Università degli Studi di Napoli Federico II
Ph.D. Program in
Industrial Product and Process Engineering
XXXVI Cycle

Thesis for the Degree of Doctor of Philosophy

Advancement in Operations Management through Deep Reinforcement Learning and Simulation

by
Maria Grazia Marchesano

Advisor: Prof. Guido Guizzi

Scuola Politecnica e delle Scienze di Base
Dipartimento di Ingegneria Chimica, dei Materialei e della Produzione Industriale

Advancement in Operations Management through Deep Reinforcement Learning and Simulation

Ph.D. Thesis presented
for the fulfillment of the Degree of Doctor of Philosophy
in Industrial Product and Process Engineering
by

Maria Grazia Marchesano

Approved as to style and content by

_____

Prof. Guido Guizzi, Advisor

_____

Università degli Studi di Napoli Federico II

Ph.D. Program in Industrial Product and Process Engineering
XXXVI cycle - Chairman: Prof. Andrea D'Anna

Candidate's declaration

I hereby declare that this thesis submitted to obtain the academic degree of Philosophiæ Doctor (Ph.D.) in Industrial Product and Process Engineering is my own unaided work, that I have not used other than the sources indicated, and that all direct and indirect sources are acknowledged as references.
Parts of this dissertation have been published in international journals and/or conference articles (see list of the author's publications at the end of the thesis).

Napoli, December 13, 2023

_____

Maria Grazia Marchesano

Abstract

The profound impact of artificial intelligence and advanced simulations has transformed various aspects of operations management, addressing complexities and optimizing processes in contemporary manufacturing and energy landscapes. This comprehensive research delves into three primary domains: production control, maintenance scheduling, and the integration of Battery Swapping Station (BSS) into smart microgrids.

In the contest of production control, the study highlights the imperative nature of effectively coordinating material flow, resources, and processes, with a keen focus on Work in Process (WIP) control. A groundbreaking technique has been introduced that integrates Reinforcement Learning and Industry 4.0 technologies, aiming to regulate throughput in a Flow Shop setting and dynamically define sequencing rules. By employing a deep Q-network (DQN), the study achieves a targeted throughput while maintaining constant WIP. A central aspect of this work is the adaptability of the proposed "intelligent" tool, which adjusts its decision-making contingent on the dynamic changes in a production environment.

Shifting focus to maintenance scheduling in Flow Shop production lines, Deep Reinforcement Learning (DRL) emerges as a game-changer. The research proposes an innovative integration of DRL and simulation tools to optimize maintenance tasks scheduling, harnessing the predictive power of simulations and the adaptive capabilities of DRL. This approach's uniqueness lies in training on a single machine and testing the resulting policy across varying machine configurations, demonstrating its robustness in diverse experimental settings.

Lastly, the research addresses the potential of integrating Battery Swapping Station (BSS) within a smart microgrid. A detailed simulation model, acting as a digital twin of the microgrid, is developed, capturing the nuances of renewable energy sources, second-life battery storage, and utilities.

The study meticulously evaluates the economic viability of this integration, emphasizing cost optimization and effective energy demand management. Notably, the initiative promotes the circular economy by championing the utilization of second-life batteries.

In conclusion, this research accentuates the transformative potential of AI-driven techniques and advanced simulations in redefining operations management paradigms across manufacturing and energy sectors.

Keywords: Operations Management, Deep Reinforcement Learning (DRL), Flow Shop, Maintenance Scheduling, Battery Swapping Station, Smart Microgrid.

Acknowledgements

The author's work has been supported by a PhD scholarship funded by Tecnologica S.r.L.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the contemporary era of digitization, Operations Management stands at the crossroads of innovation and efficiency. As industries strive to optimize their operational processes, the role of Artificial Intelligence (AI) and, more specifically, Machine Learning techniques, emerges as pivotal in propelling businesses to new heights [2]. Deep Reinforcement Learning (DRL), an advanced subset of Machine Learning, takes in the essence of decision-making by enabling models to learn optimal strategies through trial and error, all while interacting with a given environment [3]. This self-guided learning mechanism, when synergized with simulation, can revolutionize the way industries approach problem-solving, resource allocation, and demand forecasting, as well as other operations [4].

The vast spectrum of Operations Management, encompassing areas like supply chain optimization, production planning, and inventory control, requires sophisticated tools that can adapt, learn, and provide actionable insights [5]. Traditional mathematical models and heuristic methods formed the backbone of decision-making in Operations Management. However, the dynamism and complexity of modern operations, influenced by global market trends, intricate supply chains, and rapidly evolving consumer preferences, necessitate a paradigm shift [6]. This is where Deep Reinforcement Learning, with its ability to process vast datasets and adapt in real-time, promises transformative potential [7].

Simulation, a tool that has long been employed to mirror real-world processes and predict outcomes, offers a sandbox environment for DRL

models. Within these virtual confines, DRL algorithms can experiment, learn, and fine-tune strategies without the risk of real-world repercussions. This combination of DRL and simulation provides a robust framework, allowing businesses to test hypotheses, explore new operational strategies, and predict the outcomes of changes, all before actual implementation [8]. When fused with simulations, DRL can undergo training across diverse scenarios, equipping it to make well-informed decisions in real-world contexts. This dual approach optimizes intricate processes without intensive manual tuning and ensures adaptability to evolving environments and constraints, leading to heightened efficiency and reduced operational costs [9]. This thesis delves into the confluence of Deep Reinforcement Learning and simulation, exploring their transformative impact on Operations Management. Through comprehensive analysis and real-world case studies, this work aims to shed light on the potential of this synergy, charting a path for businesses pursuing operational excellence in the digital age.

1.1 Scientific Backgroud

The three works presented in the thesis share a common thread of leveraging cutting-edge technologies—specifically elements of Industry 4.0 such as Simulation and Artificial Intelligence (AI), with an emphasis on Deep Reinforcement Learning (DRL)—to revolutionize operations management within diverse systems. All three studies focus on creating adaptive systems capable of responding to changes and uncertainties in real-time, critical for both production lines and energy systems. The first two works integrates AI (particularly DRL) with sophisticated simulation models, emphasizing their combined strength in enhancing decision-making processes in complex, dynamic environments. Regarding the third work the decision-making process is given to an intelligent system composed of numerous algorithms. The central theme across all studies is optimization—whether of production sequences, maintenance schedules, or energy consumption and storage—to improve efficiency, cost-effectiveness, and responsiveness to environmental changes. The scientific background underlying this thesis is briefly presented here, starting with work in process control and job sequencing, planning, and scheduling of maintenance tasks in various scenarios, and finally, some considerations related to the performance eval-

uation of integrating a Battery Swapping Station into a smart microgrid.

1.1.1 Reinforcement Learning

In the contemporary landscape of Industry 4.0, the intricacies of modern manufacturing and logistics systems have become notably dynamic [10]. This dynamism is shaped by interconnected devices, advanced automation, fluctuating customer demands, unforeseen disruptions, and the need for real-time updates. One of the pressing challenges is addressing fluctuating demand patterns, which can significantly impact production schedules. Furthermore, there's an imperative need for real-time adjustments to unforeseen events such as machine breakdowns or material shortages [11]. This is where the application of AI in Operations Management becomes central. Specifically, RL which learns optimal strategies via trial and error, shows promise. Reinforcement Learning (RL) draws inspiration from a variety of other well-known fields that study decision-making under conditions of uncertainty. In Reinforcement Learning (RL), the problem to address is described as a Markov Decision Process (MDP) [7]. MDP is a mathematical framework to describe an environment in reinforcement learning. The RL consists of two main elements: the agent and the environment. The former interacts with the least. The environment reacts to and rewards the seen or state-like impacts of these activities. These two components are in constant communication, with the agent attempting to affect the environment by its behaviour and the environment reacting to the agent's activities. The agent picks an action and observes what happens in the environment after it takes the action. Then, it obtains a reward in correlation with the action and the state. The agent repeats the encounter numerous times and learns what action is ideal at each state [12]. The environment's response to a specific activity is determined by a model that the agent may or may not be aware of. There are various ways to develop policies to solve tasks using deep reinforcement learning algorithms, each with its own set of benefits. There are many algorithms with which the learning agent can be trained to solve a certain problem, at the top stage, a distinction exists between model-based and model-free (Figure 1.1).

Model-based learning tries to model the environment and then determine the best policy based on the model it has learnt. In model-free learn-

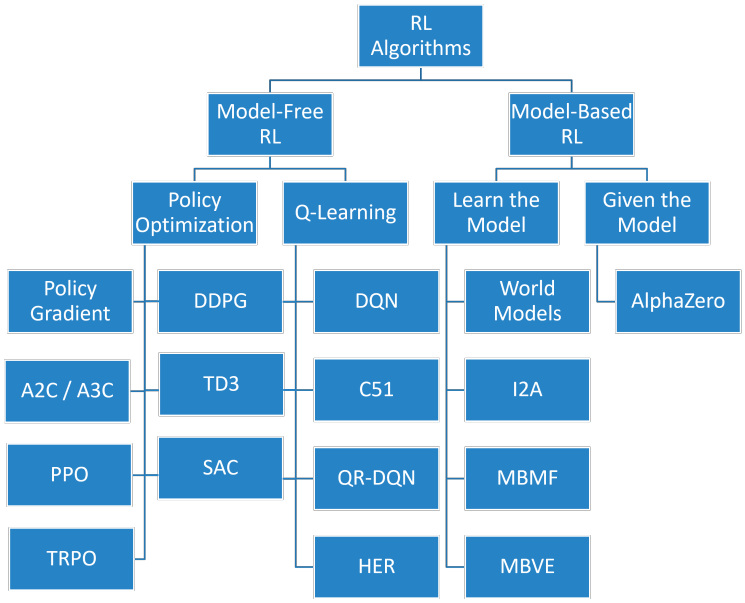

Figure 1.1. A Taxonomy of Reinforcement Learning Algorithms

ing the agent uses trial-and-error experience to draw up the best policy [7]. The approach used in this thesis is the second one considering the major scope of developing a general framework for a manufacturing planning and control system. One of the challenges of working with DRL is determining how to model the state space, action space, and reward function to properly shape the problem and choose the network hyperparameters to overcome overfitting. The learning agent receives an observation of the system's state as input and then calculates a reward based on which the system will evolve to different states to maximize the overall reward. So, in this type of problem, the question is how to represent the state, the reward, and the set of alternative actions to best depict the real system. Starting with the broader concept, this section introduces reinforcement learning, setting the foundation as a potent tool in complex decision-making scenarios. It hints at the vast potential applications ranging from production lines to energy systems. As a pivotal technology, reinforcement learning bridges the gap between theoretical aspirations and practical achievements, making it a focal point in the contemporary scientific background.

1.1.2 Flow Shop Scheduling

Referring to a manufacturing environment, the operations performed are typically known as jobs, related to the items being produced. These items can often be broken down into several tasks (or operations), each representing the smallest unit of work considered practical. In many manufacturing and assembly plants, these operations frequently need to follow the same sequence for all jobs, implying that each job must follow the same path. The machines are assumed to be connected in series in such instances, defining the environment as a Flow Shop [13].

Flow Shop Scheduling, a specific instance of Job Shop Scheduling, in which all operations are executed in a predetermined order across all jobs. Each job passes through the machines in a fixed sequence: the first operation occurs on the first machine, the second operation on the second machine (upon completion of the first), and so forth until the job is finished. The problem has seen numerous variants over the years, each prompting the development of both exact and approximate resolution algorithms [14].

However, most Flow Shop Scheduling studies have focused on deterministic cases, where all parameters, like production times, machine availability, and transportation durations, are known and consistent. In reality, several uncertainties often disrupt this predictability, affecting decision-making processes. Examples include machine failures, operator absences, stock shortages, or changes in job priority, all of which can significantly affect industrial operations [15]. In stochastic contexts, traditional assumptions for deterministic Flow Shops undergo alterations: job release dates are unpredictable, machines may malfunction, and processing times become independent random variables. The scheduling and production control problem has been handled in a variety of ways, employing a range of frameworks. The dynamic nature of flow shop environments calls for more than conventional management strategies, leading to the exploration of artificial intelligence's role in transforming these systems.

1.1.3 The Dynamics of Production Line Management

Considering the complexity of production line management, explicitly focusing on the dynamics of Work-In-Progress (WIP), throughput (TH), and cycle time (CT), interconnected through Little's Law. This law, foun-

dational in queueing theory and operations management, indicates that the product of throughput and cycle time is equal to the work in progress: $WIP = TH \times CT$. Its practical application spans individual workstations to entire production plants, given the consistency in the measurement units of WIP, TH, and CT.

The study further explores the theory in the context of CONWIP (CONstant Work In Progress) production lines, a production control system, as examined by Hopp and Spearman [1]. It considers the implications under varying operational scenarios, highlighting how performance extremes (best and worst cases) can manifest in a production line.

In the best-case scenario, the research outlines how the minimum cycle time and maximum throughput are dependent on the WIP level and certain thresholds (r_b bottleneck rate that is the rate of the workstation with the highest long-term utilization (parts per unit time or jobs per unit time); T_0 raw process time that is the sum of the long-term average process times of each workstation; W_0 critical WIP is the WIP level at which a line with given r_b and T_0 values but no variability achieves maximum throughput (r_b) with the shortest cycle time (T_0)), leading to the insight that a zero-inventory goal is impractical. Even under ideal conditions, no inventory would mean no throughput, hence zero productivity. Instead, a critical WIP (W_0) is suggested as a more realistic optimal inventory level.

The worst-case analysis, conversely, identifies conditions under which a production line would experience maximum cycle time and minimum throughput, considering both processing and waiting times. The findings underscore that even predictable process duration, if varied, can result in severe operational inefficiencies.

Furthermore, the thesis introduces the Practical Worst Case (PWC), which is an intermediary scenario incorporating maximum randomness, making it a more representative benchmark for real-world systems. It defines this case within the framework of a system state and applies conditions like balanced lines, single-machine stations, and exponential distribution of process times. The PWC scenario provides insights into the operational behaviour at extremely low and high WIP levels, emphasizing that achieving near-capacity throughput in highly variable systems requires maintaining high WIP levels, which consequently leads to longer cycle times.

BestCase:

$$CT_{min} = \begin{cases} T_0, & if \quad w < W_0 \\ \dfrac{w}{r_b}, & otherwise \end{cases} \tag{1.1}$$

$$TH_{max} = \begin{cases} \dfrac{w}{T_0}, & if \quad w < W_0 \\ r_b, & otherwise \end{cases} \tag{1.2}$$

WorstCase:

$$CT_{max} = w\dot{T}_0; \tag{1.3}$$

$$TH_{min} = \frac{1}{T_0} \tag{1.4}$$

PWC:

$$CT_{PWC} = T_0 + \frac{w-1}{r_b}; \tag{1.5}$$

$$TH_{PWC} = \frac{W\, r_b}{W_0 + W - 1} \tag{1.6}$$

This analysis bridges theoretical extremes and real-world production scenarios, establishing that systems performing better than the PWC are lean, while those performing worse may have significant inefficiencies (Figures 1.2 and 1.3). The study proposes utilizing these insights for an internal benchmarking methodology. This section underscores the operational obstacles encountered in production line management. It prepares the ground for introducing advanced technological interventions that could redefine traditional approaches. Artificial intelligence emerges as a beacon of hope, promising unprecedented improvements in the realm of production control and beyond.

1.1.4 Embracing Artificial Intelligence in Production Control

Many researchers recommend the Machine Learning, Deep Learning, or Reinforcement Learning technique to solve difficulties that may arise in production control. In [16] H. Wang et al., 2020 to address the uncertainty of the production environment in the assembly job shop, a RL algorithm called dual Q-learning, is proposed to enhance the adaptability to envi-

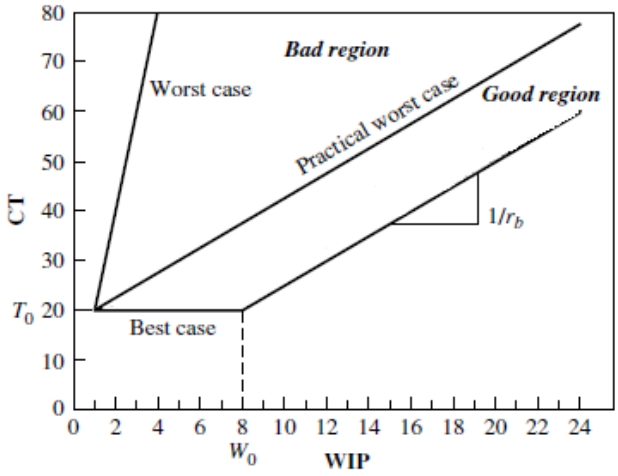

Figure 1.2. Cycle time versus WIP [1]

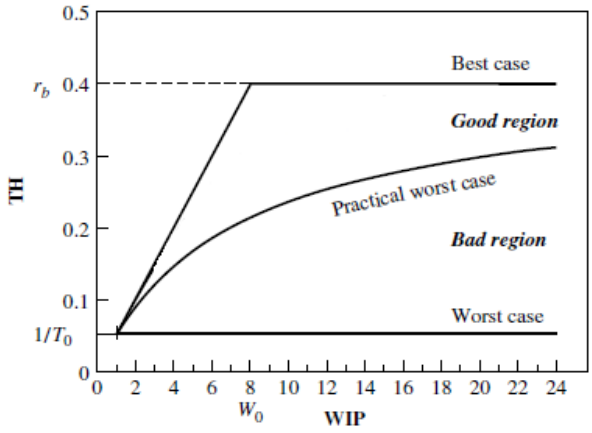

Figure 1.3. Throughput time versus WIP [1]

ronmental changes by self-learning. Moreover, in [17] Wu et al., 2020 a combination of deep neural network (DNN) and Markov decision processes (MDP) is proposed for the dynamic dispatching of re-entrant production systems. Also, in the chemical production scheduling process Hubbs et al., 2020 apply DRL to account for uncertainty and achieve online, dynamic scheduling [18]. Furthermore, to demonstrate the validity of the approaches proposed, there is a comparison with well-known heuristics or analytic models. This happens also in [19] in which the performance of the ML-based production control is compared to a static rule-based approach. There are also works in which the main contribution is the development of a four-level collaborative RL algorithm which provides a schedule for two non-identical robots for non-identical machines [20]. To reduce makespan, that is the length of time that elapses from the start of work to the end, in [21] there is the employment of a reinforcement learning (RL) technique to develop efficient robot task sequences. The Petri net model is then used to describe states, actions, and rewards. In production planning and control one of the key tasks is of course the scheduling of tasks and operations. Many methods for the problem of scheduling and production optimum control have been proposed in the literature [22]. To schedule in production systems many dispatching rules are been proposed, but many rules might apply to different situations, so, it can be difficult for the decision-maker to choose the right rule at any particular time. Furthermore, dispatch rules are confined to their local information frontiers, therefore no rule outperforms others due to differing system goals, situations, and conditions [23]. The methods and algorithms of ML, DL and RL can be used to address challenges that may arise when dealing with situations in which there is no comprehensive understanding of the system dynamics [24]. Combining the potential of the neural networks and the RL has developed the DRL that blends artificial neural networks with a reinforcement learning framework to assist software agents in learning how to accomplish their goals. It merges function approximation and goal optimization to map states and actions to the rewards they occur in. The DRL is the appropriate approach for creating a self-optimization scheduling strategy that can be employed with the accurate simulation and high-performance data supplied by a simulation tool. Because of its qualities, DRL has lately been explored and applied in manufacturing systems to handle decision-making difficul-

ties that might be problematic in today's complex and dynamic industrial systems environment [25]. In this research, the theoretical implication of the work of [1] will be used in the application of a Deep Q-Network control mechanism in a Flow Shop manufacturing environment. The objective is to meet specified throughput targets while adhering to the CONWIP principle of constant WIP, utilizing the PWC as a performance benchmark. The task now is to provide a comprehensive methodology that will lead researchers and practitioners through the application and optimization process, provide deployment instructions, and accelerate implementation in prospective use cases [26]. This transformation extends to the realm of maintenance scheduling, where AI's analytical and predictive skill unveils new horizons.

1.1.5 Maintenance Scheduling and Artificial Intelligence

Maintenance management is the key factor in maximizing the value of manufacturing assets, including productivity, reliability, and cost effectiveness. So this crucial function is responsible for planning, implementing, controlling, and enhancing maintenance activities within an organization. Maintenance scheduling determines when the planned work is to take place, who is to perform it. The goals of maintenance scheduling are to maximize the amount of work completed using available resources, to prioritize the most important work orders, to complete as many preventive maintenance tasks as possible, and to minimize the use of external resources by effectively using internal labor. When maintenance planning and scheduling are implemented together, they can offer significant benefits to an organization. These include controlling maintenance-related resources, reducing equipment downtime, reducing spare parts inventory, improving workflow, and increasing efficiency by reducing the movement of resources between areas. Maintenance can be seen as a job scheduling problem. However, it turns out that it violates the necessary conditions for the use of classical scheduling theory and so maintenance scheduling is even more complex than job shop scheduling [27]. So, this study concentrates on potential machine failures within the Flow Shop. To mitigate these, scheduled maintenance is essential; however, such actions necessitate production halts, impacting overall productivity. This dilemma is often referred to in the literature as the "joint maintenance and production scheduling prob-

lem" or "scheduling with availability constraints" [28]. Various approaches and algorithms exist to solve these scheduling problems. This thesis proposes employing AI, combined with simulation, to optimize maintenance scheduling. The manufacturing sector is on the point of transformative innovation, driven by increasing sensor integration, the Internet of Things (IoT), advancements in robotics, and an abundance of data. This surge in factory digitization compels manufacturers to reassess their existing operations and future strategies in the advent of Smart Manufacturing and Industry 4.0. AI stands poised to be instrumental in realizing these revolutionary prospects [29]. Among the tools facilitating the transition to Smart Manufacturing, such as Cloud Computing, Big Data Analytics, and 5G, Machine Learning (ML) holds a significant position [30, 31]. This work utilizes and elaborates on Reinforcement Learning. Reinforcement Learning deals with sequential decision-making, exploring how agents should perform in environments to maximize cumulative rewards. Notably, the agent learns which actions to execute without explicit guidance, instead discerning which strategies maximize its rewards through continuous interaction with its environment [32]. A specific focus of this work is Deep Reinforcement Learning, merging Reinforcement Learning with Artificial Neural Networks with extensive layers and parameters. This study implements Proximal Policy Optimization (PPO), a policy gradient method characterized by its unique clipped probability ratios forming a conservative performance estimate for the policy [33]. The aim is to train an agent capable of efficiently scheduling preventive maintenance and minimizing machine failure during processing stages. This objective was pursued by implementing a neural network with two 64-neuron hidden layers, utilizing the Python library Stable Baselines3. The neural network inputs include the probability of machine failure at the onset of each job and the time elapsed since the last maintenance. After training is completed on a single machine, the developed policy is then tested in a 5-machine flow shop environment, considering various scenarios. For the experimental phase, a Flow Shop model was constructed in Anylogic simulation software, giving the opportunity to consider different Flow Shop Scheduling problem configurations. The experiments also leveraged solutions from existing literature to enhance scientific rigour, incorporating solutions derived from [34]. In conclusion, various experiments under different Flow Shop Scheduling con-

figurations were conducted to validate the results. Notably, in all analyzed instances, AI significantly reduced machine failures by autonomously and adeptly scheduling preventive maintenance, based on real-time machine parameters. This approach demonstrated considerable promise in enhancing industrial reliability and efficiency in the face of uncertainties. This section reveals how AI revolutionizes maintenance scheduling, embodying adaptability, and foresight. Beyond production environments, these advanced scheduling techniques exhibit their versatility and impact, extending to the dynamic field of energy and transportation.

1.1.6 Simulation Approaches in Modern Energy and Transportation Systems

A transformative shift is happening also in electricity production and consumption, driven by the imperative to combat climate change and enhance energy system efficiency [35]. In this broader context, the scientific community has proposed and developed various strategies and methodologies to reduce humanity's environmental footprint, especially in the realms of transportation and energy [36]. Recent years have witnessed a significant increment in the adoption of electric vehicles (EVs). However, certain challenges continue to keep their widespread integration, and one prominent challenge is the protracted recharging times [37]. A potential remedy for this issue comes in the form of a Battery Swapping Station (BSS) [38]. These stations enable users to swiftly exchange depleted or near-depleted batteries for fully charged ones. They dramatically reduce recharging times to just a few minutes and enable efficient scheduling of recharging for depleted batteries. Consequently, the integration of BSS can substantially cut greenhouse gas emissions. To maximize these benefits, it is imperative to integrate BSS into intelligent electrical systems like smart microgrids, heavily reliant on renewable energy sources [39]. This integration can elevate transport system sustainability by harnessing renewable power generation and easing efficient energy management. Managing BSS entails two crucial aspects: individual-level management to ensure efficient use of the battery fleet in terms of recharging and decommissioning, and management of interactions with the electricity grid and other associated elements. In the existing literature, considerable attention has been dedicated to achieve the best scheduling and coordination in microgrids with

BSS [40]. A fundamental aspect to consider when integrating BSS into a microgrid is the management of energy flows throughout the system. The existing literature emphasizes the importance of integrating components such as BSS, Renewable Energy Sources, and storage units within microgrids [41].

1.2 Research outline and contributions

This thesis is structured in the form of a "three-paper" thesis, with published and forthcoming scientific papers to which I significantly contributed. The first paper (Chapter 2) delves into the complexities of Work in Process (WIP) Control, presenting methodologies and findings that advance our understanding of efficient job sequencing in a Flow Shop. The second paper (Chapter 3) picks up the baton from the first and aims to conduct a comprehensive study on the planning and scheduling of maintenance activities in a Flow Shop, ensuring optimal operations and minimal disruptions. The third paper (Chapter 4), instead, focuses on the innovative approach of integrating a Battery Swapping Station into a smart microgrid, exploring its potential benefits and challenges in modern energy systems. In the next sections, there is a much more detailed description of each research work conducted.

1.2.1 Work in process (WIP) control and job sequencing in a Flow Shop

This work presents a research that combines advancements in artificial intelligence (AI) with engineering applications, specifically targeting work in process (WIP) control within a production system and dynamically determining sequencing rules for individual machines. To achieve this, the research proposes innovative approaches that leverage Reinforcement Learning techniques and algorithms inspired by Industry 4.0 technologies. The main emphasis is on regulating throughput and WIP in a Flow Shop setting. To attain a predefined throughput objective, the study employs a deep Q-network (DQN) to determine a constant WIP amount for the system. Furthermore, a significant objective of this study is to develop an "intelligent" tool that can adapt its decision-making process in response

to changes in the production line's status. This adaptability is crucial in dealing with the dynamic nature of modern manufacturing environments. In this paper, a novel technique is introduced to define the state, action space, and reward function for the Reinforcement Learning agent. These components play a crucial role in shaping the agent's learning process and influence its decision-making. The defined state, action space, and reward function are notable contributions of this research, as they provide a solid foundation for effectively applying AI-driven control in the production system.

This paper was co-authored by Silvestro Vespoli (University of Naples Federico II), Guido Guizzi (University of Naples Federico II) and Liberatina Carmela Santillo (University of Naples Federico II), under review in *Engineering Applications of Artificial Intelligence*.

1.2.2 Planning and scheduling of maintenance activities in a Flow Shop under different scenarios

The paper puts forth a combined approach of a simulation tool and a DRL algorithm designed to streamline the planning of maintenance activities in a Flow Shop production sequence. The primary objective is to perfect maintenance schedules while augmenting productivity, achieved by merging the proficiency of simulation and DRL-driven smart decision-making. This advanced simulation tool mirrors the Flow Shop production line digitally, facilitating an accurate representation and emulation of machinery functions, task sequences, and maintenance activities. What distinguishes this methodology is that the initial learning phase is conducted on a single machine. Subsequently, the devised policy undergoes testing on a Flow Shop line, both on machines with identical Weibull parameters (α and β) and on those with different parameters. This combined approach of the simulation tool and DRL paves the way for an adept system to schedule and strategize maintenance tasks in a Flow Shop production chain. Merging the utility of simulations with DRL-informed decisions, this method promises a combined strategy to refine maintenance methods and boost production outcomes across various tested scenarios.

This paper was co-authored by Silvestro Vespoli (University of Naples Federico II), Guido Guizzi (University of Naples Federico II) and Liberatina Carmela Santillo (University of Naples Federico II), is ready to be

submitted to *Applied Intelligence*.

1.2.3 Performance evaluation of integrating a Battery Swapping Station into a smart microgrid

The integration of Battery Swapping Stations (BSSs) into smart microgrids presents an opportunity to optimize energy generation, storage, and consumption. However, there exists a gap in the literature regarding the detailed analysis of the profitability of integrating a BSS within a smart microgrid, particularly utilizing second-life batteries for storage and renewable energy sources for generation. This study aims to address this gap by employing a multi-method simulation approach to thoroughly investigate the economic viability of such integration. The simulation model developed for this study is a digital twin of the microgrid, incorporating components such as the BSS, renewable energy sources (wind and photovoltaic), second-life battery storage, and utilities. By optimizing energy flows within this model, considering the cost-effectiveness of diverse generation sources and prioritizing the utilization of renewable energy, it aims to provide a comprehensive assessment of the economic benefits. Furthermore, the simulation takes into account crucial factors including battery swapping operations, warehouse management, and battery charging scheduling. The profitability analysis undertaken in this study is grounded in the objective of minimizing total costs while effectively meeting the energy demands of residential loads. Ultimately, the integration of the BSS into the smart microgrid not only targets economic efficiency but also strives to maximize the utilization of second-life batteries, contributing to the concept of a circular economy.

This paper was co-authored by Silvestro Vespoli (University of Naples Federico II), Guido Guizzi (University of Naples Federico II) and Gabriella Ferruzzi (ENEA Italian National Agency for New Technologies, Energy and Sustainable Economic Development), pubblished in *Energies* [42].

Chapter 2

Enhancing Production Line Efficiency through Work In Process Control and Scheduling using Deep Reinforcement Learning

2.1 Introduction

Managing the coordination and regulation of materials, resources, and processes in manufacturing operations, known as production control, is a critical component in ensuring efficient and effective production [43]. Traditional production control methods, based on fixed rules and heuristics, often fall short when addressing the complexities and dynamics of modern production systems [10]. The inflexibility of these legacy approaches can lead to sub-optimal resource allocation and scheduling decisions; a salient issue considering the highly customisable and rapidly evolving landscape of the modern manufacturing sector [5].

The onset of Industry 4.0, notably the incorporation of Artificial Intelligence (AI), offers a promising solution to these challenges. Specifically, Machine Learning (ML) methodologies and algorithms have been high-

lighted in several studies as advantageous alternatives to traditional production control techniques, enabling more adaptive and efficient management [44, 45, 46]. These approaches introduce new capabilities for planning and control systems to address production scheduling and control issues [47, 19, 48, 49]. Furthermore, it has been shown that the integration of such technologies into production systems can yield considerable benefits [50, 51, 52].

One such ML methodology that shows significant promise is Reinforcement Learning (RL), particularly Deep Reinforcement Learning (DRL). DRL combines RL with deep neural networks, enabling the algorithm to learn from experience and make decisions based on predefined objectives [4, 53]. Over time, a DRL algorithm can become increasingly proficient at decision-making and achieving specified goals, improving production processes and reducing costs [54].

DRL has been already applied across various aspects of production systems, from production control to scheduling and maintenance [55, 56]. It has also demonstrated capabilities in addressing the intricate decision-making challenges present in modern dynamic manufacturing environments [25, 57, 26]. However, despite these advances, the application of DRL for production control concerning Work-In-Process (WIP) levels remains an understudied area in existing literature [58, 59].

As a matter of fact, effective management of Work-in-Process (WIP) is critical to minimize disruptions and ensure smooth production in dynamic environments [60, 61]. Controlling WIP levels to align with desired throughput can optimize the production process, maximizing output while minimizing costs [62, 63]. However, decision-making can be complex and time-consuming, especially considering fluctuating market demands and system dynamics [64, 2, 65, 12].

This work aims to address this research gap by designing a Deep Q-Network (DQN) agent, a DRL approach, to effectively regulate WIP levels and sequencing in a dynamic and complex production system. By dynamically adjusting WIP in response to desired throughput, our method provides a novel solution for optimizing production control and scheduling. We also propose a dispatching rule selection method contingent on system status to further enhance adaptability and efficiency.

Summarising, this paper addresses two key research questions:

1. How can a learning agent, implemented with a Deep Q-Network (DQN), effectively regulate Work-in-process (WIP) and sequencing in a complex and dynamic production system, considering the challenges of the variability of the currently manufacturing setting?

2. What benefits does integrating a Digital Twin with Deep Reinforcement Learning (DRL) offer in optimizing production processes and enhancing decision-making capabilities, particularly regarding real-time data utilization and adaptability to environmental changes?

By investigating these questions, this work presents a novel approach to address the challenges of managing modern production systems. This study leverages DRL to optimize production control and scheduling, contributing to this academic field while also providing practical tools for more efficient and adaptive production management aligned with Industry 4.0 paradigms [1, 6, 18, 66]. Our approach offers a valuable contribution to the literature along with practical implications for modern production facilities progressing towards Industry 4.0. The paper is structured as follows: Section 2 presents the problem formulation, Section 3 describes the proposed technique in detail, Section 4 discusses the simulation environment and the experimental plan carried out to support the proposal and the results of different configurations, and finally, Section 5 concludes the study.

2.2 Problem statement

The goal of the work is to use an artificial intelligence system for decision-making within a production system for manufacturing planning and control, which draws up a plan capable of adapting decisions to contingent situations through a trial and error process together with a mechanism of rewards and punishments. A dependable simulated environment is desirable for any Reinforcement Learning system. This is best accomplished with the assistance of reliable general-purpose simulation software with quick, consistent, and simplified links to RL algorithms.

The reinforcement learning training process is based on an artificial explorer that investigates every aspect of a simulation world. With the

right reward configuration, this method could be used to partially automate some typically repetitive portions of the verification and validation process, allowing for more extensive testing of the robustness and realism of the simulation model. Standardized RL environments are available to academics so that they can test and evaluate their algorithms on level playing fields. These commonly used environments, however, lack the variety and complexity found in real-world simulated systems. Any industry or situation can benefit from advanced training settings on a general-purpose simulation platform, which can be quickly adapted while also delivering varying levels of complexity and problems.

The purpose of this research is to investigate the use of reinforcement learning in industrial production, particularly production scheduling and control. The study will look into two aspects of RL implementation: (1) using RL for production line control to maintain a constant level of work-in-process while achieving specified performance results, and (2) using RL for dynamic production scheduling by selecting appropriate rules based on downstream conditions.

To accomplish these objectives, the research needs a dependable simulated world that can provide a quick and consistent link to RL algorithms. In figure 2.1 there is the step-by-step graphical representation of the algorithm proposed in which there is a first part that involves the steps to follow in order to train an RL system to control the production (WIP amount) and the second part that involves the scheduling issue (rule to be chosen). The learning agent is to be taught the amount of WIP that must be kept constant to achieve specified results. The second topic to be investigated is the usage of RL for dynamic production scheduling. To this aim, the learning agent has to choose the most appropriate rule based on the condition of the production line downstream of the choice of the WIP to keep constant, the output of the prior training. The study also seeks to create a mechanism of rewards and punishments for the RL system to adapt decisions to contingent circumstances.

2.2.1 Theoretical background

In literature *Hopp and Spearman* analyzed the behavior of CONWIP (CONstant Work In process) lines working in a different scenarios [1]. In particular, they proposed the Practical Worst-Case (PWC) model which

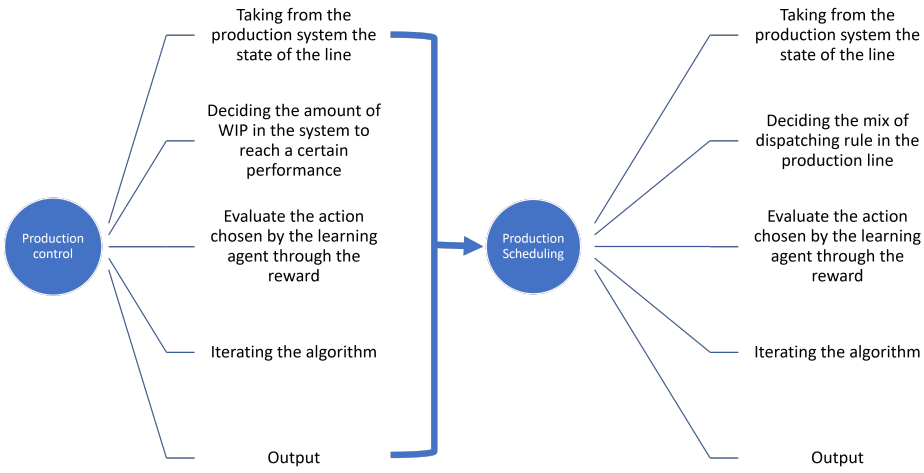

Figure 2.1. The step-by-step graphical representation of the algorithm proposed

garnered significant empirical support and has been widely implemented in diverse production settings [67]. Keeping the WIP controlled has demonstrated efficacy in mitigating the effects of fluctuations and interruptions. In particular, they deduced the relationships that link the Throughput and Work in Process as shown in Table 2.1, in particular with T_0 is the Raw Processing Time of the line (the sum of long-term average process time of each workstation); r_b represents the Bottleneck Rate of the line (it is the rate of the workstation that has the highest long-term utilization); W_0 represents the Critical WIP of the line (it is the WIP level for which a line, with a defined Raw Processing Time and Bottleneck Rate, achieve the maximum throughput and the minimum cycle time without any variability) and finally w represent the current WIP of the line.

The choice to employ the approach of the Practical Worst Case in this study allows us to evaluate the performance of the system under uncertainty. This is important because, even if there are other models available for production line management [68], they are often not utilized to regulate Work-In-Process (WIP) but they use these approaches to meet the due date constraints.

Table 2.1. Relationship in "Factory Physics"

Performance Scenario	CycleTime	Throughput
BestCase	$CT_{min} = T_0$ if $w < W_0$ $CT_{min} = \frac{w}{r_b}$ otherwise	$TH_{max} = \frac{w}{T_0}$ if $w < W_0$ $TH_{max} = r_b$ otherwise
WorstCase	$CT_{max} = wT_0$	$TH_{min} = \frac{1}{T_0}$
PWC	$CT_{PWC} = T_0 + \frac{w-1}{r_b}$	$TH_{PWC} = \frac{w\, r_b}{W_0+w-1}$

In literature to assess the problem of production control, clearing functions have also been used [69]. Clearing functions model the output capacity of a production resource as a function of its workload, enabling the prediction and optimization of resource utilization over time. Conventional methods of production control often employ predetermined rules and heuristics to carry out clearing functions, which may prove insufficient for intricate and ever-changing production systems. Hopp and Spearman's theoretical framework on practical worst-case analysis highlights the vulnerability of production systems to variability and uncertainty, resulting in potential disruptions and delays. Their results can be seen as a particular case of the clearing function when we employ Average WIP-based clearing function [70].

The objective of this research is to create a unique control strategy using reinforcement learning to analyze the current condition of the production system and its set of alternative actions, with the same objective as the one proposed in *Vespoli et al.* [61]. The system can then decide the optimum policy to accomplish a certain objective. Furthermore, to evaluate the effectiveness of the suggested technique, we compare the acquired findings with the known results of the analytical model.

2.2.2 Reinforcement Learning and its application

Even though in recent years has been developed many algorithms to perform training in a RL environment, in this work we used the Deep Q-network. The Deep Q-network (DQN) has been developed by *Mnih et al.* [71]; it combines a standard RL algorithm known as Q-Learning with a deep neural network (DNN). DQN is a reinforcement learning tool and an extension of the Q-learning technique in which a deep neural network

replaces the state-action tables. Weight adjustments depending on the loss function influence the DQN's learning of the value function 2.1:

$$L_t = (E[r + \gamma\, max_a\, Q(s_{(t+1)}, a_t)] - Q(s_t, a_t))^2 \tag{2.1}$$

in which E[r + max_a Q($s_{(t+1)}, a_t$)], represents the optimum predicted reward associated with the transition to the state $s_{(t+1)}$; r is the reward associated with the action a_t and the state s_t; is the discount factor used to balance immediate and potential reward; and Q(s_t, a_t) is the network's approximate value. In the network, back-propagation is employed to transmit the mistakes estimated by the loss function, which follows the gradient descent concept. A ε-greedy technique ([7])is used to establish the policy's behavior in order to find a compromise between exploring new states and altering current good policies. DQN has shown impressive performance in a variety of domains, including video games, robotics, and natural language processing [72]. DQN is a flexible algorithm that can be adapted to a variety of problem types and environments [54]. DQN has been shown to be effective at handling high-dimensional state and action spaces, which is a common challenge in many RL problems. DQN is better suited for discrete action spaces: DQN is a Q-learning algorithm that is specifically designed to handle discrete action spaces, which are common in many RL problems and, also in the one proposed in this work.

According to a large portion of the literature reviewed, consistency in input data is critical to the agent's ability to learn [73, 74]. Since the data necessary to accurately represent a specific system and capture the desired knowledge varies substantially between applications, most intelligent control solutions are customized. To build adaptive systems, an effective representation that incorporates all necessary information is crucial. This includes understanding time and temporal trends through observation of a time-keeping data source, awareness of the agent's own performance and how it differs from the target performance, knowledge of other agents' performance, and observation of any additional factors required to depict the system's required dynamics. By considering these characteristics, an intelligent agent can build a detailed description of the manufacturing process while taking environmental and decision-making factors into account.

These characteristics enable the intelligent agent to build a sufficiently detailed description of the manufacturing process while taking environ-

mental and decision-making factors into account. It is important when tackling this kind of problems to model in a right way the RL's features (state of observation, action space and reward fuction) [9]. In this research, we modeled the observation state by taking into consideration the features of the jobs as well as the current parameter of the line. The work's main contribution is the novel technique used to define the state, action space, and reward function [59]. In Figure 2.2, there is a representation of the proposed approach. In particular, it shows how the system works: the learning agent receives observations from the production line and jobs' attributes as inputs, and then selects an action, which could be choosing the amount of WIP to be injected in the system or selecting the dispatching rule to be performed. To evaluate the quality of the action, the system receives feedback in the form of a reward, which is modeled in two ways depending on the specific sub-problem.

The class of operations to be executed, or jobs, in the queue, have particular characteristics, and an agent attached to the line learns the appropriate policy using reinforcement learning techniques. This work contributed to the modeling of the Flow Shop line and its associated parameters, as well as the network hyperparameters [58], in such a way that training can be performed and a policy can be established. Using this policy and the simulation instrument, Deep Reinforcement Learning techniques may be used to determine whether the objective is centered or not. The goodness of the training settings is specifically evaluated by simulating the model with the policy personalized from the many configurations and evaluating the relative function of loss and reward. The environment chosen for model execution is the simulation program AnyLogic, which allows for the handling of many scenarios.

2.3 Proposed settings of DQN features

The primary advancement presented in this paper involves the development of a Deep Reinforcement Learning (DRL) methodology that effectively manages Work-in-Process (WIP) to regulate throughput and job sequencing within a production system. These two sub-problems are addressed sequentially and separately. Given the central objective of configuring the attributes of the Reinforcement Learning (RL) algorithm in

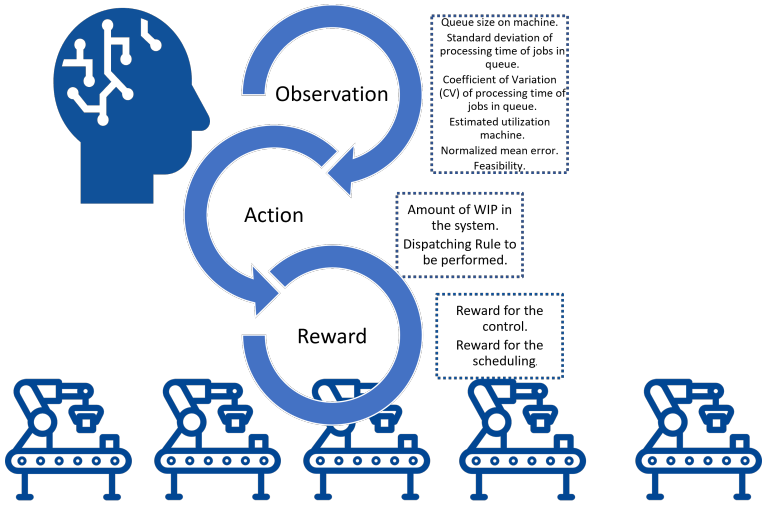

Figure 2.2. The representation of how the reinforcement learning (RL) works applied to the problems issued

this scientific paper, we now proceed to elaborate on the observation space. The observation space is consistent for both approaches, except for two parameters that are specifically tailored to address one of the sub-problems (control approach). Through empirical investigation, it has been determined that the selected state space aptly captures the essence of the problem at hand. The configuration of the state space presented in this work is based on our initial attempts [58, 59]. Dedicated sections will be allocated to explain the treatment of the action space and reward function in the two different approaches. The provided data points aim to encompass essential information in the context of regulating throughput and dynamically selecting dispatching rules. For training the DQN (Deep Q-Network) algorithm with training data, a vector of observations is employed for both sub-problems of WIP control and Dispatching Rule selection:

1. $S_0 ... S_4$: Queue size on machine i;

2. $S_5 ... S_9$: Standard deviation of processing time of jobs in queue on machine i;

3. $S_{10}...S_{14}$: Coefficient of Variation (CV) of processing time of jobs in queue on machine i;

4. $S_{15}...S_{19}$: Estimated utilization machine i;

5. S_{20}: Normalized mean error;

6. S_{21}: Feasibility.

The state space consists of machine parameters, such as those referenced in 1 and 4 ($S_0...S_4$ and $S_{15}...S_{19}$). We chose these parameters because we wanted to map the system from the perspective of production availability (i.e., the machines) and production flow (i.e., the jobs present in the system). For the latter, we used metrics referenced in 2 and 3 ($S_5...S_9$ and $S_{10}...S_{14}$) to inform the learning agent of the distribution of job processing times over time. In 5 of the list (S_{20}), we present the normalized mean error of the current throughput relative to the desired one. This metric was used to determine how to weight the variable for normalizing the mean error in state space. We chose to normalize the value between 0 and 1 because this range is more suitable for use as a parameter. It is impossible to reduce the value of a WIP below its limit, which is known as the system condition of feasibility 6 (S_{21}) in terms of the validity and qualification of the action to be performed. If WIP is positive, this variable is set to 1, otherwise, it is set to 0. We did not consider the normalized mean error (S_{20}) and feasibility (S_{21}) because they were important only for the control approach and not for the scheduling aspect. A positive work-in-process value suggests a desired situation, prompting the agent to factor in the action's feasibility before execution.

Regarding the action space, its formalization is carried out in more depth in the next section. Similar to the formalization of the agent's state representations in the observation space, the formalization of the available actions the agent can take at each step to modify its surroundings is presented. Consequently, the number of feasible actions, as well as their distribution and composition, is determined by the specific application and domain. In the Control approach, the set of actions involves determining the number of jobs to inject into the system in order to achieve a desired CONWIP value. In the Scheduling approach, the set of actions involves combining known dispatching rules for the 5-machines.

Furthermore, using reinforcement learning, the reward that must be described and also, how it is connected to the chosen actions and the environment. So, a reward function defines how the agent is rewarded and penalized for certain behaviors and their impact on its goals. A crucial factor to consider is whether the agent receives good or negative feedback while getting the reward, as well as its relative value concerning the criteria against which it is evaluated. The reward strategy must consider how these incentives should be delivered, based on which values and at what intervals, all of which will influence training and agent behavior by changing the environmental input it receives.

2.3.1 Control Approach Setting

One of the objective of this study is to regulate a Flow Shop using a Deep Reinforcement Learning (DRL) algorithm, with the aim of addressing the challenges encountered when analyzing a complex manufacturing system. Maintaining controlled levels of Work-in-Process (WIP) has proven to be effective in mitigating the impact of fluctuations and interruptions [61]. For the production line productivity control problem, the action space and reward function were formalized ad hoc. So, when dealing with the issue in this work, the action is to add some work in process to the line equal to the difference between the initial WIP value and the WIP value that the network chooses to manage to achieve the desired throughput target, scaled by a factor of 10 to be consistent with the initial WIP value, and a size of 41 for the action space (deciding to add 0, 1,...,40 jobs). In this application, the agent is provided with a reward function that gives an higher reward in relation to how close it gets to the throughput target. The purpose is to reward the agent whenever it does an activity that puts it closer to the desired behavior, the function is always positive as shown in the Figure 2.3. If the absolute value of $meanError$ is greater than a particular threshold, d, the function returns a large value of reward, which indicates a high level of reward. This means that the function strongly encourages the agent to choose actions that result in a small $meanError$ value. On the other hand, if the absolute value of $meanError$ is less than d, the function returns a value that is a linear combination of $|meanError|$ and β, that identifies the slope of the linear trend. As the absolute value of $meanError$ gets closer to d, the linear component becomes dominant

and the value of the function increases. This indicates that the function still rewards actions that result in a small *meanError* value, but the level of reward decreases as the value of *meanError* approaches d. Overall, the reward function 2.2 encourages the agent to take actions that minimize the value of *meanError*, with a particularly strong emphasis on minimizing *meanError* when its absolute value is greater than d. However, to avoid the situation where the system does not get a *meanError* equal to zero, a linear part is introduced to give the agent a specific reward value.

As a result, the calculation of this reward function is formalized in the equation 2.2 and in the Figure 2.3 is displayed the reward function with the parameter $d = 0.5$ and $\beta = 20$ (chosen after a pre-experimental campaign [58]).

$$reward_1 = \begin{cases} \dfrac{1}{|meanError|}, & if \quad |meanError| > d \\ \dfrac{\frac{1}{d} - \beta}{d} \cdot |meanError| + \beta, & if \quad |meanError| < d \end{cases} \tag{2.2}$$

Within a manufacturing setting, there are various different states and behaviors that are defined by rules. For example, in terms of both the validity and qualification of the action to be conducted, it is not possible to attempt to reduce the value of a parameter WIP below its limit. As a result, if the agent believes that the operations are unjustified, a fixed negative penalty is applied to encourage these methods to be avoided. The system is penalized if the WIP in the system, considering also the last action done by the learning agent, scaled by 10, is less than or equal to zero. With the addition of feasibility, this concept has been translated. Furthermore, in many cases, a less-than-ideal action is required in order to reach a condition in which a far more advantageous action can be taken. When the throughput objective is reached, the reward function is enabled to evaluate the biggest value.

2.3.2 Scheduling Approach Setting

This section will explain scheduling in a Flow Shop that utilizes Deep Reinforcement Learning. As a general rule, one dispatching rule may be superior to another depending on the design and conditions of the prob-

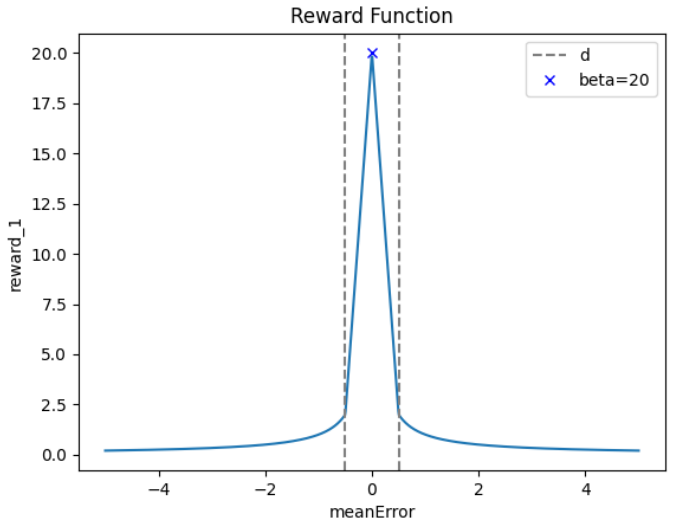

Figure 2.3. The representation of the reward function varying the value of the mean error with $d = 0.5$ and $\beta = 20$

lem. This paper considers as *Mouelhi-Chibani et al.* [75] the problem of the decision-making difficulty connected with the optimal dispatching choice for a Flow Shop. The authors believe that the system's strength comes from the fact that it does not require training but rather requires weight adjustment in the network. On the contrary, here there will be a series of experiments performed changing the hyperparameters. Notably, dispatching rules have been validated using simulation. Another commonly acknowledged result is that no dispatching rule is universally superior. Certain metrics, such as mean flow time (the number of time jobs spends in the system), do very well on certain metrics, such as SPT, while others, such as maximum job lateness, perform poorly on others. They perform poorly because they are highly dependent on system design, operating conditions, and the performance criterion used to evaluate rules. As a result, determining the optimal dispatching rules for a given situation may be challenging. In the proposed learning methodologies, simulation samples are employed as training sets. An automated system can use these sets to determine which rule selections produced positive or negative results.

Each machine's scheduling rule specifies the range of conceivable action that the system may perform. Three scheduling rules are provided: first-in, first-out (FIFO); the shortest processing time (SPT) and, the longest processing time (LPT). Considering each action as a possible rule combination to be applied to the five machines. The learning agent schedules the jobs on each machine, considering that the latter has the option of picking one of three alternative scheduling rules, which affect the system's possible operations. The set of the possible actions indicate five different rule combinations. The total number of alternatives with three criteria per machine is 3^5 (243). The recurrence with which decisions are made, and thus the action is chosen by the network, in this case, is equal to the raw time of the production line. The proposed method formulates the reward function based on the work of Hopp and Spearman, linking it to the throughput (TH) of the production line. Although the throughput of the production line cannot exceed its value at the Best Case, the link is made with the mobile throughput of the line, TH_{mobile}, which is calculated as the number of jobs processed in a certain time window over that time window. If N represents the number of jobs processed in a certain time window, and Δ_t is the length of that time window in minutes, then the mobile throughput TH_{mobile} can be calculated as:

$$TH_{mobile} = \frac{N}{\Delta_t} \tag{2.3}$$

In equation (2.3) the time window is Δ_t 240 minutes that represents the duration of an half of a working shift (8 hours = 480 minutes). The function has its intersection with the x-axis, representing the TH_{mobile} of the line, at the TH relative to the PWC, TH_{PWC}; it is asymptotic to zero because the TH is a quantity that cannot take negative values, and it has its maximum in the TH relative to the Best Case scenario 2.1. The TH corresponding to the Best Case is compensated with the highest reward and values greater than this are evenly and equally weighted. So, for values of TH_{mobile} less than or equal to r_b, the function is logarithmic and increases as TH_{mobile} increases. This means that the agent is encouraged to take actions that result in higher values of TH_{mobile} up to a certain point, after which the reward remains constant at a maximum value of $score_{max}$. On the other hand, for values of TH_{mobile} greater than r_b, the function takes a constant

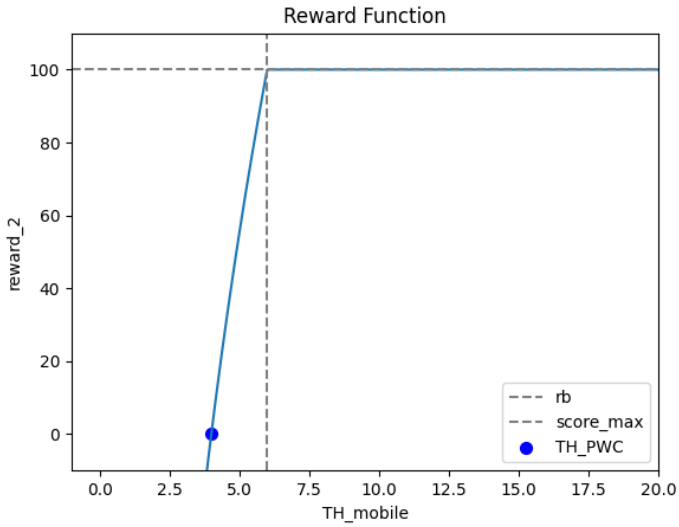

Figure 2.4. The representation of the reward function varying the value of the TH_{mobile}

value of $score_{max}$. This means that the agent is no longer incentivized to increase TH_{mobile} beyond the r_b threshold. This behavior is useful in scenarios here because it is important to balance the trade-off between maximizing TH_{mobile} and avoiding excessively high values that could lead to negative outcomes.

$$reward_2 = \begin{cases} \dfrac{\log(\frac{W_0+W-1}{W\ r_b}\ TH_{mobile})}{\log((\frac{W_0+W-1}{W})^{\frac{1}{score_{max}}})}, & if \quad TH_{mobile} \in [0; r_b] \\ score_{max}, & if \quad TH_{mobile} \in [r_b; \infty] \end{cases} \tag{2.4}$$

In (2.4), TH_{mobile} is the throughput calculated in a time window of 240 minutes, W_0 is the critical WIP of the production line, W is the amount of the WIP set constant in our CONWIP Flow Shop and r_b is the rate of the workstation that has the highest long-term utilization that is the TH in the best case (Hopp and Spearman). The $score_{max}$ in the (2.4) is

the max reward that the system can achieve and in this work we chose to set it at 100, as shown in 2.4, in order to balance the penalization and/or rewarding and the generalization of the learning.

2.4 Results and Discussion

Any Reinforcement Learning system must have a trustworthy simulated environment. This is best performed with the use of dependable general-purpose simulation software that provides rapid, consistent, and simple linkages to RL algorithms. For the experimental scenarios, we utilized a tool for Reinforcement Learning (RL) to explore different settings of the system. The environment is built with the AnyLogic simulation tool, and the RL agent is trained with rl4j from Deeplearning4j, a deep learning programming framework developed in Java. In the initial phase (configuration 1-2), we devoted considerable effort to fine-tuning the hyperparameters and adjusting the size of the hidden layer within the tool, and also the amount of episodes for the training need to be evaluated. The process allowed us to optimize the RL training. In the subsequent phase (configuration 3-4-5), our focus shifted towards modifying two important factors: the desired throughput target and the frequency at which actions are taken. By experimenting with different values for these parameters, we aimed to assess their impact on the learning process and the overall behaviour of the RL system.

The value set by hyperparameters frequently plays a key role for learning efficiency and has to be established in advance, while implementing and improving learning algorithms. In the work of *Ottoni et al.*[76] the RL performance has been examined using statistical methods, offering a rigorous method for selecting the optimal set of hyperparameters. Multiple tests have been conducted in order to determine the right number of steps per episode, as well as the total number of episodes, as well as the suitable observation and action-space. Likewise, in this work, the hyperparameter settings were determined using scientific literature and the characteristics of our case.

The hyperparameters ([77]) considered are :

- The Learning Rate has been set to 0.001, which is a reasonable value because a large coefficient (e.g., 1) causes parameters to jump,

whereas a small coefficient (e.g., 0.00001) causes them to inch along smoothly.

- The Regularization is a technique for avoiding overfitting. The "L2" regularization technique is used, which involves inserting a term into the objective function that reduces squared weights. L2 improves generalization, smoothes model output when input changes, and assists the network in ignoring weights that are unnecessary.

- The Optimizer: RMSProp (for Root Mean Square Propagation) is used as a gradient-ascent approach (as in [78]), and it is a procedure that changes the learning rate for each parameter in the network. We also compare performance using Adam (a more recently discovered updating strategy) as in *Park et al.*[79], which generates learning rates from estimations of the gradient's first and second moments.

- Max step by epoch: It specifies the maximum number of steps allowed for each episode in the learning algorithm.

- Max step: It is the maximum number of total iterations allowed in the learning algorithm. The training process will terminate when the number of steps reaches this limit.

- Max size of experience replay: This hyperparameter defines the maximum size of the experience replay buffer used in the learning algorithm. The experience replay buffer stores a collection of transitions encountered by the agent during the learning process.

- Size of batches: It refers to the number of transitions used to update the neural network weights during each iteration of the learning algorithm.

- Target update: It determines the frequency at which the target network is updated with the weights from the online network. The target network is used to stabilize the learning process.

- Num step noop warmup: This hyperparameter specifies the number of initial steps where the agent takes random actions to warm up the learning process.

- Reward scaling: It is a scaling factor applied to the rewards received by the agent. The reward scaling factor can significantly impact the efficiency of the learning process.

- Gamma: This hyperparameter determines the discount factor applied to future rewards. A higher value of gamma places more importance on long-term rewards. The value is set to 0.99. This is useful for demonstrating the convergence of a specific algorithm.

- Td-error clipping: It limits the range of the TD-error (temporal difference error) used in backpropagation during the learning process.

- Min epsilon: It is the minimum value of epsilon used in the epsilon-greedy policy during the learning process.

- Num step for eps greedy anneal: This hyperparameter specifies the number of steps after which the epsilon value is annealed (decreased) to its minimum value. The choice of how many steps used for the epsilon (ϵ) greedy annealing has a significant influence on the convergence behaviour and has been chosen so a convergence behaviour occurs within the last 10% of the training.

The algorithm in the provided code implements a Q-learning approach to train an agent in a Markov Decision Process (MDP) environment [80]. Here's a schematic explanation of how the training algorithm 1 works:

The algorithm used in this code is an off-policy reinforcement learning method. It aims to learn the optimal action-value function (Q-function) that maps states and actions to expected rewards. The agent explores the MDP environment by taking actions based on an epsilon-greedy policy, balancing exploration and exploitation. During training, the agent updates its Q-values using the observed rewards and estimates of the maximum Q-value achievable in the next state. The training process continues iteratively until a stopping criterion is met or a maximum number of steps is reached.

So, the first two experiment configurations will provide a way to identify the suitable hyperparameters, starting with a review of the state of the art and adjustments concerning the acquired outcomes to make the learning more efficient. The hyperparameters that were subject to change were

Algorithm 1 Training algorithm

1. Create an *MDP object* with the desired observation space and action space sizes.
2. Initialize the engine and the variables.
3. Define methods to retrieve the observation space, action space, current observation, reset the environment, perform an action, check if the episode is done, and create a new *MDP instance*.
4. Inside the `reset()` method:
   - Stop the engine if it already exists.
   - Create a new engine and set its properties (seed, simultaneous event selection mode, stop time).
   - Create a new object with the engine and set default parameters.
   - Start the engine and let it run for a short period to stabilize.
   - Return the current observation.
5. Try:
   - Create a *DataManager* object to record training data.
   - Define *Q-learning* configuration, including parameters like random seed, maximum steps per epoch and in total, size of experience replay, batch size, target update frequency, epsilon-greedy exploration parameters, and more.
   - Define the configuration for the *Dense Neural Network* used as the function approximator for *Q-values*.
6. Create a *QLearningDiscreteDense* object with the *MDP, DQN configuration, Q-learning configuration*, and *DataManager*.
7. Train the *Q-learning* agent using the `train()` method.
8. Get the final policy from the trained agent.
9. Save the policy to a file.
10. Close the *MDP* by stopping the engine.

Table 2.2. Hyperparameter configuration 1 and configuration 2

Hyperparameters	**Value Configuration 1**	**Value Configuration 2**
Max step by epoch	14400	57600
Max step	1440000	5760000
Max size of experience replay	1440000	5760000
Num step for eps greedy anneal	432000	1728000
L2	0-0.001	0-0.001-0.005
Optimizer	RmsProp-Adam	Adam
Hidden Layer Size	1-2	2

predominantly *Max step by epoch, Max step, Max size of experience replay* and *Num step for eps greedy anneal*. The initial setting involves a decision about the Max step hyperparameter, which has been set at 144000. This quantity is chosen considering a year of working plant: $480\frac{min}{day} * 25\frac{days}{month} * 12\frac{months}{year} = 144000$.

- The first configuration (1) sets the Max step per epoch at 14,400 steps, thereby setting the Max step at 1,440,000 steps. Similarly, the Max size of the experience replay is also set to 1,440,000. The Num step for epsilon-greedy annealing is set at 432,000 steps. The L2 parameter, indicating the strength of regularization applied, ranges from 0 to 0.001. As for the optimizer, we considered both the RmsProp and Adam optimizers. Configuration 1 can have either 1 or 2 hidden layers.

- The second configuration (2) was determined based on the results of the first experimentation. The parameters that yielded better results were taken into consideration. The Max step per epoch is quadrupled to 57,600 steps from the value in the first configuration, resulting in a Max step of 5,760,000 steps and a Max size of experience replay also set to 5,760,000. The Num step for epsilon-greedy annealing is set at 1,728,000 steps, and the L2 parameter ranges from 0 to 0.005. In this configuration, the Adam optimizer is used, and it has 2 hidden layers.

The two settings are as in table 2.2:

The following experimental configuration, on the other hand, analyzes the implementation of the model, with attention turned towards identifying the conditions that can enhance the agent's learning and understanding of the problem domain. This analysis is based on the hyperparameters set in configuration 2, as described previously. In this setting, an event was provided to the network, and we differentiate between three configurations:

- The first configuration (3) deals with 200.000-time steps of training with a triggering action of 50 minutes and a 25.000 minutes occurrence changing target that leads to good results in terms of hitting the target, low standard deviation, and mean WIP that validates Practical Worst Case hypothesis.
- The second configuration (4) deals with 1.000.000 time steps training with a triggering action of 50 minutes and 250.000 minutes occurrence changing target that leads to an even better result, especially in terms of standard deviation.
- The third configuration (5) deals with 1.000.000 time steps training with a triggering action of 50 minutes and 25.000 minutes occurrence where the targets varied cyclically. This configuration leads to results that are very similar to the second experiment configuration.

2.4.1 Control Approach Results-Configuration 1

Preliminary studies for this contribution demonstrated that the settings in the second column of table 2.2 yielded favourable results. To gauge the performance more precisely, an experimental campaign was initiated. Without loss of generality, in this paper we will focus on a production line consisting a 5 machine-Flow Shop assuming the jobs' processing time distributed as a Gamma distribution as in [63]. The Gamma distribution is characterized by the shape parameter α and scale parameter β, it enables the manipulation of variability levels. This distribution is a continuous probability distribution, encompassing the exponential distribution as a special case (when α=1). By adjusting the value of α, it becomes possible to model variability exceeding that of the exponential distribution when α<1, and variability below that of the exponential distribution when α>1. The gamma distribution's versatility empowers the

modeling of both high-variability scenarios (e.g., customized production) and low-variability scenarios (e.g., standardized production).The hypothesis proposed here are based on the work of *Hopp and Spearman* [1]. From the assessment of the system's variability identified via the Gamma distribution with α=1, thereby in essence an exponential distribution with a mean equal to 10, and considering the parameters of the Flow Shop line comprising 5 machines, the decision was made to train the network utilising a throughput target of 4. Specifically, this choice is associated with the evaluation of the minimum throughput value linked to the Worst Case scenario and the maximum throughput value linked to the Best Case scenario. Scenario configurations and formulas are summarised in the preceding Section 2.2.1; thus, a throughput of 4 suggests that we position ourselves in the intermediate case.

The formula to calculate the WIP relative to the Practical Worst Case throughput is the following (derived:derived by the equation in the table 2.1)

$$w = \frac{TH(W_0 - 1)}{r_b - TH} \tag{2.5}$$

During each training step, the state parameters of the system are observed. The experiments were conducted by varying some parameters of both the Flow Shop system and the learning ones. In particular the experiments highlighted in light blue were carried out considering an initial WIP equal to 5 while those in purple were carried out considering an initial WIP equal to 20. In addition, the size of the network was varied: in particular in Figure 2.5.a is considered a network with a hidden layer of 150 neurons while in figure 2.5.b is considered a network with two hidden layers of 300 neurons each.

It is necessary to consider not only how the reward function varies but also how WIP value behaves during the training. The output parameters considered are: the mean error intended as the difference between target throughput and current throughput, its standard deviation and lastly the mean WIP kept constant in the production system. The value 8 for mean WIP outcome corresponds to the WIP value of the practical worst case with a throughput equal to 4 as we can deduce from Equation 2.5.

We have considered as initial WIP one that is very high (20) and one that is equal to the critical WIP (5) which is the WIP level through which a

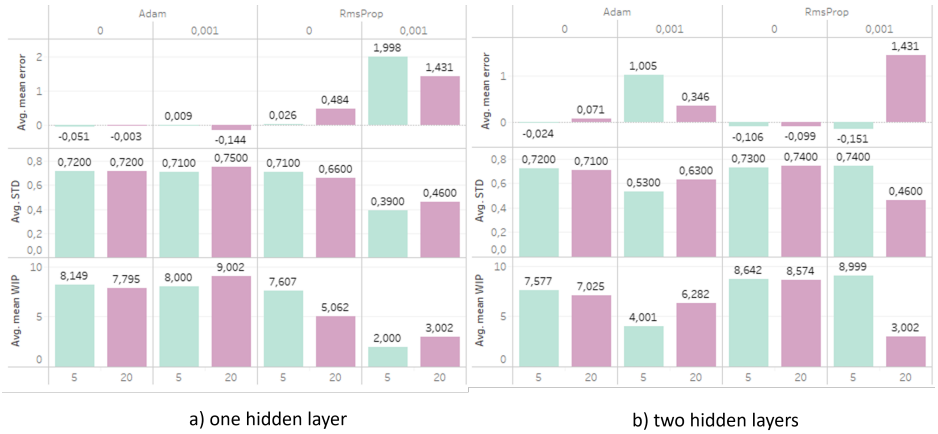

Figure 2.5. The graphical results (the mean error of the throughput, its standard deviation, and the mean WIP in the system) of the experiment of setting 1 for the two configurations with one hidden layer and two ones.

production line, with a defined Raw Processing Time and Bottleneck Rate, achieve the maximum throughput and the minimum cycle time without any variability. Results show that, excluding small variations, the WIP is on average 8 with throughput target equal to 4 in accordance to Practical Worst case formula. In terms of responsiveness of the system to changes in throughput target different than the one used to perform the training and determine the respective policies, there were not great results. The loss function represented in Figure 2.6 has on the x-axis the number of iterations and on the y-axis the value of the loss. The decreasing trend is shown and it remains constantly low until the end of the training.

2.4.2 Control Approach Results-Configuration 2

Based on the results obtained from the configuration above, it is possible to affirm that the configuration of the action space, state and reward in this manner, disregarding the varying configurations of the network in terms of size, regularisation and optimiser, enables the network to establish a certain amount of WIP without making alterations. Specifically, when a different throughput target is applied, the network does not react. Nevertheless, Adam seems to be the superior optimiser. In addition, an

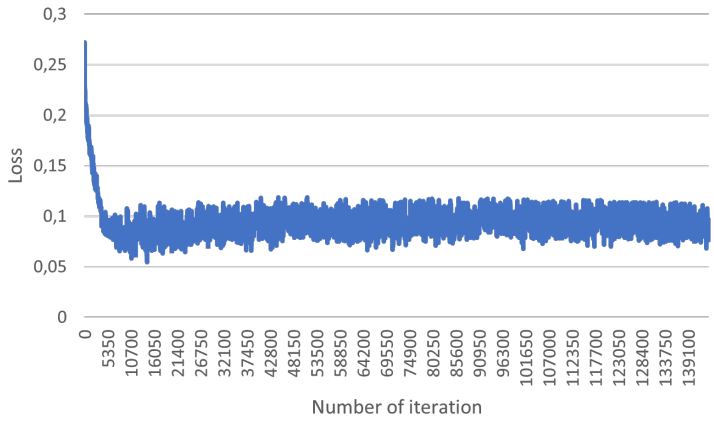

Figure 2.6. The results of the training in terms of loss function for the configuration 1

attempt has been made to extend the duration of training. The considered increment entails quadrupling the iterations and the corresponding epochs.

As a result, a new configuration of hyperparameters for the agent's learning has been established, as depicted in the third column of Table 2.2. Furthermore, from previous experiments, it could be deduced that having a network with two layers of neurons leads to globally better results in relation to hit the target. The results of this new set of experiments are summarized in Figure 2.7.

The output parameters considered are, as before, the average error intended as the difference between target throughput and current throughput, its standard deviation and average WIP. The experiments in green-blue were carried out considering an initial WIP equal to 5 while those in purple were carried out considering an initial WIP equal to 20. The results of the experiments performed with this hyperparameter setting showed that when the L2 regularization was changed to 0.005, the target was more centered and the loss function was generally better for several tests.

As the Table 2.3 shows, the system effectively do reach the target of 4 learns to keep constant in the system 8 jobs (in accordance to Practical Worst Case laws). The loss function represented in Figure 2.8 has on the x-axis the number of iterations and on the y-axis the value of the loss.

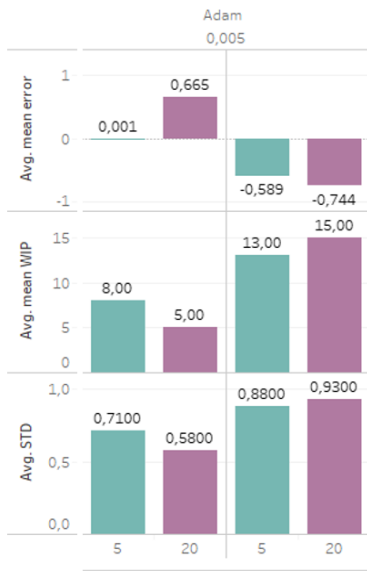

Figure 2.7. The graphical results (the mean error of the throughput, its standard deviation, and the mean WIP in the system) of the experiment of setting 2 for the two configurations with two hidden layers

Table 2.3. Experiment results - configuration 1 and configuration 2

WIPvalue	Optimizer	L2	Mean Error	Standard Dev	Mean WIP
5	Adam	0.001	0.009	0.71	8
5	Adam	0.005	0.001	0.71	8

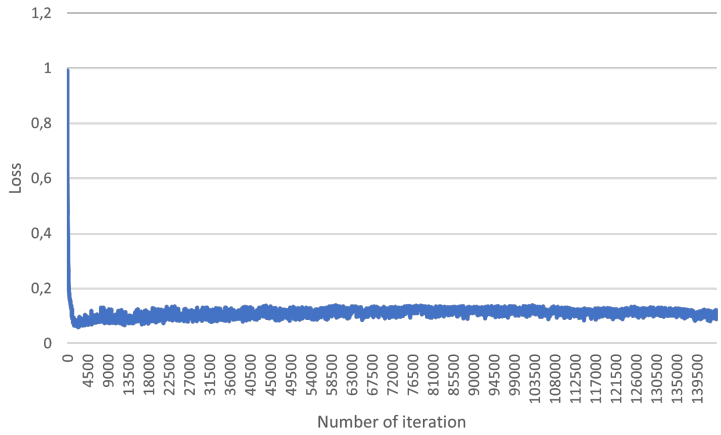

Figure 2.8. The results of the training in terms of loss function for the configuration 2

The strongly decreasing trend is shown and it remains constantly to zero value until the end of the training. We consistently observe the same phenomenon in the results: the network learns to stay at that Work-in-process (WIP) value, even when we change the target. The training of the agent has resulted in a stable policy, and for this reason, its parameters have not been further tuned. From this point onwards, we have redirected our attention towards factors that can improve the agent's acquisition of knowledge and comprehension within the problem domain.

2.4.3 Control Approach Results-Configuration 3

Considering the fact that the initial configurations yielded good results but fell short in effectively tracking the target throughput, three new configurations were developed. These new configurations were designed to provide the learning agent with a broader range of experiences encompassing various WIP and throughput values that a production system, adhering to the assumptions of the problem under consideration, could encounter. During the simulation the amount of TH to be reached has been changed to make the system experience different production line setting. The time of simulation has been set to 200.000, therefore every 25.000 iterations the

Table 2.4. Results for configuration 3

TH Target	mean Error	Standard Deviation	mean WIP	WIP PWC
2	-0.04	0.46	2.1	2
3	-0.02	0.72	5.01	4
4	0.32	0.93	12.1	8
5	0.78	1.03	19.5	20

throughput target from 2, increases of one unit; when it arrives at 6 (best case throughput) the simulation and the training stop. For every training configuration after 250 epochs, the training will be stopped early and create a policy if the mean reward does not change by more than 1% over 75 epochs. The acquired results with an initial WIP value of 10 are displayed in the table 2.4.

The value (Table 2.4) represent different target levels for Throughput (TH). The mean Error shows the average deviation from the TH Target. TH Targets 2 and 3, the errors are slightly negative, indicating that the actual throughput was marginally below the targets. For TH Targets 4 and 5, the errors are positive, indicating that the actual throughput exceeded the targets. The increasing trend in error values, especially the jump from 0.32 at TH Target 4 to 0.78 at TH Target 5, suggests a tendency for greater deviation from the target as the target increases. The values of Standard Deviation measure the consistency or variability in achieving the TH Target. Higher values, particularly for TH Targets 4 and 5, suggest more variability or fluctuation in meeting the targets. "WIP" stands for Work In Process, and these numbers represent the average number of jobs being processed. Given that "PWC" stands for "Practical Worst Case," these represent the values resulting with the laws of [1] for WIP at each TH Target. Notably, for TH Target 4, the WIP PWC is 8, which is significantly lower than the Mean WIP of 12.1. For TH Target 5, the WIP PWC closely matches the Mean WIP, suggesting that the average condition is closely aligned with the PWC scenario. The results in terms of error and mean WIP are very satisfactory in particular with the first two throughput targets.

Figure 2.9 shows the network learning process and, as epochs increase, the reward function increases significantly. This means that the network

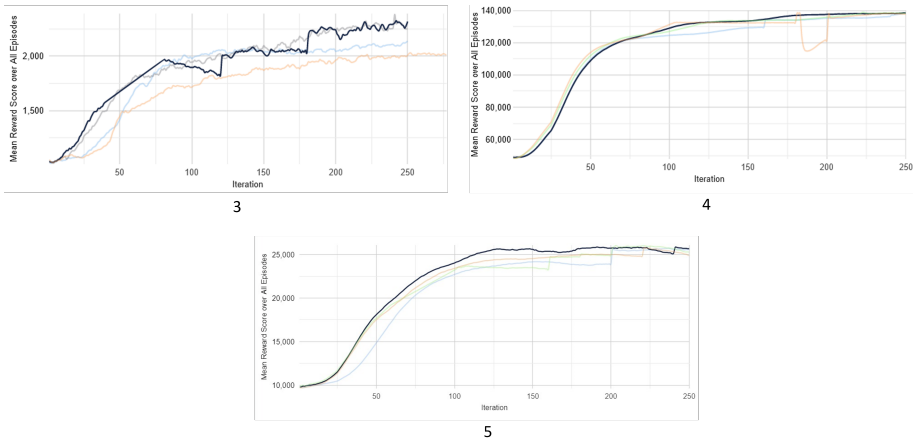

Figure 2.9. The results of the training in terms of the average reward for the three different configurations 3-4-5

managed to learn the policy with results shown in the table 2.4 mentioned before.

2.4.4 Control Approach Results-Configuration 4

To investigate if the system became more responsive, for this configuration it was chosen to increase the simulation time to 1 million steps and to modify the event with the target change every 250,000 iterations. The outcomes are shown in the Table 2.5. System variability has improved significantly, throughput are on target and mean WIP values validates the Practical Worst case hypothesis. Figure 2.9 in 4 shows the network learning process and, as epochs increase, the reward function increases significantly. The policy learnt in this configuration is smoother than the previous one and this means that the network managed to learn even better the policy with results shown in the Table 2.5 mentioned before.

The values (Table 2.5) of mean Error indicate how much the actual throughput deviated from the target. Negative values for TH Targets 2 and 3 suggest underperformance, while positive values for 4 and 5 suggest overperformance. The values of standard deviation show the consistency of performance in achieving the TH Target. Higher values indicate greater

Table 2.5. Results for configuration 4

TH Target	mean Error	Standard Deviation	mean WIP	WIP PWC
2	-0.04	0.41	2.08	2
3	-0.13	0.61	4.55	4
4	0.11	0.73	7.91	8
5	0.69	0.87	14.4	20

Table 2.6. Results for configuration 5

TH Target	mean Error	Standard Deviation	mean WIP	WIP PWC
2	-0.04	0.43	2.1	2
3	-0.12	0.67	4.8	4
4	0.16	0.77	8.91	8
5	0.72	0.91	16.9	20

variability. The PWC of the WIP values provide a benchmark for the maximum expected WIP under each target scenario. Notably, for higher TH Targets, the PWC values are notably higher than the Mean WIP (especially for TH Target 5), indicating a significant disparity between these values and PWC scenario ones. The PWC model's analysis reveals that with higher target values (4, 5), it is possible to maintain a lower average WIP, even with greater yet positive error values. This leads to increased Throughput (TH), enhancing overall productivity.

2.4.5 Control Approach Results-Configuration 5

Working with the system's temporal parameters, it was possible to conclude that the training resulted in a good learning of the net. With this configuration, a new technique was tested that directly affects a cyclical change of the target without stopping the simulation, as seen in the figure. The results are shown in the Table 2.6. The results are very similar to configuration 4.

In this case (Table 2.6), for TH targets 2 and 3, the mean errors are negative (-0.04 and -0.12), indicating that the actual TH was slightly below the target. For TH targets 4 and 5, the mean errors are positive (0.16 and 0.72), suggesting that the actual TH exceeded the target. The

increasing trend in error values, from negative to positive, suggests that as the target TH increases, the system tends to overshoot the target more. Lower values of the standard deviation (0.43 and 0.67 for targets 2 and 3) indicate more consistent performance around the target. Higher values of the standard deviation (0.77 and 0.91 for targets 4 and 5) indicate less consistency, with more fluctuation around the target. The column related to the WIP PWC represents a benchmark Work In process value for each TH Target, possibly derived from a model like the PWC (Practical Worst Case) model. The values generally align with the Mean WIP but are not identical. For example, at TH Target 5, the WIP PWC is 20, which is higher than the Mean WIP of 16.9, possibly indicating an overestimation or a more aggressive prediction by the model. Overall, these values suggest a relationship between the target throughput, the average error in achieving this target, the variability of this performance, and the amount of work being processed. Higher throughput targets seem to correlate with more variability in performance and a higher volume of work in process. Figure 2.9 in 5 shows the network learning process and, as epochs increase, the reward function increases significantly this means that the network managed to learn a good policy and its results are shown in the table 2.6 mentioned before. When the simulation is executed, the system reacts whenever the throughput target is changed. In other words, the neural network understood whether it needs to increase or decrease the amount of WIP in the system, deciding on a mean WIP value that is close to that of the Practical Worst Case.

2.4.6 Control Approach Discussion

The data analysis of throughput (TH) targets in an operational context reveals insightful trends about performance consistency and work in process (WIP). For TH Targets 2 and 3, the mean errors are slightly negative, indicating a modest underperformance, whereas for Targets 4 and 5, the errors turn positive, showing that actual throughput surpasses the targets. This shift from negative to positive errors as targets increase highlights a tendency for the system to overshoot more ambitious goals. The standard deviation values, reflecting performance variability, are lower for Targets 2 and 3, suggesting more consistent achievement of these targets. In contrast, higher values for Targets 4 and 5 indicate greater fluctuations

around these targets. In terms of WIP, the Practical Worst Case (PWC) scenario analysis provides a benchmark for maximum expected workload. Notably, at higher targets, the PWC values substantially exceed the Mean WIP, pointing to a significant disparity between the average and practical worst-case scenarios. This comprehensive analysis underscores the complex interplay between target setting, error margins, and operational consistency, offering valuable insights for optimizing throughput and managing WIP effectively. Although configuration 4 seems to be the most promising option and the overall best for the production line considered, the results obtained still show a high value for the standard deviation. Despite the significant improvement in terms of the assessment of the state, action, and reward function over previous settings, several factors could cause a difference in performance between the actual and virtual systems. For example, machine failure and maintenance are not included in the model, and failures are common in real-world systems, with maintenance processes always present. Additionally, the utilization rates of the resource pools are significantly higher than in the others when the system deals with high levels of throughput. This is mainly due to the significant variation in processing time compared to the other processes. This issue has a damaging effect on the high outputs of the process that operate at sub-capacity levels. Despite all the complexities and concerns discovered, the results validate the Practical Worst Case hypothesis and serve as a solid starting point for future, deeper investigations.

2.4.7 Scheduling Approach Results

This study involves developing a dynamic system to select processing rules for each machine and evaluating the characteristics of the DQN network. Although we explored different configurations and achieved our target, the challenge lies in applying the settings to various throughput targets. Regarding training and hyperparameter tuning, we observed that the state-action-reward setup was successful when using a the hyperparameters set in the “Configuration 2". Therefore, we divided the proposed approach into two sub-problems: first, to begin training for TH control by manipulating the WIP, and second, to configure a system with a different reward function to regulate the job dispatching rule choice for each machine by evaluating the production system’s status. For the second part,

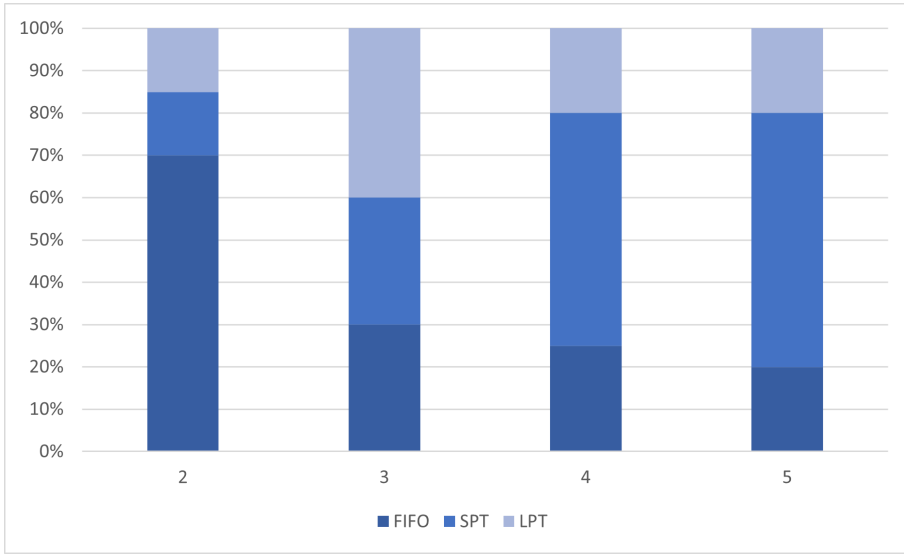

Figure 2.10. The results of the scheduling approach combined to the control approach for the Configuration 4

we used the simulation parameters from “Configuration 4" which produced the most promising results. We excluded the normalized mean error (S_{20}) and feasibility (S_{21}) since they were relevant only for the control approach and not for the scheduling part. Thus, the input layer size is smaller, with 20 nodes, the action space consists of 243 possible actions resulting from combining the three rules for the five machines, and the reward function aims to maximize throughput.

The results of using the policy generated by training for dispatching rule selection are shown in Figure 2.10. The policy takes the average WIP outcomes obtained with the policy developed for the control method, considering the throughput value of 2, 3, 4, and 5 as input. The results indicate that when the system’s work in process increases, the network favours the SPT rule, which is known to perform best when maximizing TH. Overall, since our goal was to maximize TH, the suggested tool learned to arrange an appropriate mix of DR. For lower WIP numbers, the system favours the FIFO rule, consistent with the simulation model assumptions.

2.5 Conclusions

The combination of reinforcement learning and deep neural networks proposes an effective strategy for controlling and scheduling production in Flow Shops, thereby mitigating inefficiencies in manufacturing systems. This innovative technique describes the state and action space, enabling regulation of a production system's performance in terms of work-in-process (WIP) and throughput (TH), as well as determining the dispatching rules to be employed in a 5-machine Flow Shop. To train the model, a simulation tool has been employed to acquire the dataset necessary, while the reinforcement learning procedure involved a learning agent that comprehensively examines every aspect of the simulated environment. The proposed approaches has been bifurcated into two sub-problems: controlling TH by dynamically varies WIP levels, and devising a new dispatching rule contingent on the system's status.

The neural network effectively determined the appropriate amount of WIP in the system, settling on a mean value similar to the Practical Worst Case scenarios. Subsequently, the reinforcement learning agent selects the most suitable dispatching rule based on the output of the initial training. Notably, the configuration of state, action, and reward proves effective when utilizing the hyperparameters from "Configuration 2." Specifically, "Configuration 4" has emerged as the most effective in achieving the desired Throughput target and aligning with the Practical Worst Case scenario, thereby significantly optimizing the parameters of the DQN model. The same configuration ("Configuration 4") has been then used for the Scheduling Approach. The findings indicate that as the WIP increases, the reinforcement learning approach demonstrates a preference for the Shortest Processing Time (SPT) rule, known for its superior performance in maximizing TH. Consistent with our objective of maximizing TH, the suggested tool effectively incorporates an optimal blend of Dispatching Rules (DR). Conversely, for lower WIP values, the system tends to prioritize the First-In-First-Out (FIFO) rule, aligning with the assumptions of the production model.

In the authors opinion, such a tool has the potential to serve as a control room, where simulations and digital twins can be utilized to validate decisions. This characteristic renders the method especially pertinent in

the context of Industry 4.0, wherein data from the production line can be conveyed to a central controller for analysis and informed decision-making, even amidst potential disruptions. This research constitutes an initial step towards a broader research path intended to address more intricate systems with diverse state configurations. The proposed technique, exhibiting robustness and adaptability, has the potential to offer invaluable assistance in decision-making processes in Industry 4.0. It thereby establishes itself as a dependable tool for future research and practical applications.

Chapter 3

Optimizing Flow Shop Maintenance Planning and Scheduling through Deep Reinforcement Learning: An Integrated Framework

3.1 Introduction

In the fast-paced realm of modern manufacturing, strategic maintenance planning stands as a crucial task in achieving operational optima, underscoring the need for sophisticated strategies that ensure efficient resource allocation and system reliability [81]. Maintenance scheduling, especially in complex Flow Shop systems characterized by a fixed sequence of processing tasks, is imperative to uphold system performance [82]. The dynamic nature of these systems necessitates quick responses to minimize downtime and preempt equipment failures, ensuring a seamless production continuum [83].

Maintenance scheduling has balanced a fine line between equipment reliability, associated downtime costs, and resource management, as highlighted by Paz and Leigh [27]. The convergence of these factors delineates

the plan for superior maintenance strategies, guiding the intricate decision-making processes inherent in scheduling protocols.

In this contest, Artificial Intelligence (AI), and more specifically, Deep Reinforcement Learning (DRL) a Machine Learning sub-field, has emerged as a powerful tool at navigating the complexities of such decision-making landscapes. DRL, a subset of AI, has been employed in the optimization of manufacturing systems, demonstrating remarkable advancement in scheduling, system optimization, and control approaches[26] . In the scientific literature, many studies have been conducted that employ these kinds of tools to optimize the scheduling of maintenance activities. Valet et al.[84] proposed a deep RL-based opportunistic maintenance scheduling approach. By considering the operational status of the production system, their approach schedules maintenance tasks during periods of low demand, thereby minimizing the impact on production and improving maintenance effectiveness. Yan et al.[85] developed a double-layer Q-learning algorithm for dynamic scheduling with preventive maintenance in digital twin-enabled manufacturing systems. Their approach considered the uncertainties and dynamics of the production system to maximize production efficiency while minimizing maintenance costs. By incorporating RL, manufacturers can achieve more effective real-time scheduling and maintenance decision-making, leading to improve overall system performance. Mao et al.[86] developed a hash map-based memetic algorithm to minimize the total flowtime of jobs while considering multiple maintenance activities. Although their approach did not explicitly integrate RL, the integration of such techniques could further enhance the scheduling optimization process, enabling adaptive decision-making based on real-time data and system dynamics. Predictive maintenance is another critical aspect in manufacturing systems, and AI techniques can play a significant role in this area. Huang et al.[82] proposed a DRL-based preventive maintenance policy for serial production lines. By considering the trade-off between maintenance costs and machine availability, their approach demonstrated the potential of DRL in making optimal maintenance decisions. Furthermore, the integration of different planning activities can lead to more efficient resource utilization and improved system performance. Akl et al.[81] developed a simulation-optimization approach for joint optimization of strategic workforce planning and preventive maintenance scheduling. By integrating these two

planning activities, manufacturers can achieve better coordination and optimization, resulting in improved overall performance.

While these works highlight the benefits of DRL in manufacturing systems, there are several challenges to address. Nunes et al. [87] reviewed the challenges in predictive maintenance, emphasizing the need for accurate data, effective ML models, and appropriate maintenance strategies. The authors highlight the importance of advancements in AI techniques to overcome these challenges and improve predictive maintenance practices in manufacturing systems. Furthermore, the combination of DRL and other optimization methods has shown promise in maintenance planning. Kosanoglu et al.[88] proposed a DRL-assisted simulated annealing algorithm for maintenance planning. This approach harnesses the benefits of both DRL and simulated annealing, allowing for efficient exploration of the solution space and finding optimal maintenance schedules.

In this work we introduce an integrated simulation tool and Deep Reinforcement Learning (DRL) algorithm to efficiently schedule and plan maintenance events in a production line flow shop. This approach combines the strengths of simulation and DRL to optimize maintenance processes and maximize productivity. The simulation tool creates a virtual environment that replicates the production line flow shop, allowing for modeling and simulation of machine operations, job flows, and maintenance events. This enables the evaluation of different maintenance strategies and their impact on overall performance by accurately capturing the system's dynamics and complexities. The DRL algorithm is the core intelligence of the proposed approach. It learns optimal decision-making policies by interacting with the simulated environment and maximizing cumulative rewards. The algorithm considers machine failure probabilities and scheduling constraints to make informed decisions about maintenance scheduling.

The organization of this paper is outlined as follows: Section 2 delineates the formulation of the problem under investigation. In Section 3, we elaborate on the proposed methodology with comprehensive detail. Section 4 delves into the simulation framework and experimental design implemented to validate our approach, including an analysis of various configurations and their outcomes. The paper culminates with Section 5, where we draw conclusive remarks from the study.

3.2 Proposed approach

This study introduces a tool that merges Deep Reinforcement Learning techniques with simulation for scheduling maintenance events in a Flow-Shop production line. A Flow-Shop line has "m" machines and "n" tasks, with each task needing "m" processes. Only one process can be executed on a machine at any moment, each requiring a distinct duration. Without interruptions, these processes are done sequentially on machines.

In the Flow-Shop model, the batch flow is one-way and linear, maintaining consistency across machines. Explicitly, the job's *i-th* process takes place on the *i-th* machine. Thus, in this setup, each task follows a designated order on every machine, though tasks can be arranged in varying sequences.

The assumptions include:

- Machines can fail unpredictably, following a Weibull distribution. Each machine's failure rate can vary.
- The transport time to move a job between two consecutive machines is ignored.
- Both corrective and preventive maintenance adhere to the perfect repair model.
- Maintenance is performed during machine downtimes. Specifically, immediate action follows a machine's failure for corrective maintenance, while preventive maintenance is planned between production tasks.

3.2.1 Deep Reinforcement Learning and Proximal Policy Optimization

Reinforcement Learning (RL) is a subset of Machine Learning where an agent learns to decide optimally in an environment by maximizing cumulative rewards over time. The agent employs a trial and error approach, getting feedback as rewards or penalties based on its actions. At each stage, represented by t, the agent acknowledges the current state s_t and its associated reward r_t. The agent then picks an action a_t, transitions

the environment to the next state s_{t+1}, and receives a subsequent reward r_{t+1}. The agent's main aim is to formulate a policy π that defines the best action for each state to maximize the expected reward.

A standout DRL method is the Proximal Policy Optimization (PPO) by Schulman et al. [89]. As part of Policy Gradient techniques, PPO cyclically collects environment data and maximizes a surrogate objective function using stochastic gradient ascent. PPO designs the policy without needing an environment model. By adopting Trust Region Optimization, the method ensures moderate policy updates for consistent learning. The Clipping Objective Function in PPO curtails policy update sizes to avert instability.

PPO, being an Actor-Critic method, uses neural networks for both the policy (actor) and value estimation (critic), as shown in algorithm 2. It comprises the policy network that selects actions based on the environment state and the value network predicting cumulative rewards from that point.

PPO's strengths include handling continuous action spaces often found in real-world scenarios. PPO also supports parallelism, facilitating multiple agent-environment interactions concurrently, resulting in quicker training and enhanced results. Overall, PPO offers advantages like straightforward implementation, increased adaptability, and superior empirical sample efficiency.

The integrated simulation tool and DRL algorithm offer several advantages. Firstly, they allow the assessment of maintenance strategies without disrupting real production operations, reducing risks and costs associated with experimentation. Secondly, they provide a platform to evaluate different maintenance scheduling policies under diverse scenarios, identifying optimal strategies for maximizing production efficiency and minimizing downtime. Lastly, the integration of simulation and DRL facilitates the development of adaptive maintenance planning systems that can adjust to changes in machine failure patterns, job priorities, and production demands. From a computational standpoint, conducting training sessions for the entire system could be demanding due to the complexity of the production system. Therefore, in this work, we propose conducting the training on a single machine and then assessing the effectiveness of the developed policy within a 5-machine flow shop scenario.

Algorithm 2 PPO Training algorithm

1. Initialize actor (policy) and critic (value function) neural networks.
2. Initialize hyperparameters (learning rate, number of iterations, gamma, etc.).
3. Initialize experience buffer.
4. Training Loop: Execute the training loop for a specific number of iterations:
   (a) Collect experience by interacting with the environment using the current policy.
      - Sample actions from the policy.
      - Execute actions in the environment and observe rewards and new states.
      - Store experience in the experience buffer.
   (b) Calculate estimated advantages for each time step:
      - Compute estimated values $V(s)$ from the critic.
      - Calculate advantage $A(s, a) = (R(t) + \gamma \times V(s') - V(s))$.
   (c) Calculate policy loss:
      - Calculate the ratio of new and old action probabilities (with clipping).
      - Compute surrogate loss L^{CLIP} based on the ratio.
      - Update the policy to maximize L^{CLIP}.
   (d) Calculate critic loss:
      - Compute the value loss $V(s)$.
      - Update the critic to minimize the loss.
   (e) Update the policy and critic using stochastic gradients.
5. Repeat the training loop for the desired number of iterations.

3.3 System Components and Operational Framework

To ensure a comprehensive understanding of our proposal, we first delve into the system components and operational framework. The process involves utilizing Anylogic simulation software to model the production line. This software allows for the simulation of various line operations, as well as corrective and preventive maintenance events. Additionally, it provides the capability to configure essential parameters for the Reinforcement Learning Experiment, which will then be exported and integrated into Python using the ALPyne library. Subsequently, a Reinforcement Learning (RL) algorithm will be developed in Python, and agent training will be carried out using a single-machine system to formulate a policy. This policy will undergo testing both within the Python environment and through simulation using Onnx. The utilization of Onnx for implementation facilitates policy testing across different systems, notably including testing on a Flow-Shop system consisting of 5 machines. The process of integrating various applications starts with the simulation software, where the model is tailored to suit the RL experiment (Figure 3.1). This model is then converted into a zip file, and its location is supplied as input to the ALPyne client. Among other configurable options that will be elaborated on later, the DRL algorithm is set up for training. Upon completing the training and formulating the policy, this policy is transitioned into an onnx file format. Subsequently, the Onnx Helper library aids in modifying and executing the simulation. Users have the facility to call upon a function within the model to access the onnx file, utilizing its output to gauge the system's efficacy across the entire Flow Shop.

3.3.1 The Multi-Method Simulation Model Configuration

The AnyLogic environment has streamlined the development of a Flow Shop model, enabling experiments with various problem configurations, such as the number of jobs and machines, and the probability of machine failures. This model utilizes a multi-method approach combining Discrete Event Simulation and Agent-Based modeling. This sets the foundation upon which we detail the configuration of the multi-method simulation

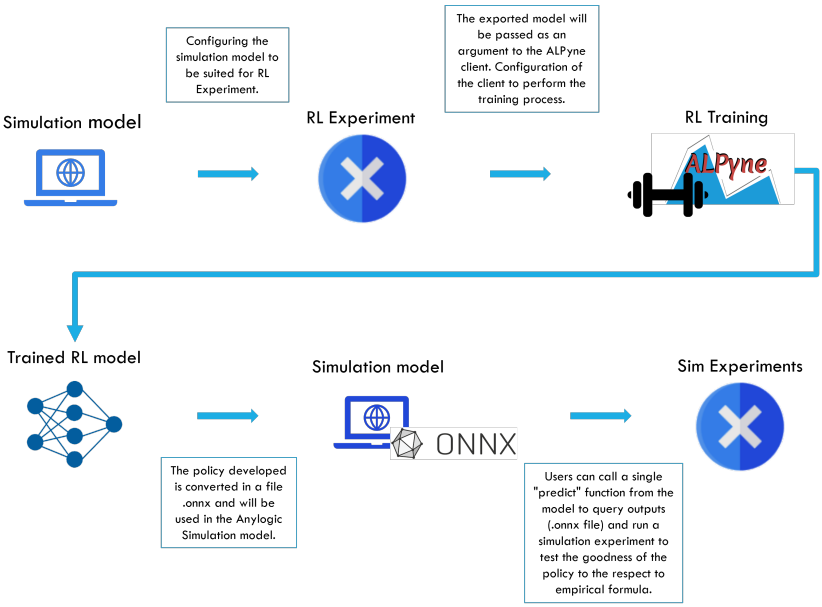

Figure 3.1. How the integration of the different platforms is achieved.

model (Figure 3.1). The Anylogic model follows a hierarchical structure, where each agent at a higher level encompasses lower-level agents. This structure allows for various levels of abstraction, with higher-level agents having a higher level of abstraction than lower-level agents. The proposed model comprises of a top-level agent, Main, and two lower-level agents, Job and Machine. At the top of the hierarchy is the agent called "Main", which acts as a container for the lower-level agents, "Job" and "Machine". Its primary role is to create the populations of jobs and machines, making it responsible for providing the environment for the other agents to function properly.

Next, at a lower level, there is the Job Agent which contains a statechart that describes the possible states in which each job may be found during the production phases. And finally, at the same level of Job Agent, there is the "Machine" Agent, the heart of the system. It represents the production stations where the jobs are processed and consists in two paths, one for production jobs and one for maintenance jobs.

The simulation model parameters were set as shown in the table 3.1:

N.jobs	50
Processing Times	triangular(20,100,50)
Corrective Mant. Time	51 min
Preventive Mant. Time	17 min
Weibull scale parameter (α)	450
Weibull scale parameter (β)	1.5

Table 3.1. Simulation Model parameters

Subsequently establishing the simulation groundwork, we transition to Configuring the Deep Reinforcement Learning (DRL) Training Library, equipping us with the necessary tools for advanced analysis.

3.3.2 Deep Reinforcement Learning (DRL) Training Library Configuration

To perform the training phase ALPyne library, an AnyLogic-Python connector is used. In particular, it is a Python library for interactively running models exported from the RL experiment, which can be used with any edition of AnyLogic (Personal Learning Edition, University, or Professional). The need to use this tool stems from the fact that, unlike other experiments, the RL experiment cannot be directly executed by end-users within AnyLogic. This is because AnyLogic does not have any embedded RL algorithms. Thus, it can only be directly executed by compatible platforms (e.g., ALPyne). To start using ALPyne, the initial step is to configure the AnyLogic model by filling in the fields related to Configuration, Observation, Action, and stopping conditions in the section dedicated to Reinforcement Learning experiments and also the *takeAction* method should be called when the agent needs to make a decision. Once the model is configured, it must be exported for the training phase, which occurs on a separate platform from Anylogic.

After the model is exported, the next step is to set up the training phase using ALPyne. This involves importing the necessary libraries and defining the observation and action spaces. The libraries to import include *ALPyneClient*, *ModelRun*, *Observation*, *Action*, and *BaseALPyneEnv*. The *ALPyneClient* initializes the ALPyne application, while *ModelRun* represents a single simulation run from the *ALPyneClient* object. *Observation*

is a read-only object representing an observation taken from the simulator, and *Action* is an object representing an action to send to the simulator. The *BaseALPyneEnv* class is also imported to simplify using ALPyne with Gym environments.

When leveraging ALPyne in conjunction with Gym environments, it's crucial to establish the observation and action spaces, including their specific data types. These elements vary based on the particular settings employed in the AnyLogic model. Subsequently, the code is required to transform the entities known as *Observation* into the format stipulated by the observation space and morph the action received from the RL framework into an *Action* entity. Furthermore, it needs to compute the numerical incentive predicated on the observation recorded post-action.

In the instance at hand, the space dedicated to Observation is categorized as a *Box* type. It is multi-dimensional, continuous, and bound by maximum and minimum thresholds, as indicated by its *high* and *low* parameters. It is explicitly a quintet-dimensional space, encompassing variables such as:

- **FP**: failure probability;
- **CJ**: The percentage of completed jobs at time t;
- **CR**: $\left(\frac{MPT}{pT+cT}\right)$, the ratio of the mean processing time to the sum of the processing time at time t and the time in corrective maintenance;
- **PR**: $\left(\frac{pT+mT}{MPT}\right)$, the ratio of the sum of the processing time at time t and the time in preventive maintenance to the mean processing time;
- **TR**: $\left(\frac{PT}{PT+CT+MT}\right)$, the ratio of the total processing time at time t to the sum of the total processing time, corrective time and preventive time at time t.

The first element discussed is the probability of failure for a single machine. Notably, this probability is updated as production jobs are processed, with each new observation refreshing the data. The agent uses this updated failure probability to make decisions about scheduling maintenance jobs. Following this, data on the completion of production jobs, expressed in percentage, is provided. This information is vital for the agent

CT (*Corrective Time*)	The total time spent for Corrective maintenance at time t
cT (*corrective Time*)	The time of a single corrective maintenance task
MT (*Preventive Time*)	The total time spent for preventive maintenance at time t
mT (*preventive Time*)	The time of a single preventive maintenance task
MPT (*Mean Processing Time*)	The jobs mean processing time
PT (*Processing Time*)	The total processing time at time t
pT (*processing Time*)	The processing time of a single job at time t

Table 3.2. Problem notation

to understand the proportion of jobs remaining (the experiment concludes when this reaches 100 %). Additionally, there's a variable valued with the average job processing time, determined after the processing times are generated. Three ratios follow this. The first ratio (CR) considers the time spent on corrective maintenance, the second (PR) accounts for the time spent on preventive maintenance, and the third encompasses all time aspects, including processing, corrective, and preventive maintenance.

The table 3.2 elucidates the symbols utilized. Concurrently, the Action space, classified as *Discrete*, admits two feasible values, namely 0 and 1, signifying the absence of maintenance and its presence, in that order. The reward algorithm is also articulated, with the method deriving the agent's incentive grounded in the data retrieved from the simulation, as follows:

$$reward = FP \cdot CR \cdot PR \cdot TR \tag{3.1}$$

The agent's ultimate goal, often depicted as the pike of product performance, relies on the meticulous calibration of several ratios within the operational framework, each of which corresponds to different facets of the maintenance schedule and overall system efficiency. These ratios, namely Corrective Ratio (CR), Preventive Ratio (PR), and Time Ratio (TR), serve

as the guiding metrics for the agent's decision-making process, ensuring that its actions contribute positively to the overarching goal of optimized productivity.

- Augmenting Corrective Ratio (CR): The CR is significantly influenced by 'cT,' which represents the duration of individual corrective maintenance tasks. Essentially, CR provides an inverse indication of system failures; a higher CR suggests fewer corrective interventions were required, indicative of a more reliable system. To enhance CR, the system must undergo a reduction in 'cT,' implying fewer and shorter corrective maintenance interventions due to fewer failures. This improvement is generally a consequence of enhanced system reliability, potentially resulting from superior components, improved operating procedures, or more effective preventive maintenance. By emphasizing activities that lead to fewer system failures, the agent not only works towards reducing downtime but also minimizes the resources expended on corrective actions, thereby contributing positively to operational efficiency.

- Amplifying Preventive Ratio (PR): In contrast, PR is associated with 'mT,' the time dedicated to preventive maintenance tasks. Unlike corrective maintenance, preventive tasks are scheduled and aim to preclude failures before they occur. Therefore, increasing 'mT' is indicative of a proactive approach, investing more time in preventive measures. While this initially appears to increase maintenance time (seemingly reducing operational efficiency), it effectively diminishes the likelihood of unplanned downtimes and disruptions, which are typically more detrimental to overall productivity. Consequently, enhancing PR involves a strategic increase in 'mT,' ensuring that the system remains in optimal working condition for as long as possible, thereby avoiding the steeper costs associated with corrective maintenance.

- Balancing with Time Ratio (TR): The TR plays a balancing role in this equation. It demands a holistic view, seeking to decrease both 'CT' (total corrective maintenance time) and 'MT' (total preventive maintenance time). At first glance, this seems contradictory to the objectives of CR and PR. However, TR represents an optimal

state of maintenance efficiency, where both preventive and corrective maintenance are minimized without compromising system reliability or lifespan. This equilibrium underscores the essence of resource optimization, requiring the agent to discern the perfect juncture between doing too little and doing too much, essentially finding the 'sweet spot' for maintenance activities.

Overlaying these aspects is the Failure Probability (FP), a crucial factor that modulates the importance of the balanced improvement in CR, PR, and TR. FP reflects the likelihood of system failures, inherently tying the objectives of reducing this probability to all maintenance activities. It represents the environmental uncertainty within which the system operates, suggesting that an absolute reduction in FP through improved system resilience would inherently enhance all other ratios. The agent's navigation through this intricate landscape of competing needs and operational constraints is a delicate act. It involves making informed, strategic decisions that balance short-term operational disruptions with long-term system reliability and performance improvements. This process, while complex, is essential in steering the system towards the maximization of product performance by continually fine-tuning these critical ratios in response to the dynamic operational environment. Concluding this sequence, we focus on the configuration of Machine Learning Model Deployment in the Simulation Model, ensuring a seamless integration of AI and simulation processes.

3.3.3 Machine Learning Model Deployment in the Simulation Model Configuration

ONNX (Open Neural Network Exchange) is an open-source format developed jointly by Microsoft and Meta to simplify the transfer of machine learning models across different frameworks and tools. By providing a standardized way of representing machine learning models, ONNX enables data scientists and engineers to train models in one framework and then deploy them in another one without the need to re-implement or integrate the model from scratch. In our case, the policy generated through ALPyne will be converted into an ONNX file and then uploaded to AnyLogic using the ONNX Helper library.

In addition, to test the policy in Anylogic with Onnx, it is necessary

to set up three functions in the software. The first function will take the observation variables as input and pass them to the "predict" function. The "predict" function will then output a value to the "action" variable, where 0 represents no maintenance and 1 represents maintenance. Finally, another function will use the output value ("action", 0 or 1) to determine whether to generate a maintenance job or not.

3.4 Experiments and Results

This section presents the results acquired. The efficacy of a policy was valued using metrics such as the number of preventive and corrective maintenance tasks, as well as the total makespan. The makespan denotes the complete duration needed to finish a task series, often utilized in optimization and planning to assess a system's proficiency. This section presents the experimental plan structured as follows:

1. Initially, we validate the formulation of state, action, and reward in alignment with the hypotheses from Branda et al.'s work [34].
2. Next, we validate the formulation of state, action, and reward, this time incorporating hypotheses of the simulation model exposed in the previous section.
3. We then proceed to scale the policy across five machines- Flow Shop, maintaining identical Weibull parameters for each.
4. Finally, we explore the Flow Shop configuration involving five distinct machines, differentiated by their respective Weibull parameters.

The hypothesis of the work [34] are the following:

- Job processing time is a Gaussian random variable with a mean of 100 and a standard deviation of 25.
- Weibull scale parameter (α) is 100.
- Weibull shape parameter (β) is 1.15.
- The average time required to carry out corrective maintenance is evenly distributed between 15 and 25.

N.jobs	50
N.machines	5
Processing Times	triangular(20,100,50)
Corrective Mant. Time	51 min
Preventive Mant. Time	17 min
Weibull scale parameter (α)	300-350-400-450-500
Weibull scale parameter (β)	1.5

Table 3.3. Simulation Model parameters different machine

- The average time required to carry out the planned maintenance is evenly distributed between 30 and 50.

Regarding the number of machines, although there are several machines in these problems, we chose to focus on solving the problem with a single machine. It follows that the Makespan value obtained in these settings (related to more than one machine) must be corrected for the case of a single machine. This correction is made considering the optimal sequence found by the algorithms proposed by Branda et al. and evaluating its effect only on the first machine of the Flow Shop. The solution considered is the one generated by the Genetic Algorithm, intending to minimise Makespan. Firstly, the sequence of jobs that constitutes the solution to problem considered in the paper was recreated within the simulation model in Anylogic. Subsequently, a series of simulations were conducted to verify the behaviour of the first flow shop machine subjected to the sequence of jobs decided by the genetic algorithm. After verifying the behaviour of the machine in the simulated environment (with the possibility of recreating fault conditions on the machine), a series of new experiments will be carried out, this time using the previously obtained policy for the evaluation of maintenance interventions. Finally, the results obtained with the use of Reinforcement Learning will be compared with the initial results from [90].

In Table 3.4, the experimental results concerning the optimal sequence of jobs obtained through the Genetic Algorithm proposed by Branda et al. ([34]) are presented.

The experiments reveal that deterministic solutions, like those in Branda et al.'s work [34], may not be optimal in real-world scenarios, such as those simulated in the stochastic Anylogic environment. This conclusion

Approach	Makespan	Number of Failures	Number of Maintenance Tasks
GA [34]	2461.9	17.5	1
DRL	2905.7	7.7	10.6

Table 3.4. The comparison of the mean values of the metrics considered

is drawn from observing the higher number of failures in the deterministic approach compared to the Deep Reinforcement Learning (DRL) method. Although the DRL approach results in a longer Makespan, it effectively reduces failures, suggesting a trade-off between Makespan length and reliability.

In terms of Makespan and machine failures (see table 3.4), the artificial intelligence-based approach significantly reduces failures compared to scenarios without scheduled maintenance. By adopting the same hypothesis as Branda et al., the AI agent effectively reduced machine failures, showcasing its ability to autonomously decide on preventive maintenance based on factors like failure probability at a given time and the duration since the last maintenance.

Furthermore, the preventive maintenance actions of the AI agent were benchmarked against the optimal maintenance strategies proposed by Branda et al. [34], using an empirical formula. The training performance of the RL algorithm, supported by the Stable Baselines3 library, is also discussed.

However, directly comparing the outcomes with Branda et al.'s paper is challenging due to differing assumptions. While Branda et al. prioritize corrective over preventive maintenance, our approach and underlying assumptions, including Weibull values and processing time distributions, differ. To facilitate this comparison, three key metrics were considered: Makespan, number of failures, and number of maintenance tasks.

3.4.1 Learning Curve and Hyperparameters

In the process of enhancing the model's learning efficacy, considerable attention was given to the configuration of the Proximal Policy Optimization (PPO) algorithm during the training phase. Hyperparameters play a critical role in determining the performance of the PPO algorithm, dictating how it will learn from the environment and adapt its strategies. While the majority of these parameters were kept at their default settings as per

the Stable Baselines3 framework, particular modifications were deemed necessary to fine-tune the learning process to our specific context and objectives.

Among the altered parameters, we adjusted the *gamma* (also known as the discount factor), which is instrumental in shaping the algorithm's strategy. It dictates the importance given to future rewards, a *gamma* value of 0.99 suggests a strategy that considers future rewards with minimal discounting, encouraging foresight in the algorithm's decision-making process.

Another critical parameter, gae_{lambda}, associated with Generalized Advantage Estimation (GAE), balances the bias-variance compromise in the estimation of the advantage function. By setting gae_{lambda} at 0.95, we aimed for a harmonious equilibrium, ensuring the model is sensitive enough to learn effectively from various scenarios without becoming skewed by anomalies in the data, thus maintaining a robust learning trajectory.

The $learning_{rate}$, set at 0.0001, was another pivotal hyperparameter. This parameter regulates the step size during the gradient descent process, which is the method the algorithm uses to adjust its internal values in response to the perceived errors in prediction. By carefully selecting a learning rate, we could ensure that the model adapts its understanding efficiently while preventing the oscillation or destabilization that can occur with rates that are too high, or the slow learning that can happen with rates that are too low. In the Figure 3.2 is displayed the goodness of the training process. Over 100,000 timesteps, the agent's training demonstrated an early and consistent growth in cumulative reward, indicating its evolving strategy. The x-axis is the Number of Timesteps and represents the cumulative number of steps or iterations in the learning process. The y-axis represents the reward value obtained by the agent. A higher value indicates a more favorable outcome, suggesting that the agent is making better decisions or predictions. Reagarding the learning trajectory, in the initial Phase (0 to approx. 20,000 timesteps) there is a rapid increase in rewards. This indicates that in the initial phases of training, the agent was quickly learning from its environment and improving its decision-making abilities. Such steep improvements are common at the beginning as the agent starts from a point of little to no knowledge. In the middle phase (20,000 to approx. 60,000 timesteps), the curve continues to rise but at a

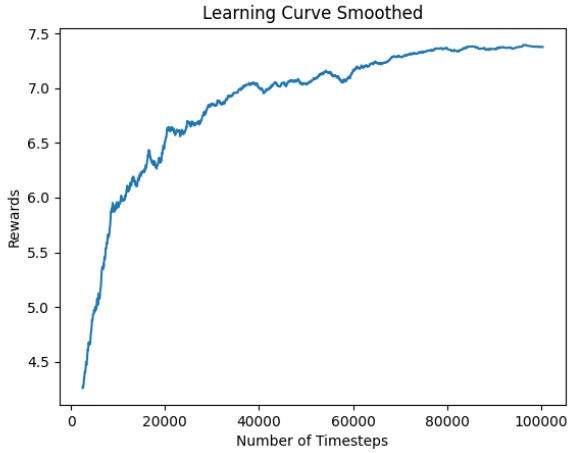

Figure 3.2. Learning curve

slower rate. This could be due to the agent refining its knowledge, dealing with more intricate situations, or approaching an optimal policy. It's typical for learning to decelerate after the initial surge as the agent starts to encounter more nuanced scenarios or diminishing returns. In the last phase (60,000 onwards), the curve begins to flatten, indicating that the agent's learning has plateaued. The rewards oscillate within a narrow range, suggesting the agent has potentially reached a near-optimal policy or strategy for the given environment or task.

3.4.2 Single Machine Results

When tested on a singular machine system, the resulting policy, over 20 replications, showed an average of 7 maintenance tasks and breakdowns each, amounting to a total makespan of approximately 3425 minutes. Interestingly, the maintenance task average was consistent with Branda et al.'s optimal calculations. The graph (Figure 3.3 showcases the failure probability across various time steps and how it correlates with maintenance activities. On the x-axis there is the time step and on the y-axis there is the Failure Probability. The height of each bar indicates the probability of failure at that specific time step. A higher bar suggests a higher risk

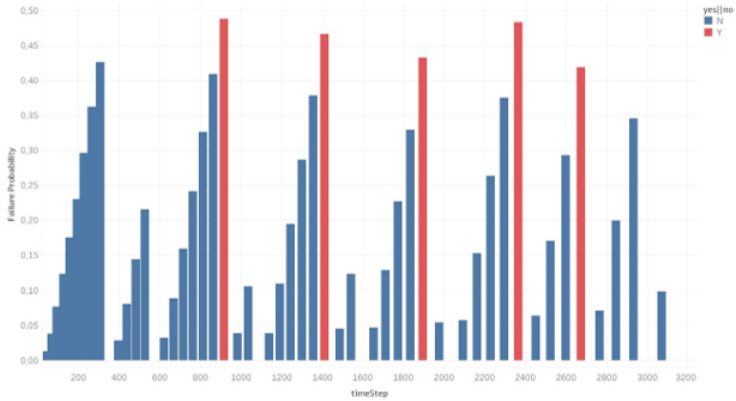

Figure 3.3. Average number of maintenance tasks.

of failure. Blue Bars ("No Maintenance"), demonstrate the failure probability when no maintenance was performed. Red Bars ("Maintenance"), showcase the failure probability when maintenance was carried out. A high red bar suggest the value of the failure probability at which the system performs maintenance. The system learns to perform maintenance at a higher value of failure probability.

3.4.3 Flow Shop Results

At this stage, the effectiveness of the policy generated was evaluated in simulation using the previously mentioned Onnx tool. First, the necessary functions were set up in the Anylogic environment. Subsequently, the policy file in the ".zip" format was converted into an ".onnx" file format to be loaded into Anylogic.

For this evaluation, the policy was tested on a configuration with five identical machines (i.e., having the same Weibull parameters). As in the previous experiments, a total of 20 replications were conducted to show how the metrics considered varied. Specifically, the optimal number of maintenance actions for each machine and the optimal number in total, considering the whole system, were compared with the actual number of maintenance actions, as well as the number of breakdowns and the makespan.

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

Figure 3.4. Average number of maintenance tasks considering the Flow Shop.

The results of all 20 replications consistently showed that the total number of actual maintenance actions was lower than the total optimal number. Furthermore, the total makespan did not increase significantly compared to the single-machine case (Table 3.5)

The figure 3.4 consists of a table illustrating the disparity between the actual number of maintenance interventions performed and the optimal number. The table provides information on the number of replications conducted, and for each replication, it presents the difference between the actual number of maintenance interventions and the optimal number calculated using an empirical formula derived from Branda et al. [91], considering both the individual machines and the system as a whole.

The graph (Figure 3.4) visually represents the trend of this difference across the 20 replications, with the most favorable outcomes evident when considering the system as a whole. The comprehensive view of the system allows for a better appreciation of the results.

Experiment	Makespan	Breakdown	Maintenance Tasks	Opt. Maintenance Tasks
Single Machine	3425,4	7,4	7,2	8
Flow Shop (5 machines)	3868,9	43,25	25,75	35,75
Flow Shop (5 different machines)	3892,8	47,6	33,85	46,9

Table 3.5. Comparison of the mean values results.

3.4.4 Flow Shop Results with Different Weibull Parameters

The policy was evaluated on five distinct machines, defined by their Weibull parameters. A comparison between two Flow Shops—one with identical machines and another with varying machines—displayed marginal differences in their outcomes. A heightened maintenance intervention count corresponded to increased failures. The summary table 3.5 reveals that there is no significant difference in the results between the two experimental settings. In a single machine setup, any maintenance or downtime affects the entire process. In contrast, in a multi-machine flow shop, maintenance can be scheduled in a staggered manner, allowing other machines to continue operating. This can lead to a more continuous workflow, despite having more machines that might individually require maintenance or experience downtime. Moreover, the table indicates that a higher number of failures corresponds to a higher number of maintenance interventions. In conclusion, our study stands out for tackling the intricate task of coordinating maintenance interventions across diverse machines within a production chain. Leveraging DRL, we've showcased the shortcomings of traditional tools in addressing this complex decision-making problem. The findings spotlight the potency of an optimized maintenance scheduling approach, surpassing conventional methods even with fewer maintenance tasks.

3.5 Conclusions

In this work the problem of maintenance scheduling using Deep Reinforcement Learning (DRL) was explored. Effective scheduling can help optimize the use of maintenance resources by reducing the total amount of time and manpower required for maintenance activities. In particular, maintenance scheduling is an essential aspect of ensuring the reliability and performance of production systems. Specifically, the problem of maintenance scheduling in a Flow Shop, a system in which production processes follow a fixed sequence and has certain constraints that make maintenance scheduling particularly challenging due to the interdependence of machines and the potential for bottlenecks, was analyzed. This was followed by an extensive exposition of the tool used to address this problem: DRL. This was chosen because it has emerged as a promising approach for optimizing maintenance scheduling in complex systems. More specifically, an integrated simulation tool and DRL algorithm was proposed, a comprehensive approach that combines the power of simulation with the intelligence of DRL to optimize the maintenance process and maximize productivity. The integrated simulation tool, Anylogic, provides a virtual environment to replicate the production line Flow Shop, enabling modeling and simulation of various machine operations, workflows, and maintenance events, while the DRL algorithm forms the central intelligence of the proposed approach. The experimental work performed consisted of a series of experiments that, through a trial-and-error process, served to develop the proposed DRL algorithm. First, the training was performed on a single-machine system.Then the policy that returned the best results was further implemented on a 5-machine system, both identical and different. The solutions derived from Branda et al. (2021) [90] were further enhanced through the integration of DRL. This approach is particularly designed to address the stochastic nature of real-world scenarios, which has been realistically simulated using the Anylogic software. Given the fact that the work mentioned supports different hypothesis we moved to futher experiments. The metrics used to evaluate the results obtained were: makespan, number of breakdowns, number of actual maintenance, and number of optimal maintenance. Comparison of the latter two showed how the usual tools often used for maintenance planning are ineffective in addressing this

decision problem, underscoring the need for new methodologies. In particular, they do not lend themselves well in systems that are closer to reality. Consequently, DRL appears to be a potential option for optimizing maintenance planning and scheduling in multi-machine manufacturing systems. Possible future developments could aim to study the adaptability and effectiveness of the proposed approach in more complex production contexts. This will require implementing a multi-product production system and considering different levels of stochasticity in the model. In addition, the investigation will include the integration of machine set-up times, implementation of stochastic maintenance times, maintenance costs, and other variables that contribute to a more complex and realistic problem scenario. In addition to the objectives already mentioned, possible future development could also focus on resource allocation for maintenance activities. This will involve determining the manpower, equipment, and other resources needed to perform maintenance tasks. By incorporating resource constraints, the study aims to establish a practical and realistic maintenance planning strategy. The goal to be pursued should be to test and refine the proposed integrated approach for maintenance planning and scheduling in multi-machine production systems through comprehensive experiments. The results of these trials will not only contribute to the advancement of Deep Reinforcement Learning techniques, but also provide insights into the practical application of these techniques in real production scenarios. In this way, the research will lead to the improvement of operational efficiency and productivity in practical settings, while also taking into account the complex and realistic factors mentioned above.

Chapter 4

Battery Swapping Station Service in a Smart Microgrid: A Multi-Method Simulation Performance Analysis

4.1 Introduction

A new paradigm of electricity production and consumption is being defined in the global landscape to fight the climate crisis and increase the overall efficiency of the energy system. Within this broader context, various strategies and methodologies have been proposed and developed by the scientific community to mitigate the human impact on the environment, particularly in transportation and energy [92].

In recent years, there has been a noticeable increase in the adoption of electric vehicles (EVs). However, certain challenges still hinder their widespread integration, and one of these challenges is the issue of extended recharging times. A potential solution to address this concern is offered by Battery Swapping Stations (BSSs) [93]. These stations allow users to swiftly exchange their depleted or nearly depleted batteries for fully charged ones [94]. They enable the reduction of recharging times to just a few minutes and allow for efficient scheduling of recharging times for depleted batteries [95]. As a result, the integration of BSSs can contribute

to a significant reduction in greenhouse gas (GHG) emissions. However, to maximize the benefits of BSSs, it is essential to integrate them into intelligent electrical systems, such as smart microgrids, that heavily rely on renewable energy sources [96]. Such integration can further enhance the sustainability of the transport system by utilizing renewable power generation and facilitating efficient energy management [97]. The management of BSSs should focus on two aspects: individual-level management to ensure the efficient utilization of the battery fleet in terms of recharging and decommissioning, and management of interactions with the electricity grid and other associated elements [98].

In the literature, considerable attention has been given to researching the optimal scheduling and coordination in microgrids with BSSs. In [38], the authors introduce a bi-level optimal scheduling model to address the coordination challenges between an isolated microgrid (IMG) and electric vehicle BSSs in scenarios involving multiple stakeholders. To solve the model, they devise a hybrid algorithm called JAYA-BBA, combining a real/integer-coded JAYA algorithm and the branch and bound algorithm (BBA). The research in [99] presents an optimal dispatch strategy for a grid-connected microgrid featuring a battery swapping station and renewable energy sources. By utilizing particle swarm optimization (PSO) and a penalty function method, the authors aim to minimize power interaction with the distribution network, enhancing the efficiency and effectiveness of the microgrid system. The authors in [41] introduce a binary integer linear programming (BILP) model to jointly plan electric vehicle battery swapping stations and distribution grids with centralized charging. Their methods incorporate probabilistic estimation, integer programming, and nonlinear optimization to address planning and optimization challenges. The paper in [100] employs a decision matrix to assess economically viable options for scheduling battery swapping stations within smart microgrids. This comprehensive approach examines factors such as demand response, renewable energy sources, and market interactions, shedding light on the potential profitability and success of battery swapping stations.

A key point to discuss when considering BSS integration in a migrogrid is the energy management flow through the system. In the literature, [101] proposes a real-time energy management model for a smart-community microgrid equipped with battery swapping and renewable energy sources.

Their real-time energy management algorithm, rooted in Lyapunov and queueing theory, ensures robust performance by adapting to uncertain future information. In [37], the authors put forth a method to optimize power source utilization within a network of flexible loads. Their approach employs Lagrange and KKT multipliers to tackle the optimization problem, considering constraints and uncertainties. The paper in [102] delves into the optimal power flow dynamics of microgrids featuring BSSs and Var Compensators (VCs). The authors examine energy arbitrage opportunities through BSSs and enhance voltage stability using VCs. The paper in [36] introduces a Cyber-Physical Energy Management System for optimizing the sizing and operation of networked nanogrids with a BSS. By utilizing various methodologies, the authors tackle challenges related to energy management, cyber defense, and robust optimization to design efficient and resilient nanogrids. The authors in [35] focus on optimal sizing of photovoltaic generation and battery-based energy storage in off-grid nanogrids. Through robust optimization approaches, they strive to address uncertainties and optimize capacity planning. The authors propose a mathematical model to optimize battery swapping station (BSS) loads for capacity enhancement in distribution systems. By maximizing demand response, they aim to improve utility operator performance.

Both the existing literature and the problem formulated here recognize the importance of integrating various components such as BSSs, Renewable Energy Sources, and storage units. This study highlights how these components interact within the microgrid and contribute to its overall operation and profitability. The authors in [103] present an optimized placement strategy for battery swap stations (BSSs) in microgrids with micro-pumped hydro storage, photovoltaic, wind, and geothermal distributed generators. Through mathematical formulations and optimization techniques, they address challenges associated with EV charging times and cost-effectively size BSSs for maximal efficiency. The study in [39] proposes a coordinated planning approach that integrates the charging and swapping stations with the active distribution network based on electric vehicle spatial–temporal load forecasting. Using algorithms and case studies, the authors explore the efficient spatial–temporal distribution of EV charging, providing valuable insights into planning and operational strategies.

In Table 4.1, there is a summary of the literature referenced in this

work in which are highlighted the methods and the main focuses of the different papers. The last row shows the method used in this work and the main focus in order to compare the proposal and the existing literature. While the existing literature employs various optimization techniques, algorithms, and mathematical models, this study also adds a simulation-based approach. This allows for a detailed analysis of the microgrid's operation, considering multiple components and their interactions in a dynamic manner. The use of Discrete Event Simulation and Agent-Based modeling adds a layer of realism and complexity to the analysis. The management of energy flows is a common theme across the literature and this study. The optimization of energy transfers, balancing supply and demand, and utilizing renewable energy sources are discussed in both contexts. Unlike the referenced literature, the method proposed here for scheduling, coordinating, and managing energy flows between the BSS and the microgrid utilizes two algorithms. The first algorithm rationalizes energy flows among different grid components, while the second algorithm sizes the storage formed by the second-life batteries. Downstream of these two algorithms, the discrete event simulation (DES) and agent-based multi-method simulation approach are used to implement the resulting decisions.

Integrating a BSS and a microgrid requires coordination and synchronization between the two entities. In many cases, game theory tools and models have been utilized to address the problem [98]. Various coordination models have been proposed, such as peer-to-peer and leader–follower models, which facilitate energy exchange and revenue generation for prosumers [104]. Ni et al. [105] introduce an alternative architecture called a charging-swapping-storage integrated station (CSSIS). The management system employs the Stackelberg Game, with the microgrid acting as the leader to determine the optimal operating point and maximize profit by considering the cost of internal energy exchange with the CSSIS.

Based on the scientific literature presented here, we explore the profitability of integrating a Battery Swapping Station within a microgrid in different settings. Compared to the previously mentioned approaches, the distinguishing feature of this work is the utilization of batteries that have reached the end of their useful life for electric cars. These batteries are repurposed for a second-life battery storage system [106] to improve the management of the battery fleet, considering their gradual degradation

even when discharging energy to the grid. The difference between first- and second-life batteries can be summed up as in [107], which presents a comparative analysis between new batteries used in electric vehicles (EVs) and second-life batteries repurposed for stationary applications, specifically energy storage systems (ESSs) [108, 109]. This integration involves implementing a management system that utilizes decommissioned batteries [40] from BSS users to create an Energy Storage System. The Energy Storage System serves as storage for two renewable energy power plants, namely photovoltaic and wind power plants, while also considering the presence of consumer loads within the microgrid. To examine this, we conduct a multi-method simulation [110] performance analysis of battery swapping station services in a smart microgrid. A similar approach is taken in [111], but the paper lacks a consideration of charging efficiency and battery degradation in its calculations and does not include renewable energy sources in its proposal. Performance is assessed by developing a digital model of a smart microgrid, which includes a battery swapping station, storage, renewable energy sources [112], and utilities. This study shares common research objectives with the existing literature, aiming to optimize microgrid operations through BSS integration. However, it uniquely focuses on evaluating the economic viability of this integration using a simulation-based analysis, incorporating DES and Agent-Based modeling for depth and realism. Notably, the study explores off-grid operation to enhance self-sufficiency and employs second-life batteries, adding an additional layer of sustainability and economic complexity to the analysis. In summary, this work aims to answer the following research questions:

- RQ1: How can battery swapping stations be optimally managed and integrated with renewable energy sources and microgrids to efficiently utilize the battery fleet?

- RQ2: What is the profitability and performance analysis of integrating a Battery Swapping Station within a microgrid using decommissioned batteries from electric vehicles to create a second-life battery storage system, and how does this impact the management of the battery fleet and gradual degradation?

The evaluation includes several key aspects:

Table 4.1. The survey of the existing literature and the methodological focus of the present work.

Reference	Method	Focus
Li, et al., 2018 [38]	Bi-level programming approach combining JAYA and BBA algorithms.	Optimizing scheduling between isolated microgrid (IMG) and BSS.
Yan, et al. 2019 [101]	Real-time energy management model using Lyapunov and queueing theory.	Integrating battery swapping, renewables, and residential load supply.
Infante, et al. 2022 [37]	Optimization using Lagrange and KKT multipliers with constraints.	Optimization of power source utilization in flexible load networks.
Jordehi, et al. 2020 [102]	Analyzing energy arbitrage opportunities and voltage stability enhancement.	BSS and Var Compensators for optimal power flow in microgrids.
Jordehi, et al. 2021 [103]	Mathematical optimization for BSS placement with mixed energy sources.	Efficient BSS placement considering various distributed generators.
He, et al. 2023 [39]	Algorithmic coordination of charging and swapping based on EV load forecasting.	Coordinated planning and management of EV charging and swapping.
Garcia-Guarin, et al. 2020 [100]	Decision matrix for assessing BSS scheduling strategies in smart microgrids.	Profitability and success of BSS integration in microgrids.
Ban, et al. 2019a [36]	Cyber-physical energy management with robust optimization for nanogrids.	Optimal sizing and operation of nanogrids with battery swapping.
Ban, et al. 2019b [35]	Formulation and optimization of PV and storage sizing in off-grid nanogrids.	Efficient sizing of renewable energy and storage components.
Song, et al. 2019 [99]	Optimization algorithm for dispatch strategy in grid-connected microgrids.	Minimizing power interaction with distribution network using BSS.

Reference	Method	Focus
Alharbi, et al. 2023 [113]	Mathematical model for optimizing BSS load flexibility in distribution.	Enhancing capacity of distribution systems with BSS load optimization.
Shaker, et al., 2023 [41]	Probabilistic estimation, integer programming, and nonlinear optimization.	Joint planning of BSS and distribution grid with centralized charging.
This work	Optimization of energy flows and size storage in a system involving a BSS and a microgrid. The resulting decisions implemented through DES and an AB multi-method simulation approach.	This work aligns with the existing literature in terms of optimizing microgrid operation with BSS integration, but it adds a distinct focus on evaluating profitability through simulation-based analysis.

- Market Share Served: This refers to the number of users served by the swapping station.

- User Behavior: User behavior is reflected in the state of charge (SOC) of the battery when brought for swapping.

- Electricity Consumption: This aspect encompasses the energy demand from the conventional grid and the withdrawal of energy from battery storage.

The paper is structured as follows. The first section focuses on the operating principles of each agent involved separately, as well as their interactions. The second section provides detailed definitions of the different aspects of the system and describes the hypotheses on which the work is based. The third section examines the relevant cost structures of the model, presents simulation results, and concludes the study.

4.2 Problem Formulation

The objective of this study is to evaluate the profitability of integrating a Battery Swapping Station (BSS) into a smart microgrid system. A multi-method simulation approach was employed for the analysis, in particular, AnyLogic software version 8.8.1. was utilized to create a digital twin of the microgrid. The approach employed combines Discrete Event Simulation ([114]) and Agent-Based modeling. The scheme of the digital twin model is as shown in Figure 4.1.

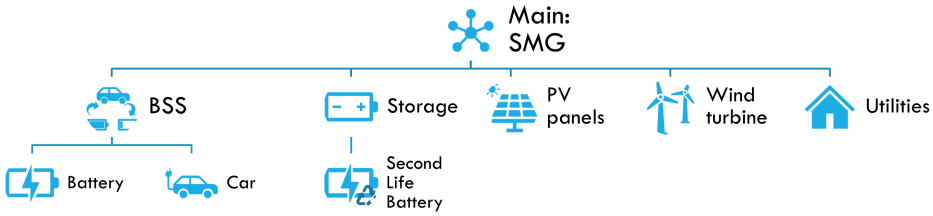

Figure 4.1. The simulation model hierarchy

In the simulation model, the top-level agent (Main) represents the smart microgrid, which consists of several components. These include the BSS containing two low-level agents: Battery and Car. Additionally, there is the Second-life Battery Storage unit, the two Renewable Energy Sources (RESs)—PV panels and a wind turbine—and finally, the Utilities component responsible for incorporating residential loads. The microgrid can operate in two modes: grid-connected or off-grid. In the grid-connected mode, it supplements internal generation with power from the traditional grid during periods of high demand. However, this study focuses on achieving off-grid operation, aiming for greater independence from the utility grid in terms of generation. Surplus renewable energy generated is utilized to charge the storage batteries, reducing the need to sell energy surplus to the utility grid. Consequently, there is a unidirectional energy flow from outside to inside the microgrid. The second-life battery storage can be recharged in two ways: overnight and daytime recharging. Overnight recharging occurs during off-peak hours when energy prices are lower. The process minimizes the cost of external energy and takes advantage of low demand from utilities. Overnight recharging is conducted

in standard mode, requiring approximately 5 h for a complete recharge (0.2 C). Daytime recharging takes place during the remaining hours of the day, utilizing renewable sources to recharge the batteries in the BSS. Each hour, an assessment takes place to determine if surplus energy is available after meeting the demand. If an energy surplus exists, a decision is made whether to use it for a standard recharge on multiple batteries or a fast recharge on a smaller number of batteries. If the surplus is insufficient to charge all discharged batteries with a standard recharge, a fast recharge is performed on as many batteries as possible. To perform a profitability analysis of the integration of BSSs into a smart microgrid, we identified two critical elements for optimization: minimizing the cost of energy exchange across all system components and determining the storage composition in terms of the number and mix of types. The overall operation of the digital twin involves managing generation and demand, simulating the optimal transfer of energy between them, and aligning it with the activities of the BSS. The management of energy flows in the microgrid is influenced by consumer demand and cost-effectiveness. Energy usage from different sources is determined by economic considerations while considering availability. In this study, microgrid management can be based on either a system where the microgrid is owned by a distribution system operator (DSO) or a consortium of users who act as both producers and passive consumers. The difference lies in the profit distribution, where the DSO profits in the first case, while individual users benefit from the cost difference between the current generation and external grid energy supply in the second case.

4.3 System Components and Operational Framework

To delve into the workings of a Battery Swap Station (BSS) integrated into a smart micro-grid, it is important to understand the system components involved. The system comprises a BSS, renewable energy sources (specifically a wind turbine and a photovoltaic panel farm), a storage system consisting of second-life batteries, and utilities, including residential loads.

In the BSS, users (the Car agents) can swap their depleted batteries

with fully charged ones, allowing them to continue using electric vehicles without the need for extended charging periods. The station manages the logistics of battery storage and retrieval.

The renewable energy sources play a vital role in the micro-grid.

The storage system, composed of second-life batteries, serves as an essential component of the micro-grid. It enables the storage of surplus energy generated by the renewable sources during periods of high production. This stored energy can be utilized during times of increased demand or when the renewable sources are less productive, ensuring a steady and reliable power supply.

The utilities represent the various loads within the micro-grid, including residential consumers. By integrating the BSS and renewable energy sources into the micro-grid, these utilities benefit from sustainable and efficient energy resources.

The simulation operates as follows (Figure 4.2). Each hour, a certain amount of energy is generated from the renewable sources and storage, while a certain amount of energy is required by the residential utilities and the BSS. These energy demands and supplies are compared and various logical processes are executed. The distribution of energy from generation to utilities is prioritized based on the unit cost of each energy source, with renewables having the highest priority, followed by storage and then the utility grid. When the renewable energy supply meets or exceeds demand, the demand can be fully met and the surplus energy is used to charge the second-life batteries in storage. If the renewable energy supply is less than demand, energy from storage or the utility grid is used to meet the demand. To determine which second-life battery to discharge to meet utility needs, the batteries are prioritized based on their state of health (SOH). The second-life batteries can be charged in two modes: nocturnal or diurnal. In the nocturnal mode, the batteries are charged using energy from the utility grid when it is at its lowest price. In the diurnal mode, surplus energy generated by the renewables is used to charge the batteries, either in a standard or fast mode depending on the amount of surplus energy available. As the batteries have a limited lifespan, the storage must be replenished periodically. The simulation model monitors the SOH of batteries in both the storage and the BSS and transfers batteries from the BSS to storage once they reach the end of their first life, to maximize the

use of batteries in line with the principles of the circular economy. The batteries in the simulation are modeled as agents, with a state chart that tracks their status, including fast charging, standard charging, discharging, storage, and end of life. The BSS model simulates various operations, including swapping, warehouse management, and battery charging process scheduling. The model takes into account technical and economic evaluations to account for all associated costs and revenues. An optimization model is employed every hour to select the generation source that meets users' requirements. The primary objective of this model is to achieve economic efficiency by determining the most cost-effective energy source.

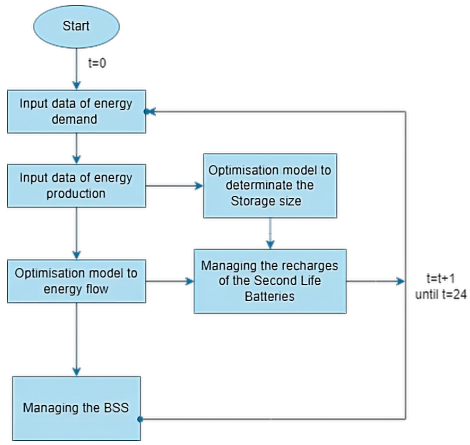

Figure 4.2. The simulation procedure

The optimization model incorporates costs associated with energy generation but does not consider revenues received from utilities for each generation.

To incorporate the optimization model into AnyLogic, the GLPK (GNU Linear Programming Kit) library was utilized. By leveraging the capabilities of the GLPK library, the digital twin's optimization model can effectively determine the optimal generation source, considering the objective of minimizing total costs. This integration ensures that the twin operates in an economically efficient manner, making informed decisions

on energy sourcing and minimizing overall costs.

$$z_1 = C_{WT} \cdot kWh_{WT}(t) + C_{PV} \cdot kWh_{PW}(t) + C_{SLBs} \cdot kWh_{SLBs}(t) + C_{UG}(t) \cdot kWh_{UG}(t) \tag{4.1}$$

$$kWh_{WT}(t) + kWh_{PW}(t) + kWh_{SLBs}(t) + kWh_{UG}(t) = Dem_{RL}(t) + BSS_{Limit}(t) \tag{4.2}$$

$$0 \leq kWh_{Wt}(t) \leq P_{Wt}(t) \tag{4.3}$$

$$0 \leq kWh_{PW}(t) \leq P_{PW}(t) \tag{4.4}$$

$$0 \leq kWh_{SLBs}(t) \leq kWh_{2discharge}(t) \tag{4.5}$$

$$0 \leq kWh_{UG}(t) < kWh_{UG_{Limit}} \tag{4.6}$$

The generation cost of a plant can be represented by a quadratic function of the generated power [115]. For example, for the i-th plant, it would be as follows:

$$C(P_i) = \alpha \cdot P^2(t) + \beta \cdot P_i(t) + \gamma \tag{4.7}$$

The coefficients α, β, and γ refer to maintenance, fuel, and labor costs, respectively. The cost per kWh of renewable generation is approximately zero, which is due to the absence of fuel usage and infrequent maintenance.

The objective function (Equation (4.1)) represents the total cost of generation offered to utilities. It includes costs associated with wind turbine generation (C_{WT}), photovoltaic (PV) generation (C_{PV}), energy from the storage composed of Second-Life Batteries (SLBs) (C_{SLBs}), and energy sourced from the utility grid ($C_{UG}(t)$).

Equation (4.2) represents the energy balance constraint, where the sum of energy generated from wind turbines ($kWh_{WT}(t)$), photovoltaic systems ($kWh_{PW}(t)$), storage composed of Second-Life Batteries ($kWh_{SLBs}(t)$), and the utility grid ($kWh_{UG}(t)$) equals the demand from utilities ($Dem_{RL}(t)$) and the limit of the Battery Swapping Station ($BSS_{Limit}(t)$).

Equations (4.3)–(4.6) represent constraints on the energy generation from each source, ensuring that the generated energy does not exceed the maximum power capacity of wind turbines ($P_{Wt}(t)$) and photovoltaic

systems ($P_{PW}(t)$), and the available energy for discharge from the storage ($kWh_{2discharge}(t)$). The constraint in Equation (4.6) ensures that the energy sourced from the utility grid ($kWh_{UG}(t)$) is within the limit ($kWh_{UG_{Limit}}$).

The generation cost of a plant can be represented by a quadratic function (Equation (4.7)), where α, β, and γ represent coefficients associated with maintenance, fuel, and labor costs, respectively. The cost per kWh of renewable generation is considered zero due to the absence of fuel usage and infrequent maintenance.

The cost of energy obtained from the utility grid is based on the average household user's payment, adjusted for the time of day. A 10% reduction is applied to exclude transmission costs from high or very high voltage to medium voltage, benefiting residential users participating in the microgrid.

The cost of energy from the storage composed of Second-Life Batteries (SLBs) is considered to be zero, since the costs are treated as initial investment costs, not on a per kWh basis.

The objective function represents the total cost of generation offered to utilities, while the decision variables represent the generation rates from each source to meet the utilities' demands. The cost coefficients in the model prioritize renewables and storage over externally sourced energy. The optimization of the objective function is subject to constraints, including the availability of each source at a specific time and the requirement to meet the entire demand.

4.3.1 Battery Swapping Station Operations

The battery swapping station model simulates different operations including the swapping process, warehouse management, and the battery charging process scheduling. The swapping demand is hourly [97], as in Figure 4.3, and refers to a general swap that must be associated with a specific battery category according to percentage defined in Table 4.2.

Table 4.2. Car categories.

Car Category	Percentage [%]	Capacity [kWh]
A	16.9	20
B	35.0	30
C	46.0	40
D	2.1	100

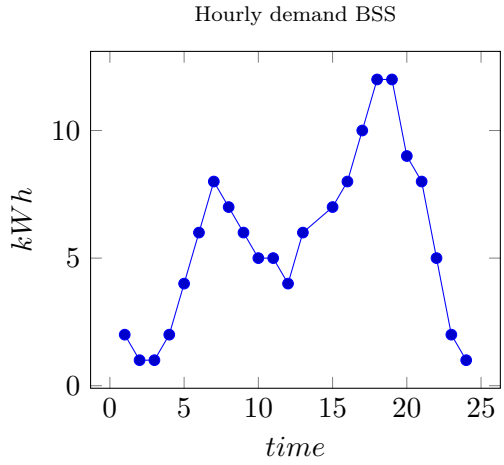

Figure 4.3. The hourly demand for the BSS

The swapping process involves cars and batteries. The model generates an agent, which is a car, and then checks to see if there is a suitable battery for that agent. If there is one, the discharged battery begins charging and the warehouse battery reaches the mounting state. The model also accounts for backlog. The purposes of the scheduling process are many; overall, it provides for charging a specified amount of batteries based on demand forecast for the next hour, and it allows for fast charging of the fewest number of batteries possible as well as charging as many batteries as possible whenever there is a surplus of energy generation. Without loss of generality, the interarrival time of cars is generated in the model following a Gamma distribution. The Gamma distribution is characterized by shape parameter α and scale parameter β, allowing for the manipulation

of variability levels. This continuous probability distribution includes the exponential distribution as a special case, as in [38] and in the present work (when $\alpha = 1$). By adjusting the value of α, it becomes possible to model variability that exceeds the exponential distribution when $\alpha < 1$, and variability that is lower than the exponential distribution when $\alpha > 1$. The versatility of the Gamma distribution enables the modeling of both high-variability and low-variability scenarios. The charging process can be categorized as either fast or standard. Fast charging fully charges a battery within one hour, utilizing an electric current equal to the battery's capacity (referred to as 1 C). On the other hand, standard charging (referred to as 0.2 C) takes approximately five hours to complete.

4.3.2 Renewable Energy Sources and Their Operation Parameters

The operations of Renewable Energy Sources (RESs), the wind turbine and the photovoltaic panel farm, are based on assumptions derived from the scientific literature. The wind turbine is considered to have a rated power of 500 kW and its rated output is modelled as in [38].

$$P^{\mathrm{WT}} = \begin{cases} 0 & v < v_{in}, v > v_{out} \\ {P_*}^{\mathrm{WT}} \cdot \frac{v - v_{in}}{v_* - v_{in}} & v_{in} \leq v \leq v_* \\ {P_*}^{\mathrm{WT}} & v_* \leq v < v_{out} \end{cases} \tag{4.8}$$

Its functioning parameters are described in Table 4.3 as in [116].

Table 4.3. Wind turbine parameters

Wind Turbine Parameters	
Speed	**[m/s]**
cut-in speed	2.5
cut-out speed	25
rated speed	12

The model includes just one wind turbine but it could also include more. In that case, the interaction between them must be taken into

Table 4.4. Solar farm panel parameters

Solar Farm Panels Parameters	
Efficiency (η_{PV}) [%]	14
Area (A_{PV}) [$\frac{m^2}{kW}$]	25.7
Mean Solar radiation (ζ) [$\frac{kW}{m^2}$]	0.266257

Table 4.5. Weibull distributions

Parameter	**Wind**	**Solar**
λ (scale parameter)	2	284
α (shape parameter)	12	7.31

account [116]. The solar farm provides for about 500 kW peak power and its rated output is modelled as in [38].

$$P_{PV} = A_{PV} \cdot \eta_{PV} \cdot \zeta \tag{4.9}$$

where A_{PV} is the irradiation area of each individual panel, η_{PV} is the panel's conversion efficiency, and ζ is the solar irradiation.

Its functioning parameters are described in Table 4.4:

It is meant to be made of polycrystalline panels whose efficiency is set at about 14% [117]. The photovoltaic plant is thought to operate only during certain hours of the day, from 5 a.m. to 8 p.m., whereas the wind turbine is supposed to work 24 hours per day. Wind and solar power systems are weather-dependent, which is why they are modelled using a stochastic model, which is, in both cases, the Weibull distribution (Equation (4.10)).

$$f(v, \lambda, k) = \frac{k}{\lambda} \cdot (\frac{x}{\lambda})^{k-1} \cdot e^{-(\frac{x}{\lambda})^{k-1}} \tag{4.10}$$

The distribution for modeling solar power [118] and wind power [116] is constructed using different shape and scale parameter values, as shown in Table 4.5.

The plants have been sized on the mean hourly demand coming from both utilities and BSS; residential loads have been modelled referring to a

summer day [115], as in Figure 4.4.

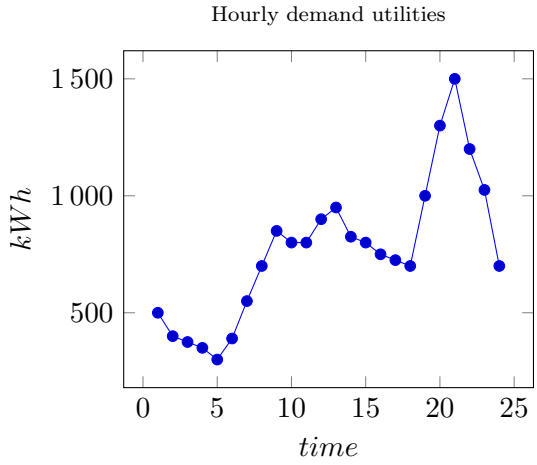

Figure 4.4. The hourly demand for the utilitie

4.3.3 Modeling and Operation of Energy Storage System

The energy storage system is primarily designed to address the unpredictability of wind and solar energy while facilitating peak shaving and load leveling. It operates by storing surplus energy when available (charging the batteries) and discharging the stored energy when the supply does not meet the demand from utilities.

The size of the storage, determined through an optimization model, represents the number of available batteries. Considering the average hourly demand and the need to address the unpredictability of wind and solar energy, as well as demand variability, the objective function of the model aims to minimize the gap between supply and demand. This allows the smart microgrid to primarily operate in an off-grid mode.

The optimization model is inspired by the one described in [99], but in this case, it runs only once using mean values instead of 24 times per day (hourly). Constraints primarily pertain to the various types of batteries. The optimization problem is solved using the GNU Programming Kit Linear (GPLK) library, connected with the AnyLogic simulator.

The provided equations and explanations enable a mathematical repre-

sentation of the optimization problem and provide a framework for finding the optimal solution in terms of battery categories and quantities. This contributes to the analysis of the system's profitability and efficiency. The objective function denoted in Equation (4.11) aims to be minimized.

$$z_2 = |P_{BSS} + P_{RL} + \Delta_{BSS} + \Delta_{RL} - P_{WT} - P_{PV} - (C_A \cdot SOH_{SLB}) \cdot N_A - (C_B \cdot SOH_{SLB}) \cdot N_B - (C_C \cdot SOH_{SLB}) \cdot N_C - (C_D \cdot SOH_{SLB}) \cdot N_D| \tag{4.11}$$

$$(C_A \cdot SOH_{SLB}) \cdot N_A + (C_B \cdot SOH_{SLB}) \cdot N_B + (C_C \cdot SOH_{SLB}) \cdot N_C + (C_D \cdot SOH_{SLB}) \cdot N_D \geq \Delta_{BSS} + \Delta_{RL} + P_{BSS} + P_{RL} - P_{WT} - P_{PV} \tag{4.12}$$

$$N_C > N_B \tag{4.13}$$

$$N_B > N_A \tag{4.14}$$

$$N_A > N_D \tag{4.15}$$

$$1 \leq N_A \tag{4.16}$$

$$1 \leq N_B \tag{4.17}$$

$$1 \leq N_C \tag{4.18}$$

$$1 \leq N_D \tag{4.19}$$

i = A,B,C,D refers to battery categories, and C_i represents each category's battery capacity, as in Table 4.2. The objective function (Equation (4.11)) represents the total cost of generation offered to utilities (z_2). It includes terms related to the energy demand from the battery swapping station (P_{BSS}), energy demand from residential loads (P_{RL}), variations in the battery swapping station energy demand (Δ_{BSS}), variations in the residential load energy demand (Δ_{RL}), energy supply from the wind turbine (P_{WT}), energy supply from the photovoltaic panel farm (P_{PV}), and costs associated with the second-life batteries in each category (C_A, C_B, C_C, C_D) multiplied by the state of health of the second-life batteries (SOH_{SLB}) and the number of batteries in each category (N_A, N_B, N_C, N_D).

The goal is to find values for N_A, N_B, N_C, N_D that minimize the ob-

jective function z_2 while satisfying the constraints specified in Equations (4.12)–(4.19). These constraints define relationships between the numbers of batteries in different categories, ensuring certain conditions are met. For example, the constraint $N_C > N_B$ indicates that the number of batteries in category C must be greater than the number in category B.

Second-life batteries have previously been used in electric vehicles during their first lives. The state of health (SoH) is a parameter that can be utilized to determine the beginning and end of a battery's life. The SoH is calculated by dividing the discarded capacity ($C_{discarded}$) by the total capacity (C_{total}) and multiplying by 100. The state of health is calculated as follows [119]:

$$SOH[\%] = \frac{C_{discarded}}{C_{total}} \cdot 100 \tag{4.20}$$

For second-life batteries, the first life typically lasts from 100% SoH to 80% SoH, while the second life continues until 60% SoH. However, the end of the first life can be defined differently based on various factors [120].

The energy storage system comprises a combination of second-life batteries with different percentages. The number of batteries in the Battery Swapping Station (BSS) and the storage batteries are proportional to the current percentage of electric vehicles on the road. Each battery category has a specific capacity, as shown in Table 4.2, which provides data from the Association of Foreign Automotive Companies Operating in Italy in the Distribution and Sales of Motor Vehicles in 2019.

The aging process of batteries and second-life batteries is taken into account by considering the lost cycles, which are influenced by factors such as operating temperature, depth of discharge, state of charge, and electric current [121]. In this study, the aging process assumes a linear relationship between lost cycles and decreasing capacity, as the total number of lost cycles is relatively low compared to the threshold at which a non-linear aging process would occur [122]. Batteries and second-life batteries are assumed to operate within a state of charge (SOC) range of 20% to 80%, which is modeled using a Beta distribution [122].

When discharging second-life batteries to meet utility needs, the order of discharge is determined based on the SoH of the batteries. The batteries can be ordered either ascendingly or descendingly, resulting in a uniform

or uneven level of degradation.

Second-life batteries can be charged in two modes: nocturnal or diurnal. Nocturnal charging involves recharging the batteries using energy from the utility grid at its lowest price, leading to a full recharge in standard mode for all discharged batteries. Diurnal charging, on the other hand, utilizes surplus energy generated by renewable plants. If the surplus energy is sufficient to increase the SOC of all discharged batteries by at least 20%, a standard charge is performed. Otherwise, the available surplus energy is used to charge as many batteries as possible in fast mode.

Both second-life batteries and batteries have a limited SoH and eventually need to be discarded. To optimize the use of batteries and adhere to the principles of the circular economy, the digital twin employs a mechanism to transfer batteries from the BSS to the storage when a battery from the BSS reaches the end of its first life and a battery from the storage reaches the end of its second life.

To facilitate the processes described above, batteries are modeled as agents with a state chart representing the different states they can transition through, including fast charging, standard charging, discharging, warehouse, and end of life. The batteries can change states on an hourly basis as the charging and discharging processes are managed on an hourly schedule, with batteries spending no more than an hour in the charging or discharging state, depending on the required SOC for a full charge.

4.4 Results

Anylogic was used to run simulations over a 20-year period. The decision to run the simulation over a 20-year period is rooted in the fact that this duration aligns with the expected lifetime of the key components within our system, including the PV panels and the wind power plant. The values assigned to parameters and variables vary across all simulation settings. In particular, those varying are the size of the energy storage (the number of batteries installed), range anxiety [38, 123] (represented as the SOC with which customers bring batteries back), utility grid energy usage limitations, the number of batteries the BSS uses to serve the customers, and a parameter that manages the degradation mode of batteries as uniform or uneven. Data collected through simulations are discussed in

the following sections. Simulations of the global digital twin's operation were conducted across 162 scenarios. These scenarios were differentiated by varying certain model parameters, including the following:

- Range anxiety of station customers: This parameter represents the state of charge at which batteries are returned to the station. The scenarios included 30%, 50%, and 70% of the nominal capacity of the battery.

- Number of batteries for customer service at the swap station: Scenarios were tested with 400, 500, and 600 batteries available for customer use.

- Number of second-life batteries installed in storage: Different scenarios considered 14, 22, and 30 second-life batteries integrated into the storage system.

- Limit of energy that can be drawn from the grid: Scenarios were evaluated with limits of 1800 kWh, 2500 kWh, and unlimited energy withdrawal from the grid.

- Uniform degradation of batteries/non-uniform degradation of batteries: This parameter explored the impact of batteries degrading uniformly or non-uniformly over time.

By exploring these various scenarios, the simulations aimed to analyze the performance and outcomes of the integrated system under different conditions and configurations.

4.4.1 Payback Period

The payback period is approximately 2 years; as shown in Figure 4.5, it increases as the number of second-life batteries installed in the storage increases. Each of them involves a purchasing and decommissioning cost, as well as the cost of a charger. It is also demonstrated that as the SOC increases, the payback period decreases. This is due to the lower amount of energy supplied by the BSS to the customers.

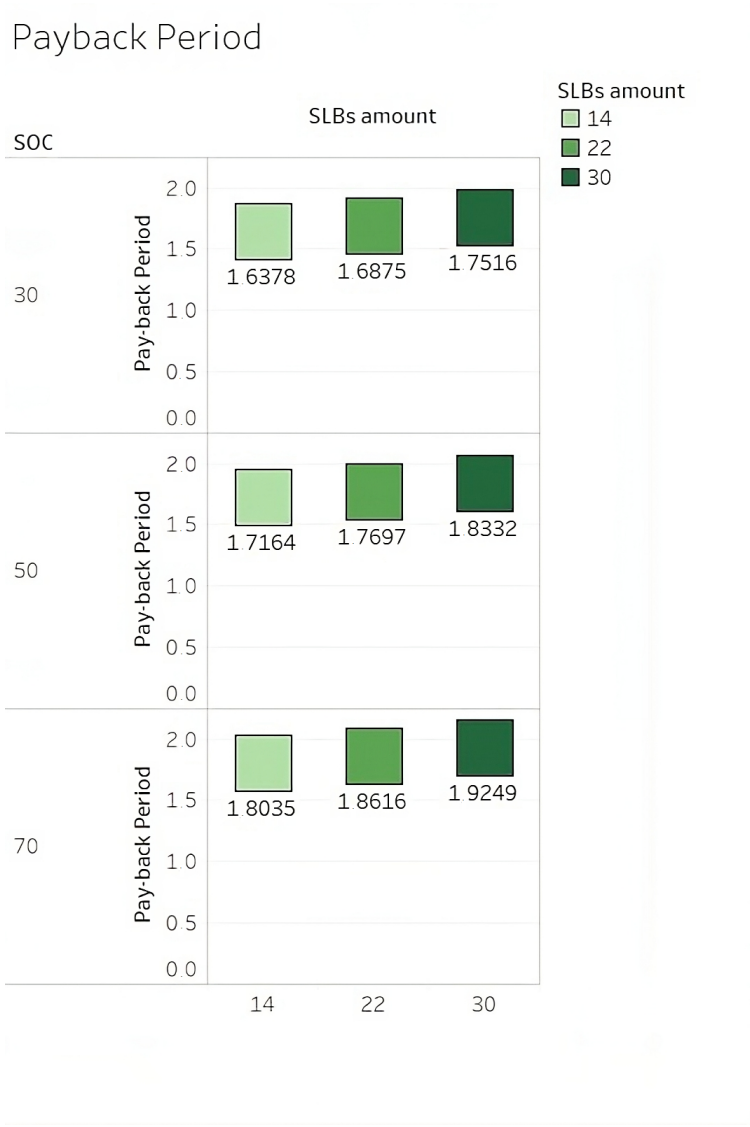

Figure 4.5. Overall Payback Period

4.4.2 Cash Flows

According to a positive payback period, the cash flows (Figure 4.6) are positive too ; they also turn out to be much higher than investments.

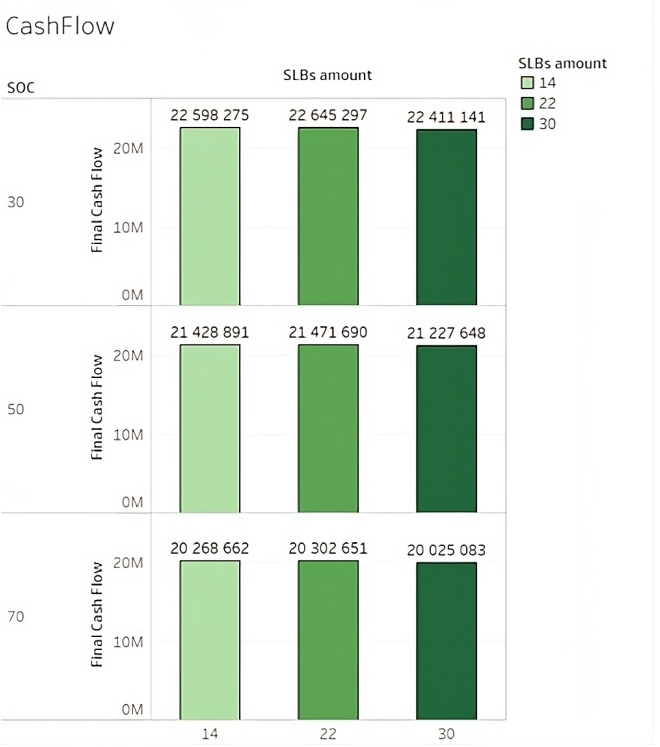

Figure 4.6. Final cash flow

4.4.3 Revenues

Revenues are sensitive to changes in range anxiety; as it rises, revenues fall because the BSS sells less energy to customers. Range anxiety represents the SOC with which the batteries are brought back to the swapping station.

Revenues can be also divided into amounts derived from the energy sold in the BSS and the amounts derived from the microgrid operation (actually from the energy sold to the residential utilities).

Microgrid revenues are higher than BSS revenues for a few reasons (Figure 4.7) . First, there is an higher demand, and second, a limit has been imposed to the BSS. Unlike residential loads, BSS loads can go into backlogs; those backlogs actually represent free swapping for customers. Residential backlogs could have been predicted using a demand side management model (Figure 4.8) .

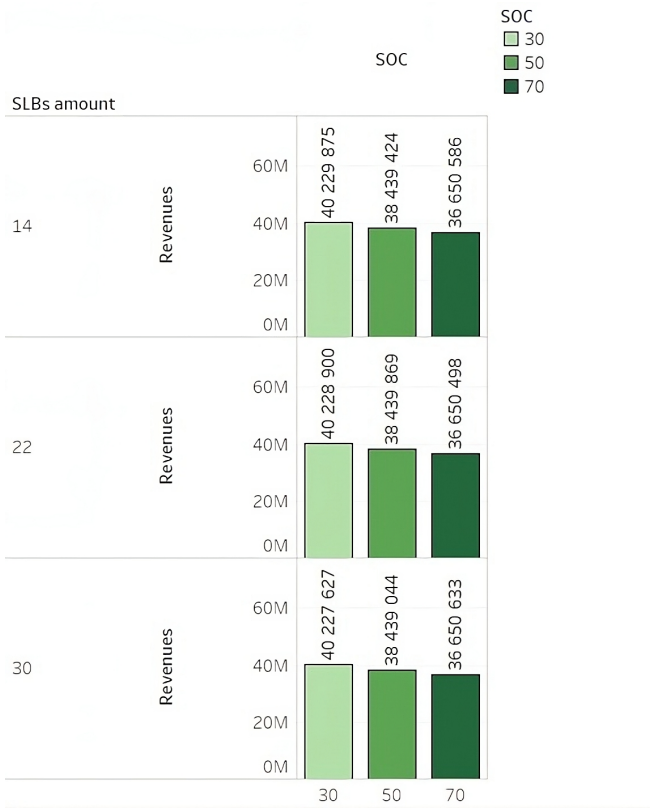

Figure 4.7. Revenues

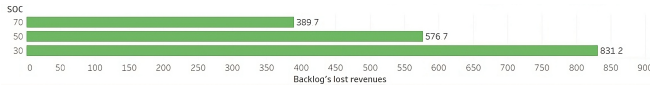

Figure 4.8. Backlog's lost revenues

4.4.4 Degradation Costs

The degradation cost of the batteries installed in the BSS is directly associated with the number of life cycles lost by each battery and, consequently, with the state of charge (SOC) at which they have been recharged (Figure 4.9) . A higher SOC indicates lower utilization of the batteries, resulting in a reduced degradation cost. In other words, when the batteries are charged to a higher SOC, they experience less degradation over their lifespan, leading to cost savings in terms of battery degradation.

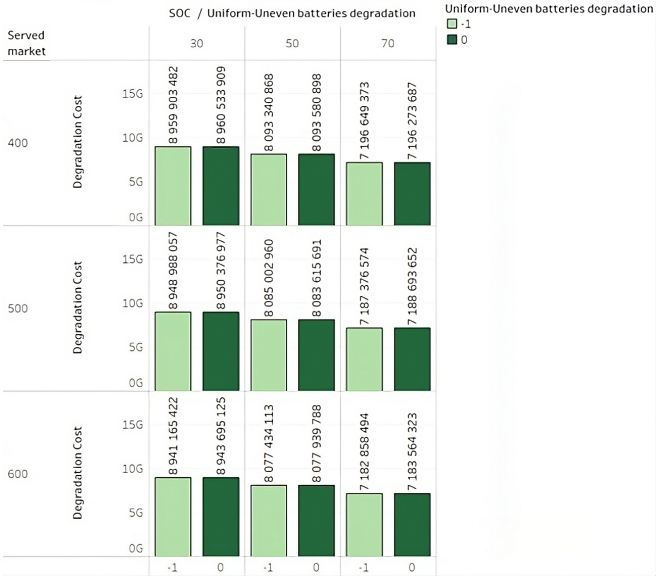

Figure 4.9. Degradation cost

4.4.5 Utility Grid Energy

Energy can be also traded from the utility grid. This happens whenever renewable sources do not meet energy demand; this source of energy relates to an higher unit cost (Figure 4.10) . Since the aim of the smart microgrid was mainly to work in off-grid mode, this observation was expected to occur as the number of storage batteries increased. However, this effect could have been reached only if the sizes of renewables also increased along

with the amount of storage batteries; otherwise, there would not be much energy to be charged into storage.

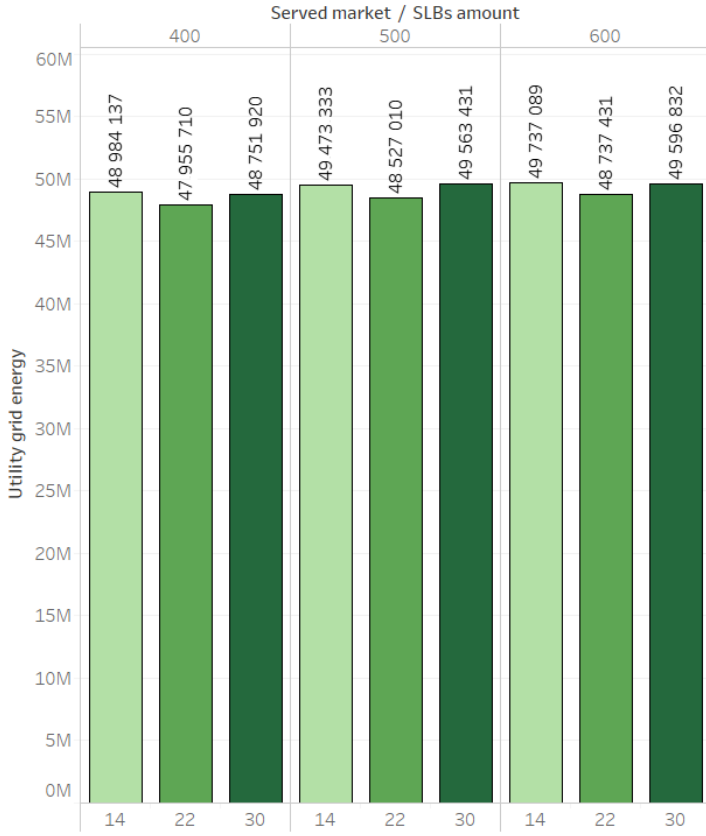

Figure 4.10. Utility grid energy

4.4.6 Discarded Batteries

Using a uniform degradation model for second-life batteries makes the most out of the batteries. This way, a lower number of batteries reach their end of life (Figure 4.11).

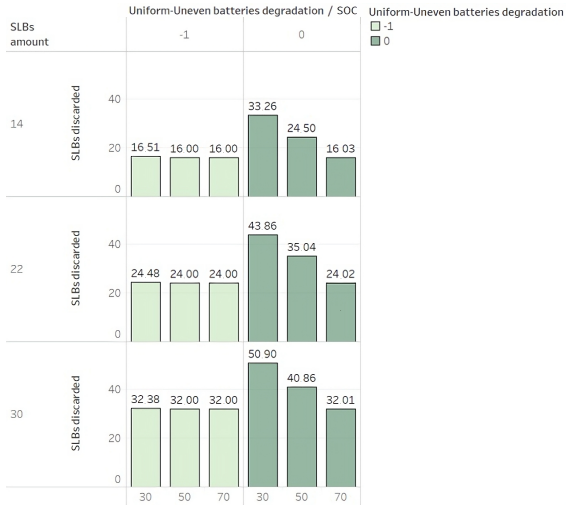

Figure 4.11. Discarded second-life batteries

The number of batteries which reach their end of life is different as the number of batteries in the BSS increases (Figure 4.12) ; this is because a higher number of them means they are less likely to be charged and discharged often, and so they will also be less likely to be discarded quickly.

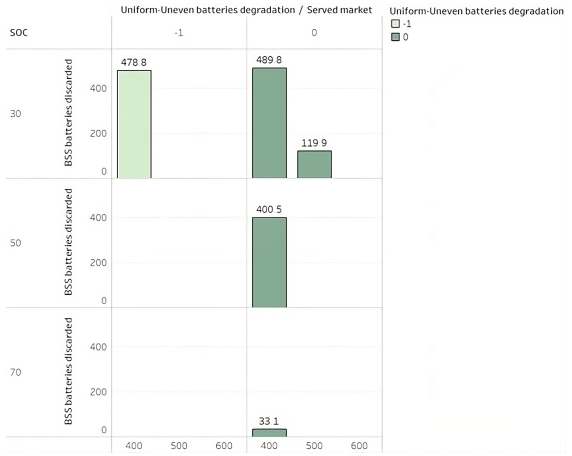

Figure 4.12. Discarded BSS Batteries

4.4.7 Comparing Paybacks of BSS in Smart Micro-Grid and as a Stand-Alone Entity

Simulations results show that inserting a BSS in a smart microgrid is a profitable business (Figure 4.13). It seems especially profitable when compared to the operations of a BSS connected to the utility grid (Figure 4.5). Not only are the revenues higher, but the overall payback period is also much shorter.

Figure 4.13. Paybackperiod—connected to utility grid

What we have learned from the analysis of the data collected through digital twin simulations suggests that the business investigated in this study is profitable. The cash flows generated by the system are significant, and the investment in the Battery Swap Station (BSS) is expected to be recovered in a short period. This indicates that integrating the BSS within a microgrid and combining it with alternative energy sources can yield better results compared to operating the BSS as a standalone entity.

Additionally, the reuse of second-life batteries which were previously used in electric vehicles proves to be a positive aspect of the business. These batteries have a lower market price compared to new electric batteries, and their longevity is considerable when they are managed. As a result, the cost of installing a storage unit using second-life batteries is optimal and fulfills its intended purpose. This approach allows for the full utilization of battery packs that would otherwise be prematurely recycled. The analyses conducted highlight the importance of a well-managed approach to the use of batteries in terms of degradation. By carefully managing the degradation process, optimal results can be achieved in terms of the timing of battery depletion. This knowledge can inform decision-making processes and contribute to the overall efficiency and success of the battery swapping system. Overall, the findings suggest that the integrated operation of a BSS within a microgrid, combined with the reuse of second-life batteries, has the potential to create a profitable and sustainable business model.

4.5 Conclusions

The BSS integrated into a smart micro-grid consists of a battery swapping station, renewable energy sources (wind turbine and photovoltaic panel farm), a storage system comprising second-life batteries, and utilities such as residential loads. Just like the existing literature, this study aims to optimize the operation of microgrids with BSS integration. However, here the emphasis is on evaluating the profitability of this integration, which involves comprehensive simulation-based analysis using AnyLogic software. This extends beyond optimization to consider economic viability. The proposal allows for efficient battery swapping, harnessing renewable energy, optimizing energy storage, and providing reliable and sustainable electricity to meet the demands of various consumers within the micro-grid. The results of the study demonstrate that contextualizing a battery swapping station within a microgrid and managing the two entities in a coordinated manner is more profitable compared to managing the station as a single entity. The coordinated management approach includes activities such as energy sales to residential consumers, the utilization of second-life batteries, and maximizing the use of renewable energy through storage. These strategies result in increased cash flows and significantly reduce the pay-

back period. The payback period for the integrated management approach is approximately 2 years considered as a mean value, compared to a mean value of 5 years for single management. The profitability of the investment is further confirmed through calculations of Net Present Value and Internal Rate of Return.

Reusing batteries from the battery swapping station in storage proves to be a viable business strategy. It enables the avoidance of premature battery disposal and extends their overall lifespan. The impact of the storage included in the model, in terms of energy delivered compared to the total, is approximately 6%. This value could be increased by sizing renewable plants differently. This can be achieved by following the evaluation provided through a sensitivity analysis regarding the impact of varying the peak power of renewable plants. This analysis may influence the percentage of energy used by each source and, consequently, impact the revenues.

The study also highlights the importance of considering the timing of battery depletion and how they are handled. Managing batteries with a uniform degradation pattern minimizes the number of depleted batteries at any given time, allowing for efficient replenishment of the storage system. On the other hand, a non-uniform degradation pattern may lead to suboptimal utilization of batteries, resulting in more spent batteries than necessary.

The limitations of the proposed approach mainly stem from the fact that the simulation model relies on certain functional assumptions that may not always hold true. While the study focuses on evaluating the profitability of microgrid BSS integration, it may not encompass all relevant economic factors or external influences, such as policy changes and technological advancement. To assess these limitation, there are several potential directions to further develop this work, and the following options are particularly interesting:

- Sensitivity analysis regarding the utilization of renewable sources.
- Load prediction for the Battery Swapping Station (BSS).
- Management of a cluster of microgrids.

The first point aims to investigate how the variation in the nominal capacity of renewable plants impacts the cash flows of the model. This

analysis would involve resetting the parameters defined in the current digital twin model. The second point involves adopting an alternative strategy to the current approach. It would entail removing the power constraint of the BSS and prioritizing its loads over residential ones. Consequently, it may be necessary to develop a demand-side management model for residential loads. The third point explores the management of a cluster of microgrids, allowing for energy trading between them. The objective is to optimize the coordination of loads in each microgrid and minimize reliance on the traditional grid.

These directions offer promising avenues for further research and can significantly enhance the current model. They provide opportunities to explore sensitivity, optimize load management, and investigate the benefits of interconnecting microgrids for energy trading.

Chapter 5

Conclusions

The integration of advanced technologies in Operations Management highlights the transformative potential of Artificial Intelligence (AI) and simulation tools, particularly within the context of Industry 4.0. In the exploration of Operations Management's evolution through the lens of Deep Reinforcement Learning (DRL) and simulation, it becomes evident that the convergence of these technologies promises a new epoch of operational efficiency and innovation. DRL, with its adaptive learning capabilities, coupled with the power of simulation, offers a robust mechanism that transcends traditional boundaries, making complex decision-making more intuitive and data-driven. This fusion not only optimizes current processes but also paves the way for designing novel strategies to meet unforeseen challenges.

By designing suited state, action, and reward function configurations for DRL, the research delivers methods specifically adapted for complex manufacturing scenarios. Furthermore, the detailed analysis of microgrid operations, facilitated by the combined application of Discrete Event Simulation and Agent-Based modeling, elucidates dynamic multi-component interactions with unprecedented clarity.

These investigative efforts significantly underscore the far-reaching impacts of AI and simulation in operations management, with specific revelations concerning enhanced production control, strategic maintenance planning, and energy grid management. The insights gleaned not only confirm the instrumental role of these technologies in current industrial

landscapes but also lay a robust groundwork for an adaptive Industry 4.0 ecosystem. By harnessing these technological advancements, industries are assured to evolve towards a state characterized by greater agility, efficiency, and environmental sustainability.

In this direction, we identify the contribution of the first paper presented (Chapter 2) in the integration of Reinforcement Learning with Deep neural networks to overseeing and organizing production within Flow Shop environments, effectively addressing various production system shortcomings. This cutting-edge approach delineates both the state and action spaces, facilitating the optimization of a production network's metrics, including work-in-progress (WIP) and throughput (TH). It also guides the selection of appropriate dispatching rule in a 5-machine Flow Shop context. Investigating in the methodology used that it is splited into two distinct sub-issues: dynamically adjusting WIP levels to manage TH and crafting novel dispatching criteria based on real-time system conditions. The DRL mechanism adeptly calibrated the optimal WIP quantity within the network, aligning the average with that observed in Practical Worst Case instances. Following this, the Reinforcement Learning entity identified the optimal dispatching protocol, drawing on insights gained during preliminary training. The trio of state, action, and reward configurations, especially under certain configurations, proved advantageous. Most prominently, one of the configuration stood out for its efficacy in reaching the Throughput objectives and mirroring the Practical Worst Case conditions, hence optimizing the DQN model's variables significantly. This configuration extended its utility to the Scheduling strategy as well.

Considering the Flow Shop configuration, the second work (Chapter 3) contributes in optimizing maintenance scheduling via Deep Reinforcement Learning. Central to the proposed solution is the integration of a simulation instrument and a DRL componet. This strategy unites simulation with DRL's adaptive intelligence, optimizing maintenance coordination to booster productivity. The chosen simulation platform, Anylogic, mirrors the flow shop's production line, operational flows, and maintenance occurrences. Concurrently, the DRL component operates as the brain behind the optimization. Experimental analysis in this research involved iterative tests refining the DRL part. Initial experiments explored various reward functions within a single-machine framework, followed by compara-

tive analysis of the outcomes. Subsequently, the most efficacious policy was deployed in a 5-machine setting, encompassing homogeneous and heterogeneous systems. Evaluation metrics encompassed makespan, breakdown frequency, actual versus optimal maintenance number of tasks performed, highlighting conventional maintenance planning tools inadequacies in such realistic, intricate systems. Thus, DRL emerges as a formidable contender for enhancing maintenance planning and scheduling in expansive manufacturing setups.

Focusing specifically on the unique application of simulation as a robust tool capable of handling complex activities within a Battery Swapping Station in a smart microgrid. In the last paper presented (Chapter 4), the decision-making process is entrusted to two algorithms that oversee energy flows and determine the sizing of storage composed of second-life batteries. Incorporated within a smart micro-grid, the Battery Swapping Station (BSS) configuration involves a ensemble of renewable energy source units (including a wind turbine and solar panels), a storage utilizing second-life batteries, and consumer utilities. While parallel studies have pursued the optimization of microgrids incorporating BSS, this investigation distinguishes itself by probing the economic feasibility of such integration, deploying an exhaustive simulation methodology via AnyLogic software, thereby transcending mere operational optimization to assess financial sustainability. Findings underscore that integrating a BSS within a microgrid, ensuring their unified management, yields higher profitability than isolated operations. This integrated control encompasses aspects like residential energy sales, second-life battery exploitation, and renewable energy maximization through strategic storage, collectively fostering enhanced revenue streams and diminishing investment recovery times. Notably, the integrated model achieves an average payback period of approximately two years, substantially shorter than the five-year period associated with standalone structures, with further financial viability evidenced through Net Present Value and Internal Rate of Return indices. Employing second-hand batteries from the BSS for storage purposes emerges as a profitable commercial move, avoiding early battery disposal and prolonging their operational life.

In essence, the studies provide a compelling narrative of transition and potential within key sectors, driving them towards a new phase of opera-

tional excellence. The profound implications revealed through this research signal a paradigm shift, where AI and simulation emerge as pivotal forces in optimizing industrial operations, fortifying sustainability measures, and ultimately, sculpting the future contours of global industries. The three papers significantly advance the scientific research on the fusion of simulation with intelligent systems, notably Artificial Intelligence and Deep Reinforcement Learning. Although the promising findings from these studies highlight the effectiveness and potential these solutions hold for both researchers and practitioners, they also indicate several areas that future research needs to further explore.

Beyond the specific topics discussed in each chapter, these primary areas are suggested for future research:

- Extended System Complexity: The next logical step is to test how well this approach works in more complex manufacturing settings. This means we need to look at production processes that have many different configurations, face various unexpected changes, and deal with real-world issues like specific machine setups, unpredictable maintenance needs, costs, and other elements that make industry operations challenging.

- Resource Optimization in Maintenance: The next phase of experimentation will involve assessing the resources dedicated to maintenance activities, encompassing an evaluation of the necessary manpower, equipment, costs, and other essential resources for conducting maintenance interventions. Further effort will be devoted to formulating a more pragmatic and realistic maintenance planning strategy that takes into account resource constraints.

- Strategic Enhancements for Energy System Optimization: The future development plan focuses on optimizing energy systems through three strategic adjustments: analyzing the financial implications of varying renewable plants' capacities, re-strategizing load prioritization by removing BSS power constraints and potentially revising residential demand-side management, and enhancing microgrid interconnectivity for efficient load coordination and reduced traditional grid reliance.

The methodology articulated in this research signifies a leap towards virtualized decision-support environments, resonating with the digital nerve of Industry 4.0. It emphasizes synthesizing real-time manufacturing insights navigating the unpredictabilities of industrial dynamics. This foundational work serves as a springboard for advancing systems in different operational scenarios.

In conclusion, this research marks significant advancements in industrial and energy management systems, employing cutting-edge methodologies like Deep Reinforcement Learning and simulation models. The innovative approach of integrating DRL with control and maintenance scheduling in flow shop systems addresses complexities and interdependencies, optimizing resources and uptime. Similarly, the strategic incorporation of a Battery Swapping Station (BSS) within a smart micro-grid, focusing on coordinated management, has proven to be economically viable, ensuring sustainable and efficient energy use. These methodologies, particularly when applied in the context of Industry 4.0, pave the way for more resilient, adaptable, and efficient systems. The promising results underscore the potential for further exploration and application in more complex scenarios, enhancing operational efficiency across various industries. Future research, building on these foundations, could delve deeper into real-time adaptability, multi-system coordination, and broader resource management, contributing substantially to the fields of industrial maintenance and sustainable energy management.

Bibliography

[1] W.J. Hopp and M.L. Spearman, *Factory Physics: foundation of manufacturing management*, Second edition edn, Irwin/McGraw-Hill, 2001. ISBN ISBN 0-256-24795-1.

[2] J.A. Palombarini and E.C. Martínez, End-to-end on-line rescheduling from Gantt chart images using deep reinforcement learning, *International Journal of Production Research* (2021), 1–30. doi:10.1080/00207543.2021.2002963.

[3] N.O. Fernandes, M. Thürer, T.M. Pinho, P. Torres and S. Carmo-Silva, Workload control and optimised order release: an assessment by simulation, *International Journal of Production Research* **58**(10) (2020), 3180–3193. doi:10.1080/00207543.2019.1630769.

[4] A. Esteso, D. Peidro, J. Mula and M. Díaz-Madroñero, Reinforcement learning applied to production planning and control, *International Journal of Production Research* (2022). doi:10.1080/00207543.2022.2104180.

[5] D. Ivanov, B. Sokolov, W. Chen, A. Dolgui, F. Werner and S. Potryasaev, A control approach to scheduling flexibly configurable jobs with dynamic structural-logical constraints, *IISE Transactions* **53**(1) (2021), 21–38. doi:10.1080/24725854.2020.1739787.

[6] A. Grassi, G. Guizzi, L.C. Santillo and S. Vespoli, A semi-heterarchical production control architecture for industry 4.0-based manufacturing systems, *Manufacturing Letters* **24** (2020), 43–46. doi:10.1016/j.mfglet.2020.03.007.

[7] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, Second edition edn, The MIT Press Cambridge, 2018. ISBN ISBN 978-0-262-19398-6.

[8] J. Heger and T. Voss, Dynamically adjusting the k-values of the ATCS rule in a flexible flow shop scenario with reinforcement learning, *International Journal of Production Research* (2021), 1–15. doi:10.1080/00207543.2021.1943762.

[9] D. Shi, W. Fan, Y. Xiao, T. Lin and C. Xing, Intelligent scheduling of discrete automated production line via deep reinforcement learning, *International Journal of Production Research* **58**(11) (2020), 3362–3380. doi:10.1080/00207543.2020.1717008.

[10] A. Dolgui, S.E. Hashemi-Petroodi, S. Kovalev and M.Y. Kovalyov, Profitability of a multi-model manufacturing line versus multiple dedicated lines, *International Journal of Production Economics* **236** (2021), 108113. doi:https://doi.org/10.1016/j.ijpe.2021.108113.

[11] A. Dolgui and D. Ivanov, 5G in digital supply chain and operations management: fostering flexibility, end-to-end connectivity and real-time visibility through internet-of-everything, *International Journal of Production Research* **60**(2) (2022), 442–451. doi:10.1080/00207543.2021.2002969.

[12] H. Hu, X. Jia, Q. He, S. Fu and K. Liu, Deep reinforcement learning based AGVs real-time scheduling with mixed rule for flexible shop floor in industry 4.0, *Computers and Industrial Engineering* **149**(August) (2020), 106749. doi:10.1016/j.cie.2020.106749.

[13] H. Emmons and G. Vairaktarakis, *Flow shop scheduling: theoretical results, algorithms, and applications*, Vol. 182, Springer Science & Business Media, 2012.

[14] H.H. Miyata and M.S. Nagano, The blocking flow shop scheduling problem: A comprehensive and conceptual review, *Expert Systems with Applications* **137** (2019), 130–156.

[15] E. González-Neira, J. Montoya-Torres and D. Barrera, Flow-shop scheduling problem under uncertainties: Review and trends, *International Journal of Industrial Engineering Computations* **8**(4) (2017), 399–426.

[16] H. Wang, B.R. Sarker, J. Li and J. Li, Adaptive scheduling for assembly job shop with uncertain assembly times based on dual Q-learning, *International Journal of Production Research* (2020), 1–17. doi:10.1080/00207543.2020.1794075.

[17] C.H. Wu, F.Y. Zhou, C.K. Tsai, C.J. Yu and S. Dauzere-Peres, A deep learning approach for the dynamic dispatching of unreliable machines in reentrant production systems, *International Journal of Production Research* **58**(9) (2020), 2822–2840. doi:10.1080/00207543.2020.1727041.

[18] C.D. Hubbs, C. Li, N.V. Sahinidis, I.E. Grossmann and J.M. Wassick, A deep reinforcement learning approach for chemical production scheduling, *Computers and Chemical Engineering* **141** (2020). doi:10.1016/j.compchemeng.2020.106982.

[19] C. Hofmann, C. Krahe, N. Stricker and G. Lanza, Autonomous production control for matrix production based on deep Q-learning, *Procedia CIRP* **88** (2020), 25–30. doi:10.1016/j.procir.2020.05.005.

[20] K. Arviv, H. Stern and Y. Edan, Collaborative reinforcement learning for a two-robot job transfer flow-shop scheduling problem, *International Journal of Production Research* **54**(4) (2016), 1196–1209. doi:10.1080/00207543.2015.1057297.

[21] J.-h. Lee, H.-j. Kim and J.-h. Lee, Reinforcement learning for robotic flow shop scheduling with processing time variations variations, *International Journal of Production Research* (2021), 1–23. doi:10.1080/00207543.2021.1887533.

[22] A. Dolgui, D. Ivanov, S.P. Sethi and B. Sokolov, Scheduling in production, supply chain and Industry 4.0 systems by optimal control: fundamentals, state-of-the-art and applications, *International Journal of Production Research* **57**(2) (2019), 411–432. doi:10.1080/00207543.2018.1442948.

[23] A. Oukil and A. El-Bouri, Ranking dispatching rules in multi-objective dynamic flow shop scheduling: a multi-faceted perspective, *International Journal of Production Research* **59**(2) (2021), 388–411. doi:10.1080/00207543.2019.1696487.

[24] H. Kim, D.E. Lim and S. Lee, Deep Learning-Based Dynamic Scheduling for Semiconductor Manufacturing with High Uncertainty of Automated Material Handling System Capability, *IEEE Transactions on Semiconductor Manufacturing* **33**(1) (2020), 13–22. doi:10.1109/TSM.2020.2965293.

[25] K. Xia, C. Sacco, M. Kirkpatrick, C. Saidy, L. Nguyen, A. Kircaliali and R. Harik, A digital twin to train deep reinforcement learning agent for smart manufacturing plants: Environment, interfaces and intelligence, *Journal of Manufacturing Systems* **June** (2020), 1–21. doi:10.1016/j.jmsy.2020.06.012.

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

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

[28] C. Varnier and N. Zerhouni, Scheduling predictive maintenance in flow-shop, in: *Proceedings of the IEEE 2012 Prognostics and System Health Management Conference (PHM-2012 Beijing)*, IEEE, 2012, pp. 1–6.

[29] J.F. Arinez, Q. Chang, R.X. Gao, C. Xu and J. Zhang, Artificial intelligence in advanced manufacturing: Current status and future outlook, *Journal of Manufacturing Science and Engineering* **142**(11) (2020).

[30] L. Wang, From intelligence science to intelligent manufacturing, *Engineering* **5**(4) (2019), 615–618.

[31] J. Patterson and A. Gibson, *Deep learning: A practitioner's approach*, " O'Reilly Media, Inc.", 2017.

[32] M. Van Otterlo and M. Wiering, Reinforcement learning and markov decision processes, in: *Reinforcement learning*, Springer, 2012, pp. 3–42.

[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, Proximal policy optimization algorithms, *arXiv preprint arXiv:1707.06347* (2017).

[34] A. Branda, D. Castellano, G. Guizzi and V. Popolo, Metaheuristics for the flow shop scheduling problem with maintenance activities integrated, *Computers & Industrial Engineering* **151** (2021), 106989.

[35] M. Ban, J. Yu and M. Shahidehpour, Optimal sizing of PV and battery-based energy storage in an off- grid nanogrid supplying batteries to a battery swapping station, *Journal of Modern Power Systems and Clean Energy* **7**(2) (2019), 309–320. doi:10.1007/s40565-018-0428-y.

[36] M. Ban, M. Shahidehpour, J. Yu and Z. Li, A Cyber-Physical Energy Management System for Optimal Sizing and Operation of Networked Nanogrids With Battery Swapping Stations, *IEEE Transactions on Sustainable Energy* **10**(1) (2019), 491–502. doi:10.1109/TSTE.2017.2788056.

[37] W. Infante and J. Ma, Multistakeholder Planning and Operational Strategy for Electric Vehicle Battery Swapping Stations, *IEEE Systems Journal* **16**(3) (2022), 3543–3553. doi:10.1109/JSYST.2021.3123254.

[38] Y. Li, Z. Yang, G. Li, Y. Mu, D. Zhao, C. Chen and B. Shen, Optimal scheduling of isolated microgrid with an electric vehicle battery swapping station in multi-stakeholder scenarios: A bi-level programming approach via real-time pricing, *Applied Energy* **232**(April) (2018), 54–68. doi:10.1016/j.apenergy.2018.09.211.

[39] C. He, J. Zhu, A. Borghetti, Y. Liu and S. Li, Coordinated planning of charging swapping stations and active distribution network based on EV spatial-temporal load forecasting, *IET Generation, Transmission & Distribution* (2023). doi:10.1049/gtd2.12915.

[40] Y. Deng, Y. Zhang and F. Luo, Operational Planning of Centralized Charging Stations Using Second-Life Battery Energy Storage Systems, *IEEE Transactions on Sustainable Energy* **3029**(c) (2020), 1–1. doi:10.1109/tste.2020.3001015.

[41] M.H. Shaker, H. Farzin and E. Mashhour, Joint planning of electric vehicle battery swapping stations and distribution grid with centralized charging, *Journal of Energy Storage* **58**(November 2022) (2023), 106455. ISBN ISBN 6135785311. doi:10.1016/j.est.2022.106455.

[42] M.G. Marchesano, G. Guizzi, S. Vespoli and G. Ferruzzi, Battery Swapping Station Service in a Smart Microgrid: A Multi-Method Simulation Performance Analysis, *Energies* **16**(18) (2023). doi:10.3390/en16186576.

[43] O. Battaïa, A. Dolgui and N. Guschinsky, Optimal cost design of flow lines with reconfigurable machines for batch production, *International Journal of Production Research* **58**(10) (2020), 2937–2952. doi:10.1080/00207543.2020.1716092.

[44] J. Lee, H. Davari, J. Singh and V. Pandhare, Industrial Artificial Intelligence for industry 4.0-based manufacturing systems, *Manufacturing Letters* **18** (2018), 20–23. doi:10.1016/j.mfglet.2018.09.002.

[45] R. Rai, M.K. Tiwari, D. Ivanov and A. Dolgui, Machine learning in manufacturing and industry 4.0 applications, *International Journal of Production Research* **59**(16) (2021), 4773–4778. doi:10.1080/00207543.2021.1956675.

[46] L. Zhou, Z. Jiang, N. Geng, Y. Niu, F. Cui, K. Liu and N. Qi, Production and operations management for intelligent manufacturing: a systematic literature review, *International Journal of Production Research* **60**(2) (2022), 808–846. doi:10.1080/00207543.2021.2017055.

[47] A. Kusiak, Convolutional and generative adversarial neural networks in manufacturing, *International Journal of Production Research* **58**(5) (2020), 1594–1604. doi:10.1080/00207543.2019.1662133.

[48] J. Xu, S. Liang, X. Ding and R. Yan, A zero-shot fault semantics learning model for compound fault diagnosis, *Expert Systems with Applications* **221**(January) (2023), 119642. doi:10.1016/j.eswa.2023.119642.

[49] J. Deng, S. Sierla, J. Sun and V. Vyatkin, Reinforcement learning for industrial process control: A case study in flatness control in steel industry, *Computers in Industry* **143**(June) (2022), 103748. doi:10.1016/j.compind.2022.103748.

[50] P. Priore, B. Ponte, J. Puente and A. Gómez, Learning-based scheduling of flexible manufacturing systems using ensemble methods, *Computers and Industrial Engineering* **126**(September) (2018), 282–291. doi:10.1016/j.cie.2018.09.034.

[51] Y. Li, J. Wang, Z. Huang and R.X. Gao, Physics-informed meta learning for machining tool wear prediction, *Journal of Manufacturing Systems* **62**(June 2021) (2022), 17–27. doi:10.1016/j.jmsy.2021.10.013.

[52] Q. Wang, W. Jiao, P. Wang and Y.M. Zhang, A tutorial on deep learning-based data analytics in manufacturing through a welding case study, *Journal of Manufacturing Processes* **63**(May 2020) (2021), 2–13. doi:10.1016/j.jmapro.2020.04.044.

[53] C.P. Nielsen, A. Avhad, C. Schou and E. Ribeiro da Silva, Control system architecture for matrix-structured manufacturing systems, *Computers in Industry* **146**(December 2022) (2023), 103851. doi:10.1016/j.compind.2023.103851.

[54] J. Leng, C. Jin, A. Vogl and H. Liu, Deep reinforcement learning for a color-batching resequencing problem, *Journal of Manufacturing Systems* **56** (2020), 175–187. doi:https://doi.org/10.1016/j.jmsy.2020.06.001.

[55] A. Valet, T. Altenmüller, B. Waschneck, M.C. May, A. Kuhnle and G. Lanza, Opportunistic maintenance scheduling with deep reinforcement learning, *Journal of Manufacturing Systems* **64** (2022), 518–534. doi:https://doi.org/10.1016/j.jmsy.2022.07.016.

[56] J.P. Usuga-Cadavid, S. Lamouri, B. Grabot and A. Fortin, Using deep learning to value free-form text data for predictive maintenance, *International Journal of Production Research* **60**(14) (2022), 4548–4575. doi:10.1080/00207543.2021.1951868.

[57] A. Kuhnle, J.P. Kaiser, F. Theiß, N. Stricker and G. Lanza, Designing an adaptive production control system using reinforcement learning, *Journal of Intelligent Manufacturing* (2020). doi:10.1007/s10845-020-01612-y.

[58] M.G. Marchesano, G. Guizzi, L.C. Santillo and S. Vespoli, A Deep Reinforcement Learning approach for the throughput control of a Flow-Shop production system, *IFAC-PapersOnLine* **54**(1) (2021), 61–66. doi:https://doi.org/10.1016/j.ifacol.2021.08.006.

[59] M.G. Marchesano, G. Guizzi, L.C. Santillo and S. Vespoli, Dynamic Scheduling in a Flow Shop Using Deep Reinforcement Learning, in: *Advances in Production Management Systems. Artificial Intelligence for Sustainable and Resilient Production Systems*, A. Dolgui, A. Bernard,

D. Lemoine, G. von Cieminski and D. Romero, eds, Springer International Publishing, Cham, 2021, pp. 152–160. ISBN ISBN 978-3-030-85874-2.

[60] J. Lohmer and R. Lasch, Production planning and scheduling in multi-factory production networks: a systematic literature review, *International Journal of Production Research* (2020), 1–27. doi:10.1080/00207543.2020.1797207.

[61] S. Vespoli, G. Guizzi, E. Gebennini and A. Grassi, A novel throughput control algorithm for semi-heterarchical industry 4.0 architecture, *Annals of Operations Research* (2021). doi:10.1007/s10479-021-04184-z.

[62] A. Dolgui, F. Sgarbossa and M. Simonetto, Design and management of assembly systems 4.0: systematic literature review and research agenda, *International Journal of Production Research* **60**(1) (2022), 184–210. doi:10.1080/00207543.2021.1990433.

[63] S. Vespoli, A. Grassi, G. Guizzi and V. Popolo, Generalised Performance Estimation in Novel Hybrid MPC Architectures: Modeling the CONWIP Flow-Shop System, *Applied Sciences* **13**(8) (2023). doi:10.3390/app13084808.

[64] G. Zhou, Z. Chen, C. Zhang and F. Chang, An adaptive ensemble deep forest based dynamic scheduling strategy for low carbon flexible job shop under recessive disturbance, *Journal of Cleaner Production* **337**(28) (2022), 130541. doi:10.1016/j.jclepro.2022.130541.

[65] D. Huang, Z. Mao, K. Fang and L. Chen, Solving the shortest path interdiction problem via reinforcement learning, *International Journal of Production Research* (2021), 1–18. doi:10.1080/00207543.2021.2002962.

[66] S. Chen, S. Fang and R. Tang, A reinforcement learning based approach for multi-projects scheduling in cloud manufacturing, *International Journal of Production Research* **57**(10) (2019), 3080–3098. doi:10.1080/00207543.2018.1535205.

[67] M.L. Spearman, D.L. Woodruff and W.J. Hopp, CONWIP Redux: reflections on 30 years of development and implementation, *International Journal of Production Research* **60**(1) (2022), 381 – 387–. doi:10.1080/00207543.2021.1954713.

[68] D. Mezzogori, G. Romagnoli and F. Zammori, A new perspective on Workload Control by measuring operating performances through an economic valorization, *Scientific Reports* **12**(1) (2022), 1–17. ISBN ISBN 0123456789. doi:10.1038/s41598-022-17968-5.

[69] H. Missbauer and R. Uzsoy, Order release in production planning and control systems: challenges and opportunities, *International Journal of Production Research* **60**(1) (2022), 256–276. doi:10.1080/00207543.2021.1994165.

[70] H. Missbauer and R. Uzsoy, *Production Planning with Capacitated Resources and Congestion*, Springer New York, 2020. doi:10.1007/978-1-0716-0354-3.

[71] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg and D. Hassabis, Human-level control through deep reinforcement learning, *Nature* **518**(7540) (2015), 529–533. doi:10.1038/nature14236.

[72] H. Van Hasselt, A. Guez and D. Silver, Deep reinforcement learning with double Q-Learning, *30th AAAI Conference on Artificial Intelligence, AAAI 2016* (2016), 2094–2100. ISBN 9781577357605.

[73] T. Zhou, D. Tang, H. Zhu and Z. Zhang, Multi-agent reinforcement learning for online scheduling in smart factories, *Robotics and Computer-Integrated Manufacturing* **72**(November 2020) (2021), 102202. doi:10.1016/j.rcim.2021.102202.

[74] M.A. Dittrich and S. Fohlmeister, Cooperative multi-agent system for production control using reinforcement learning, *CIRP Annals* **69**(1) (2020), 389–392. doi:10.1016/j.cirp.2020.04.005.

[75] W. Mouelhi-Chibani and H. Pierreval, Training a neural network to select dispatching rules in real time, *Computers and Industrial Engineering* **58**(2) (2010), 249–256. doi:10.1016/j.cie.2009.03.008.

[76] A.L.C. Ottoni, E.G. Nepomuceno, M.S. de Oliveira and D.C.R. de Oliveira, Tuning of reinforcement learning parameters applied to SOP using the Scott–Knott method, *Soft Computing* **24**(6) (2020), 4441–4453. doi:10.1007/s00500-019-04206-w.

[77] J. Patterson and A. Gibson, *Deep Learning: A Practitioner's Approach*, First edit edn, O'Reilly, 2017. ISBN ISBN 1491914254.

[78] I.B. Park, J. Huh, J. Kim and J. Park, A Reinforcement Learning Approach to Robust Scheduling of Semiconductor Manufacturing Facilities, *IEEE Transactions on Automation Science and Engineering* **17**(3) (2020), 1420–1431. doi:10.1109/TASE.2019.2956762.

[79] J. Park, J. Chun, S.H. Kim, Y. Kim and J. Park, Learning to schedule job-shop problems: representation and policy learning using graph neural network and reinforcement learning, *International Journal of Production Research* (2021), 1–18. doi:10.1080/00207543.2020.1870013.

[80] G. De Giacomo, M. Favorito, F. Leotta, M. Mecella and L. Silo, Digital twins composition in smart manufacturing via Markov decision processes, *Computers in Industry* **149** (2023). doi:10.1016/j.compind.2023.103916.

[81] A.M. Akl, S.E. Sawah, R.K. Chakrabortty and H.H. Turan, A Joint Optimization of Strategic Workforce Planning and Preventive Maintenance Scheduling: A Simulation–Optimization Approach, *Reliability Engineering and System Safety* **219** (2022), 108175. doi:10.1016/j.ress.2021.108175.

[82] F.H. Huang, Understanding user acceptance of battery swapping service of sustainable transport: An empirical study of a battery swap station for electric scooters, Taiwan, *International Journal of Sustainable Transportation* **14**(4) (2020), 294–307. doi:10.1080/15568318.2018.1547464.

[83] J. Hu, Y. Wang, Y. Pang and Y. Liu, Optimal maintenance scheduling under uncertainties using Linear Programming-enhanced Reinforcement Learning, *Engineering Applications of Artificial Intelligence* **109** (2022), 104655. doi:10.1016/j.engappai.2021.104655.

[84] A. Valet, T. Altenmüller, B. Waschneck, M.C. May, A. Kuhnle and G. Lanza, Opportunistic maintenance scheduling with deep reinforcement learning, *Journal of Manufacturing Systems* **64** (2022), 518–534. doi:10.1016/j.jmsy.2022.07.016.

[85] Q. Yan, H. Wang and F. Wu, Digital twin-enabled dynamic scheduling with preventive maintenance using a double-layer Q-learning algorithm, *Computers and Operations Research* **144** (2022), 105823. doi:10.1016/j.cor.2022.105823.

[86] J.Y. Mao, Q.K. Pan, Z.H. Miao, L. Gao and S. Chen, A hash map-based memetic algorithm for the distributed permutation flowshop scheduling problem with preventive maintenance to minimize total flowtime, *Knowledge-Based Systems* **242** (2022), 108413. doi:10.1016/j.knosys.2022.108413.

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

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

[89] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, Proximal Policy Optimization Algorithms, *arXiv preprint* (2017).

[90] A. Branda, D. Castellano, G. Guizzi and V. Popolo, Dataset of metaheuristics for the flow shop scheduling problem with maintenance activities integrated, *Data in Brief* **36** (2021), 106985.

[91] A. Branda, D. Castellano, G. Guizzi and V. Popolo, Metaheuristics for the flow shop scheduling problem with maintenance activities integrated, *Computers and Industrial Engineering* **151** (2021), 106989. doi:10.1016/j.cie.2020.106989.

[92] Z. Ding, W. Tan, W.J. Lee, X. Pan and S. Gao, Integrated Operation Model for Autonomous Mobility-on-Demand Fleet and Battery Swapping Station, *IEEE Transactions on Industry Applications* **57**(6) (2021), 5593–5602. doi:10.1109/TIA.2021.3110938.

[93] F. Zhang, H. Lyu, Y. Ji, M. Wong, C. Kuai and J. Fan, Battery swapping demand simulation for electric micromobility vehicles considering multi-source information interaction and behavior decision, *Journal of Cleaner Production* **414**(October 2022) (2023), 137525. doi:10.1016/j.jclepro.2023.137525.

[94] F. Ahmad, M.S. Alam, I.S. Alsaidan and S.M. Shariff, Battery swapping station for electric vehicles: Opportunities and challenges, *IET Smart Grid* **3**(3) (2020), 280–286. doi:10.1049/iet-stg.2019.0059.

[95] M. Neroni, E.M. Herrera, A.A. Juan, J. Panadero and M. Ammouriova, Battery Sharing: A Feasibility Analysis through Simulation, *Batteries* **9**(4) (2023), 1–14. doi:10.3390/batteries9040225.

[96] S.R. Revankar and V.N. Kalkhambkar, Grid integration of battery swapping station: A review, *Journal of Energy Storage* **41**(May) (2021), 102937. doi:10.1016/j.est.2021.102937.

[97] Z.S. Hosseini, M. Mahoor and A. Khodaei, Battery Swapping Station as an Energy Storage for Capturing Distribution-Integrated Solar Variability, in: *2018 North American Power Symposium, NAPS 2018*, 2019. ISSN 9781538671382. doi:10.1109/NAPS.2018.8600598.

[98] X. Hu, Z. Yang, J. Sun and Y. Zhang, Optimal pricing strategy for electric vehicle battery swapping: Pay-per-swap or subscription?, *Transportation Research Part E: Logistics and Transportation Review* **171**(January) (2023). doi:10.1016/j.tre.2023.103030.

[99] J. Song and L. Suo, Optimal Energy for Grid-Connected Microgrid with Battery Swapping Station and Wind Photovoltaic and Energy Storage, *Proceedings of 2019 IEEE 3rd International Electrical and Energy Conference, CIEEC 2019* (2019), 1469–1473. ISBN 9781728116754. doi:10.1109/CIEEC47146.2019.CIEEC-2019533.

[100] J. Garcia-Guarin, W. Infante, J. Ma, D. Alvarez and S. Rivera, Optimal scheduling of smart microgrids considering electric vehicle battery swapping stations, *International Journal of Electrical and Computer Engineering* **10**(5) (2020), 5093–5107. doi:10.11591/IJECE.V10I5.PP5093-5107.

[101] J. Yan, M. Menghwar, E. Asghar, M. Kumar Panjwani and Y. Liu, Real-time energy management for a smart-community microgrid with battery swapping and renewables, *Applied Energy* **238** (2019), 180–194. doi:10.1016/j.apenergy.2018.12.078.

[102] A.R. Jordehi, M.S. Javadi and J.P.S. Catalão, Energy management in microgrids with battery swap stations and var compensators, *Journal of Cleaner Production* **272** (2020), 122943. doi:10.1016/j.jclepro.2020.122943.

[103] A. Rezaee Jordehi, M.S. Javadi and J. P. S. Catalão, Optimal placement of battery swap stations in microgrids with micro pumped hydro storage systems, photovoltaic, wind and geothermal distributed generators, *International Journal of Electrical Power and Energy Systems* **125**(August 2020) (2021), 106483. doi:10.1016/j.ijepes.2020.106483.

[104] K. Balu and V. Mukherjee, Optimal allocation of electric vehicle charging stations and renewable distributed generation with battery energy storage in radial distribution system considering time sequence characteristics of generation and load demand, *Journal of Energy Storage* **59**(October 2022) (2023), 106533. doi:10.1016/j.est.2022.106533.

[105] K. Ni, Z. Wei, H. Yan, K.Y. Xu, L.J. He and S. Cheng, Bi-level optimal scheduling of microgrid with integrated power station based on stackelberg game, *2019 4th International Conference on Intelligent Green Building and Smart Grid, IGBSG 2019* (2019), 278–281. ISBN 9781728121482. doi:10.1109/IGBSG.2019.8886314.

[106] D. Kamath, R. Arsenault, H.C. Kim and A. Anctil, Economic and Environmental Feasibility of Second-Life Lithium-Ion Batteries as Fast-Charging Energy Storage, *Environmental Science and Technology* **54**(11) (2020), 6878–6887. doi:10.1021/acs.est.9b05883.

[107] E. Hossain, D. Murtaugh, J. Mody, H.M.R. Faruque, M.S.H. Sunny and N. Mohammad, A Comprehensive Review on Second-Life Batteries: Cur-

rent State, Manufacturing Considerations, Applications, Impacts, Barriers Potential Solutions, Business Strategies, and Policies, *IEEE Access* **7** (2019), 73215–73252. doi:10.1109/ACCESS.2019.2917859.

[108] D. Kamath, S. Shukla, R. Arsenault, H.C. Kim and A. Anctil, Evaluating the cost and carbon footprint of second-life electric vehicle batteries in residential and utility-level applications, *Waste Management* **113** (2020), 497–507. doi:10.1016/j.wasman.2020.05.034.

[109] Z. Liu, Y. Chen, R. Zhuo and H. Jia, Energy storage capacity optimization for autonomy microgrid considering CHP and EV scheduling, *Applied Energy* **210** (2018), 1113–1125. doi:10.1016/j.apenergy.2017.07.002.

[110] A. Uribe, M. Fernández-Montoya, J. Vargas, G. Osorio-Gómez and A. Montoya, Discrete event simulation for battery-swapping station sizing for hybrid and electric motorcycles, *Journal of Cleaner Production* **390**(January) (2023), 136155. doi:10.1016/j.jclepro.2023.136155.

[111] J. Sindha, J. Thakur and M. Khalid, The economic value of hybrid battery swapping stations with second life of batteries, *Cleaner Energy Systems* **5**(October 2022) (2023), 100066. doi:10.1016/j.cles.2023.100066.

[112] M. Ahmadi Jirdehi and V. Sohrabi Tabar, Risk-aware energy management of a microgrid integrated with battery charging and swapping stations in the presence of renewable resources high penetration, cryptocurrency miners and responsive loads, *Energy* **263**(PA) (2023), 125719. doi:10.1016/j.energy.2022.125719.

[113] W. Alharbi, A.S.B. Humayd, R.P. Praveen, A.B. Awan and V.P. Anees, Optimal Scheduling of Battery-Swapping Station Loads for Capacity Enhancement of a Distribution System, *Energies* **16**(1) (2023), 1–12. doi:10.3390/en16010186.

[114] E. Kremers, J. Gonzalez De Durana and O. Barambones, Multi-agent modeling for the simulation of a simple smart microgrid, *Energy Conversion and Management* **75**(July) (2013), 643–650. doi:10.1016/j.enconman.2013.07.050.

[115] G. Ferruzzi, G. Graditi, F. Rossi and A. Russo, Optimal Operation of a Residential Microgrid: The Role of Demand Side Management, *Intelligent Industrial Systems* **1**(1) (2015), 61–82. doi:10.1007/s40903-015-0012-y.

[116] S. Xia, X. Luo, K.W. Chan, M. Zhou and G. Li, Probabilistic Transient Stability Constrained Optimal Power Flow for Power Systems with Multiple Correlated Uncertain Wind Generations, *IEEE Transactions on Sustainable Energy* **7**(3) (2016), 1133–1144. doi:10.1109/TSTE.2016.2520481.

[117] N.S.M. Hussin, N.A.M. Amin, M.J.A. Safar, R.S. Zulkafli, M.S.A. Majid, M.A. Rojan and I. Zaman, Performance Factors of the Photovoltaic System: A Review, *MATEC Web of Conferences* **225** (2018), 1–8. ISBN ISBN 2018225030. doi:10.1051/matecconf/201822503020.

[118] E.M.A. Mokheimer, A. Al-Sharafi, M.A. Habib and I. Alzaharnah, A new study for hybrid PV/Wind off-grid power generation systems with the comparison of results from homer, *International Journal of Green Energy* **12**(5) (2015), 526–542. doi:10.1080/15435075.2013.833929.

[119] Y. Zou, X. Hu, H. Ma and S.E. Li, Combined State of Charge and State of Health estimation over lithium-ion battery cell cycle lifespan for electric vehicles, *Journal of Power Sources* **273** (2015), 793–803. doi:https://doi.org/10.1016/j.jpowsour.2014.09.146.

[120] L.C. Casals, B. Amante García and C. Canal, Second life batteries lifespan: Rest of useful life and environmental analysis, *Journal of Environmental Management* **232**(October 2017) (2019), 354–363. doi:10.1016/j.jenvman.2018.11.046.

[121] V. Muenzel, J. De Hoog, M. Brazil, A. Vishwanath and S. Kalyanaraman, A multi-factor battery cycle life prediction methodology for optimal battery management, *e-Energy 2015 - Proceedings of the 2015 ACM 6th International Conference on Future Energy Systems* (2015), 57–66. ISBN 9781450336093. doi:10.1145/2768510.2768532.

[122] J. Yang, F. Guo and M. Zhang, Optimal planning of swapping/charging station network with customer satisfaction, *Transportation Research Part E: Logistics and Transportation Review* **103** (2017), 174–197. doi:10.1016/j.tre.2017.04.012.

[123] N. Zhang, Y. Zhang, L. Ran, P. Liu and Y. Guo, Robust location and sizing of electric vehicle battery swapping stations considering users' choice behaviors, *Journal of Energy Storage* **55**(September) (2022). doi:10.1016/j.est.2022.105561.

Author's publications

1. Marchesano, M.G.; Guizzi, G.; Vespoli, S.; Ferruzzi, G. Battery Swapping Station Service in a Smart Microgrid: A Multi-Method Simulation Performance Analysis. Energies 2023, 16, 6576. https://doi.org/10.3390/en16186576.
2. Marchesano M.G., Tortora E., Guizzi G., Vespoli S., Santillo L.C. Integrated Approach for Maintenance Planning and Scheduling in a Flow Shop Using Deep Reinforcement Learning(2023) Frontiers in Artificial Intelligence and Applications Volume 371: New Trends in Intelligent Software Methodologies, Tools and Techniques, 263 – 274. DOI 10.3233/FAIA230241.
3. Marchesano, M.G., Staiano, L., Guizzi, G., Castellano, D., Popolo, V. Deep Reinforcement Learning Approach for Maintenance Planning in a Flow-Shop Scheduling Problem (2022) Frontiers in Artificial Intelligence and Applications, 355, pp. 385-399. DOI: 10.3233/FAIA220268.
4. Salatiello, E., Guizzi, G., Marchesano, M.G., Santillo, L.C. Assessment of performance in Industry 4.0 enabled Job-Shop with a due-date based dispatching rule (2022) IFAC-PapersOnLine, 55 (10), pp. 2635-2640. DOI: 10.1016/j.ifacol.2022.10.107.
5. Marchesano, M.G., Guizzi, G., Popolo, V., Converso, G. Dynamic scheduling of a due date constrained flow shop with Deep Reinforcement Learning (2022) IFAC-PapersOnLine, 55 (10), pp. 2932-2937. DOI: 10.1016/j.ifacol.2022.10.177.
6. Marchesano, M.G., Salatiello, E., Guizzi, G., Santillo, L.C. A Reinforcement Learning approach in Industry 4.0 enabled production system (2022) Proceedings of the Summer School Francesco Turco, 7 p.
7. Marchesano, M.G., Vespoli, S., Guizzi, G., Popolo, V., Grassi, A. A Performance-based dispatching rule for decentralised manufacturing

planning and production control system (2021) Frontiers in Artificial Intelligence and Applications, 337, pp. 581-593. DOI: 10.3233/FAIA210055.

8. Marchesano, M.G., Guizzi, G., Castellano, D., Di Nardo, M. On Reinforcement Learning in production control and its potentiality in manufacturing (2021) Proceedings of the Summer School Francesco Turco, 6 p.
9. Marchesano, M.G., Guizzi, G., Santillo, L.C., Vespoli, S. A deep reinforcement learning approach for the throughput control of a flow-shop production system(2021) IFAC-PapersOnLine, 54 (1), pp. 61-66. DOI: 10.1016/j.ifacol.2021.08.006.
10. Popolo, V., Gallo, M., Grassi, A., Marchesano, M.G. Exploiting the Full Potential of I4.0 Technologies for Products EOL Recovery Process (2021) IFIP Advances in Information and Communication Technology, 632 IFIP, pp. 307-316. DOI: 10.1007/978-3-030-85906-0_35.
11. Marchesano, M.G., Guizzi, G., Santillo, L.C., Vespoli, S. Dynamic Scheduling in a Flow Shop Using Deep Reinforcement Learning (2021) IFIP Advances in Information and Communication Technology, 630 IFIP, pp. 152-160. DOI: 10.1007/978-3-030-85874-2_16