (iii) Decision making. It should be emphasized that these steps are iterative and continuous. CBM is an ongoing process that entails constant monitoring of the asset's condition, analyzing data, making maintenance decisions, and implementing appropriate actions based on the current and projected health of the asset.

Research within the context of CBM has been growing for many years. References such as (Duffuaa et al., 1999), (Tsang et al., 1999) and (Jardine et al., 2006) conducted an early study on CBM, which aimed to develop a model for implementing CBM in industrial settings. Following

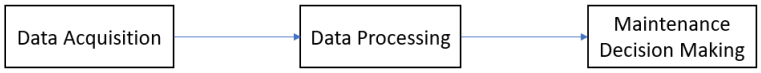

Figure 2.1: Condition-based monitoring steps.

this initial study, the field of CBM has experienced remarkable growth in research, focusing on advancing theoretical foundations and practical applications. During the early 2000s, mathematical models were the primary solution for addressing CBM problems, as demonstrated by the following research studies.

In subsequent years, researchers consistently advocated for various statistical approaches in CBM problem-solving. An outstanding research contribution (Panagiotidou & Tagaras, 2010) draws attention to the potential of an integrated approach that combines Statistical Process Control (SPC) and CBM to optimize maintenance decision-making and improve equipment reliability and availability. The authors stress the importance of data sharing between the two methods and the significance of accurate and reliable data to ensure the effectiveness of this integrated approach.

Building upon this progress, the increasing reliance on wind energy underscores the necessity for effective condition-based monitoring systems, especially since wind turbines are frequently positioned in remote locations, making their maintenance more challenging. In a notable paper, (Tautz-Weinert & Watson, 2017) presented an extensive study on using SCADA data for condition monitoring of operational wind turbines. They emphasized that SCADA systems, when used appropriately, can be instrumental in early fault detection, reducing the cost implications of major repairs or component failures.

Given the progress in industrial systems, the integration of sensor devices for comprehensive CM data collection, and the abundance of vast and diverse datasets, there is a notable shift towards leveraging ML techniques to enhance predictive analytic and decision-making processes in the CBM program. The subsequent section will provide an in-depth exploration of the utilization of ML in the context of CBM.

2.2 Overview of Machine Learning and Reinforcement Learning Approaches in CBM

Researchers have increasingly shifted their focus towards ML-based algorithms, driven by their potential to address complex system challenges effectively. These algorithms can dissect intricate patterns and relationships within extensive datasets, enabling more precise predictions and informed decision-making in demanding contexts. Consequently, ML-based solutions have gained substantial prominence across diverse fields, from healthcare and finance to industrial applications such as CBM and fault diagnosis (Ali & Abdelhadi, 2022).

ML's capacity to process and analyze vast amounts of data at unprecedented speeds has revolutionized decision-making in Industry 4.0, empowering businesses to make real-time adjustments and strategic choices based on actionable insights (Nguyen, Do, Vosin, & Iung, 2022). As industries increasingly embrace ML-powered solutions, they are staying competitive and laying the foundation for a more agile, efficient, and responsive future.

A comprehensive breakdown of the machine learning process and its intricate steps is presented in Figure 2.2 (Fausing Olesen & Shaker, 2020). It visually represents the fundamental stages within the ML process. It begins with Data Acquisition, where sensor installation, database integration, and meticulous experiment setup establish the crucial foundation by ensuring the availability of high-quality data. Subsequently, Data Preprocessing plays a pivotal role in refining the collected data, addressing issues such as noise reduction, handling missing data, and conducting feature engineering to extract valuable insights.

The workflow advances to Model Selection and Training, where a suitable ML model, including classical techniques like Artificial Neural Networks (ANN), Support Vector Machines (SVM), K-Nearest Neighbours (KNN), and Decision Trees (DT), is chosen and trained on the prepared dataset. The model's generalization abilities and performance on unseen data are scrutinized to ensure its predictive accuracy. Moving forward, Parameter Tuning fine-tunes the model's parameters to optimize its performance, ultimately enhancing its predictive capabilities. Finally, the process

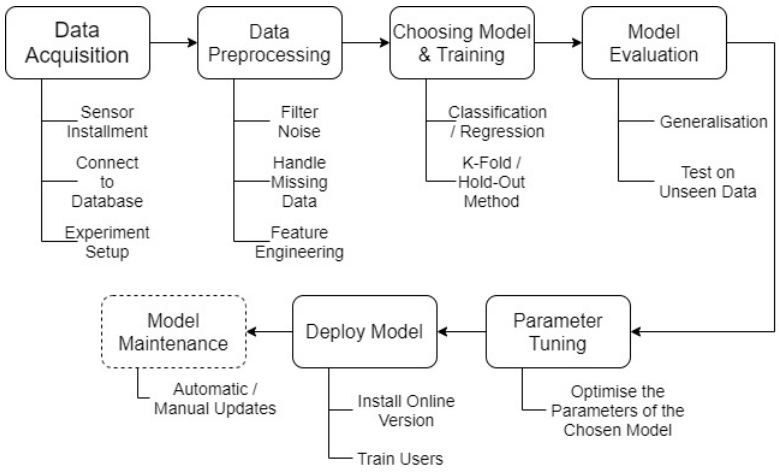

Figure 2.2: Steps of Machine Learning

culminates in Deploying the Model, which involves installing an online version for real-time predictions and user training.

In the context of AI-enhanced maintenance strategies, classical ML techniques have proven effective. At the same time, the broader realm of ML, including DL, offers a subset built upon artificial neural networks. Techniques such as Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), DRL, and autoencoders are gaining prominence as essential tools for maintaining diverse equipment and systems. This end-to-end approach ensures that ML solutions are effectively implemented, resulting in actionable insights and valuable outcomes (R. Zhao et al., 2019).

RL, another facet of ML, involves an agent learning optimal decision-making through interactions with an environment and receiving feedback in the form of rewards or penalties. RL is particularly advantageous in scenarios requiring the acquisition of optimal behaviour through iterative trial and error. These domains within AI are reshaping numerous industries by ushering in advanced automation, predictive analytics, and intelligent decision-making systems (Nguyen et al., 2022).

For example, Widodo et al. (Widodo & Yang, 2007) conducted pioneering research on utilizing ML algorithms in CBM. Their study explored the application of these algorithms to enhance the effectiveness of CBM strategies, marking an important milestone in integrating ML techniques

into CBM practices. The paper presents a methodology using SVMs to classify machine conditions based on vibration data. Experimental results on a gearbox dataset demonstrate the effectiveness of SVMs in accurately categorizing gearbox conditions and detecting faults. The authors discuss the advantages and limitations of SVMs in machine condition monitoring and fault diagnosis. They conclude that SVMs can effectively improve the reliability and availability of machines by enhancing machine condition monitoring and fault diagnosis. The groundbreaking work by Widodo et al. in utilizing ML algorithms for CBM sparked significant interest among researchers in the field. Subsequently, several studies were conducted, including (Banerjee & Abraham, 2018), (Gryllias & Antoniadis, 2012), and (Bordoloi & Tiwari, 2014). These researchers further explored and expanded upon the application of ML techniques in CBM, contributing to the growing body of knowledge in the field.

Researchers later began employing more sophisticated DL algorithms, such as ANN and Long Short-Term Memory (LSTM) (H. Chen, Liu, Chu, Liu, & Xue, 2021), to address CBM problems. These advanced algorithms allowed for more intricate analysis and modelling of complex systems, enabling improved accuracy in condition monitoring, fault detection, and maintenance decision-making within CBM. Additionally, integrating transfer learning techniques and ensemble models further enhanced the robustness and generalization of DL-based CBM approaches, making them more suitable for real-world industrial applications.

Coraddu et al.(Coraddu et al., 2016) proposed a methodology that utilizes ML algorithms, particularly ANNs, to predict the remaining useful life of naval propulsion plants. Through experiments conducted on a dataset specific to naval propulsion plants, the authors demonstrated the effectiveness of their approach. The results indicated that ANNs can accurately forecast the remaining useful life of these plants, thereby enhancing the efficacy of CBM strategies. After that, Wei et al.(Wei, Zhao, He, & He, 2019) employ a Markov process to simulate the deterioration of the system and account for the impact of zoned shock events on its condition. Additionally, they introduce a maintenance optimization model that incorporates the system's condition and the associated maintenance costs. This model aims to identify the most effective maintenance actions for achieving optimal system performance.

Having explored the machine learning techniques in CBM, it is natural to progress to an even

more dynamic area of focus: RL. While ML provides the foundation for predictive insights, RL has the potential to leverage these insights for decision-making in CBM tasks. The following section will explore how RL fits into the CBM landscape and understand its real-world applications.

2.2.1 Overview of Reinforcement Learning Algorithms

In recent years, the synergy between RL and CBM has emerged as a prolific area of research. RL, a branch of machine learning, has emerged as a promising tool for optimizing CBM tasks. Traditional RL involves an agent that learns by interacting with its environment to maximize cumulative rewards. Popular RL algorithms include Q-learning, Deep Q Networks (DQN), and Proximal Policy Optimization (PPO), among others (Sutton & Barto, 2018).RL has been used to derive policies that balance the continued operation of deteriorating systems and preventive maintenance actions. Such policies are crucial for systems where early maintenance can be costlier than operating a slightly degraded system(Wang, 2002).

The scholarly work by (Marugán, 2023)provides a literature review on the application of RL algorithms for the maintenance of engineering systems, particularly in the context of CBM. The author reviewed the application of RL in many different fields, such as transportation systems, manufacturing and production systems, civil infrastructures, etc. Consequently, these insights frequently galvanize researchers, spurring subsequent investigations and studies to address the identified challenges and expand the domain's boundaries.

In another notable contribution, (Álvarez et al., 2019) presents a maintenance model that incorporates CBM strategies while considering resource constraints on the number of inspections. They offered a stochastic dynamic programming model to determine the most opportune moment to inspect and execute preventive maintenance over each component of a non-redundant system. Later, Smith and Doe (Hamer et al., 2021) presented a framework that utilizes RL to deduce optimal CBM decision policies. Their work showcased that RL can be effectively harnessed to adapt to varying system conditions and dynamically choose between repair, replacement, or continued operation based on real-time data.

Several researchers have adopted a more sophisticated approach to CBM issues, aiming to address more intricate systems. This has led to the utilization of DL algorithms. Zhang et al. (Zhang

& Si, 2020) proposed a deep RL-based CBM to overcome high- and low-dimensional problems. Both stochastic and economic dependencies are taken into consideration. They mapped the system degradation directly to the maintenance decision without having a maintenance threshold.

While the previous section shed light on the foundational applications of RL in CBM, modern complex systems often consist of multiple interdependent components, demanding a more intricate approach. As systems grow in complexity, so does the need for sophisticated strategies to navigate the nuanced interactions between these components. In the subsequent section, we will explore how RL and its advanced counterpart, DRL, are being harnessed to address the unique challenges and optimize maintenance tasks in multi-component systems.

2.3 Dynamic Maintenance of Identical Manufacturing Systems

In this section, we delve into the dynamic maintenance of identical manufacturing systems, where CBM plays a pivotal role in ensuring operational efficiency and reliability. Identical manufacturing systems pose unique challenges in terms of maintenance, as uniformity can sometimes lead to unexpected failures.

CBM, as depicted in Figure 2.3, starts with data collection using sensors to monitor equipment conditions. After collecting data, the process moves to diagnosis for fault detection and identifying operational anomalies. The next step is prognosis, which estimates the equipment's Remaining Useful Life (RUL). This predictive aspect provides insights into the machinery's expected life. The process concludes with the Maintenance Decision Support System (MDSS), which consolidates the gathered information to recommend optimal maintenance actions based on current conditions and future predictions.

While CBM encompasses a broad spectrum of activities, this thesis narrows its focus on the maintenance decision-making processes for multi-component systems. The potential of reinforcement learning agents is thoroughly explored, examining their aptitude in navigating complex maintenance decisions.

As we journey deeper into maintenance, it becomes evident that Condition-Based Maintenance is not just a methodology; it's the future. The forthcoming section will focus on the literature

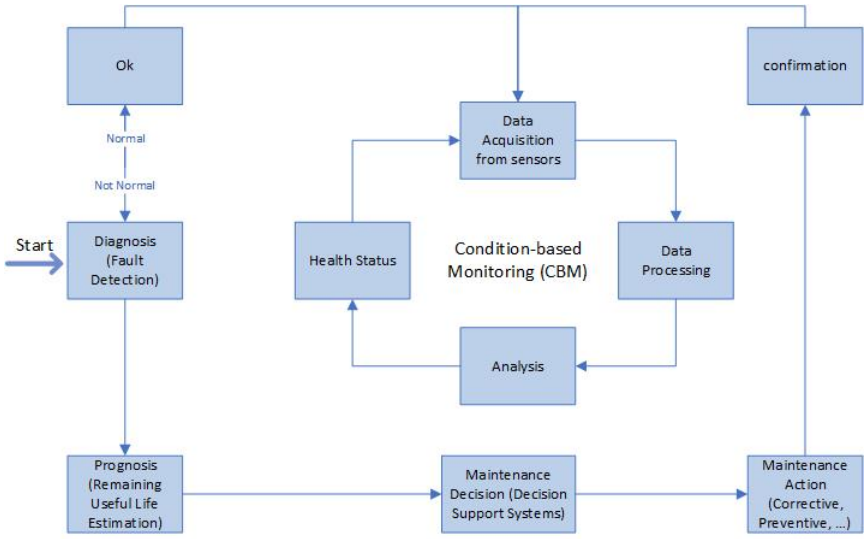

Figure 2.3: Condition-based Monitoring flowchart.

surrounding RL and DRL, elucidating their roles in the dynamic maintenance of multi-component systems, which is the main focus of this thesis.

2.3.1 Application of Reinforcement Learning in Maintenance Scheduling

This section explores how RL is applied to maintenance scheduling, a critical aspect of industrial asset management. RL's adaptive capabilities are leveraged to optimize schedules and minimize downtime, offering intelligent, data-driven solutions that enhance equipment lifespan and improve system reliability and sustainability.

There are various maintenance methodologies for fault diagnosis and prognosis, but the two common approaches are model-based maintenance and data-driven maintenance. Model-Based Maintenance (MBM) is a maintenance strategy that uses a mathematical model of a system to diagnose and predict faults. Several research studies have investigated MBM in many sectors.

MBM has been used in the energy sector to monitor wind turbines and predict component breakdowns. The reference (Barriento & Achcar, 2019) tried to find an optimal maintenance strategy using statistical methods like ANOVA. The authors defined a fixed threshold for doing maintenance actions, which is not accurate and precise. MBM is used to forecast gearbox failures in wind turbines, which lowers maintenance costs and downtime, according to research by (Song et al., 2018).

Also, (Rasay et al., 2022) modelled a process with interrelated stages. They formulated the problem based on renewal theory and considered both process and equipment conditions. Based on the concept of opportunistic maintenance, they developed a novel integrated SPC and maintenance planning framework. There are various restrictions with MBM, one of which is that it requires accurate models of the system under observation. It may be time-consuming and costly to develop these models. Moreover, systems with unpredictable behaviour or operating conditions that change quickly may not perform well with MBM.

On the flip side, Data-Driven Maintenance (DDM) is an approach that uses historical data to detect and diagnose faults in a system. Unlike MBM, the system's physical characteristics are not necessary for DDM. DDM analyses data and looks for patterns that point to the beginning of defects using a variety of statistical and ML approaches. As statistical models may not be able to accurately predict future failures or maintenance needs, particularly in complex systems with many interacting components, the main focus of researchers is on developing advanced ML and AI-based solutions for maintenance management problems. We will now conduct a literature review focused explicitly on DDM models.

Traditional approaches to maintenance scheduling rely on pre-determined schedules based on time or usage. Still, these methods can lead to inefficiencies if maintenance is performed too early or too late. For example, (Hu, Miao, Zhang, Liu, & Pan, 2021) took spare component storage cost into account and proposed a novel RL method that can handle numerous maintenance strategies without prior knowledge. They simulated different maintenance scenarios to generate enough training data for an RL-driven strategy. Even in more recent research, several types of research focus on simple systems. However, the increased complexity of modern machines brings new challenges for modelling and analyzing their failure behaviours.

Numerous studies have historically centred on simplistic systems, like single-unit systems, with early examples from researchers such as (Paraschos et al., 2020). This trend persists in contemporary works, as demonstrated by recent research like (Xiao, Yan, Kou, & Wu, 2021), where a single-unit system is analyzed, and a mathematical model is presented to minimize the anticipated total expense. They only assumed two failure modes for their supposed system and considered production loss while it was under maintenance. Also, they present different inspection policies,

including periodic inspection policies and condition-based inspection policies. The effectiveness of their algorithm is under the various simplicity assumptions that they assumed.

Similarly, (X. Zhao & Wang, 2022) considered two identical components that operate in parallel and degrade over time with a bivariate Wiener process. The maintenance model is presented using the MDP for both the finite and infinite planning horizons. The authors also consider the impact of system performance, such as system availability, on maintenance decisions. Notably, their algorithm seemed effective on systems with two identical components. Unlike this paper, motivated by the challenges of solving high-dimensional MDPs, (Ahadi & Sullivan, 2019) introduced an approximate dynamic programming algorithm in their proposal. The algorithm addresses the selective maintenance problem in a series-parallel system that consists of only binary-state components.

Furthermore, it is worth noting that previous research studies have primarily focused on single-unit systems, employing highly effective algorithms such as DRL. In contrast, these algorithms have demonstrated their capability to handle more intricate and complex systems compared to traditional approaches.

In conclusion, this section has explored the application of RL in maintenance scheduling, highlighting its potential to optimize schedules, minimize downtime, and enhance equipment lifespan. While RL is a cutting-edge approach, we've also touched upon the broader landscape of maintenance methodologies, from MBM to DDM. Contemporary research predominantly focuses on single-unit systems, yet there is a growing interest in addressing the complexities of more intricate systems with advanced algorithms, such as DRL. As we transition to the next chapter, we delve further into DRL's exciting prospects in the maintenance scheduling context, examining how this advanced technique is shaping the future of industrial asset management.

2.3.2 Application of Deep Reinforcement Learning in Maintenance Scheduling

As industries march forward in the automation and data-driven decision-making age, DRL has carved a significant niche, especially in complex optimization problems. Maintenance scheduling, a critical component in ensuring the smooth operation of machinery and systems, is undergoing a paradigm shift with the infusion of DRL. With the capability to model and learn from intricate system interactions, DRL presents a leap in our ability to schedule maintenance activities in a manner

that's both efficient and responsive to real-time changes. This new horizon holds the potential to drastically reduce downtimes, enhance system longevity, and ultimately elevate the entire operational workflow.

Speaking of DRL, (Valet et al., 2022) proposed a new method for maintenance scheduling in complex production systems using DRL. Also, they test their result with different traditional methods as a benchmark. (Y. Chen et al., 2021) propose a framework that combines DRL with the dynamic loading strategy for repairable multi-state systems. With a continuous action space and a mixed integer discrete continuous state space, they formulated their problem as an MDP. The deep deterministic policy gradient algorithm is used to solve their problem. Similarly, (Yousefi, Tsianikas, & Coit, 2022) defined every state as the exact level of degradation and proposed a dynamic maintenance model. The maintenance team uses a neural network as a decision-making tool to determine the optimum maintenance course of action depending on the system's current level of degradation. Researchers have attempted to address even more complex problems by introducing additional constraints into their research, which can help guide their investigations and narrow down potential solutions. These constraints may be inherent to the problem itself or intentionally imposed by researchers to explore specific aspects of the problem, such as time constraints, resource constraints, or other limitations.

More similar to the present research, (Zhang & Si, 2020) proposed a novel approach that utilizes DRL to optimize imperfect maintenance in multi-component systems with load sharing. Load sharing refers to the ability of operational components in a system to distribute or bear the load of failed components, and this concept is integrated into the decision-making process. The authors assumed the system's health states could be restored by implementing imperfect maintenance and corrective maintenance actions. The authors' findings indicate that the proposed algorithm performs well when dealing with large-scale problems.

Several researchers have directed their attention toward investigating various dependencies in maintenance planning. (J. Chen & Wang, 2023) focused on multi-component systems and examined three types of dependencies: stochastic dependency, economic dependency, and structural dependency. They employed the compound Poisson process to model the degradation process of components. Furthermore, they used a fixed threshold to determine the failure level of the components.

Their study primarily considered a limited number of cost components, such as the cost of replacement and inspection. To evaluate the effectiveness of maintenance actions, they compared the actual maintenance costs incurred in four different system states with the Q-values calculated by the agent based on the identified state.

Additionally, (Nguyen, Do, Voisin, et al., n.d.) delved into maintenance problems by examining multi-state components influenced by state interactions and economic independence. State interactions refer to the effect of one component's degradation on others. The researchers considered two types of state observations: partially observed and fully observed, resulting in a substantial policy space. To address these complexities, they introduced a weighted QMIX algorithm, a multi-agent algorithm designed to handle state intersections and optimize decision-making in such scenarios.

In conclusion, the literature review has provided a comprehensive overview of the existing research on the topic, highlighting the complexities and nuances of the problem. Through thoroughly examining the relevant literature, researchers have identified gaps, limitations, and opportunities for further exploration. The review has also highlighted how some researchers have attempted to tackle these complex problems by incorporating additional constraints into their research. The insights gained from the literature review will serve as a solid foundation for the current study, laying the groundwork for the investigation to build upon and contribute to the existing body of knowledge.

Chapter 3

Reinforcement Learning-Based Dynamic Maintenance Model for Large-Scale Identical Parallel Manufacturing Systems

This chapter presents a novel and innovative machine learning-based framework tailored for the intricate realm of maintenance decision-making, with a particular emphasis on multi-unit systems. In the context of this framework, we introduce an RL approach designed to address the dynamic maintenance challenges faced by multi-component parallel systems that contend with stochastic degradation and random failures.

In this complex landscape, each unit's condition deteriorates independently, driven by a three-state homogeneous Markov process. These states encompass the health of the unit, its unhealthy state, and the unfortunate event of failure. The dynamics of the system, marked by the intricate interplay among individual component states, are elegantly captured using the Birth/Birth-Death process. This modelling strategy enables a comprehensive understanding of how the overall system state emerges from various combinations of individual component states, paving the way for effective maintenance strategies.

Crucially, we delve into the heart of our approach—formulating the optimal maintenance policy as a MDP. This MDP framework encapsulates the essence of dynamic maintenance decision-making, with the goal of cost minimization as the guiding principle. To navigate the complexities of this framework and derive actionable insights, we employ a Q-learning algorithm—a well-established and powerful RL technique. This choice proves instrumental in streamlining the decision-making process and delivering substantial time and cost savings compared to traditional MDP approaches.

One key distinction that sets RL apart in this context is its remarkable efficiency, mainly when dealing with large state-spaces. In this scenario, traditional MDP solutions often struggle to converge to the optimal policy within reasonable timeframes. In the face of such challenges, RL emerges as a decisive need in the arsenal of maintenance decision-makers, providing practical and effective solutions.

To bring these concepts to life, we present a compelling numerical example that vividly illustrates how RL can be harnessed to derive the optimal maintenance policy for the multi-component parallel system under scrutiny. Through this example, we showcase the feasibility and practicality of our RL-based approach, shedding light on its potential to revolutionize maintenance decision-making for complex, multi-unit systems in various industrial domains.

The chapter is structured in the following manner: Section 3.1 lays the groundwork for the problem, offering a comprehensive overview of RL principles and the birth/birth-death process. In Section 3.2, a numerical experiment is presented, testing the algorithm introduced in the prior section. Section 3.3 concludes the chapter and delves into potential prospects.

3.1 Problem Statement

Consider a production system consisting of M identical, parallel units. Deterioration of each unit occurs independently according to a three-state homogenous Markov process such that each unit has three states: healthy, unhealthy, and failure, denoted as states 0, 1, and 2, respectively. Healthy and unhealthy states are operational, and failure state is not operational. At the start of the planning horizon, all units are in a healthy state and gradually deteriorate. The system state is

defined using two interacting populations: (i) The number of units in the unhealthy state and (ii) The number of units in the failure state. More specifically, at a specific time as t, the state of the system is defined as $s_t = (i, j)$ where i is the number of units in the failure state, and j is the number of units in the unhealthy state such that $0 \leq i + j \leq M$, $i, j \geq 0$. It means that the number of units in the healthy state is $M - i - j$. Accordingly, the total number of system states is equal to $(M + 2)(M + 1)/2$. The Birth/Birth-Death stochastic process describes the interaction among the system states as described in the next section. Each system unit is monitored continuously so that updated information about the deterioration of each unit is available.

3.1.1 Birth/Birth-Death Process

Let $N_0(t)$, $N_1(t)$, and $N_2(t)$ be the number of units in state 0, the number of units in state 1, and the number of units in state two at time t, respectively. We take into the account a bivariate homogeneous Markov process $\{N_2(t), N_1(t)), t \geq 0\}$ with the state space $S = \{(i, j) : 0 \leq i, j \leq M, 0 \leq i + j \leq M\}$, to trace $N_2(t)$ and $N_1(t)$, because $N_0(t) + N_1(t) + N_2(t)$ remains constant as noted. For this purpose, we will apply a birth/birth-death process, a subclass of competition processes with two interacting populations of failure and unhealthy units, $(N_2(t), N_1(t))$, whose first population is rising $N_2(t)$. All possible transitions occur with the following probabilities:

$$P((N_2(t + dt), N_1(t + dt))(i, j + 1)|(N_2(t), N_1(t))(i, j)) = \lambda_{ij}^{(1)} dt + o(dt) \tag{1}$$

$$P((N_2(t + dt), N_1(t + dt))(i, j + 1)|(N_2(t), N_1(t))(i, j)) = \lambda_{ij}^{(2)} dt + o(dt) \tag{2}$$

$$P((N_2(t + dt), N_1(t + dt))(i, j + 1)|(N_2(t), N_1(t))(i, j)) = \gamma_{ij} dt + o(dt) \tag{3}$$

$$P((N_2(t+dt), N_1(t+dt))(i, j+1)|(N_2(t), N_1(t))(i, j)) = 1-(\lambda_{ij}^{(1)} + \lambda_{ij}^{(2)} + \gamma_{ij})dt+o(dt) \tag{4}$$

This expression denotes the likelihood that the system will move from the state (i, j) to state $(i, j + 1)$ within an infinitesimally brief interval, dt. The term $\lambda_{ij}^{(1)} = (M - i - j)p_{01}\lambda_0$ estimates the probability of transitioning units from a healthy condition to state 1, indicating a birth in the population of unhealthy units. Similarly, $\lambda_{ij}^{(2)} = (M - i - j)p_{02}\lambda_0$ gives the chance of moving healthy units to state 2, pointing to a birth in the population failed units. Meanwhile, $\gamma_{ij} = jp_{12}\lambda_1$

establishes the likelihood of changing units from an unhealthy state to state 2. This represents a simultaneous event where the number of unhealthy units decreases (akin to a "death" in this group), and there is a birth in the failed units. given i units in state 2 and j units in state 1.

The leaving rates of the states can be obtained by:

$$V_{ij} = \begin{cases} \lambda_{ij}^{(1)} + \lambda_{ij}^{(2)} + \gamma_{ij} = (M-i-j)\lambda_0 j\lambda_1 & 0 \le i \le M-1, 1 \le j \le M-i-1, \\ \lambda_{ij}^{(1)} + \lambda_{ij}^{(2)} = (M-i)\lambda_0 & 0 \le i \le M-1, j = 0 \\ \gamma_{ij} = j\lambda_1 & 0 \le i \le M-1, j = M-i \\ 0 & i = M, j = 0 \end{cases} \tag{5}$$

The state of the continuous-time Markov chain $\{(N_2(t), N_1(t))\}$ just after a state transition is described by the discrete-time Markov chain $\{(N_{2,n}, N_{1,n})\}$ whose one-step transition probabilities $p_{(i,j),(i',j')}$ at the end of sojourn times are derived as follows:

- When $0 \le i \le M-1$ and $1 \le j \le M-i-1$, the one-step transition probabilities are given by:

$$p_{(i,j),(i',j')} = \begin{cases} \frac{\lambda_{ij}^{(1)}}{\lambda_{ij}^{(1)}+\lambda_{ij}^{(2)}+\gamma_{ij}} & i' = i, j' = j+1 \\ \frac{\lambda_{ij}^{(2)}}{\lambda_{ij}^{(1)}+\lambda_{ij}^{(2)}+\gamma_{ij}} & i' = i+1, j' = j \\ \frac{\gamma_{ij}}{\lambda_{ij}^{(1)}+\lambda_{ij}^{(2)}+\gamma_{ij}} & i' = i+1, j' = j+1 \end{cases} \tag{6}$$

- When $0 \le i \le M-1$ and $j = 0$, the one-step transition probabilities are given by:

$$p_{(i,j),(i',j')} = \begin{cases} \frac{\lambda_{ij}^{(1)}}{\lambda_{ij}^{(1)}+\lambda_{ij}^{(2)}} & i' = i, j' = j+1 \\ \frac{\lambda_{ij}^{(2)}}{\lambda_{ij}^{(1)}+\lambda_{ij}^{(2)}} & i' = i+1, j' = j \end{cases} \tag{7}$$

- When $0 \le i \le M-1$ and $j = M-i$, a transition is possible to the state with $i' = i+1$ and $j' = j-1$ with probability 1:

$$p_{(i,j),(i',j')} = 1 \tag{8}$$

Uniformization method can obtain the probabilities $p_{(i,j),(i',j')}$ via:

$$p_{(i,j),(i',j')}(t) = \sum_{n=0}^{\infty} e^{-vt} \frac{(vt)^n}{n!} p^{-(n)}_{(i,j),(i',j')} \tag{9}$$

where the probabilities $p^{-(n)}_{(i,j),(i',j')}$ can be recursively computed from:

$$\overline{p}^{-(n)}_{(i,j),(i',j')} = \sum_{(k,h)\in S} \overline{p}^{-(n-1)}_{(i,j),(k,h)} \overline{p}_{(k,h),(i',j')}, \qquad n = 1, 2, ... \tag{10}$$

starting with $p^{-(0)}_{(i,j),(i,j)} = 1$ and $p^{-(0)}_{(i,j),(i',j')} = 0$ for $(i,j) \neq (i',j')$. This completes the derivation of the transition probabilities. In the next section, we provide a detailed description of the proposed RL algorithm.

3.1.2 Review of RL and Q-learning

Generally speaking, ML models can be classified into three major categories: (i) Supervised learning, (ii) Unsupervised learning, and (iii) Reinforcement learning. RL is a branch of ML where an agent learns how to behave in an environment by performing specific actions and receiving rewards or penalties in return. Unlike supervised learning, where the correct decisions are explicitly provided, RL is about learning from trial-and-error experience. An agent makes observations, takes actions based on those observations, and receives feedback through rewards or penalties. The objective is to find a policy that maximizes the agent's expected cumulative reward over time.

At its core, RL operates on a simple principle: learn by interacting with an environment. It mirrors how humans and animals learn through trial and error, refining their decisions based on consequences. In RL, decisions aren't guided by explicit labels or by being shown examples. Instead, they're driven by feedback from the environment in the form of rewards or penalties.

RL is inherently temporal, unlike many ML methods that often assume independence between samples. Decisions made now can have consequences far into the future, and understanding these long-term effects is crucial for effective learning. This is captured in the return or cumulative future reward, which an agent seeks to maximize.

3.1.2.1 Reinforcement Learning Components

RL stands distinctively in the vast landscape of ML, characterized by its interactive learning paradigm. While its foundational principles, which revolve around learning through interaction and feedback, echo throughout various learning theories, the magic of RL truly materializes through the interplay of its core components (Figure 3.1). Each component in the RL framework serves as a pillar, holding up the tower of this dynamic form of learning. Just as understanding the intricate workings of a watch requires a detailed examination of each cog, spring, and hand, a comprehensive grasp of RL mandates a deep dive into its constituent parts. This section endeavours to elucidate these components, shedding light on their roles, nuances, and collective synergy that powers the RL mechanism.

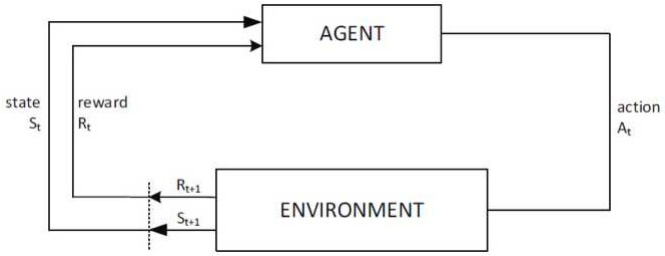

Figure 3.1: Foundation of RL

- **Agent**: At the forefront of the RL paradigm is the agent, an autonomous entity that learns and makes decisions. Think of the agent as the learner or decision-maker, constantly interacting with its environment, adapting its strategies based on feedback, and evolving its understanding to achieve its objectives. The agent embodies algorithms that process its experience to improve its policy over time.

- **Environment**: The world through the eyes of the agent is its environment. It's a dynamic entity that reacts to the agent's actions, providing new situations (states) and feedback (rewards). The environment encapsulates all the challenges and nuances the agent must understand and navigate to succeed in its learning journey.

- **Action** (A): The agent's means to influence its environment are the actions. Actions can

span a vast spectrum, from simple discrete decisions like "move left" or "jump" to complex continuous actions like controlling the speed and direction of a car. The collection of all possible actions is known as the action space, which can vary in complexity depending on the problem.

- **State (S)**: If actions are the means, states are the context. States define the current situation or scenario in which the agent finds itself. They capture vital information, giving the agent a snapshot of the environment at a particular moment. The state space, a collection of all potential states, provides the backdrop against which the agent evaluates its decisions.
- **Reward (R)**: Serving as the feedback mechanism, rewards are immediate signals given to the agent after taking an action in a particular state. They guide the learning process, incentivizing beneficial actions and discouraging unfavourable ones. The reward structure is crucial, as it shapes the agent's learning trajectory. A well-designed reward system can significantly accelerate learning, while a poorly designed one can mislead the agent.

3.1.2.2 Policies and Value Functions in RL

Navigating the intricate spaces of the RL world requires both a direction (what to do next) and a sense of the terrain (what the future might hold). Policies and Value Functions serve these roles, guiding the agent through its learning journey. Both determine how an agent will act and learn over time.

- **Policy (π)**: A policy is a strategy that the agent employs to determine the next action based on the current state. It can be seen as a mapping from states to probabilities of selecting each possible action.
    - **Deterministic vs. Stochastic Policies**: In a deterministic policy, there's a defined action that the agent will always take for a given state. On the other hand, a stochastic policy provides a probability distribution over actions, allowing for exploration and capturing uncertainties in decision-making.
    - **Policy Improvement and Iteration**: RL often aims to find an optimal policy that maximizes the expected reward over time. Agents can iteratively refine their policies through

mechanisms like policy iteration, evaluating and improving them to converge to an optimal strategy.

- **Value Function**: While the policy directly maps actions, the value function offers a forecast, estimating how good a particular state or action is in terms of expected future rewards.
    - **State Value Function ($V(s)$)**: This is a measure of how good it is for an agent to be in a particular state. More formally, it's the expected return an agent can obtain, starting from state s and following its current policy thereafter. It reflects the long-term desirability of states.
    - **Action Value Function ($Q(s, a)$)**: This quantifies the value of taking a specific action a in state s. It considers the immediate reward and weighs in the potential future rewards the agent can get by following its policy after taking the action.
    - **The Role of Bellman Equation**: Both the State and Action Value Functions often satisfy recursive relationships known as the Bellman equations. These equations relate the value of a state (or state-action pair) to the values of its successor states (or state-action pairs). They form the foundational bedrock for many RL algorithms, allowing iterative methods to estimate and optimize value functions. For a given policy π, the state value function $V_\pi(s)$ can be expressed using the Bellman equation as:

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V_\pi(s')] \tag{11}$$

        * $\pi(a|s)$ is the probability of taking action a in state s under policy π
        * $p(s', r|s, a)$ denotes the probability of transitioning to state s' and receiving reward r when taking action a in state s.
        * γ is the discount factor which dampens future rewards, capturing the idea that immediate rewards are generally preferred over distant ones.

While policies direct actions, value functions guide the updates to these policies. By estimating future rewards, the value function provides insights into how beneficial different states and actions

are. This information, in turn, can be employed to adjust and improve the policy, making it more aligned with the goal of maximizing cumulative rewards.

3.1.2.3 Q-Learning Algorithm

RL presents agents with a myriad of choices in intricate environments. Q-learning, as an off-policy algorithm, offers an approach to navigate these choices with the goal of optimizing long-term rewards. At the intersection of learning and decision-making, Q-learning is a beacon, guiding agents to effective policies even without a model of the environment.

Q-learning operates on the premise of learning the optimal action-value function directly. This function, denoted as $Q(s, a)$, quantifies the expected return of taking action a in state s, followed by an optimal policy thereafter. Unlike other algorithms, Q-learning learns about the optimal policy regardless of how the agent explores the environment or what policy it currently follows.

The heart of Q-learning is its iterative update rule, which refines the Q-values based on observed rewards and estimated future returns:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)] \tag{12}$$

- r is the immediate reward
- $\max_{a'} Q(s', a')$ estimates the future reward.
- α is the learning rate.
- γ is the discount factor.

One of the central challenges in RL is the trade-off between exploration and exploitation. This is the dilemma of whether the agent should try new actions (exploration) or stick to what it believes to be the best action (exploitation). Figure 3.2 shows how this trade-off works. A popular approach to address this is the ϵ-greedy policy, where the agent takes a random action with probability ϵ and the best-known action with probability $1 - \epsilon$. Over time, ϵ can be decayed to shift from exploration to exploitation.

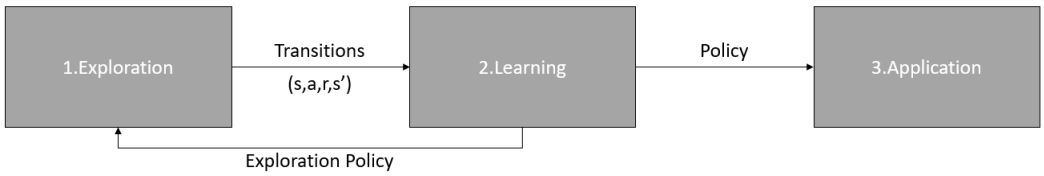

Figure 3.2: Exploration Vs. Exploitation (Wiering & Van Otterlo, 2012)

In this model-free RL algorithm, the Q-value of the action a at the state s_t of the agent at time slot t, denoted by $Q(s_t, a_t)$, is calculated as:

$$Q(s_t, a_t) = \mathbb{E}_\pi\{\sum_{t=0}^{H-1} \gamma^t r_{t+1} | s_0 = 0, a_0 = 0, a_t = \pi(s_t)\} \tag{13}$$

First of all, we initialize the Q-table with zero. Then, the Q-value will be updated in each time step as follows:

$$Q(s_t, a_t) \leftarrow (1 - \lambda)Q(s_t, a_t) + \lambda(r_t + \lambda max Q(s_{t+1}, a_{t+1})) \tag{14}$$

With the growth of DL, neural networks have been used as function approximators for the Q-values. DQNs can handle vast and continuous state spaces. DQNs provide a paradigm shift from traditional tabular Q-learning, allowing RL agents to function efficiently in environments with large or continuous state spaces. The nexus of Q-learning and DL in DQNs leads to agents capable of complex decision-making, but it also introduces unique challenges.

3.1.3 Reinforcement Learning for Maintenance Policy

We provide an RL approach for the CBM policy optimization of the system described in the previous section. Every RL framework consists of five main different components, namely, (i) Agent, (ii) Action-space, (iii) State-space, (iv) Reward, and (v) Transition probabilities. The MDP for proposed RL problem can be defined as a tuple (S, A, P, R, λ) where S is the states, A is the action set $A = \{a_0, a_1, a_2, a - 3\}$, P is the transition probability, and R is the reward function. Let's define each component in detail:

- **Agent**: The management system, as the intelligent component of the system, acts as the agent

and interacts with the environment based on a set of maintenance decisions.

- **State-space**: In the proposed problem, a component could be in three different states: (i) Healthy state, (ii) Unhealthy state, and (iii) Failed state. Therefore, the state of the system is in the form of (N_2, N_1), in which N_2 represents the number of failed units and N1 represents the number of units in unhealthy units.

- **Action-space**: At each decision epoch, four actions are available:

  - Action 0, i.e., do nothing denoted by a_0.
  - Action 1, i.e., conducting reactive maintenance (RM) on the failed units, denoted by a_1.
  - Action 2, i.e., conducting preventive maintenance (PM) on the unhealthy units denoted by a_2.
  - Action 3, i.e., conducting PM on the unhealthy units and RM on the failure units, denoted by a_3

  At each decision epoch, the agent makes an action which results in the system transition such that action 0 takes one epoch, and after that the system transits from $S_t = (i, j)$ to $S_{t+1} = (i', j')$ with the probability $p^{a_0}_{(i,j),(i',j')}$ obtained from Equation1 for a decision epoch π as follows:

$$p^{a_0}_{(i,j),(i',j')} = p_{(i,j),(i',j')}(\pi) \tag{15}$$

  Action 1 takes one epoch and after that, the system transits from $S_t = (i, j)$ to $S_{t+1} = (0, j)$ with probability one. This state transition implicitly assumes that if action 1 is selected, all units in the operating state remain unchanged. Action 2 takes one epoch and after that, the system transits from $S_t = (i, j)$ to $S_{t+1} = (i, 0)$ with probability one. This state transition also implicitly assumes that if action 2 is selected, all units in the operating state remain unchanged. Action 3 takes one epoch and after that, the system transits from $S_t = (i, j)$ to $S_{t+1} = (0, 0)$ with probability one.

- **Reward**: The reward function in the proposed model is a function of different cost components, which is denoted by the following equation:

$$\begin{aligned} R_t(s,a) = &-[M - N_1(zt) - N_2(t)]C_0\Delta t - N_1(t)C_1\Delta t \\ &- \Delta t C_D max\{0, D - [M - N_1(t) - N_2(t)]P_0 - N_1(t)P_1\} \\ &\Delta t C_E max\{0, [(M - N_1(t) - N_2(t))P_0 + N_1(t)P_1] - D\} \\ &- [C_K + C_F N_2(t)]\beta_2 - [C_K + C_P N_1(t)]\beta_3 - [C_K + C_P N_1(t) + C_F N_2(t)]\beta_4 \end{aligned} \tag{16}$$

Where:

$$\beta_1 + \beta_2 + \beta_3 + \beta_4 \leq 1 \tag{17}$$

In Equation 16, the production rate of a unit in the healthy and unhealthy states are denoted by P_0 and P_1, respectively such that $P_0 \geq P_1$. Also, the operational cost of a unit in states 0 and 1 are C_0 and C_1 ($C_1 \geq C_0$). The constant demand rate is denoted by D, which should be satisfied. The production rate of the system depends on its operational state, so for state $s_t = (i, j)$, the production rate of the system is $(M - i - j)P_0 + jP_1$. We assume there is an economic dependency among the units. Maintenance action needs to send crew members and incurs a high fixed set-up cost as C_k. Hence, simultaneous maintenance of several units is more cost-effective than conducting maintenance for one unit. It is assumed the cost of RM $(C_R M)$ is larger than the cost of PM, i.e., $(C_R M \geq C_P M)$. RM and PM are perfect, meaning they return the unit to the as-good-as-new state. The objective is to minimize the expected total cost during a finite planning horizon (T). In addition, the cost rate of lost production for unsatisfied demand is also considered, which occurs when demand exceeds the production level (C$_D$. On the other hand, the excess production can be sold at a lower price as C_E.

- **Transition probabilities**: Let $p^a_{(i,j),(i',j')} = P(S_{t+1} = (i', j')|S_t = (i, j), a)p^{a_0}_{(i,j),(i',j')}$ be the transition probability of being in state (i', j') at time $t + 1$, if the system was in state (i, j) at time t, and action a is chosen. The transition probabilities can be calculated from Equations.[1-4].

3.2 Numerical Experiment

Consider a system with three components, i.e., $M = 3$. As previously mentioned, each component can be in three different states. As a result, the overall system state is equal to 10, which can be shown in the form of (N_1, N_2), which N_1 and N_2 represents the number of failed units and number of units in warning states, respectively. Table 3.1 shows the state-space of the proposed model.

Table 3.1: State-Space

Num. of state	State
0	(0,0)
1	(0,1)
2	(0,2)
3	(0,3)
4	(1,0)
5	(1,1)
6	(1,2)
7	(2,0)
8	(2,1)
9	(3,0)

The agent will start from state 0 and choose an action. The agent moves among states based on the transition probabilities defined in Equations [1-4]. Every single action changes the state of the system. Figure 3.3 shows the state-action space for the system under study, including ten states and four actions: 0, 1, 2, and 3. Each time, the system can transit from state i, for ($i \in \{0, 1, 2, \ldots, 9\}$) to state j, for ($j \in \{0, 1, 2, \ldots, 9\}$) with action a, ($a \in \{0, 1, 2, 3\}$) with the probability P_{ij}^{a}. As it can be seen from Figure3.3, blue arrows show the transition from state i to state j when the "no repair" action is selected.

It is worth mentioning that most of the input parameters are taken from (Azizi & Salari, 2023) and (Salari & Makis, 2017). They are related to wind turbine gearbox data. The following Table 3.2 shows input parameters that are considered in our model:

By considering the transition probability formulation and values that are as follows (Tables3.3-3.6), reward function, and input cost components, the results of the RL implementation are shown in Figure3.4.

The code is run for 1000 decision epochs. In each decision epoch, the agent will go through 500

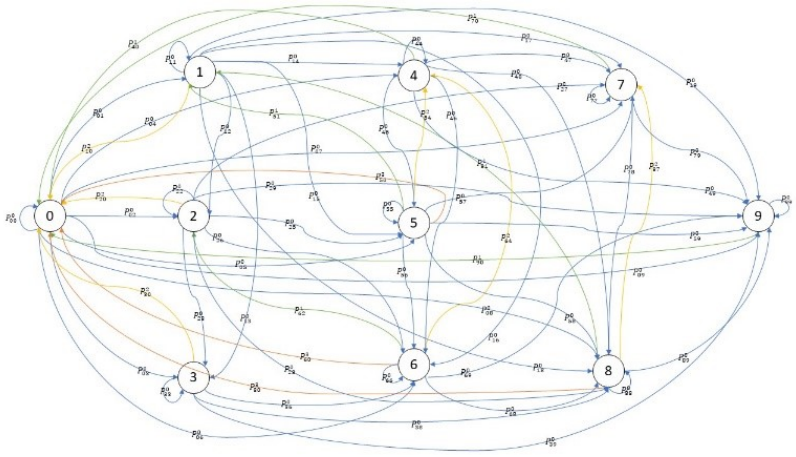

Figure 3.3: RL State-Action Space

Table 3.2: Input parameters

Parameters	Value
C_0	1000
C_1	2000
C_D	20000
P_0	0.99
P_1	0.01
C_E	12000
D	2000
C_K	800
C_F	10000
C_P	1000
t	1
M	3

Table 3.3: Transition probabilities for action 0

	(0,0)	(0,1)	(0,2)	(0,3)	(1,0)	(1,1)	(1,2)	(2,0)	(2,1)	(3,0)
(0,0)	0.1225	0.1932	0.1016	0.0178	0.1792	0.1885	0.0496	0.0875	0.046	0.0142
(0,1)	0	0.1353	0.1423	0.0374	0.1113	0.2491	0.1002	0.1086	0.0893	0.0265
(0,2)	0	0	0.1796	0.0787	0	0.2549	0.2023	0.1011	0.1731	0.0493
(0,3)	0	0	0	0.1653	0	0	0.4077	0	0.3352	0.0918
(1,0)	0	0	0	0	0.2446	0.2593	0.0682	0.2406	0.1265	0.0587
(1,1)	0	0	0	0	0	0.2725	0.1433	0.2241	0.2508	0.1093
(1,2)	0	0	0	0	0	0	0.3012	0	0.4952	0.2036
(2,0)	0	0	0	0	0	0	0	0.4966	0.2611	0.2423
(2,1)	0	0	0	0	0	0	0	0	0.5488	0.4512
(3,0)	0	0	0	0	0	0	0	0	0	1

Table 3.4: Transition probabilities for action 1

	(0,0)	(0,1)	(0,2)	(0,3)	(1,0)	(1,1)	(1,2)	(2,0)	(2,1)	(3,0)
(0,0)	0	0	0	0	0	0	0	0	0	0
(0,1)	0	0	0	0	0	0	0	0	0	0
(0,2)	0	0	0	0	0	0	0	0	0	0
(0,3)	0	0	0	0	0	0	0	0	0	0
(1,0)	1	0	0	0	0	0	0	0	0	0
(1,1)	0	1	0	0	0	0	0	0	0	0
(1,2)	0	0	1	0	0	0	0	0	0	0
(2,0)	1	0	0	0	0	0	0	0	0	0
(2,1)	0	1	0	0	0	0	0	0	0	0
(3,0)	1	0	0	0	0	0	0	0	0	0

Table 3.5: Transition probabilities for action 2

	(0,0)	(0,1)	(0,2)	(0,3)	(1,0)	(1,1)	(1,2)	(2,0)	(2,1)	(3,0)
(0,0)	0	0	0	0	0	0	0	0	0	0
(0,1)	1	0	0	0	0	0	0	0	0	0
(0,2)	1	0	0	0	0	0	0	0	0	0
(0,3)	1	0	0	0	0	0	0	0	0	0
(1,0)	0	0	0	0	0	0	0	0	0	0
(1,1)	0	0	0	0	1	0	0	0	0	0
(1,2)	0	0	0	0	1	0	0	0	0	0
(2,0)	0	0	0	0	0	0	0	0	0	0
(2,1)	0	0	0	0	0	0	0	1	0	0
(3,0)	0	0	0	0	0	0	0	0	0	0

Table 3.6: Transition probabilities for action 3

	(0,0)	(0,1)	(0,2)	(0,3)	(1,0)	(1,1)	(1,2)	(2,0)	(2,1)	(3,0)
(0,0)	0	0	0	0	0	0	0	0	0	0
(0,1)	0	0	0	0	0	0	0	0	0	0
(0,2)	0	0	0	0	0	0	0	0	0	0
(0,3)	0	0	0	0	0	0	0	0	0	0
(1,0)	0	0	0	0	0	0	0	0	0	0
(1,1)	1	0	0	0	0	0	0	0	0	0
(1,2)	1	0	0	0	0	0	0	0	0	0
(2,0)	0	0	0	0	0	0	0	0	0	0
(2,1)	1	0	0	0	0	0	0	0	0	0
(3,0)	0	0	0	0	0	0	0	0	0	0

	0	1	2	3
0	-5.610326e+07	0.0	0.0	0.0
1	-5.436832e+07	0.0	-39966000.0	0.0
2	-6.303102e+07	0.0	-39987600.0	0.0
3	-6.288400e+07	0.0	-40009200.0	0.0
4	-5.278500e+07	-39973200.0	0.0	0.0
5	-6.197207e+07	-39993800.0	-39984800.0	-39994800.0
6	-6.211208e+07	-40014400.0	-40006400.0	-40016400.0
7	-4.447280e+07	-40002000.0	0.0	0.0
8	-6.418854e+07	-40022600.0	-40003600.0	-40023600.0
9	0.000000e+00	-40030800.0	0.0	0.0

Figure 3.4: Q-table

time steps. In other words, the agent will take 500 different states and actions per decision epoch.

Each value in this table represents the value of the Q-table per each decision epoch. For example, if the system is in state = 4 and the agent takes action 1, the q-value would be -39973200. Also, zeros mean the agent never took that action for that specific state. For example, based on the assumptions that we had, the agent will never take action 1 for the states that do not have any failed units. As mentioned before, the optimal policy can be extracted from the Q-table. For each state, the maximum value of the Q-table would be the optimal action. The optimal policy for this problem is shown in Table 3.7.

Table 3.7: Optimal policy

State	Optimal action
0	0
1	2
2	2
3	2
4	1
5	2
6	2
7	2
8	2
9	1

The reward and cost functions are shown in Figure3.5 and 3.6, respectively. As shown, the

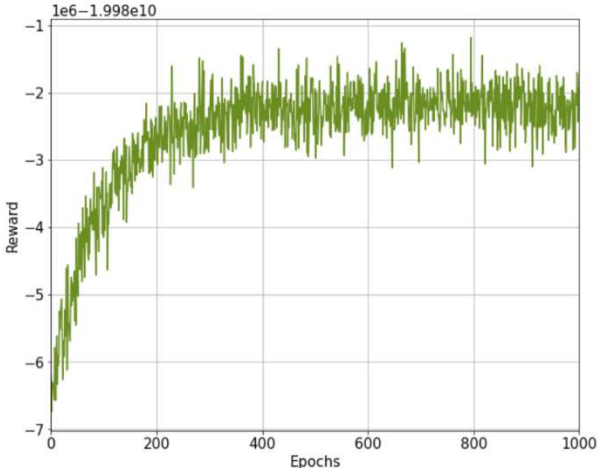

Figure 3.5: Reward Function

reward and cost functions converge at epoch 400.

Figure3.6 shows the cost convergence of the proposed problem through different epochs. Each point in the figure shows the cumulative maintenance cost in each epoch. As shown, it can be seen that after 400 epochs, the cost converges. Therefore, the maintenance's cumulative cost equals 39963148 (approximately).

In addition, we can see the frequency of actions in every decision epoch in Figure3.7. For instance, during the 600th decision epoch, the agent opted against selecting action 3. Instead, it equally picked actions 1 and 2 (150 times each) and chose action 0 on 220 occasions.

As we can observe in Figure3.7, at the beginning when the system state is healthy, the most frequent action taken by the agent is action 0, i.e., do nothing which is a reasonable behaviour. When the system starts to deteriorate over time, the frequency of other actions is increased, which is expected. Mainly, when the cost converges, on average agent takes 240 times action 0, 120 times action 1, 140 times action 2 and almost 0 times action 3.

It is worth mentioning that there are 500 time steps in each epoch, meaning the summation of different actions that the agent takes should equal 500. The processor of the computer that we used to run the RL code is Intel(R) Core (TM) i7-8700 CPU @ 3.20GHz and 16GB RAM. Also, our computational time for this problem is less than 5 minutes.

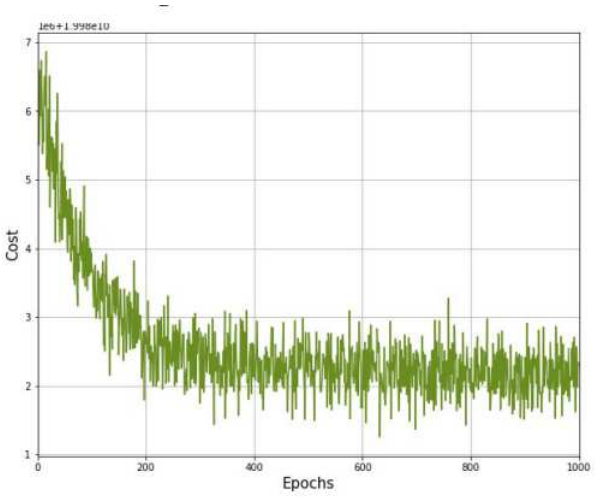

Figure 3.6: Cost Function

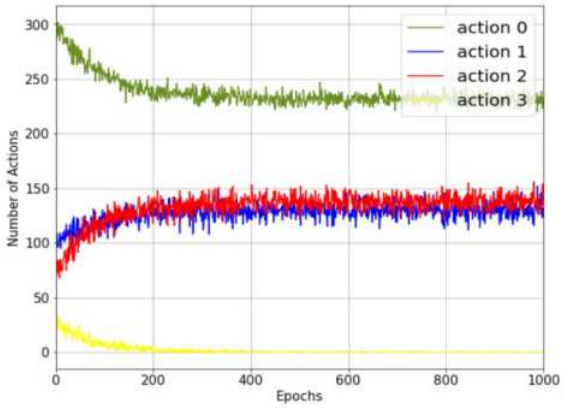

Figure 3.7: Number of different actions per decision epoch

3.2.1 Comparison of Proposed Model with Statistical Model

To evaluate the effectiveness of our proposed model, the comparison is made with a paper developed by (Azizi & Salari, 2023), in which a novel statistical method is designed to find the optimal maintenance policy for a multi-component system. The authors applied a heuristic algorithm to see the optimal policy and total maintenance cost. Based on the results, the best policy is to perform maintenance actions when one failed unit exists. In other words, it is optimal to stop the system and do the corrective maintenance when we have one failed unit, regardless of the number of unhealthy units.

It is important to note that we utilized identical input parameters for a comparative analysis between our proposed model and the one presented in (Azizi & Salari, 2023), followed by the execution of statistical code. The outcomes of this comparison are meticulously documented in Table 3.8. The results unequivocally indicate that our proposed method achieves a lower optimal average cost when contrasted with the cost reported in (Azizi & Salari, 2023). Furthermore, concerning computational efficiency, both algorithms exhibit similar processing speeds.

Table 3.8: Comparisons with different Algorithms

Algorithm	Total Cost
Proposed Algorithm	39963148
(Azizi & Salari, 2023)	39981000
Run-to-Failure	39992000

In addition, we compared our model with the run-to-failure model, i.e., only reactive maintenance will be performed on the system when all components fail. As the result shows, the proposed model leads to more cost savings.

3.2.2 Sensitivity Analysis

In this section, we perform the sensitivity analysis. We investigate the effect of input parameters on results. In particular, we present the sensitivity analysis on the main three cost parameters, namely, C_F, C_P, and C_K. In this regard, the actual values of these parameters described in Table 3.2 are multiplied by a coefficient, δ ranging from 0.5 to 3. The results are shown in the following

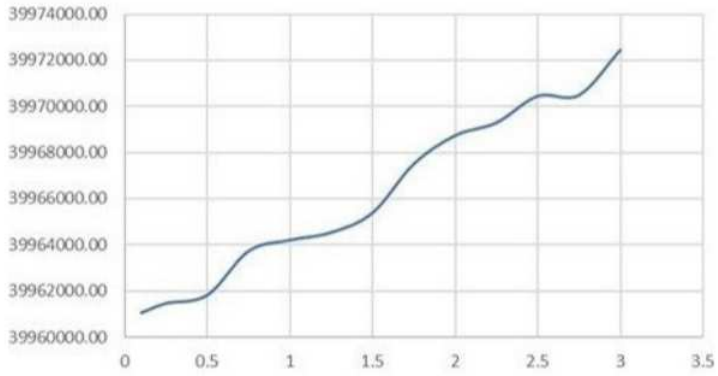

Figure 3.8: Total maintenance cost with different input parameters

graph.

The y-axis shows the total maintenance cost, and the x-axis shows the value of the coefficient. For example, if we increase all the costs above by doubling them, i.e., $\delta = 2$, the total maintenance cost will increase to 39969447. On the other hand, if we decrease the input costs, the total maintenance cost will decrease accordingly, which is a rational behaviour. This completes our numerical experiment part.

3.3 Conclusion and Future Perspective

In summary, our contribution in this chapter has centred on developing an RL-based framework meticulously designed to optimize maintenance decision-making, primarily driven by the overarching objective of cost minimization. We have situated our study within a complex parallel multi-unit system, where individual components operate independently and are susceptible to random failures. These components navigate through three distinct states: healthy, unhealthy, and failed, forming the intricate landscape for our maintenance strategy. To realize our objective, we harnessed the power of a Q-learning algorithm to discern the optimal maintenance policy that minimizes costs while ensuring system reliability.

In our quest to demonstrate the practicality and effectiveness of our proposed model, we have gone beyond theory. A comprehensive numerical example, enriched with sensitivity analysis, has been meticulously crafted to shed light on the model's real-world applicability. Through these

empirical exercises, we have uncovered noteworthy insights, underlining the substantial reductions in total maintenance costs achievable through our innovative approach.

Yet, our journey does not end here. Our sights are set on even greater horizons as we peer into the future. Specifically, we envision extending our model's applicability to systems comprising more than the current three components. While we believe that our proposed model exhibits effectiveness within a certain threshold of components, the emergence of vast state-spaces due to an increasing number of components may pose convergence challenges in a timely fashion.

Therefore, as a compelling avenue for future exploration, we aspire to delve into developing DL-based solutions tailored to tackle large-scale maintenance management quandaries. In particular, our focus will be directed toward creating DQN models, where we anticipate harnessing the prowess of neural networks to navigate the complexities of managing maintenance in expansive industrial systems. This exciting direction holds the promise of further advancing the field of maintenance optimization and expanding its applicability to ever-growing, intricate systems in diverse industrial domains.

Chapter 4

Deep Reinforcement Learning for Dynamic Maintenance Policies of Large-Scale Identical Parallel Manufacturing Systems

In this chapter, we delve into a maintenance decision-making framework tailored for multi-unit systems, leveraging the capabilities of machine learning. Specifically, we employ deep reinforcement learning to model the dynamic maintenance needs of a multi-unit parallel system in the presence of stochastic degradation and random failures. Each unit within the system independently follows a three-state homogeneous Markov process, categorizing its condition as healthy, unhealthy, or failed. By amalgamating individual unit states, we construct an overarching system state and explore the interactions among these states using the bivariate Birth/Birth-Death process. The central aim of our framework is to minimize maintenance costs while ensuring sustained system reliability. To address the intricacies of this challenge, we adopt the MDP framework for formulating and solving the optimal maintenance policy. Moreover, we employ the DDQN algorithm, a cutting-edge deep reinforcement learning technique, to enhance the practicality and efficiency of our solution, offering advantages in terms of both time and cost savings. To illustrate the framework's applicability,

we provide a numerical example showcasing its effectiveness in determining optimal maintenance policies for the examined multi-unit system.

4.1 Problem Statement

Consider a production system consisting of M identical, parallel units. Deterioration of each unit occurs independently according to a three-state homogeneous Markov process such that each unit has three states: healthy, unhealthy, and failure, denoted as states 0, 1, and 2, respectively. Healthy and unhealthy states are operational, and failure states are not operational. At the start of the planning horizon, all units are in a healthy state and gradually deteriorate. The system deterioration state is defined using two interacting populations. More specifically, the state of the system is defined as (N_2, N_1) where N_2 is the number of units in the failure state, and N_1 is the number of units in the unhealthy state and $0 \leq N_1 + N_2 \leq M$. It means that the number of units in the healthy state is $N_0 = M - N_2 - N_1$. According to this definition for system state, the total number of system states is: $\frac{(M+1)(M+2)}{2}$.

The Birth/Birth-Death stochastic process describes the interaction among the system states. Generally speaking, we can describe a bi-variate birth/birth-death Markov process as the governing dynamics of a system consisting of two types of particles, in which one out of three possible events can occur in infinitesimal time: (1) a new type 1 particle enters the system; (2) a new type 2 particle enters the system; or (3) a type 1 particle becomes a type 2 particle. In this system, N_1 and N_2 track the number of type 1 and type 2 particles, respectively. Compared to a birth/birth-death process, a birth-death process tracks only the size of a univariate population, which is unsuitable for modelling the state of the system under study. For more details about the Birth/Birth-Death process, see (Salmani et al., 2023).

Suppose that the production rate of a unit in the healthy and unhealthy states are p_0 and p_1, respectively, such that $p_0 > p_1$. A constant demand rate, D, per unit time, should be satisfied. The cost rate of lost production for unsatisfied demand, C_D, occurs while demand exceeds the production level. On the other hand, the excess production can be sold at a lower price as C_E. In addition to the cost parameters described so far, the following costs also occurred. The operational

cost of a unit in states 0 and 1 are C_0 and C_1 ($C_1 \geq C_0$). An economic dependency exists among the units: maintenance action needs to send crew members and incurs a high fixed set-up cost as C_K. Hence, simultaneous maintenance of several units is more cost-effective than conducting maintenance for one unit. For a unit that is in the failure state, corrective maintenance (CM) should be conducted to bring the unit back to the healthy state, while preventive maintenance (PM) is conducted for an unhealthy unit. It is assumed that the cost of CM (C_{CM}) is larger than the PM cost $C_{PM} : (C_{CM} > C_{PM})$. CM and PM are perfect, returning the unit to the as-good-as-new state. We assume there is no delay between the scheduled maintenance time and the actual implementation of maintenance actions.

We provide a policy for the system, which directly maps the system state upon each decision epoch to the maintenance decision space. The decision epochs are scheduled periodically at pre-specified time points as $t, 2t, 3t, ...$ to reveal the deterioration of the system.

At each decision epoch, four actions are available for maintenance:

- Action 0 means do nothing
- Action 1 means conducting corrective maintenance (CM) on the failed units
- Action 2 means conducting preventive maintenance (PM) on the unhealthy units
- Action 3 means conducting PM on the unhealthy units and CM on the failure units.

More details about these actions are provided in the following sections, where the deep reinforcement algorithm is provided.

The paper aims to propose a CBM policy so that given the state of the system at each decision epoch as j^{th}, i.e., $(N_{2,j}, N_{1,j})$, the policy prescribes an action to minimize the expected cost during the planning time of maintenance contract. In the next section, we propose a customized DRL approach to obtain the optimal policy efficiently.

4.1.1 Markov Decision Process

Different operational research (OR) methods/techniques have been proposed to tackle sequential decision-making problems in deterministic and stochastic situations, including Markov Decision

Process, Dynamic programming, Stochastic optimization, decision trees, etc. MDP is employed to analyze the decision-making process in a stochastic environment while Markov property holds. Markov Decision Process plays a vital role in RL as almost all algorithms of RL and DRL have been developed based on MDP. Thus, this section reviews the main components and concepts of MDP.

Roughly speaking, an MDP can be represented by the following tuple (S, A, P, R, λ). In the following, these elements are described:

- S: is the finite set of states.
- A: is the finite set of actions
- P: is a transition probabilities matrix. More precisely, given that the system/environment is in state s, and the agent takes action a, with probability $P^a_{ss'}$, the system transits to state s'. Indeed, it is a conditional probability which can be denoted as follows: $P(S_{t+1} = s'|S_t = s, A_t = a)$. For more details about the transition probabilities, see (Azizi & Salari, 2023).
- R: is immediate reward. It can be described as follows: given that the system is in the state s, and the agent takes action$a$, the immediate reward is $R^a_S = E(R_{t+1}|S_t = s, A_t = a)$.
- λ: is discount factor. It is a number between zero and one, $\lambda \in [0, 1]$, whose concept returns to the engineering economy principle. More precisely, it corresponds to the concept of interest rate. It means that the present value of a given money value as F under the interest rate i% is $F(1 + i)^{-1}$. From the mathematical point of view, it is commonly employed in the MDPs to avoid the infinity of the solutions.

Agent and environment are two main components of RL. Given that the environment is in state $s \in S$, the agent takes action a from set $A(a \in A)$, observes immediate reward R^a_s and with probability $P^a_{ss'}$, the environment transits to state s'. This process continues until the agent reaches the terminal state.

The ultimate goal of an MDP is determining the optimal policy to maximize(minimize) the expected returns (costs) while the stages can be finite or infinite. Hence, policy is a fundamental concept in MDP, RL, and DRL. Given a specific state of MDP, a policy prescribes which action should be selected among the eligible actions of that state. A policy fully describes the behaviour

of the agent in MDP. Given that the system is in the state s, under the policy π, the probability of taking action a is $P(A_t = a|S_t = s)$. Hence, a policy is a map from the states of the system to the action set: $\pi : S \rightarrow A$.

The other two important concepts in MDP are the state-value and action-value functions. Under policy π, the state-value function of state s is denoted as $V_\pi(s)$. It computes the expected return given that the system is in state s and the agent follows policy π. The action-value function is the expected return given that the system is in state s, action a is taken, and hereafter, policy π is employed. It is denoted by $Q_\pi(s, a)$. According to the Bellman optimality equation, the state-value and action-value functions can be decomposed into immediate reward plus the discounted value of successor states. Under policy π, and given the system is in state s and the agent takes action a, the Bellman equation can be presented as follows:

$$q_\pi(s, a) = R_s^a + \lambda \sum_{s' \in s} P_{ss'}^a V_\pi(s') \tag{18}$$

According to the theorem of MDP in (Azizi & Salari, 2023), for any MDP, an optimal deterministic policy exists that is better than or equal to all other policies. The optimal policy specifies the optimal action-value function.

There are no closed-form solutions for the Bellman optimality equation. However, many iterative methods and algorithms are presented in this regard: value iteration, policy iteration, Q-learning and SARSA.

4.1.2 Reinforcement learning and Q-learning

As stated before, the base of RL is MDP. Q-learning is a well-known algorithm of RL which can effectively solve the small and medium-sized MDPs. It uses the action-value function to determine the optimal policy. More specifically, the Q-values for action-state pairs are initialized at zero as a matrix/table. According to the exploration/exploitation process, the agent takes action, and the Q-values are updated according to the following Equation 19:

$$Q(s, a) \leftarrow Q(s, a) + \theta[r + \lambda * Max_a Q(s', a) - Q(s, a)] \tag{19}$$

In this equation, $0 \leq \theta \leq 1$ is a learning rate and $0 \leq \lambda \leq 1$ is the discount factor. Exploration means that the agent randomly takes an action from the available action of the current state. In contrast, exploitation means that the agent uses a Q-table to select the best action. The trade-off between exploration and exploitation uses a parameter as ϵ.

Table 4.1 provides the Q-learning algorithm. For more detail about the procedure of Q-learning, the reader is referred to (Yousefi et al., 2022).

Table 4.1: The algorithm of Q-learning

Q-learning algorithm
Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
Initialize s
Repeat (for each step of the episode):
Choose a from s using policy derived from Q (e.g., ϵ greedy)
Take action a, observe r, s'
$Q(s, a) \leftarrow Q(s, a) + \theta[r + \lambda * Max_a Q(s', a) + Q(s, a)]$

4.1.3 Deep Reinforcement Learning

For large-scale problems, as the size of MDP increases, i.e., the cardinality of A and S sets, the efficacy of Q-learning significantly decreases to find the optimal policy in a reasonable time. Apart from that, the inefficiency of Q-learning is more obvious for problems with continuous state/action spaces. To overcome the lack of scalability of Q-learning to handle large-scale problems or continuous space problems, the action-value function can be parameterized to approximate/estimate the true value of the action-value function. Given the parameter θ, the action-value function under policy π is approximated as follows:

$$Q(s, a, \theta) \approx Q_\pi(s, a) \tag{20}$$

Hence, $Q(s, a, \theta)$ is approximated action-value function parameterised with trainable parameter θ. Different function approximators exist, linear combinations of features, neural networks, and decision trees, among others. DQN is a prevalent method to train parameter θ and consequently

finds a suitable approximation for the true action-value function. As its name indicates, it employs deep neural networks to find optimal values for θ.

A deep network is a neural network with multiple hidden layers; each layer has various neurons. The current state of the environment is specified at the input layer of the neural network so that each element of the current state vector of the environment is fed into a neuron of the input layer. The approximate action-value function is computed at the output layer for each available action. Hence, the number of neurons at the output layer is usually set to the number of actions. Some researchers state that standard DQN algorithms suffer from overestimating the action-value function. Hence, to address the problems of standard DQNs, double DQN (DDQN) is proposed. In standard DQN, one deep neural network is employed, while in DDQN, two networks are used: an original training network or online network and a target network. The hyperparameters of the two networks are exactly similar, meaning the number of hidden layers and neurons in each layer are identical.

Parameters of online network and target network are denoted as θ and θ', respectively. The target network is used for policy evaluation, while the online network selects actions given the current state. While the parameter of the online network is updated in each training step, θ' of the target network is frozen, and only after a specified number of iterations as z, the parameter of the online network is copied into the target network. The goal of this point is to stabilize the learning process. It is used batch training to train the neural network. Previous transition steps are recorded in a reply memory. A minibatch of transitions from the reply memory is randomly selected to train the online network.

Training of the online network means that an optimization algorithm is conducted to minimize the squared loss, i.e., $[Q_{target} - Q(s, a, \theta)]^2$, concerning parameter θ. In each training step, according to the gradient descent method, the parameter is updated using the following equation:

$$\theta \leftarrow \theta + \alpha[Q(s, a; \theta) - Q_target]\nabla_\theta Q(s, a; \theta) \tag{21}$$

While α is the learning rate.

4.2 Deep Reinforcement Learning for Maintenance Planning of the current system

In the following, we propose an RL approach for the CBM policy optimization of the system described in the previous section. We show that the system deterioration and maintenance process can be formulated as an MDP. In the context of our CBM, the observation of S upon the j^{th} decision epoch is denoted as $s_j = (n_{2,j}, n_{1,j})$, and $a_j \in \{0, 1, 2, 3\}$ represents the decisions/actions. The state transition probability P determines the distribution of system state s_{j+1} given the state observation s_j at the j^{th} decision epoch and the chosen action a_j. When the next state is observed, the immediate reward $R_{s_j}^{a_j}$ is also determined. The immediate reward is defined as the negative of the loss described in the following according to our CBM.

According to the costs mentioned above, the reward function in the proposed model is a function of the cost components given by:

$$Reward = R_1 + R_2 \tag{22}$$

where:

$$R_1 = (C_K + C_{CM} \times N_2) \times B_1 - (C_K + C_{PM} \times N_1) B_2 - (C_K + C_{CM} \times N_2 + C_{PM} \times N_1) B_3 \tag{23}$$

in which B1, B2, and B3 are indicator variables,and

$$R_2 = -(M - N_1' - N_2') * C_0 - N_1' * C_1 - C_D * max(0, D - ((M - N_1' - N_2') * p_0 + N_1' * p_1)) \\ + C_E * max(0, ((M - N_1' - N_2') * p_0 + N_1' * p_1) - D), \tag{24}$$

where n_1' and n_2' are equal to the value of n_1 and n_2 after performing the corresponding action. That's mean $n_1' = n_1$ and $n_2' = n_2$ if $a = 0$; $n_1' = n_1$ and $n_2' = 0$ if $a = 1$; $n_1' = 0$ and $n_2' = n_2$ if $a = 2$; and $n_1' = 0$ and $n_2' = 0$ if $a = 3$. In other words, the first part in Equation 22 calculates the cost of maintenance actions using three dummy variables. The second part calculates the operational costs of the system, the expected costs of unsatisfied demand, and the profit obtained

from the production exceeding the demand level. Notably, the second and third terms of R_2 cannot be non-zero simultaneously, i.e., the multiply of these two terms is always zero.

4.3 Numerical Study

In a system of 30 units, an agent initiates operations from a starting state of (0, 0). The system's state changes dynamically with each agent's action, governed by a complex stochastic model known as the Birth/Birth-Death process. This model encapsulates various transitional states, simulating units' addition, degradation, or repair. Significantly, the agent's decisions are influenced by a set of cost parameters outlined in Table 4.2, which incorporate variables such as the financial burden of maintenance activities and system downtimes.

The Birth/Birth-Death process serves as a mathematical foundation and adds layers of complexity to the agent's decision-making. Given the scale of 30 units, each action can result in a chain of consequences affecting the system's overall health. Therefore, the agent is tasked with navigating this intricate landscape, making choices that are not just immediate fixes but also aligned with long-term objectives like cost-efficiency and system reliability. This multi-unit, multi-state environment presents a challenging yet rich scenario for adaptive, strategic decision-making.

Table 4.2: Input Parameters Of The Example

Parameter	**Value**
D	9000
p_0	450
p_1	250
C_0	5
C_1	10
C_D	0.8
C_E	0.04
C_K	500
C_{CM}	40
C_{PM}	1
M	30

Other input parameters are also associated with the DL model, which is given in Table 4.3.

Moreover, in the DDQN-based DRL, a neural network with two fully connected hidden layers

Table 4.3: Hyper Parameters Of The DDQN

Parameter	Value
α	0.01
γ	0.5
ϵ	0.9
Number of iterations	50000
Batch size	64
Number of layers	3

(128 neurons in each layer) is chosen. Since the dimension of the states is two, the size of the input layer is 2, and since the number of actions is four, the size of the output layer is 4. The experience memory and minibatch size are 100,000 and 64, respectively. Also, the neural network duplication happens after every episode, which means that after every episode, the target network will be copied from the online network. At the beginning of a training simulation, ϵ starts at 0.9, and near the end, it becomes a minimal value. The future reward discount factor is set to be $\gamma = 0.5$.

As previously stated, our model's primary objective is to minimize long-term costs. We executed the model on a "pc model," which took approximately two and a half days to complete. Figure 4.1 indicates the outcomes of the run based on the input parameters specified in Table 4.2 and Table 4.3.

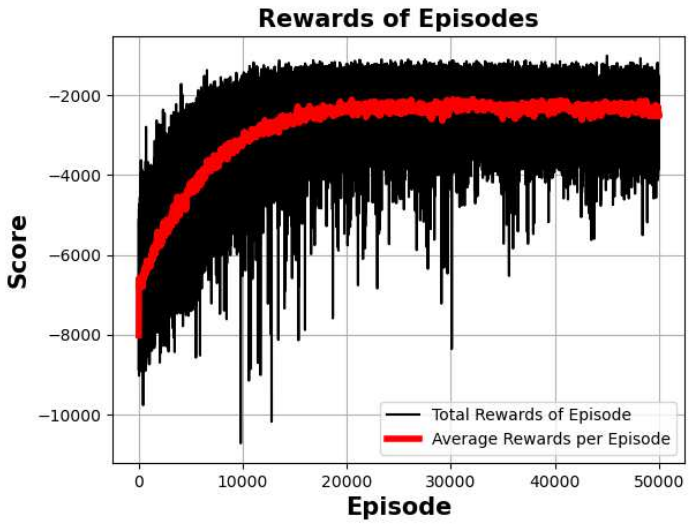

Figure 4.1: Convergence of the DDQN Algorithm.

Figure 4.1 illustrates the outcome of our proposed model. We executed the model for a total of 50,000 iterations to observe its convergence. As depicted in the figure, it is evident that as the iterations progress, the agent consistently improves its decision-making capabilities, resulting in higher rewards.

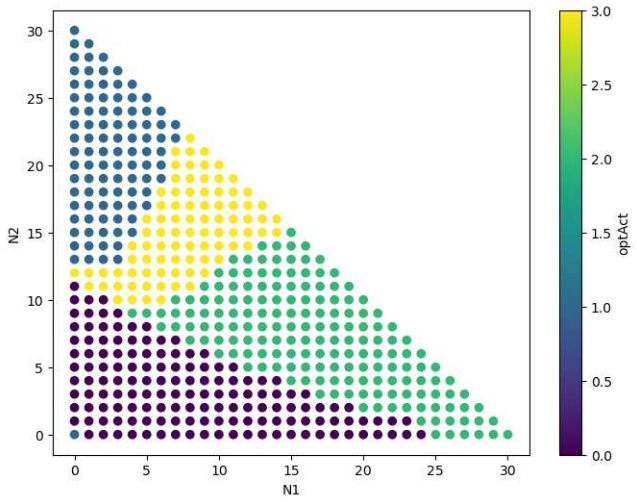

Figure 4.2: Varied Action Selections Based on Different Values of N_1 and N_2.

In Figure 4.2, distinct colours represent different actions, which are explained on the right side of the figure. The graph illustrates that when the number of failed and unhealthy units is low, it is preferable to take no action (action = 0). As the number of failed units increases while still having a small number of unhealthy units, the optimal action is to perform corrective maintenance (action = 1). Similarly, when the number of unhealthy units rises while the number of failed units remains low, the optimal approach is implementing preventive maintenance (action = 2). Lastly, the yellow area indicates that when there is a high number of failed and unhealthy units, the optimal strategy involves conducting corrective maintenance on failed units and preventive maintenance on unhealthy units simultaneously (action = 3).

Figure 4.3 enhances our understanding of the agent's decision-making process, enabling us to discern how it selects different actions based on varying states.

This graph displays the reward function values for various actions in different scenarios. As depicted in the image, the vertical axis represents the reward function value, while the horizontal axis

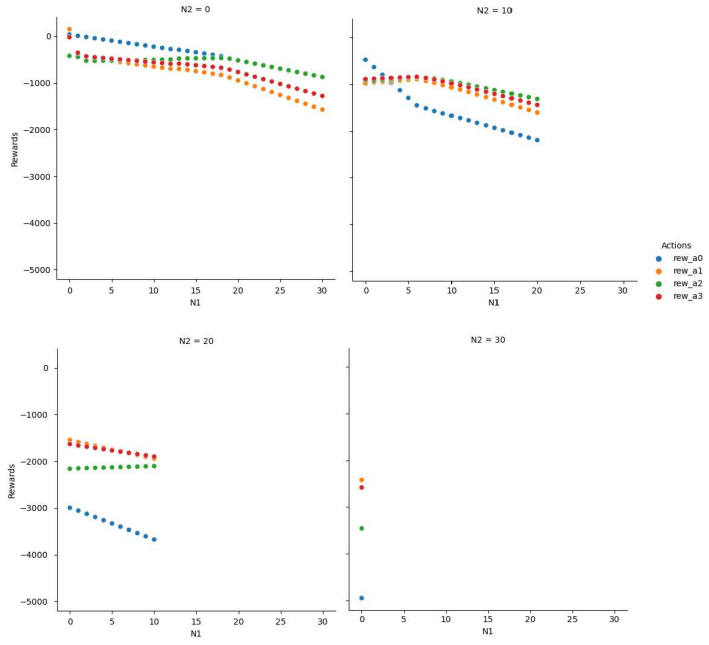

Figure 4.3: reward function's value as influenced by varying values of N_1 and N_2.

represents the number of unhealthy units. Each quadrant corresponds to a different numbers of failed units. Like Figure 4.1, the first quadrant of Figure 4.2 demonstrates that when the number of failures is low, action zero yields a higher reward. As the number of unhealthy units increases, the optimal action shifts towards preventive maintenance (action = 2). It is worth mentioning that the figure's size diminishes as we progress along the graph. This occurs because the total number of units remains constant, and an increase in the number of failures (N_2) will result in a reduction of the feasible area for the horizontal axis (N_1).

Figure 4.4 serves as a complementary graph to Figure 4.3, illustrating the optimal course of action for various scenarios involving different numbers of failed and unhealthy units.

Figure 4.4 visually presents the optimal action, determined by the value of the reward function, for each state. In Figure 4.3, it is observed that when the number of failed units is zero, and the total number of unhealthy units exceeds 15, both action zero and action two display similar values in terms of reward. However, Figure 4.4 reveals that, in this case, the optimal action is action 3.

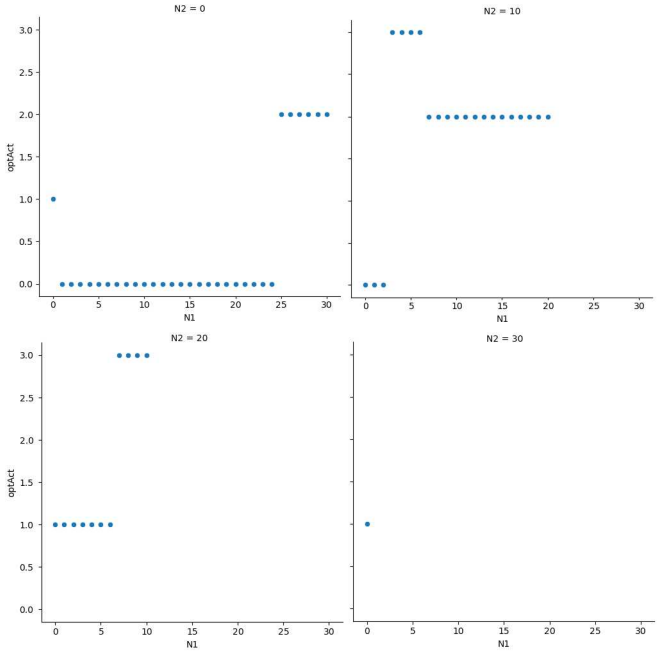

Figure 4.4: Optimal Action Corresponding to Different Values of N_1 and N_2.

4.4 Sensitivity Analysis

To accomplish our research objectives, we employed a comprehensive sensitivity analysis methodology. We carefully selected an appropriate approach that would provide meaningful insights into the parameter sensitivities while considering the nature of our model and the available resources. The methodology involved systematically varying the input parameters and observing the corresponding changes in the model's output. By following this approach, we could assess the model's sensitivity in a controlled and structured manner.

We have a range of input parameters encompassing cost components and DDQN network parameters. Observing the nature of our reward function, it becomes evident that the parameters C_{CM} (cost of corrective maintenance) and C_{PM} (cost of preventive maintenance) exert a significant influence on the reward value. Therefore, to assess the robustness of our model in the face of parameter variations, we conducted a sensitivity analysis specifically targeting these parameters in two phases. The goal was to determine whether our model remains resilient and reliable when these parameters are altered. Table 3.4 shows the different values of these parameters that we analyzed:

Table 4.4: Hyper Parameters Of The DDQN

Phase 1: Fixed C_{CM}	Scenario 1	C_{PM} = 5
	Scenario 2	C_{PM} = 10
	Scenario 3	C_{PM} = 20
Phase 2: Fixed C_{PM}	Scenario 1	C_{CM} = 30
	Scenario 2	C_{CM} = 10

Across both phases and all the scenarios, we observed convergence in the reward figure, indicating the stable performance of our model. Therefore, we will shift our focus away from analyzing the reward figure and instead delve into examining the agent's decision-making process in various states. This will provide valuable insights into how the agent's actions and choices adapt to different system conditions.

In the initial phase, we held the cost of corrective maintenance (C_{CM}) constant and assessed the robustness of our model by varying the value of the cost of preventive maintenance (C_{PM}). Specifically, with C_{CM} (cost of corrective maintenance) fixes at 40, we examined different values of C_{PM}. The corresponding values are presented in Table 3.4.

In the first scenario of the first phase, the cost of preventive maintenance was raised to 5. As indicated in the lower right section of Figure 4.5, when the cost of preventive maintenance (C_{PM}) is higher, the agent decides to do nothing instead of performing preventive maintenance to minimize the overall cost. Moreover, as illustrated in the upper left portion of Figure 4.5, the agent prioritizes corrective maintenance (action = 1) over a combination of preventive and corrective maintenance (action = 3). This behaviour is influenced by the increase in the cost of preventive maintenance (C_{PM}).

In the context of the second scenario, we undertook a deliberate modification by elevating the cost attributed to preventive maintenance. Our rationale behind this adjustment was to examine how this change would influence the agent's decision-making dynamics, with a specific focus on the frequency of selecting action two. It was our expectation that, given the increased cost associated with preventive maintenance, the agent would exhibit a decreased propensity to opt for action two in comparison to the outcomes observed in the first scenario. Figure 4.6 serves as a graphical

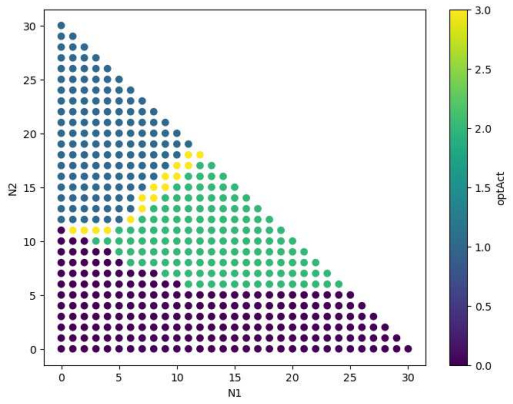

Figure 4.5: Varied Action Selections Based on Different Values of N_1 and N_2.

representation of the results stemming from this scenario, elucidating the agent's response to this alteration in cost dynamics. This visual representation provides a comprehensive insight into how the agent navigates its decision space and strategically adapts its choices in reaction to the heightened preventive maintenance costs, shedding light on its ability to optimize its decision-making process in varying environmental conditions.

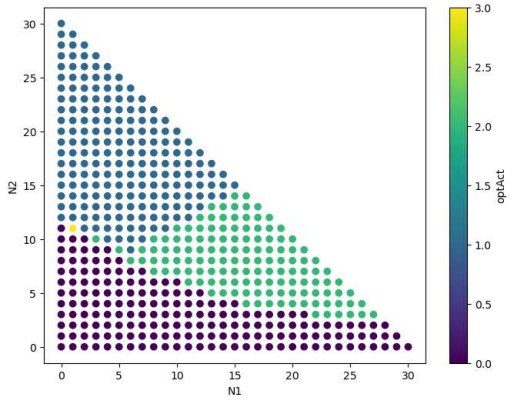

Figure 4.6: Varied Action Selections Based on Different Values of N_1 and N_2.

In Figure 4.6, we present a graphical representation of the agent's decision-making pattern, shedding light on a notable trend that aligns with our initial expectations. This visual depiction offers valuable insights into how the agent navigates its decision space in response to varying costs associated with different actions. The observed behaviour in the figure showcases a clear and discernible pattern. As the cost linked to taking action two steadily rises, the agent exhibits a growing

inclination towards selecting action zero and action three. This tendency indicates a deliberate strategic shift on the part of the agent, as it recognizes that the increased cost of action two makes it a less attractive choice within the given decision context. This phenomenon underscores the agent's capacity to adapt and optimize its decision-making process in response to changing environmental conditions or cost structures, showcasing its ability to balance trade-offs effectively and efficiently, ultimately aiming to maximize its expected utility or achieve its predefined goals.

Lastly, Figure 4.7 illustrates the third scenario of the initial phase in the sensitivity analysis. The analysis reveals that the agent favours executing maintenance activities on the system to preempt any additional expenses arising from frequently choosing action 0. By proactively engaging in preventive maintenance, the agent aims to mitigate the long-term costs associated with system failures or downtimes, which could potentially escalate if action 0 is repeatedly selected. This strategy suggests a calculated approach by the agent to optimize the system's performance while also keeping an eye on cost efficiency.

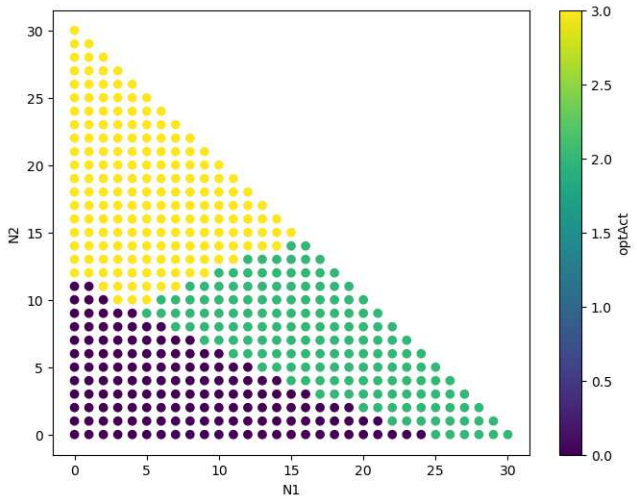

Figure 4.7: Varied Action Selections Based on Different Values of N_1 and N_2.

In the second phase, we held the cost of preventive maintenance (C_{PM}) constant and assessed the robustness of our model by varying the value of the cost of corrective maintenance (C_{CM}). We fixed the cost of preventive maintenance to 1 and changed the value of the cost of corrective maintenance as shown in Table 4.4. In the first scenario of the second phase, we expect that when

the cost of corrective maintenance is reduced, there should be a tendency for the agent to perform both corrective and preventive maintenance. This preference is illustrated in the top left portion of Figure 4.8, where the agent favours the simultaneous execution of both maintenance actions (action = 3) instead of choosing action 1.

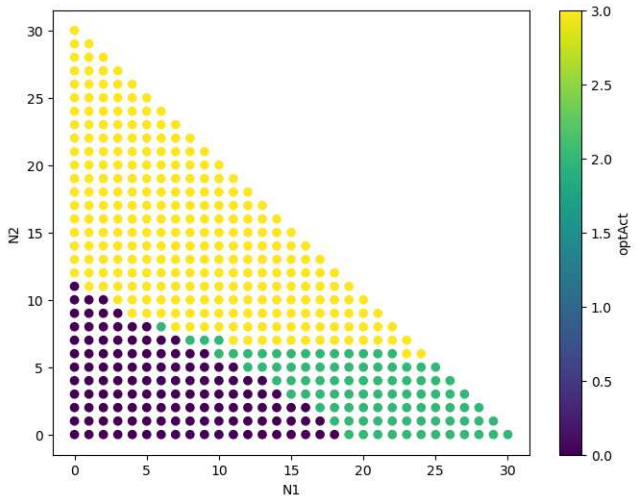

Figure 4.8: Varied Action Selections Based on Different Values of N_1 and N_2.

In the second scenario of the second phase, the cost of corrective maintenance was set to 10. Interestingly, the agent exhibited an even stronger preference for action 3 in this scenario (Figure 4.9).

By comparing the bottom right portion of Figure 4.8 with the original results presented in Figure 4.2, it can be inferred that when the costs of both preventive and corrective maintenance are relatively small and similar, the agent tends to either choose not to take any action or perform both actions simultaneously (action = 3). This decision pattern can be attributed to the higher crew cost (C_K) in comparison to the costs of corrective maintenance (C_{CM}) and preventive maintenance (C_{PM}). This observation is demonstrated in Figure 4.9. At the beginning of each episode, when all components are in a healthy state, the agent tends to refrain from taking any action. However, as the system degrades over time, the cost associated with action 3 (represented by the red points) becomes significantly lower than the costs of other actions. This can be attributed to the relatively high crew cost, which incentivizes the agent to select action three more frequently.

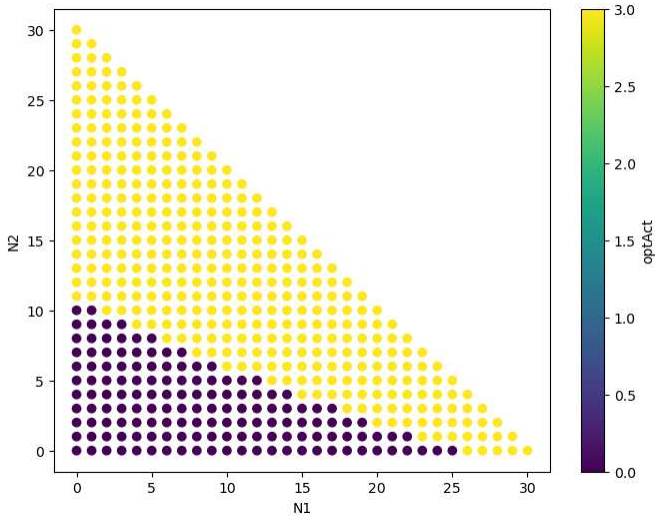

Figure 4.9: Varied Action Selections Based on Different Values of N_1 and N_2.

4.4.1 Conclusion

In conclusion, we proposed a deep RL-based framework for maintenance decision-making with the goal of cost minimization. We considered a large parallel multi-unit system. Units are subject to random failures and are independent of each other. Each unit could be in three different states, namely, healthy, unhealthy, and failed. Since the number of units is enormous, leading to a large state-space, we used a DDQN algorithm to obtain the optimal maintenance policy for the system. In the end, we provided a numerical example to evaluate the effectiveness of the proposed model compared to the traditional methods.

Figure 4.10: reward function's value as influenced by varying values of N_1 and N_2.

Chapter 5

Summary and Future Research Direction

In the course of this thesis, we have delved into the intricacies and complexities of maintenance scheduling, underscoring the benefits of utilizing optimization algorithms. As we near the conclusion of our investigation, it is crucial to synthesize our discoveries, acknowledge our contributions, and ponder the future prospects of this domain. This closing chapter aims to furnish a thorough recapitulation of the essential findings and propose potential avenues for future research in the realm of maintenance scheduling.

5.1 Summary of Thesis Contributions

Throughout this research on maintenance scheduling, this thesis has made specific and valuable contributions to the field. This section will provide a clear breakdown of what this thesis has added to the existing knowledge on the topic.

(i)Dynamic Maintenance for a Large Scale Identical Parallel Manufacturing Systems Using Reinforcement Learning : This study proposed an RL-based algorithm for managing large-scale identical parallel manufacturing systems. Unlike other research in this field that primarily focuses on a system's degradation level, our approach considers the number of failed or unhealthy

units. This shift in perspective transforms the problem into a more nuanced and dynamic maintenance challenge, requiring a unique analytical lens and solution strategy. We evaluated the algorithm's efficacy on a three-unit parallel system and observed its effectiveness in managing three components. To address this issue, we employed Q-learning, which has demonstrated superiority over other methods, such as value iteration. We incorporated several terms into our reward function during the decision-making process to optimize the agent's performance.

We formulated our problem based on a set of simplifying assumptions. Simplifying assumptions are essential premises employed to simplify complicated issues, making them more manageable for analysis or solution. For instance, one such premise is that the count of healthy and unhealthy units remains constant during maintenance procedures. After testing our model on a three-component system, we acknowledged its limitations. Therefore, in the subsequent chapter, we relaxed some simplifying assumptions to enhance our model's capacity to handle more intricate systems and yield superior outcomes.

(ii) Deep Reinforcement Learning for Dynamic Maintenance Policies of Large-Scale Parallel Manufacturing Systems: In this study, attention is drawn to developing a robust maintenance decision-making framework designed for multi-unit systems, harnessing ML techniques. DRL is employed, exploring the intricacies of a maintenance model tailored for systems comprising multiple parallel units. These units are exposed to stochastic degradation and unforeseen failures given their operational conditions. This research establishes that individual deterioration is experienced independently by each unit, and this deterioration is represented using a three-state homogeneous Markov process. As a result, at any given instance, a unit is categorized as being in a healthy, unhealthy, or completely failed state. The cumulative state of the entire system is deduced when states of individual units are integrated. The interaction among these system states is meticulously represented using the bivariate Birth/Birth-Death process. In pursuit of cost efficiency, the Markov decision process framework is relied upon to pinpoint the most advantageous maintenance policy. The DDQN algorithm is incorporated, making the RL solution proposed appear more pragmatic and efficient, especially when benchmarked against traditional MDP methods in terms of both time and cost. To showcase the practical implications and robustness of this approach, a numerical example is provided, wherein the potential of RL in identifying the optimal maintenance strategy for the

system in focus is highlighted.

This research modified several original assumptions to capture the system's complexity better. For instance, the number of operational and non-operational units can change during maintenance. If more units become non-operational during maintenance, the current repair action will also address those units.

Compared to the earlier study, there is a significant change in how rewards are determined. Previously, a singular reward structure was employed. Now, this has been divided into two distinct sections. This division was necessary because the subsequent state did not exclusively determine the action taken. The initial section of the reward is calculated based on the number of units that were non-operational at the start. The latter section is influenced by the outcomes of the chosen action, represented by variables n_1 and n_2. This two-part reward system offers a more accurate evaluation and enhances the agent's performance.

One of the standout features of our methodology is the intricate attention to detail we've afforded when constructing our reward function. While the conventional approach in many existing studies has been to zero in on the cost of production as a primary consideration, our approach offers a broader perspective. We've meticulously included diverse factors that can influence the system's performance. This spans from the tangible costs of unmet demand to the financial gains from surplus production. Additionally, we've considered the tangible expenses linked to deploying maintenance crews and the costs associated with operating in two distinct modes. Beyond these, the model also factors production rates, a critical variable in determining system efficiency. This multifaceted approach to structuring our reward function ensures that our agent isn't just making decisions based on a narrow subset of parameters. Instead, it is equipped with a holistic understanding of the system, enabling it to make strategic decisions aligned with optimizing the reward function.

One significant aspect of this research that warrants special attention is the extensive testing on a system comprising 30 identical components functioning concurrently. This represents a notable advancement when viewed in the context of existing literature on the subject. In many prior studies, systems of this size and complexity weren't frequently examined, which often limited their applicability to real-world, large-scale scenarios. This expansion in scale bolsters the relevance of our findings to larger manufacturing setups and sets a benchmark for future research endeavours

in this domain. The ability to handle and derive insights from such a vast system underscores the robustness and adaptability of the methodologies adopted in this study.

5.2 Future Research

Potential avenues for future exploration in this field include:

- In the subsequent phases of research, there is a pressing need to scale up and refine the current model to embrace systems of larger magnitudes. A promising direction would be to engineer a more encompassing algorithm designed with the capability to seamlessly manage systems embedded with 'M' identical units that function side by side in a parallel fashion. Such an expansion not only builds upon the foundational work of this study but also paves the way for broader applications in diverse industrial settings. This evolution would significantly propel the field forward, bridging the gap between theoretical models and complex real-world systems.

- In future studies, it would be advantageous to actualize system designs via regular inspection intervals. Following each inspection period, a meticulous system assessment should be executed. This approach would not only enhance the system's reliability over time but also provide insights into potential areas of improvement, ensuring that the system consistently meets operational standards. Adopting a systematic and iterative evaluation process would strengthen the bridge between theoretical design and practical performance, offering valuable feedback for continual refinement.

- Looking ahead, there is potential to shift from a model that computes immediate rewards to one that gauges the average compensation during subsequent inspections using integral calculations. This would allow a more holistic view of system performance over a given period rather than instantaneous snapshots. Incorporating this method can offer a continuous and cumulative understanding of the system's behaviour, which might be pivotal in refining the maintenance strategies and ensuring sustained operational efficiency. This refined approach would be valuable to future research endeavours, providing depth and accuracy to reward

evaluations.

- In future research endeavours, it would be crucial to delve into the varying timeframes required for distinct maintenance actions. It is pivotal to recognize that each maintenance task, contingent on its intricacy and specifications, might demand different durations. A cursory diagnostic evaluation may be accomplished swiftly, but intricate repairs or component replacements could be time-intensive. These disparities can substantially influence the overall effectiveness of a maintenance strategy, leading to potentially extended downtimes and affecting system productivity. Integrating these time nuances into maintenance planning algorithms will enhance their applicability and efficiency.

References

Ahadi, K., & Sullivan, K. M. (2019). Approximate dynamic programming for selective maintenance in series–parallel systems. *IEEE Transactions on Reliability*, *69*(3), 1147–1164.

Ali, A., & Abdelhadi, A. (2022). Condition-based monitoring and maintenance: state of the art review. *Applied Sciences*, *12*(2), 688.

Álvarez, C., López-Campos, M., Stegmaier, R., Mancilla-David, F., Schurch, R., & Angulo, A. (2019). A condition-based maintenance model including resource constraints on the number of inspections. *IEEE Transactions on Reliability*, *69*(3), 1165–1176.

Amari, S. V., McLaughlin, L., & Pham, H. (2006). Cost-effective condition-based maintenance using markov decision processes. In *Rams'06. annual reliability and maintainability symposium, 2006.* (pp. 464–469).

Ambani, S., Li, L., & Ni, J. (2009, May). Condition-based maintenance decision-making for multiple machine systems. , *131*.

Azar, K., Hajiakhondi-Meybodi, Z., & Naderkhani, F. (2022). Semi-supervised clustering-based method for fault diagnosis and prognosis: A case study. *Reliability Engineering & System Safety*, *222*, 108405.

Azizi, F., & Salari, N. (2023). A novel condition-based maintenance framework for parallel manufacturing systems based on bivariate birth/birth–death processes. *Reliability Engineering & System Safety*, *229*, 108798.

Banerjee, T. P., & Abraham, A. (2018). A support vector machine based approach to real time fault signal classification for high speed bldc motor. In *Intelligent systems design and applications: 17th international conference on intelligent systems design and applications (isda 2017) held*

in delhi, india, december 14-16, 2017 (pp. 836–845).

Barriento, V. F., & Achcar, J. A. (2019). Statistical analysis of equipment maintenance time in the food industry: a case study to identify sources of impact on performance. *INGENIARE-Revista Chilena de Ingeniería*, *27*(1).

Bordoloi, D. J., & Tiwari, R. (2014). Optimum multi-fault classification of gears with integration of evolutionary and svm algorithms. *Mechanism and Machine Theory*, *73*, 49–60.

Chen, H., Liu, H., Chu, X., Liu, Q., & Xue, D. (2021). Anomaly detection and critical scada parameters identification for wind turbines based on lstm-ae neural network. *Renewable Energy*, *172*, 829–840.

Chen, J., & Wang, Y. (2023). A deep reinforcement learning approach for maintenance planning of multi-component systems with complex structure. *Neural Computing and Applications*, *35*(21), 15549–15562.

Chen, Y., Liu, Y., & Xiahou, T. (2021). A deep reinforcement learning approach to dynamic loading strategy of repairable multistate systems. *IEEE Transactions on Reliability*, *71*(1), 484–499.

Coraddu, A., Oneto, L., Ghio, A., Savio, S., Anguita, D., & Figari, M. (2016). Machine learning approaches for improving condition-based maintenance of naval propulsion plants. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, *230*(1), 136–153.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*, 273–297.

de Jonge, B., Teunter, R., & Tinga, T. (2017). The influence of practical factors on the benefits of condition-based maintenance over time-based maintenance. *Reliability engineering & system safety*, *158*, 21–30.

Duffuaa, S. O., Raouf, A., & Campbell, J. D. (1999). Planning and control of maintenance systems. *John Willey and Son, New York*.

Fausing Olesen, J., & Shaker, H. R. (2020). Predictive maintenance for pump systems and thermal power plants: State-of-the-art review, trends and challenges. *Sensors*, *20*(8), 2425.

Feng, Y., Qiu, Y., Crabtree, C. J., Long, H., & Tavner, P. J. (2013). Monitoring wind turbine gearboxes. *Wind energy*, *16*(5), 728–740.

Gryllias, K. C., & Antoniadis, I. A. (2012). A support vector machine approach based on physical

model training for rolling element bearing fault detection in industrial environments. *Engineering Applications of Artificial Intelligence*, *25*(2), 326–344.

Hamer, R., Waterson, P., & Jun, G. T. (2021). Human factors and nuclear safety since 1970–a critical review of the past, present and future. *Safety Science*, *133*, 105021.

Hart, E., Clarke, B., Nicholas, G., Kazemi Amiri, A., Stirling, J., Carroll, J., ... Long, H. (2020). A review of wind turbine main bearings: design, operation, modelling, damage mechanisms and fault detection. *Wind Energy Science*, *5*(1), 105–124.

Hu, Y., Miao, X., Zhang, J., Liu, J., & Pan, E. (2021). Reinforcement learning-driven maintenance strategy: A novel solution for long-term aircraft maintenance decision optimization. *Computers & industrial engineering*, *153*, 107056.

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.

Marugán, A. P. (2023). Applications of reinforcement learning for maintenance of engineering systems: A review. *Advances in Engineering Software*, *183*, 103487.

Nguyen, V. T., Do, P., Voisin, A., et al. (n.d.). Multi-agent deep reinforcement learning-based maintenance optimization for multi-dependent component systems. *Multi-Agent Deep Reinforcement Learning-Based Maintenance Optimization for Multi-Dependent Component Systems*.

Nguyen, V.-T., Do, P., Vosin, A., & Iung, B. (2022). Artificial-intelligence-based maintenance decision-making and optimization for multi-state component systems. *Reliability Engineering & System Safety*, *228*, 108757.

Panagiotidou, S., & Tagaras, G. (2010). Statistical process control and condition-based maintenance: A meaningful relationship through data sharing. *Production and operations management*, *19*(2), 156–171.

Paraschos, P. D., Koulinas, G. K., & Koulouriotis, D. E. (2020). Reinforcement learning for combined production-maintenance and quality control of a manufacturing system with deterioration failures. *Journal of Manufacturing Systems*, *56*, 470–483.

Rasay, H., Azizi, F., Salmani, M., & Naderkhani, F. (2023). A reinforcement learning algorithm for

optimal dynamic policies of joint condition-based maintenance and condition-based production. In *2023 ieee international conference on prognostics and health management (icphm)* (pp. 200–204).

Rasay, H., Naderkhani, F., & Azizi, F. (2022). Opportunistic maintenance integrated model for a two-stage manufacturing process. *The International Journal of Advanced Manufacturing Technology*, *119*(11-12), 8173–8191.

Salari, N., & Makis, V. (2017). Comparison of two maintenance policies for a multi-unit system considering production and demand rates. *International Journal of Production Economics*, *193*, 381–391.

Salmani, M., Azizi, F., Rasay, H., & Naderkhani, F. (2023). Dynamic maintenance for a large scale identical parallel manufacturing systems using reinforcement learning. In *2023 annual reliability and maintainability symposium (rams)* (pp. 1–8).

Song, Z., Zhang, Z., Jiang, Y., & Zhu, J. (2018). Wind turbine health state monitoring based on a bayesian data-driven approach. *Renewable energy*, *125*, 172–181.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tautz-Weinert, J., & Watson, S. J. (2017). Using scada data for wind turbine condition monitoring–a review. *IET Renewable Power Generation*, *11*(4), 382–394.

Tsang, A. H., Jardine, A. K., & Kolodny, H. (1999). Measuring maintenance performance: a holistic approach. *International Journal of Operations & Production Management*, *19*(7), 691–715.

Valet, A., Altenmüller, T., Waschneck, B., May, M. C., Kuhnle, A., & Lanza, G. (2022). Opportunistic maintenance scheduling with deep reinforcement learning. *Journal of Manufacturing Systems*, *64*, 518–534.

Wang, H. (2002). A survey of maintenance policies of deteriorating systems. *European journal of operational research*, *139*(3), 469–489.

Wei, G., Zhao, X., He, S., & He, Z. (2019). Reliability modeling with condition-based maintenance for binary-state deteriorating systems considering zoned shock effects. *Computers & Industrial Engineering*, *130*, 282–297.

Widodo, A., & Yang, B.-S. (2007). Support vector machine in machine condition monitoring and fault diagnosis. *Mechanical systems and signal processing*, *21*(6), 2560–2574.

Wiering, M. A., & Van Otterlo, M. (2012). Reinforcement learning. *Adaptation, learning, and optimization*, *12*(3), 729.

Xiang, L., Yang, X., Hu, A., Su, H., & Wang, P. (2022). Condition monitoring and anomaly detection of wind turbine based on cascaded and bidirectional deep learning networks. *Applied Energy*, *305*, 117925.

Xiao, H., Yan, Y., Kou, G., & Wu, S. (2021). Optimal inspection policy for a single-unit system considering two failure modes and production wait time. *IEEE Transactions on Reliability*.

Yousefi, N., Tsianikas, S., & Coit, D. W. (2022). Dynamic maintenance model for a repairable multi-component system using deep reinforcement learning. *Quality Engineering*, *34*(1), 16–35.

Zhang, N., & Si, W. (2020). Deep reinforcement learning for condition-based maintenance planning of multi-component systems under dependent competing risks. *Reliability Engineering & System Safety*, *203*, 107094.

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., & Gao, R. X. (2019). Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, *115*, 213–237.

Zhao, X., & Wang, Z. (2022). Maintenance policies for two-unit balanced systems subject to degradation. *IEEE Transactions on Reliability*, *71*(2), 1116–1126.