Scheduling maintenance for large offshore wind farms with Deep Q-Learning: a moderate success

Bachelor Thesis
Business Analytics & Quantitative Marketing

Student: Robbert Rog (492751)
Supervisor: Rommert Dekker
Second Assessor: Wilco van den Heuvel

Abstract

Simultaneously optimizing maintenance strategies for multiple wind turbines is a complicated mathematical problem. This paper presents the results of a Deep Q-Learning (DQL) implementation for solving this issue. Additionally, the paper outlines two major challenges when implementing DQL: the credit assignment problem and the risk of unlearning. After building a custom environment, the agent was trained and benchmarked against strategies for one wind turbine, individually applied to the entire farm. The results indicate that, in specific problem instances, grouping preventive maintenance (PM) can save up to 50% because of the reduction in set-up costs. Moreover, the agent discovered that grouping of PMs might overrule the importance of time-varying costs. However, for small wind farms and low set-up costs, the model performs worse than the benchmark and sometimes converges to a "No PM"-policy. This research is limited, in the sense that it only considers a single objective: the long-term average maintenance costs. Future research should focus on developing multi-objective DQL agents that also consider different business elements, such as inventory management and energy demand.

Erasmus School of Economics
Netherlands
May 2022

Contents

Table 1. *Important abbreviations used in this paper*

Abreviation	Definition
p-ARP	Period Age Replacement Policy; as defined by Schouten et. al (2022)
DQL	Deep Q-Learning; Reinforcement Learning technique
DQN	Deep Q-Network; Neural Network for approximating q-values
MDP	Markov Decision Process
PM	Preventive Maintenance; maintaining a non-defective wind turbine
CM	Corrective Maintenance; maintaining a defective wind turbine

1 Introduction

Wind energy becomes an increasingly important factor in renewable energy strategies. Europe is projected to increase its wind power supply by 50% in the years 2022 to 2026. An increasing number of these installations is planned to be offshore, at sea. The rough weather conditions and strong sea currents make maintenance a core element of a wind turbine's cost composition; maintenance attributes between 25-30% of the life cycle cost for offshore wind turbines (Röckmann et al., 2017). To foster the necessary growth of wind power, adequate maintenance strategies should be developed to suppress this cost component. Performing maintenance on an offshore wind turbine is particularly costly compared to onshore. Due to its location on the water, it is challenging to perform maintenance on the turbines. To do so, a platform at the wind farm needs to be set up from which operational activities can be run, which is an expensive process. To meet the expanding demand for wind energy, increasingly large wind farms are being developed further and further from shore, making maintenance an even more prevalent cost component. One example is the Hornsea Wind Farm, consisting of 174 wind turbines located 120 km from the UK's east coast. Daily commuting to the wind farm will no longer be feasible for technicians with such substantial distances. As a result, schedulers should consider grouping the maintenance of wind turbines into one passage to the site, saving on set-up costs for the aforementioned platform.

There are two major types of maintenance. When a component fails and unexpected maintenance is required, it is called corrective maintenance (CM). However, one can also perform maintenance before a component has failed to be ahead of the situation, referred to as preventive maintenance (PM). This could be done based on condition, which provides valuable information but is hard to predict and monitor, or on a time basis, which is easy to schedule and does not rely on potentially inaccurate sensor data. Both CM and PM require the wind turbine to shut down, causing a significant production loss. As wind speeds vary throughout the year, so does production and thus production loss. To illustrate, the average wind speed on the North Sea is 11.5 m/s in February, whereas it averages 7.9 m/s in June (Coelingh et al., 1996). Logically, production loss is much greater in winter than in summer, making it desirable to execute preventive maintenance in the latter. The wind energy industry is not the only sector experiencing time-varying maintenance costs. Solar power, railway providers, or any industry where production or demand is varying throughout the year have to account for this, making the problem relevant cross-industry.

Research on general maintenance policies is commonplace, though few consider time-varying maintenance costs. Wind turbine maintenance studies are often considering constant cost cases, like in Ciang et al. (2008). Moreover, the strategies are often condition-based, which require sensor data and are unpredictable due to exogenous condition changes. A recent work by Schouten et al. (2022) considers three

maintenance policies under time-varying costs. The inclusion of a time-varying cost structure makes these methods novel, though they only consider one wind turbine. With the projected growth of wind energy, the sizes of wind farms are expected to increase. With the increasing distance from shore, a platform often needs to be installed from which operational activities can be run. Maintenance of multiple turbines can be combined into one passage to the wind farm, allowing cost savings on transportation and platform set-up costs. This illustrates the need for a jointly optimized maintenance strategy.

Usually, the maintenance problem is modeled as a Markov Decision Process where at each time step, there is a decision to do preventive maintenance or not. When multiple wind turbines are included, a concurrent Markov Decision Process can be used to model this. The problem, though, is that the state - and action space grow exponentially with the number of wind turbines. As a result, analytically solving the problem leads to exploding computation times, making it an infeasible strategy for larger wind farms. In recent years, a technique called Deep Q-Learning (DQL) has surged in popularity due to its ability to make decisions in large state spaces. This popularity was induced by AlphaZero, a reinforcement learning AI that dominantly beat all the top chess engines at its release (Silver et al., 2018). Recent advances in research show its potential in optimizing large Markov Decision Processes. As shown by Huang et al. (2020), a DQL agent can learn to combine preventive maintenance sessions to save on logistical costs in a manufacturing setting.

This paper will investigate the ability of a DQL agent to autonomously schedule preventive maintenance for a large offshore wind farm under time-varying costs. By using deep learning techniques, the need to analytically solve a concurrent Markov Decision Process is eliminated. Moreover, time-varying costs are implemented to account for the varying production loss throughout the year. In a maintenance setting, the combination of Deep Q-Learning and time-varying costs is a novel contribution to the literature. Our research question is as follows.

How does a Deep Q-learning agent perform in optimizing maintenance policies for wind farms with set-up costs for maintenance and time-varying production losses?

To answer this question, a simulation study is performed. Reliability data for a large offshore wind farm is produced. Then, a time-varying cost structure is defined, which is used to train the agent. The optimal policy for a single wind turbine as introduced in Schouten et al. (2022) is computed for these cost structures. Individually applying this policy to each wind turbine provides us with a benchmark cost to compare the DQL agent against. The behavior of the autonomous agent is analyzed using various statistics, providing insights into when and how to combine maintenance. We find that, in specific instances, the DQL agent finds cost savings up to 50%. However, its success is highly dependent on the setting, the DQL agent does not outperform the benchmark for less favorable problem instances.

This paper continues as follows. The next section provides an overview of the existing related literature. In section 3, the data generation is discussed further. Section 4 explains the optimal single wind turbine maintenance policy, introduces the Deep Q-learning architecture, and describes the experimental set-up. Finally, in section 5, the simulation study results are presented, after which a conclusion is drawn.

2 Literature Review

The upcoming section outlines the relevant literature and identifies the gap this paper aims to solve. There is a wide variety of articles available on maintenance policy. First, the economic relevance of maintenance costs for wind farms is explained. Second, the literature regarding preventive maintenance is summarized, with a focus on wind farms. Afterward, studies on time-varying costs about maintenance downtime are discussed. Subsequently, relevant papers that illustrate the feasibility of Deep Q-Learning are mentioned.

2.1 Maintenance Costs for Wind Turbines

Throughout the years, the production of wind turbines has become notably cheaper. This can mainly be attributed to more efficient labor, cheaper materials, and a better legal/political environment (Elia et al., 2020). Innovations are a large driver for these lower production costs. With manufacturing becoming more efficient, operation and maintenance costs become the main determinant for the levelized cost of wind energy. This is particularly true for offshore wind farms, as logistical costs for maintenance are high compared to onshore. As wind energy is expected to be the supplier of approximately 50% of our energy by 2050, keeping maintenance costs low is essential for a successful energy transition (Association et al., 2011). A recent review by Ren et al. (2021) illustrates the multi-dimensionality of the offshore wind turbine maintenance problem. The first challenge relates to the scheduling of maintenance under uncertainty; the lifetime of wind turbine components is stochastic. Once maintenance is scheduled, the next step is bringing the right personnel and material to the site. These logistical operations are not only costly, but they also emit a significant source of greenhouse emissions, which is detrimental to the sustainable nature of wind energy (Arvesen et al., 2013), and thus minimizing them is good for both economic and climate purposes.

2.2 Preventive Maintenance for Wind Turbines

Preventive maintenance strategies for offshore wind turbines are often condition-based, relying on the availability of accurate sensor data to estimate the condition of the wind turbine (Ciang et al., 2008). However, these condition-based methods suffer from a variety of issues. Firstly, sensor data might not always accurately reflect the true condition of a component. This can cause an over-or underestimation of the failure rate of a component. Moreover, a component's condition can change almost instantaneously due to exogenous, unpredictable events related to the environment. Especially further on the sea where the current is strong and the weather is rough, this becomes an issue with condition-based maintenance. This leads to a lack of plannability, which is crucial for smooth operation and lowering maintenance costs. The lead-time on to-be-replaced components is large, causing unnecessarily long downtime periods and thus production losses. This illustrates the need for methods that do not rely on conditions. The rest of this paper will focus exclusively on models that only use a statistical deterioration process; failure occurs unexpectedly with a probability dependent on the component's age. As early as 1965, preventive maintenance policies based on age and time were introduced (Barlow and Proschan, 1965). Age-based maintenance means that preventive maintenance is performed whenever a component reaches a critical age. Time-based maintenance simply schedules the next preventive maintenance with predetermined intervals. Later, Ross (1970) showed that, under some light conditions, age-based maintenance was

the optimal policy with respect to long-term average costs, when conditional information is not used. The need for preventive maintenance is illustrated by the clear advantages over unexpected, corrective maintenance: easier scheduling, avoidance of downtime during important periods, more efficient vessel and crew usage, and many more (Karyotakis, 2011).

2.3 Time-Varying Costs in Maintenance

In particular, the downtime in certain periods can be problematic; a wind turbine's output is highly dependent on the wind speed, which varies over the year. However, many discrete replacement and maintenance policies consider constant costs in each period (Nakagawa, 1984). For wind turbines, the power output is dependent on the wind speed in a non-linear fashion (). Moreover, wind speeds vary heavily over the year: on the east coast of the United Kingdom, February has a mean wind speed of around 11.5 m/s whereas July's mean wind speed is 8.2 m/s (Coelingh et al., 1996). As maintenance requires the wind turbine to temporarily shut down, production losses differ greatly from month to month. One important work that accounts for this variation is the one by Schouten et al. (2022). They consider a single-component wind turbine with time-varying costs. Indeed, their analysis shows that July yields a low critical age at which preventive maintenance is performed. Under certain conditions, preventive maintenance is actually exclusively performed in July. This clearly illustrates that time-varying maintenance costs play a significant role in the decision process for a single wind turbine.

Traditionally, a Markov Decision process is used to model the failure and maintenance decisions, in which the failure rate is dependent on the age of the component. In the paper by Schouten et al. (2022), three policies are adapted to account for time-varying costs. First, a period Age Replacement Policy (p-ARP) is instituted, which replaces the component when it reached a critical age. However, to account for time-varying costs, the optimal maintenance age can differ from period to period. One advantage of this is that it yields the lowest costs theoretically, although maintenance is hard to schedule ahead and thus the policy is practically hard to implement. The second policy, on the other hand, has much easier scheduling: the Block Replacement Policy (BRP). This method schedules preventive maintenance with a predetermined interval. Though the interval can differ in length, it shall repeat itself after a certain amount of periods. This makes the maintenance logistically more plannable, though it might lead to less optimal results. The last considered policy is the Modified Block Replacement Policy (MBRP), which is essentially a combination of the two. Preventive maintenance is scheduled in advance according to predetermined intervals but it is only executed if the component exceeds a period-varying critical age. This strikes a balance between the optimum and plannability. Indeed, their numerical experiments verify the expectations: a lower critical maintenance age in summer, where wind speeds are low, leads to a cost reduction of around 23% compared to a model that does not account for time-varying costs. The MBRP costs are only slightly higher than that of the p-ARP, illustrating the trade-off between plannability and cost savings.

2.4 Deep Q-Learning

The aforementioned strategies only account for one wind turbine. With the increasing importance of wind energy and the growing sizes of wind farms, an approach is required that handles the maintenance optimization for a large number of wind turbines. Modeling this mathematically becomes an issue as

both the state- and action space grow exponentially, making the original numerical methods slow and computationally expensive. This is where Deep Q-learning Networks (DQN) fit into the picture. In short, regular Q-learning contains a Q-table, in which expected returns are stored for each state-action pair based on simulations (Watkins and Dayan, 1992). This expected return for a given state-action pair is referred to as a q-value. The class of Q-learning algorithms is large, as it has been extended numerous amount of times; a comprehensive overview is presented by Jang et al. (2019). An issue with traditional Q-learning arises in applications with an exponentially increasing state space; as the state space becomes large, computer memory cannot contain the enormous Q-table anymore. Moreover, certain states might rarely be visited during training, thus, a large number of simulations is required to get reliable estimates of the q-values.

Deep Q-Learning (DQL) eliminates the need for a Q-table, as it uses a Deep Neural Network to approximate q-values for a given state. This way, it can estimate expected returns in states it has never seen before, based on patterns learned in the simulation. For a long time, little was known about the theoretical foundations of DQNs; their empirical success was enough to justify their usage. Recently, a novel theoretical analysis shows that the approximated q-values indeed converge to the true state-action values under mild conditions (Fan et al., 2020). The most promising results were shown in a game-like setting. When the DQN is appropriately specified, it was able to reach superhuman performance in various board games, such as chess and go (Silver et al., 2018). Moreover, the technique was applied to various old-school video games in which it played far above the human level (Hester et al., 2018). More recently, its problem-solving capabilities are extended to real-life issues as researchers realized many real-life problems can be described by an action-reward game. In a portfolio management setting, a DQN was able to outperform ten traditional investment strategies (Gao et al., 2020). Robotics has been another field of successful implementation, especially in optimizing their control system (Degrave et al., 2019).

However, DQNs have their challenges and limitations. In many applications, including this one, long-range dependencies occur in the environment. This means that the reward the agent receives does not occur directly as a result of the action. Take our problem, for example, where the benefits of scheduling PM are not directly visible. On the contrary, scheduling PM has costs attached to it whereas the benefits are less concrete and occur in the future; the wind turbine is less likely to break down. This issue is referred to as the credit assignment problem and remains a challenge for Deep Reinforcement Learning applications (Sutton and Barto, 1998). Another challenge DQLs face is long training times. In complicated settings, sometimes several millions of simulations are required for the DQN to gain an understanding of the problem. To reduce this number, an adequate exploration strategy needs to be set, which is challenging and varies from problem to problem (Arulkumaran et al., 2017). Fortunately, it is often possible to implement knowledge of the problem into the model design to speed up this training process. This has successfully been done in numerous physics applications (Kansky et al., 2017) and will later be used in our model as well.

Deep Q-Learning has been applied to the maintenance setting as described earlier. In a manufacturing setting, DQLs show that executing multiple maintenance tasks simultaneously leads to cost reductions for serial product lines (Huang et al., 2020). This idea is human-originated; when maintenance needs to be performed on one of the machines in the production chain, the production faces downtime anyway. The authors argue that a thorough understanding of the problem is required to successfully implement

the DQL. As shown in an electricity grid application, a DQL model outperforms experts by avoiding CMs as much as possible, leading to higher expected rewards (Rocchetta et al., 2019). In their work, the authors show that, in this problem setting, their DQL decisions converge to those according to the optimal Bellman Equation. A recent study shows its potential in a multiple component maintenance problem (Yousefi et al., 2022). Indeed, the DQN is able to recognize patterns in a large state space instead of directly mapping each state-action pair to a value. The literature shows promising results for maintenance applications. However, only constant cost structures are considered that do not vary over time. Applying this technique to a setting with time-varying costs, such as wind turbine production loss, contributes to the existing literature.

3 Data

This section describes the data used in this research paper. To verify the validity of the Deep Q-Learning approach, a simulation study will be performed. For each wind turbine, a vector l_i containing lifetimes is generated a priori.

$$l_i = [x_1^i, x_2^i, \ldots, x_{uN}^i] \tag{1}$$

where x_j^i is the j'th lifetime of wind turbine i, N is the number of periods in a cycle, and u is the maximum amount of cycles we want to consider in one simulation. These variables are drawn from a discretized Weibull distribution: the Weibull draws are simply rounded upward, providing us values between 1 and ∞. These lifetime vectors are linked to a wind turbine and stored in a data set at the beginning of each simulation. Once a component fails or PM is performed on a wind turbine, the time until failure will be set to the next entry in the corresponding lifetime vector. For example, if the next lifetime is one, the turbine will fail the period after repair. After training the agent, the same procedure is used to generate equal data sets for comparing the DQL against benchmark policies. This ensures a fair comparison between the optimal single wind turbine policy presented by Schouten et al. (2022) and the DQL agent as the environments behave identically. This simulation is done 100 times for each model instance, in which the DQL performance is measured against p-ARP, MBRP, and a policy that never schedules PM, referred to as a "No PM" policy.

4 Methodology

This section outlines the methodology of the experiment. First, the problem setting is formally described. Mathematical notation for the state space, action space, and cost structure are introduced. Afterward, the optimal policy for a single wind turbine (Schouten et al., 2022) is discussed. The third subsection introduces the Deep Q-learning adaptation used to create the autonomous agent. Thereafter, adjustments to the methodology as a result of experimentation are elaborated upon. Finally, the experimental setup to test the performance of our agent is explained.

Table 2. *Mathematical notation for the methodology*

Abreviation	Definition
I	State space
I_1	Set of periods within one year
I_2	Set of wind turbine ages
N	Number of periods within a year
n	Number of turbines on the wind farm
A	Action space
a_t	Action in period t; schedules maintenance for $t + 1$
s_t	State at the start of period t
$\hat{\mathbb{E}}[s_{t+1}]$	Estimated expected state at the start of $t + 1$ resulting from maintenance in period t
$c_{cm}(i_1)$	CM costs in period i_1
$c_{pm}(i_1)$	PM costs in period i_1
$\bar{c}_{cm}$	Average CM cost
$\bar{c}_{pm}$	Average PM cost
π_{ij}	Transition Probability from state i to state j
π	Policy chosen by Deep Q-Network
$q_\pi(s, a)$	Q-function; denotes the value of taking action a in state s under policy π
γ	Discount rate
r_{t+1}	Reward received in $t + 1$ from action a_t
e_t	Experience tuple; used to train the Deep Q-Network
ϵ_t	Exploration rate at time t
d	Epsilon decay rate
ω	Rate for Exponential Learning Rate Decay
m_t	Number of wind turbines surpassing the critical maintenance age in period t

4.1 Markov Decision Process

The problem at hand is to find a maintenance strategy for a large offshore wind farm such that maintenance costs can be kept low. This problem can be modeled as a Markov Decision Process (MDP). More specifically, in month t we decide how much maintenance should be scheduled in month $t + 1$. Earlier strategies were appropriate for onshore wind turbines, where logistical costs are low, or for a single offshore wind turbine. The root of the problem lies within the large logistical costs required to get to the site and to set up a platform from which operations can be run. Logistical costs become more of a concern now that offshore wind farms are moving further away from shore. As a result, the maintenance crew needs to sail a long time before arriving on-site, making daily commuting impossible. Consequently, a set-up structure is used, where logistical set-up costs are incurred once when PM is performed. Then, PMs can be executed with a predetermined additional cost. For both the crew and the operational activities, temporary platforms are often installed when maintenance needs to be performed. The cost structure will be defined more formally later on. Let us first have a look at the state space and action space.

4.1.1 State Space

The state space of the MDP consists of two parts. Firstly, the current period. As we consider time-varying production losses throughout the year, different periods yield different costs for maintenance. Let $I_1 \subseteq \mathbb{N}^+$, $I_1 = \{1, 2, ..., N\}$ where N represents the number of periods in a year. Moreover, $I_2 \subseteq \mathbb{N}^n$ denotes the set of wind turbine ages, where n is the number of wind turbines at the wind farm. i_2 is thus

a vector with ages in which a failed wind turbine has an age of zero. This already illustrates the issue with regular Q-learning or linear models; the state space grows exponentially with the number of wind turbines on the site. The state space is thus given by $I = I_1 \times I_2 \subseteq \mathbb{N}^+ \times \mathbb{N}^n$.

4.1.2 Action Space and Transition Probabilities

There are two types of maintenance that can be performed. The first one is corrective maintenance (CM), which is performed on a defective wind turbine. Traditionally, CM is rather expensive due to long lead times on new components; this causes the downtime of the wind turbine to be relatively long. Moreover, CM occurs unexpectedly, making planning rather abrupt and expensive. Besides CM, there is also preventive maintenance (PM) which can be executed before the failure of a wind turbine. As PM is planned and does not occur unexpectedly, the lead time on required components can be accounted for, and a lower downtime is necessary to perform the operation.

For a single wind turbine in period i_1 with component age i_2, the action space can be defined as follows:

$$A(i_1, i_2) = \begin{cases} \{1\} \text{ if } i_2 \in \{0, M\} \\ \{0, 1\} \text{ otherwise} \end{cases} \tag{2}$$

Here, 1 corresponds to executing maintenance, and 0 corresponds to not performing maintenance. This means that CM is always performed once the component has age 0, i.e. it has failed. PM is always performed when the turbine reaches some predetermined large age M. Note that M might be equal to ∞, this means we never perform PM by default. For the action space as given above, the transition probabilities for a single wind turbine are defined as follows:

$$\pi_{(i_1,i_2)(j_1,j_2)}(0) = \begin{cases} 1 - p_{i_2} \text{ for } j_1 = i_1 + 1 \pmod N, j_2 = i_2 + 1, i_2 \notin \{0, M\} \\ p_{i_2} \text{ for } j_1 = i_1 + 1 \pmod N, j_2 = 0, i_2 \notin \{0, M\} \\ 0 \text{ else} \end{cases} \tag{3}$$

$$\pi_{(i_1,i_2)(j_1,j_2)}(1) = \begin{cases} 1 - p_1 \text{ for } j_1 = i_1 + 1 \pmod N, j_2 = i_2 + 1 \\ p_1 \text{ for } j_1 = i_1 + 1 \pmod N, j_2 = 0 \\ 0 \text{ else} \end{cases} \tag{4}$$

where $p_{i_2} = \mathbb{P}(X = i_2 | X \geq i_2)$ defines the failure probability of the component at age i_2 and mod is the modulo operator. As can be seen from the equations above, defining the transition probabilities for each action when multiple wind turbines are considered simultaneously becomes rather complicated. This leads to a concurrent Markov Decision Process, which is computationally hard to solve exactly.

One note needs to be given regarding the failure probability. As discussed by Schouten et al. (2022), the failure rate should be increasing in the component's age for stable results; this paper will adopt this assumption as well as a Weibull distribution is used to generate the wind turbines' lifetimes. Consequently, for a group of wind turbines, it is always best to perform preventive maintenance on the oldest turbines.

For example, consider two wind turbines, one with age 1 (just maintained) and one with age 20. If we were to perform PM on either of the two, the latter yields a higher expected return because it is expected to fail much sooner than the prior.

This observation greatly simplifies the action space for larger wind farms: the question is no longer which wind turbines need PM, but how many. As a result, the action space grows linearly with the number of wind turbines. For period i_1 and component age vector i_2, the action space becomes as follows:

$$A(i_1, i_2) = \{0, 1, ..., n\} \tag{5}$$

Here, the action is the number of wind turbines on which PM will be performed. As mentioned earlier, it is always best to perform PM on the components with the highest age. Thus, for action a, PM will be executed on the oldest a wind turbines. Note that it could occur that wind turbines break down right when they were scheduled for preventive maintenance. The PM will still take place on the next oldest wind turbine and costs are still incurred. In the extremest of cases, a large number of wind turbines can break down, making it not possible to perform all scheduled PMs. A simplifying assumption is made that costs are then still incurred, even if a PM can not take place. For each action, the transition probabilities are defined but mathematically complicated. This is where a model-free technique, such as Deep Q-learning, becomes a possible solution, which is further discussed in section 4.3.

4.1.3 Cost Structure

A similar cost structure as in Schouten et al. (2022) will be used for the costs per PM and CM. Namely, since CM occurs unexpectedly and has a longer downtime, the costs of CM are much higher than the costs of PM. More formally, the costs can be defined as follows:

$$c_{cm}(i_1) = \bar{c}_{cm} + \Delta_{cm} \cos\left(\frac{2\pi i_1}{N} + \phi\right) \tag{6}$$

$$c_{pm}(i_1) = \bar{c}_{pm} + \Delta_{pm} \cos\left(\frac{2\pi i_1}{N} + \phi\right) \tag{7}$$

Here, $\bar{c}_m$ is the mean cost for maintenance type , Δ_m indicates the amplitude of monthly cost variations, $\phi = -\frac{2\pi}{N}$ is a constant to shift the costs such that the lowest costs are incurred in July. Traditionally, Δ_m is a percentage of $\bar{c}_m$. The cost structure will be extended by means of logistical set-up costs. These set-up costs will be incurred for every CM but only once when PM is performed, regardless of the number of PMs. As CMs are often large and unexpected operations, it is not unreasonable to incur logistical costs for each CM. This defines the total costs for a scheduled PMs and b required CMs as follows:

$$c_{total}(i_1, a, b) = \begin{cases} b \times (c_{cm}(i_1) + c_{set-up}) \text{ , if } a = 0 \\ b \times (c_{cm}(i_1) + c_{set-up}) + a \times c_{pm}(i_1) + c_{set-up} \text{ , if } a \geq 1 \end{cases} \tag{8}$$

4.2 Single Wind Turbine

The maintenance policy for a single wind turbine with time-varying maintenance costs has been mathematically optimized. A linear formulation for the constant cost case was introduced by Nakagawa (1984), which was later adopted and optimized for time-varying costs (Schouten et al., 2022). This is called the Period Age Replacement Policy (p-ARP). However, the downside of this policy is the lack of plannability, as will be discussed later. A counter-part of p-ARP is the Period-dependent Modified Block Replacement Policy (MBRP) which stems from the original Block Replacement Policy. This policy has a much more plannable nature, which will become clear from the discussion.

4.2.1 Period Age Replacement Policy

The p-ARP performs preventive maintenance when the wind turbine has reached a predetermined critical maintenance age. However, the critical maintenance age can differ from period to period. In months with low production loss, such as July, a lower critical maintenance age might be favorable due to cheaper preventive maintenance. Under the assumption of an increasing failure rate of the component, p-ARP yields an optimal age-based policy. The proof can be found in Schouten et al. (2022), who formally define p-ARP as follows:

Definition 4.1 *(p-ARP). A p-ARP is a policy in which in each period $i_1 \in I_1$ of the year, there is a period-dependent critical maintenance age $t(i_1) \in N\backslash\{0\}$ at or above which PM is performed (Schouten et al., 2022).*

Mathematically, the p-ARP can be formulated as a linear programming problem. To solve this problem, a distinction needs to be made between the states with a failed component and the states without. Let $I^b = I_1 \times \{0\}$ denote the set of states with a failed component. Then, using the cost function as described in section 4.1.3, the LP formulation becomes as follows:

$$
\begin{aligned}
\min \quad & \sum_{i=(i_1,i_2)\in I\backslash I^b} c_p(i_1)x_{i,1} + \sum_{i=(i_1,i_2)\in I} c_c(i_1)x_{i,1} \\
\text{s.t.} \quad & \sum_{a\in A(i)} x_{i,a} - \sum_{j\in I}\sum_{a\in A(j)} \pi_{ji}(a)x_{j,a} = 0, && \forall i = (i_1,i_2) \in I \\
& \sum_{i_2\in I_2}\sum_{a\in A(i_1,i_2)} x_{i_1,i_2,a} = \frac{1}{N}, && \forall i_1 \in I_1 \\
& x_{i,a} \geq 0, \quad \forall i = (i_1,i_2) \in I, a \in A(i)
\end{aligned}
$$

Here, $c_p(i_1)$ and $c_c(i_1)$ are simply the preventive and corrective maintenance cost as defined in 4.1.3, with set-up costs included. In this instance, the decision variables $x_{i,a}$ can be interpreted as the long-run probability that the system is in state i and action a is chosen. The objective function then represents the long-run average costs. Converting the LP solution into a policy is simple. Suppose an optimal solution to yields $x^*_{i,a}$. Let $I_{lp} = \{i | x^*_{i,a}$ for some action a$\}$ and define the strategy R^* by $R^*(i) = a$ if $x^*_{i,a} > 0$. Then, for all other states i an action a is chosen such that $p_{ij}(a) > 0$ for $j \in I_{lp}$ and add the states i to the set I_{lp} until no remains. Now, in case the policy R^* performs PM for state (i^0_1, i^0_2) then it is optimal to do PM for any $i_2 \geq i^0_2$ as well, allowing us to construct an optimal p-ARP from the solution.

4.2.2 Period-dependent Modified Block Replacement Policy

The second policy that will be considered as a benchmark is the Period-dependent Modified Block Replacement Policy (MBRP). Maintenance is scheduled with fixed intervals that repeat every m years, though the maintenance is only performed when the component surpassed a time-dependent critical maintenance age. Again, the summer months favor lower critical maintenance age, as maintenance is cheaper due to lower production losses. The advantage of this method is that the possible PM sessions are known well in advance but can be canceled if they are deemed unnecessary. Formally, the MBRP is defined as:

Definition 4.2 *(MBRP). A Period-dependent MBRP is a policy with a finite cycle of m years, in which PM can only be performed at times $T_1, T_2, ..., T_n \leq mN$, for some $n \in N^+$. PM is performed at occasion k if a critical maintenance age $t_{(k)}$ has been reached, where $t_{(k)} \leq T_k - T_{k-1}$ for $k = 2, 3, ..., n$ and $t_{(1)} \leq T_1 - T_n + mN$ (Schouten et al., 2022).*

To create an LP formulation for this problem, three new variables $y_i \in \{0, 1\}$, $z_i \in \{0, 1\}$, and $t_i \in N^+$ are added to the model. The variable y_i determines when PMs are scheduled, i.e. the intervals, whereas z_{i_1,i_2} checks whether the maintenance is actually performed at age i_2 in period i_1. The definition of these variables can be found in equation 9 and 10 respectively. t_i is added to distinguish the policy from a p-ARP as can be seen in definition 4.2. For a more thorough explanation of the formulations, read the original paper by Schouten et al. (2022). The LP formulation becomes as follows.

$$\min \sum_{i=(i_1,i_2)\in I\setminus I^b} c_p(i_1)x_{i,1} + \sum_{i=(i_1,i_2)\in I} c_c(i_1)x_{i,1}$$

$$\text{s.t.} \quad \sum_{a\in A(i)} x_{i,a} - \sum_{j\in I}\sum_{a\in A(j)} \pi_{ji}(a)x_{j,a} = 0, \qquad \forall i = (i_1, i_2) \in I$$

$$\sum_{i_2\in I_2}\sum_{a\in A(i_1,i_2)} x_{i_1,i_2,a} = \frac{1}{N}, \qquad \forall i_1 \in I_1$$

$$z_{i_1,i_2} - y_{i_1} \leq 0 \qquad \forall i_1 \in I_1, \forall i_2 \in I_2$$

$$z_{i_1,i_2} - z_{i_1,j_2} \leq 0 \qquad \forall i_1 \in I_1, \forall j_2 \in I_2 : i_2 < j_2$$

$$t_{i_1} + j_1 y_{j_1} + mNY_{j_1} \leq mN + i_1 \qquad \forall i_1, j_1 \in I_1 : j_1 < i_1$$

$$t_{i_1} + j_1 y_{j_1} \leq mN + i_1 \qquad \forall i_1, j_1 \in I_1 : j_1 > i_1$$

$$My_{i_1} - Mz_{i_1,i_2} \leq M - 1 - i_2 \qquad \forall i = (i_1, i_2) \in I$$

$$Mz_{i_1,i_2} + t_{i_1} \leq M + i_2 \qquad \forall i = (i_1, i_2) \in I$$

$$x_{i,a} \geq 0, \qquad \forall i = (i_1, i_2) \in I, a \in A(i)$$

$$z_i \in \{0, 1\} \qquad \forall i \in I$$

$$y_{i_1} \in \{0, 1\} \qquad \forall i_1 \in I_1$$

$$t_{i_1} \in N^+ \qquad \forall i_1 \in I_1$$

$$y_{i_1} = \begin{cases} 1, \text{ if we maintain preventively in } i_1 m \in I_1 \\ 0, \text{ else.} \end{cases} \tag{9}$$

$$z_i = z_{i_1,i_2} = \begin{cases} 1, \text{ if we maintain for age } i_2 \in I_2 \text{ in period } i_2 \in I_1 \\ 0, \text{ else.} \end{cases} \tag{10}$$

Just as in the previous strategy, $c_p(i_1)$ and $c_c(i_1)$ are simply the preventive and corrective maintenance cost as defined in 4.1.3, with set-up costs included.Again, constructing a policy from the LP solution is simple. The variable y_{i_1} simply tells us which periods to perform PM in. Then, the variable z_{i_1,i_2} indicates if PM should actually be performed in i_1 with age i_2. If z_{i_1,i_2} equals 1 for i_2^0 then performing PM scheduled in period i_1 should be done for any $i_2 \geq i_2^0$. This is more accurately reflected in the variable t_{i_1} which tells us the critical maintenance age for maintenance scheduled in period $i1$.

4.3 Deep Q-learning

Deep Q-Learning (DQL) is a form of Reinforcement Learning: training an agent to make decisions autonomously in a given environment. This section discusses DQL and the corresponding Deep Q-Networks (DQNs) in more detail. First, the learning environment and reward system are described. Afterward, q-values are introduced. The following sub-sections discuss different elements of the training procedure: replay memory, policy network, target network, and the epsilon greedy strategy. Lastly, problems with the initial methodology are explained after which the adjusted methodology is presented.

4.3.1 Environment

In Deep Q-Learning, the autonomous agent makes a decision based on the environment and gets a reward in return. The agent learns by means of simulation, where each simulation consists of a number of time steps. These simulations are better known as episodes. Figure 1 graphically represents a time step within this episode.

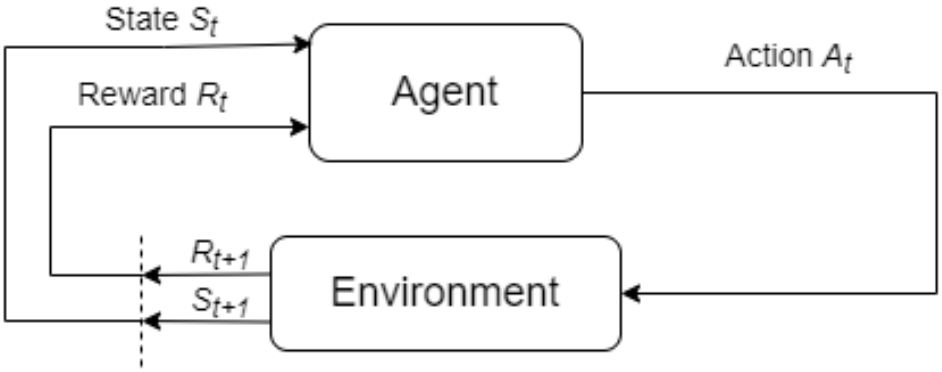

Figure 1. Graphical representation of a time step within an episode.

The state - and action space are previously described in sections 4.1.1 and 4.1.2 respectively. A brief summary: the state is characterized by the period of the year $i_1 \in I_1 = \{1, ..., N\}$ and the component

ages of the wind turbines $i_2 \in I_2$, $I_2 \subseteq \mathbb{N}^n$ where n is the number of wind turbines on-site such that i_2 is an $n \times 1$ vector. At each time step, the agent decides which action to take, i.e. how many PMs to schedule in the next month. Thus, in period t, the agent will schedule PMs for period $t+1$. The transition probabilities, however, become extremely complicated for such large state and action spaces. This is where DQL comes into play. It is a model-free technique, which means there is no need to specify the transition probabilities. Note that DQL does not replace the transition probabilities, it simply learns from simulated experiences for which we do not need to specify those probabilities. Instead, we use simulated data as described in section 3.

4.3.2 Q-values and Q-functions

The agent uses an action-value function, also referred to as the q-function, to determine it's best action. This function describes the value of taking action a in state s by means of the expected discounted return. More formally, the q-function for a given policy π is defined as:

$$q_\pi(s,a) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | S_t = s, A_t = a] \tag{11}$$

where $0 < \gamma < 1$ is the discount factor and r_t the reward obtained at time step t. A policy is simply defined as a strategy that selects the action. Note that r_t is a random variable as we are unsure how many wind turbines break down, thus how many CMs will be performed; the reward becomes random. The value that is outputted by this function for a given s and a is referred to as the q-value. The agent aims to find the optimal policy for decision-making, which yields the highest expected return for each state. That is:

$$q_*(s,a) = \max_\pi q_\pi(s,a) \tag{12}$$

Determining the optimal policy of this expression is hard and requires a definition of the transition probabilities. However, one fundamental property of q_* is that it must satisfy the following equation, known as the Bellman Optimality Equation.

$$q_*(s_t,a_t) = \mathbb{E}[r_{t+1} + \gamma \max_{a_{t+1}} q_*(s_{t+1}, a_{t+1})] \tag{13}$$

This equation states that, for each state-action pair at time t, the discounted return is equal to the expected return in the next period plus the expected discounted return obtained from the best action in state s_{t+1}, the next state. This equation is used during training, which will be further explained in section 4.3.5.

4.3.3 Policy Network

One technique well-known for its ability to approximate functions is Neural Networks. The policy network is a neural network used in Deep Q-learning to approximate $q_*(s,a)$ for all possible actions. Figure 2 shows an example of a policy network for the wind farm maintenance problem. Generally, the

state will be propagated through multiple fully connected layers, after which the q-values for each action will be estimated at the output layer.

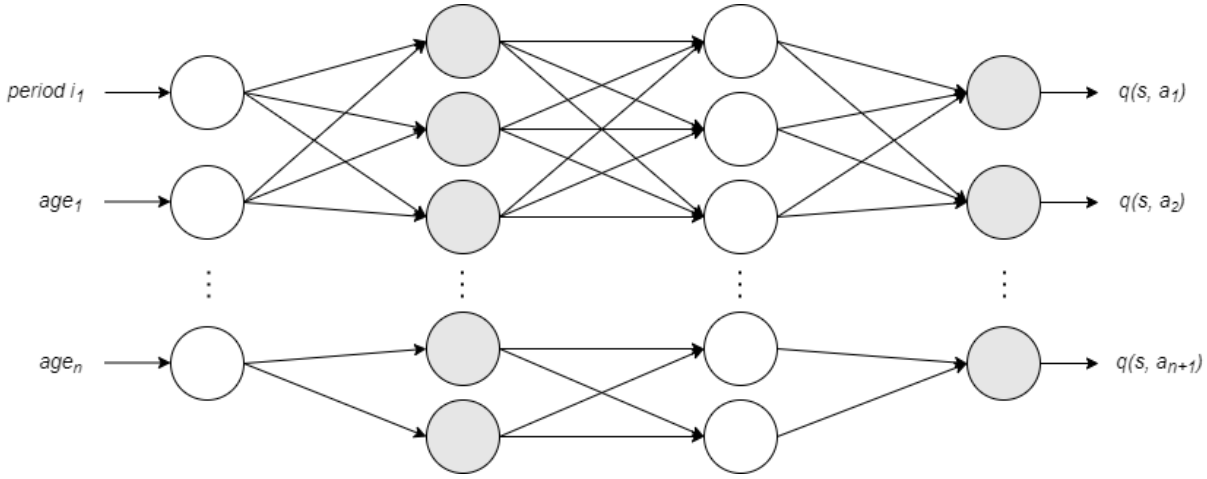

Figure 2. An example of a policy network in the wind turbine environment.

In period t, the agent schedules maintenance for period $t + 1$. To support this decision-making, the most logical input for the network is $\mathbb{E}[S_{t+1}]$. As this technique is model-free, i.e. the transition probabilities are unknown for the agent, it uses a heuristic to acquire the expected next state: the agent simply assumes that all wind turbines will age without breaking down. Thus, the following equations allow us to define the input of the policy network.

$$s_t = [i_1, age_{1,t}, ..., age_{n,t}] \tag{14}$$

$$\hat{\mathbb{E}}[s_{t+1}] = [i_1 + 1 \bmod \mathrm{N}, age_{1,t} + 1, ..., age_{n,t} + 1] \tag{15}$$

where s_t is the state in period t after maintenance has been performed, $age_{1,t}$ is the age of the oldest wind turbine in period t, and $age_{n,t}$ is the age of the youngest wind turbine in period t. The ages are sorted to keep the input consistent; at last, the location of an age variable does not contain relevant information. For deciding which action to take, we are mainly interested in the oldest wind turbines after all, and not the exact wind turbine. The input will then be propagated through multiple fully connected layers, for which the number of layers and the number of neurons per layer are hyperparameters to be tuned. Ultimately, the output layer provides us with the approximation of the q-value. More specifically, an $(n+1) \times 1$ vector outputs the approximated q-values for each action. Note that, for n wind turbines, there are $n + 1$ actions as one can also decide not to perform preventive maintenance at all. The index of the output vector, starting at zero, simply corresponds to the action, i.e. the number of PMs scheduled for period $t + 1$.

4.3.4 Replay Memory and Loss Functions

Traditionally, Neural Networks are trained on large data sets to acquire good results. However, to train the policy network, we will make use of a technique called replay memory to construct a data set for our

problem instance. This replay memory consists of experiences of the agent with the reward attached to it. For our problem setting, an experience is defined as:

$$e_t = (\hat{\mathbb{E}}[s_{t+1}], a_{t+1}, r_{t+1}, s_{t+1}) \tag{16}$$

This tuple stores the input $\hat{\mathbb{E}}[s_{t+1}]$, the action taken a_t, the reward (costs) resulting from it r_{t+1}, and the state following it s_{t+1}. These experiences are stored in a data set, namely the replay memory. The replay memory has a limited size, only the latest memories can be stored. Not only does this limit memory usage, but it also ensures that older, possibly less relevant experiences will be forgotten over time. Random samples from the replay memory will be taken to perform training. Would we use experiences in their sequential order, the training data would be highly correlated, which leads to inefficient training of the policy network.

The replay memory will be used to train the model. For an experience tuple e_t, state $\hat{\mathbb{E}}[s_{t+1}]$ will be passed through the policy network, after which the q-value for action a_t will be determined. Now, we obtain an approximation of the q-value but we have yet to find the unknown target q-value to compute the loss. The loss L is defined as:

$$L = q_*(\hat{\mathbb{E}}[s_{t+1}], a_t) - q(\hat{\mathbb{E}}[s_{t+1}], a_t) \tag{17}$$

where $q_*(s_t, a_t)$ is the q-value under the optimal policy and $q(s_t, a_t)$ is the q-value outed by the model. Now, recall that $q_*(s_t, a_t)$ follows the Bellman Optimality equation (equation 13), which allows us to rewrite the loss as follows:

$$L = \mathbb{E}[r_{t+1} + \gamma \max_{a_{t+1}} q_*(s_{t+1}, a_{t+1})] - q(\hat{\mathbb{E}}[s_{t+1}], a_t) \tag{18}$$

where γ is a discount parameter. Note that r_{t+1} can simply be retrieved from the experience tuple, but $\max_{a_{t+1}} q_*(s_{t+1}, a_{t+1})$ is still unknown. To obtain an approximation of this term, we simply put state s_{t+1} from e_t through the policy network and we select the action with the highest estimated q-value.

4.3.5 Target Network

The aforementioned approach suffers from one problem and that is that the same neural network is used to make the prediction and generate the target. As a result, after updating the weights, the computed target q_* will move in the same direction as q since they share the same network. This makes the training process like a dog chasing its own tail, which would result in an unstable network.

To resolve this issue, a new network is introduced: the target network. The target network is an exact copy of the policy network, though the weights remain frozen. These weights are updated every certain C steps, allowing the model to learn from newer, more accurate targets yet preventing the aforementioned issue. Here, C is a hyperparameter to be optimized, though it is common practice to set this somewhere between 5 and 15 episodes. A graphical representation of this training process is given in Figure 3.

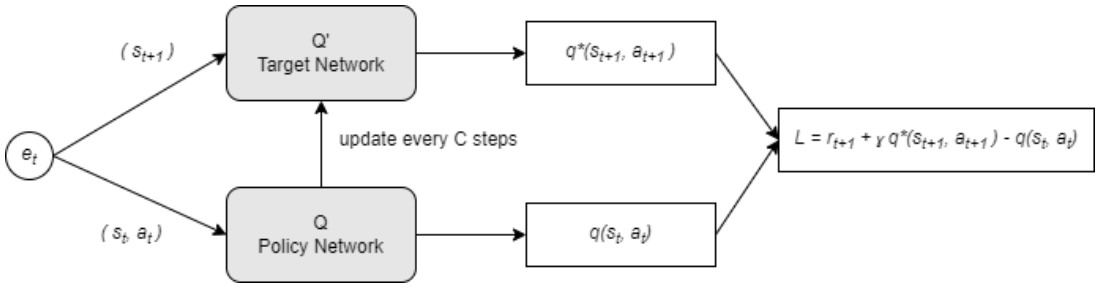

Figure 3. Simplified representation of the training process of a DQN

4.3.6 Epsilon Greedy Strategy

Let us now discuss one last important concept: the epsilon greedy strategy. If the DQL agent would always select the action with the highest estimated q-value, the network likely gets stuck into a local minimum. Therefore, at each step prior to selecting an action, the agent decides whether to go for exploration or exploitation. The latter is simply selecting the action with the highest estimated q-value. Exploration, however, randomly selects an action to explore this option further and possibly acquire new, insightful experiences. Whether exploration or exploitation is performed, is determined randomly before selecting an action. The exploration rate does not remain constant over time, instead, it follows a decaying scheme. The following decaying scheme will be used in this application:

$$\epsilon_t = \epsilon_{end} + (\epsilon_{start} - \epsilon_{end}) \times exp(-td) \tag{19}$$

where ϵ_t is the exploration rate at episode t, ϵ_{end} is the minimum exploration rate, ϵ_{start} is the starting exploration rate, and d is the decay-rate. All of which are tunable hyperparameters. However, it is custom to set ϵ_{end} close to 0 and ϵ_{start} close to 1. The selected scheme is simple to implement and yields sufficient results, such as in Hariharan and Paavai (n.d.), albeit in a different application. The most influential hyperparameter is the decay-rate d. At the beginning of training, it is favorable to explore many options. However, as the network learns more, exploitation can become a healthy part of training by reaffirming some of the good strategies the agent has already learned. Therefore, the probability of selecting exploration will decrease over time exponentially. Selecting this decay-rate such that this trade-off is managed adequately is a large challenge in DQL.

4.4 Adjusted Methodology

Throughout the experiment, various observations indicated a need to adjust the methodology. Diagnosing the behavior and iteratively improving the model design is a crucial part of the process in any Deep Learning project. This section outlines the methodological adjustments that were executed throughout the development of the final results. The result section will numerically illustrate the importance of the model adjustments.

4.4.1 Increasing Experience variety with Random Starting States

The aforementioned epsilon greedy strategy describes the trade-off between exploitation and exploration. However, the first implementation of this strategy was lacking in multiple dimensions. As the agent randomly selects an action at the beginning of training, certain states will rarely be observed. To illustrate this, let us consider the simple case with one wind turbine. At the early stages of training, the exploration rate is close to one meaning it will randomly select to maintain the turbine or do nothing. Since on average turbines only break down at a relatively large age, say after 24 months, the agent would need to choose not to maintain for many subsequent months in order for it to observe a breakdown; the probability of this happening is virtually zero. When the exploration rate finally lowers and the agent learns to abstain from maintenance for a longer period of time, the DQN has already adopted a no-maintenance policy. At this moment, the exploration rate is already at a low level, meaning the agent is not likely to perform maintenance due to exploration and its experiences with maintenance are scarce. This prevents the agent from exploring the long-term benefits of preventive maintenance.

To improve the effectiveness of the epsilon greedy strategy, two major improvements are implemented. First, the starting ages of the wind turbines are initialized randomly as an integer between one and the first lifetime, which is drawn from a Weibull Distribution. This helps to increase the variety of states the agent experiences, and thus the replay memory will be more diverse. In the literature, such as van Hasselt et al. (2016), this is common practice. Secondly, states in which the agent decided to not schedule maintenance will no longer be saved to the replay memory. This will help to keep the replay memory more diverse and might allow the DQN to more accurately approximate the q-values of a wider variety of states. It will prevent the DQN from overestimating the action ”0 PMs”, as otherwise the memory gets flooded with these 0-actions and many 0-costs as a result; wind turbines do not break down every period.

4.4.2 Restricting Action Space with p-ARP Knowledge Injection

Although the prior adjustments sound good on paper, in practice the agent simply does not realize the long-term benefits of maintenance. In fact, as the result section will illustrate, the agent never schedules maintenance at all. To help the understanding of these benefits, knowledge regarding the optimal scheduling for one wind turbine can be injected into the model for guided exploration. At first, the optimal period Age Replacement Policy will be calculated for the given parameters. Then this policy will become part of the agent's decision-making process. First, the agent will check how many wind turbines will pass the critical maintenance age next period. As the setting mainly allows for benefits in scheduling additional maintenance, the action space will be restricted accordingly. Let m_{t+1} be the number of wind turbines that will surpass the critical maintenance age in period $t+1$, assuming they do not fail. Now, the action space in period t is defined as:

$$A_t(m_{t+1}) = \{m_{t+1}, ..., n\} \tag{20}$$

Intuitively, the model can now be described as a two stage approach. First, it is decided whether next period requires any maintenance at all. The p-ARP provides a lower bound for scheduling next month's PMs. Consecutively, the agent decides exactly how many turbines to perform maintenance on. This

combines the knowledge regarding optimal single wind turbine policies with the DQN's ability to recognize potential logistical cost savings. Techniques to combine domain knowledge with neural networks are already common practice in fields like physics, though, to the best of our knowledge, we are the first to implement this in a maintenance setting.

4.4.3 Decaying Learning Rate

Even after the previous adjustments, the model still showed interesting but poor behaviour. More specifically, during the exploration stage the agent seems to discover significant cost savings due to randomly scheduling additional maintenance. However, as the number of episodes passes and the exploration rate decreases, the agent will select more of what it believes to be the best policy. Although cost savings would be expected, the contrary can occur when the agent did not adequately learn from the prior experiences. The newly generated experiences might be far from optimal and the replay memory will get flooded with these in the later stages of training. As a result, the agent will unlearn some of the cost savings it discovered during exploration and learn a sub-optimal policy.

To solve this issue, a decaying learning rate for the DQN is implemented. After each episode, the learning rate is multiplied by $0 < \omega < 1$. The learning rate determines the step size during weight optimization: a high learning rate means larger weight updates and a low learning rate means smaller weight updates. In practice, this means that a large learning rate allows the model to learn faster at the risk of converging to a local non-optimal minimum. This has an additional effect on our application: when the experience memory gets flooded with less optimal experiences, the agent will also rapidly forget the cost-savings it discovered during exploration. To reap the benefits of fast learning at the start of training but also prevent the agent from unlearning good policies, a decaying learning rate seems appropriate. There are many decay schedules, however, the exponential decay yields sufficient results, and hence no further experimentation was done. The decaying learning rate is applied to all parameters and ADAM is used for optimization.

4.5 Evaluation

To evaluate the aforementioned methodology, numerical experiments are performed. First, Gurobi (Gurobi Optimization, LLC, 2022), a Python package, is used to solve the LP formulation of the p-ARP for the given set-up costs. Applying this policy to each wind turbine individually will provide us with a naive benchmark strategy. Furthermore, the MBRP policy as discussed in section 4.2.2 is used as another benchmark as it is better at exploiting the set-up cost structure. Moreover, MBRP has large plannability, which makes it easier to schedule maintenance in advance. The p-ARP and MBRP for all instances are found in Appendix A and B respectively. Lastly, the agent's performance will be compared against a pure no-PM strategy. Various statistics are used to compare the behavior of the agent with the strategies.

To train the agent, a custom environment is built in Python. This environment can also be used to compare the policies in a simulation. The data for simulations will be generated as described in section 3. The same random seed will be used for each policy, ensuring a comparison under similar circumstances. PyTorch (Paszke et al., 2017) provides a library with all necessary tools for developing the Deep Q-Network, which is used for this implementation. Little resources were available for hyperparameter

tuning and thus its efforts were limited; the selected hyperparameters can be found in Appendix E. All operations are run on a NVIDIA GeForce GTX 1650 GPU supported by an Intel Core i5-10300H CPU.

5 Results

This section presents the results of the previously described simulation experiment. First, the monthly costs per wind turbine are shown for various problem instances and maintenance statistics are analyzed for one of the problems. After that, the impact of the adjusted methodology is illustrated. Lastly, the output of the DQN for various states is analyzed to obtain more insights into the valuation of the DQN. For the rest of the results section, the parameters are set as follows: $\alpha = 36$, $\beta = 2$, $\bar{c}_{cm} = 50$, $\bar{c}_{pm} = 10$, $\Delta = 20\%$.

The results from Schouten et al. (2022) were replicated and found to be the same. The implemented policies for a single wind turbine are used as a benchmark for our model. More specifically, p-ARP is applied to each turbine individually, where set-up costs are only shared at random. The MBRP is applied to the wind farm as a whole, which allows for frequently sharing the set-up costs. The critical maintenance age determined by p-ARP is given in Appendix A for various levels of c_{set-up}. The MBRP policies can be found in Appendix E. Lastly, a "No PM" policy is used as the baseline, which simply never schedules PM and waits for the wind turbines to break down.

5.1 Simulation Experiment

To verify the validity of our Deep Q-Learning approach, a simulation experiment is performed. The agent is compared to the No PM, p-ARP, and MBRP under equal circumstances as explained in section 3. First, the monthly costs for various problem instances are computed to compare the effect of the number of wind turbines and set-up costs on the performance. Afterward, the behavior of the DQL agent is analyzed and compared against the benchmarks for some additional simulated runs.

5.1.1 Monthly costs per wind turbine

Let us first investigate the monthly cost per wind turbine for various numbers of wind turbines. Table 3 shows the results from the simulation experiment. Here, $\hat{\mu}$ is the average monthly cost per wind turbine computed after 100 simulations of 30 years (time steps are in months), $\hat{\sigma}$ is the corresponding variance of these averages, and n is the number of wind turbines. The starting states are initialized randomly each run, though they are the same for each policy to ensure comparability. The performance of the Deep Q-Network (DQN) is measured against a "No PM"-policy, Modified Block Replacement Policy (MBRP), and Period Age Replacement Policy (p-ARP) for various instances of the model. Here, the other policies are applied to each wind turbine individually, and cost savings on PM-grouping thus occurs at random.

The first expected, economic interpretation of the results can be found in the relation between set-up costs and the monthly cost per wind turbine. Evidently, as the set-up cost increases, the cost per wind turbine increases as well for the benchmark policies. Moreover, for the p-ARP and MBRP, the costs decrease as the number of wind turbines increases. Here, preventive maintenance is sometimes grouped

Table 3. *Monthly cost per wind turbine for various wind farm sizes.* $\hat{\mu}$ *is the average monthly cost per wind turbine over the 100 simulations,* $\hat{\sigma}^2$ *the corresponding variance.* $\alpha = 36$, $\beta = 2$, $\bar{c}_{cm} = 50$, $\bar{c}_{pm} = 10$, $\Delta = 20\%$.

	c_{set-up}	5		10		20		30		50		time*
n		$\hat{\mu}$	$\hat{\sigma}^2$	$\hat{\mu}$	$\hat{\sigma}^2$	$\hat{\mu}$	$\hat{\sigma}^2$	$\hat{\mu}$	$\hat{\sigma}^2$	$\hat{\mu}$	$\hat{\sigma}^2$	
1	No PM	1.634	0.243	1.783	0.288	2.079	0.390	2.483	0.172	**3.109**	0.791	-
	MBRP	1.358	0.082	1.599	0.094	2.034	0.123	2.417	0.153	**3.109**	0.236	-
	p-ARP	**1.356**	0.082	**1.597**	0.094	**2.031**	0.120	**2.415**	0.152	**3.109**	0.236	-
	DQN	1.759	0.034	2.141	0.029	2.079	0.390	2.417	0.508	**3.109**	0.791	0.5
3	No PM	1.624	0.108	1.817	0.039	2.123	0.051	2.430	0.065	2.430	0.065	-
	MBRP	**1.146**	0.029	**1.291**	0.033	**1.580**	0.048	**2.030**	0.056	**2.216**	0.076	-
	p-ARP	1.154	0.029	1.298	0.034	1.721	0.043	2.034	0.057	**2.216**	0.076	-
	DQN	1.624	0.108	1.800	0.098	2.066	0.173	2.412	0.171	2.409	0.163	0.5
12	No PM	1.632	0.020	1.781	0.023	2.077	0.031	2.374	0.041	2.967	0.063	-
	MBRP	1.051	0.007	1.139	0.008	**1.521**	0.012	1.938	0.020	2.436	0.031	-
	p-ARP	1.065	0.007	1.147	0.009	1.526	0.012	1.750	0.017	2.436	0.031	-
	DQN	**0.932**	0.004	**1.019**	0.005	1.533	0.009	**1.531**	0.020	**2.264**	0.019	0.5
32	No PM	1.627	0.011	1.775	0.013	2.071	0.018	2.367	0.024	2.959	0.037	-
	MBRP	**1.055**	0.003	**1.122**	0.004	**1.498**	0.006	1.704	0.008	2.334	0.013	-
	p-ARP	1.059	0.004	1.128	0.004	1.504	0.006	1.709	0.008	2.334	0.013	-
	DQN	1.277	0.003	1.265	0.003	1.580	0.010	**0.849**	0.016	**1.421**	0.005	2.6
64	No PM	1.629	0.008	1.777	0.009	2.074	0.013	2.370	0.017	2.963	0.026	-
	MBRP	**1.045**	0.002	**1.122**	0.002	1.489	0.003	1.690	0.004	2.313	0.006	-
	p-ARP	1.062	0.002	1.129	0.002	1.494	0.003	1.695	0.004	2.313	0.006	-
	DQN	1.186	0.002	1.387	0.003	**1.403**	0.005	**0.933**	0.001	**1.326**	0.004	3.9

Note. * convergence time is in hours, the model is considered converged when the exploration rate is below 0.03. The difference in ϵ_{decay} is the main determinant of convergence time, n not so much.

at random, which allows for cost savings on the set-up costs. As the number of wind turbines increases, this grouping occurs more frequently. As can be seen in Appendix A, the p-ARP only maintains in two months, meaning that the random grouping of maintenance occurs rather frequently. Appendix B shows that the optimal MBRP policy simply convergences to a p-ARP that maintains once per year in August. For $c_{set-up} = 50$, the two policies become equal, hence, the identical results. These PM grouping effects are amplified by large wind farms, leading to an economy of scale effects due to the shared set-up costs. However, the DQN does not seem to exactly follow this trend. In certain instances, the model performs better than others, leading to situations where the costs increase as the number of wind turbines increases, or where the costs decrease even though the set-up cost increases. One possible explanation is the limited hypertuning efforts. Hyperparameters for each number of wind turbines were chosen after some experimentation with $c_{set-up} = 30$; perhaps the selected set of hyperparameters better fits $c_{set-up} = 30$ than $c_{set-up} = 20$. These hyperparameters can be found in Appendix E.

Consequently, the performance of the DQN seems to be extremely reliant on the setting. One of the main determinants of its relative monthly cost per wind turbine is the number of wind turbines. For smaller wind farms, there are little to no cost savings possible by sharing set-up costs. This becomes visible in the performance of the DQN. For $n = 1$, the DQN learns to schedule maintenance frequently in cases with low set-up costs. This makes the policy more expensive than "No PM", however, it keeps the variance in monthly average costs low. When the set-up costs increase, the DQN becomes hesitant to schedule PM

at all and the costs become close to - or equivalent to that of the No PM policy. Notably, the p-ARP and MBRP costs are the same for $c_{set-up} = 50, n = 1$ as the critical maintenance age becomes so high that the single wind turbine always breaks down before the site is visited. The DQN does not beat p-ARP or MBRP for $n = 1, 3$.

That being said, significant cost savings are discovered for most cases with $n = 12$. From this point onward, the DQN learns to rather aggressively schedule maintenance as it can do so rather cheaply. Intuitively, the agent learns that the marginal costs of scheduling additional maintenance are low compared to the initial preventive maintenance. The cost savings vary between 7.0% - 14.3%. This effect becomes even stronger for certain cases with $n = 32$. For $c_{set-up} = 5, 10, 20$ the model performs close to the p-ARP and MBRP. However, for the higher levels of set-up costs, large cost savings are found; these savings reach up to 50%. It exploits the economic property of the setting, as the marginal cost is low for additional maintenance. The agent rarely maintains one wind turbine, instead, it visits the site rather often to maintain a batch of wind turbines. This causes large savings on the set-up costs, leading to lower overall costs. A more detailed explanation of this behavior can be found in the upcoming sections. Note that, in this case, the p-ARP and MBRP perform poorly. Their critical maintenance age becomes high due to the large set-up costs. As a result, wind turbines frequently fail before this age is even reached, making their monthly average costs close to that of the No PM policy.

5.1.2 Maintenance Statistics

To better understand where the cost savings are coming from, let us take a look at the behavior of the DQN in a favorable problem instance. The following section will consider a wind farm with 32 wind turbines and $c_{set-up} = 30$, the rest of the parameters can be found in the description of Table 3. Table 4 shows the total PM statistics for 100 simulations with a length of 360 months.

Table 4. *PM grouping statistics for a wind farm with 32 turbines over 100 30-year simulations*

PMs at once	1 - 5	6 - 10	11 - 15	16 - 20	21 - 25	26 - 30	30+	Total	CMs/year
DQN	3245	1	2	3	51	25	36	3363	**3.81**
p-ARP	164	144	558	131	2	0	0	999	7.99
MBRP	367	391	13	0	0	0	0	771	8.95
No PM	0	0	0	0	0	0	0	0	11.36

The table shows the number of times the site was visited and how many PMs were performed during that visit. As the numbers illustrate, the DQN schedules maintenance much more frequently than the other strategies. In total, it visits the site more than three times as often as the p-ARP. As a result, its average CMs per year is the lowest of all strategies by a landslide. Most of these visits include between 1 and 5 PMs at once. Of these 3245 visits, 3226 visits include two PMs, making it the most frequently occurring number of PMs per period. In the end, the DQN rarely deems it worth it to do only one PM and almost always performs at least a second one. In a problem instance with a relatively large number of wind turbines, there are many possibilities to do so as the second oldest wind turbine is likely to be of relatively old age as well. This explains why the performance is so strong compared to other problem instances with a smaller number of wind turbines. Another remarkable observation is the fact that the DQN occasionally schedules a large group of PMs at once, sometimes even the entire wind park. However, operations larger than 20 PMs only occur on 112 occasions over 36000 simulated

months. During these months, even wind turbines with age 1 are maintained, which cannot logically be explained.

Upon further inspection of the ages at which the model performs PM, an interesting trend can be seen. Figure 4 shows a histogram of the ages at which PM is performed. The figure shows a clear trend: DQN performs PM at a low age, whereas p-ARP and MBRP only maintain at larger ages. The difference between p-ARP and MBRP can be explained by looking at the policies more carefully. The MBRP only performs maintenance in August on wind turbines aged over 37. After maintenance, the next time this wind turbine reaches this age is in August 4 years later, assuming it did not break down. At this point, the wind turbine will be of age 48. For the DQN, the preventive maintenance age is mostly between 6 and 15 months. This might seem surprisingly low at first, but the DQN always performs maintenance on multiple wind turbines. The ability to split set-up costs drastically lowers the cost per PM, which the agent exploits in its decision. In fact, if the set-up cost is shared among 5 wind turbines, p-ARP gives a critical maintenance age of 17 in August. The DQN correctly recognizes that performing PM at a lower age gives favorable results if PMs are combined, due to the shared set-up cost.

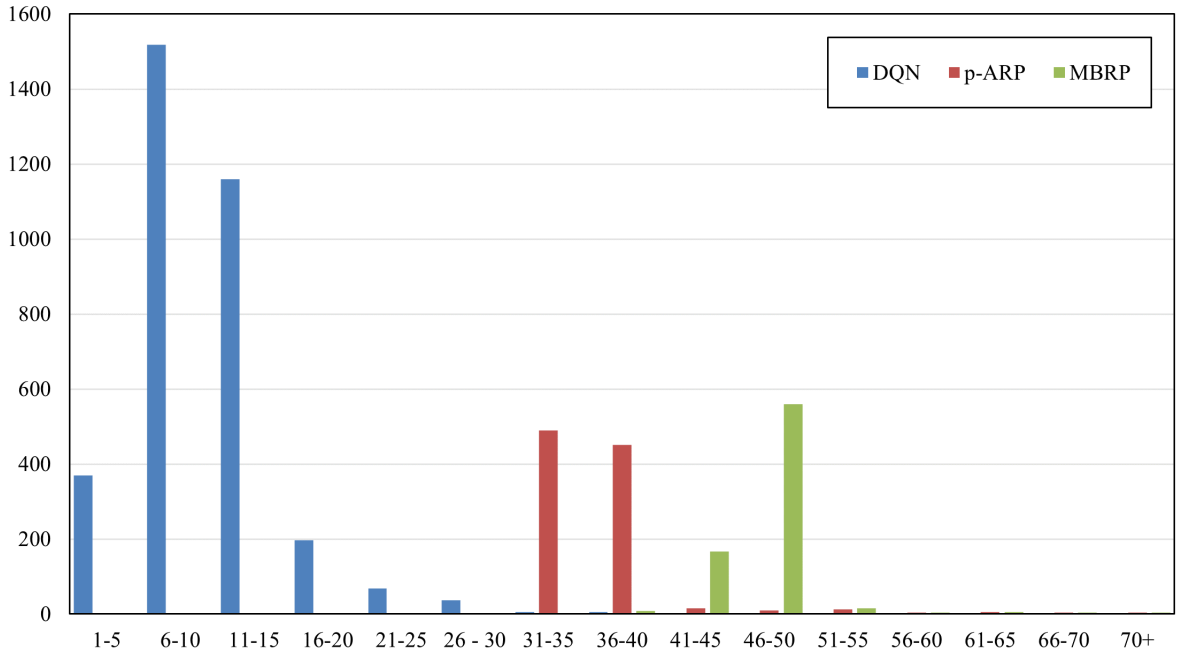

Figure 4. Histogram of the ages at which PM is performed for DQN, p-ARP, and MBRP. The problem parameters are $c_{set-up} = 30$, $\Delta = 20\%$, $\bar{c}_{cm} = 50$, $\bar{c}_{pm} = 10$, $\alpha = 36$, $\beta = 2$.

Another interesting result is found when looking at the ages of the *i'th PM*. The *i'th PM* is defined as the *i'th* oldest wind turbine when i or more PMs are performed at once. Figure 5 shows the average age of the *i'th PM* for the DQN, p-ARP, and MBRP. The trend followed by the p-ARP and MBRP is as expected. As these policies only schedule maintenance after the wind turbine is expected to exceed a large critical age, the line starts off high and shows a slight decrease. Note that the line can reach points lower than the critical maintenance age. As PMs are scheduled for t+1, one of the wind turbines could break down. Maintenance is not canceled, meaning that the next oldest wind turbine is maintained, which

has an age lower than the critical maintenance age. This causes a downward sloping trend in both lines. At the end, both the p-ARP and MBRP lines shoot upwards, though this is caused by the low number of observations in which these large numbers of PMs occur. The DQN shows an insightful trend: the average age of the first two PMs is low, after which the age shoots upwards at the third PM, subsequently following a downward sloping trend. As discussed earlier, the agent schedules plentiful maintenance operations consisting of 2 PMs. As it is able to share the set-up cost, the cost per PM is lower than for a single PM. Consequently, the agent often visits the site to maintain the two oldest wind turbines, even though their age is low. Surprisingly, the average maintenance age of the third PM is rather large. More than 2 PMs are rarely scheduled, and if they are, it is predominantly a large operation of more than 21 PMs. This explains the almost linear downward trend: when these PMs are performed in large groups, it is only logical that the 12th oldest wind turbine is slightly younger than the 11th oldest one.

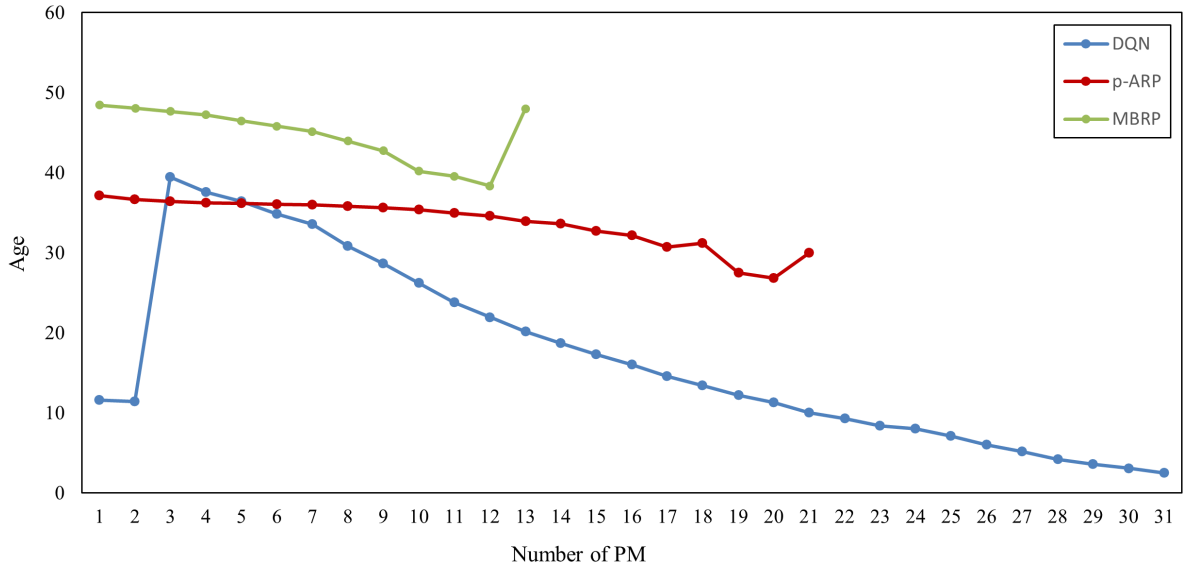

Figure 5. Average age of the *i'th PM*, which is defined as the *i'th* oldest wind turbine when i or more PMs are performed in one period. The parameters are $c_{set-up} = 30$, $\Delta = 20\%$, $\bar{c}_{cm} = 50$, $\bar{c}_{pm} = 10$, $\alpha = 36$, $\beta = 2$.

Figure 5 also gives us insight into the limited economic understanding of the DQN. The marginal cost of the second and third PM are equal; set-up costs are already incurred after the first PM. Therefore, it is surprising to see that the average maintenance age of the third PM is notably higher than the second. Of course, the marginal benefits are lower as well, considering the third oldest wind turbine is always slightly younger than the second one, though this does not explain the large observed jump. Ideally, the DQN would have better captured the relation between marginal cost/benefits and maintenance age.

5.2 Base Model vs. Knowledge Injection vs. Decaying Learning Rate

To illustrate the importance of the decaying learning rate, a comparison is made between the base model, the model with knowledge injection, and the model with a decaying learning rate. The specification of these models can be found in section 4.4.

5.2.1 Convergence of the models

The original challenges of the Deep Q-Network in this application are best illustrated by Figure 6. This graph shows the total cost moving average over the past 20 episodes for all models throughout training. In this instance, there are 32 turbines on the wind farm and an episode takes 30 years.

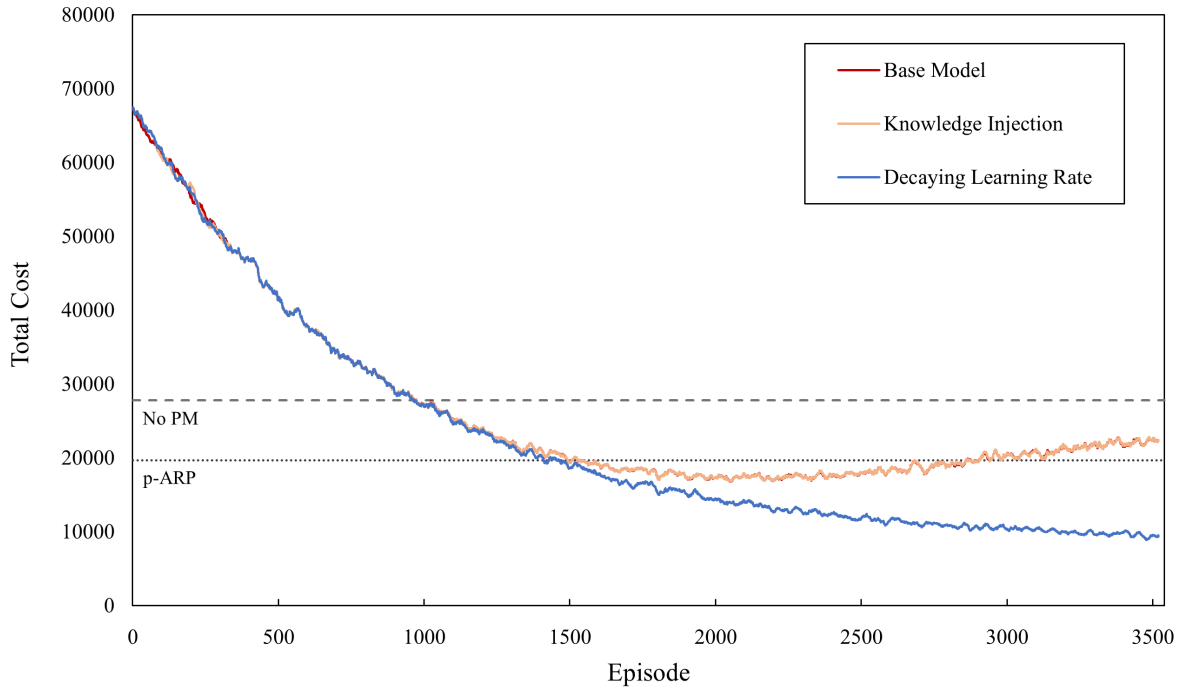

Figure 6. Total cost moving average over the last 20 episodes (30 years per episode), $\omega = 0.9$, $n = 32$. The models are defined in section 4.4; the Base Model and Knowledge Injection follow almost the exact same trajectory, causing the limited visibility of the red line.

As the figure illustrates, the base model and the model with knowledge injection behave almost identically. The costs decreases as the number of episodes rises, up until 2000 episodes. At this point, the exploration rate is 13.6%, which means that the agent randomly selects an action with 13.6% and selects its best action with 86.4%. Up until this point, the model has been able to reach lower costs than p-ARP due to randomly scheduling additional PMs. However, as the exploration rate gets lower later on in the training process, the agent selects its best action and the costs rise again. This means that the DQN has not yet adequately learned from the exploration. Its replay memory gets flooded with new, sub-optimal actions on which the DQN is retrained. This causes the DQN to ”forget” the beneficial experiences from the past, thus unlearning the previously seen policy. Due to the learning rate not decaying, the weights update aggressively in this direction and the agent is strongly reinforcing a sub-optimal policy as costs rise again above the benchmark.

In this instance, the model with a decaying learning rate of $\omega = 0.9$ does not suffer from this issue. This strong decay means that the model will first aggressively learn from its successes. Though as the training proceeds, new experiences cause the weights to update less. This prevents the DQN-weights

from converging to a local, sub-optimal minimum, which occurs for the base and knowledge injection model. Perhaps, the same could be established by lowering the exploration decay, allowing the model more time for exploration. However, this would require much more episodes for the model to converge, which drastically increases the running time.

5.3 State Analysis

To better understand the decision-making of the agent, the DQN-outputs for various states are analyzed in this section. First, let us look at a more general state for 32 wind turbines with $c_{set-up} = 30$. In this state, there are 5 wind turbines aged 50, 4 turbines aged 35, and 6 turbines aged 10; the rest has age 1. The output of the DQN for all time periods is shown in Figure 7. Recall that the output corresponds to the estimated q-values for each action in the corresponding state. The vertical axis denotes the months for which PM is scheduled and the horizontal axis corresponds to the number of scheduled PMs.

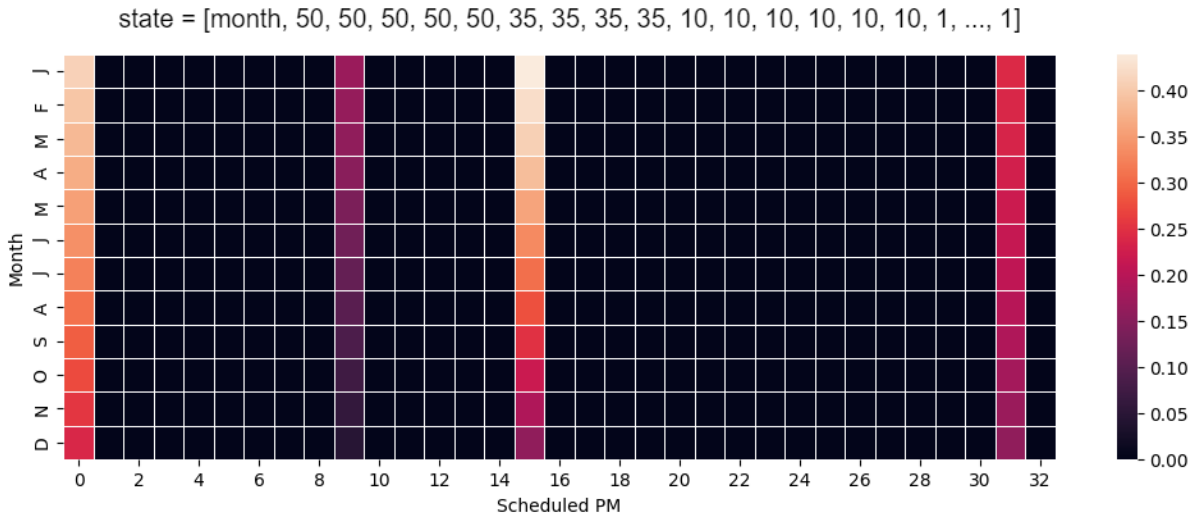

Figure 7. Output of the policy network for various months, represented as a heat map of the estimated q-values for each possible action. For this instance, there are 32 wind turbines and $c_{set-up} = 30$. The other parameters are $\Delta = 20\%$, $\bar{c}_{cm} = 50$, $\bar{c}_{pm} = 10$, $\alpha = 36$, $\beta = 2$.

As the heat map shows, the policy network always decides to perform 15 PMs. It is unsure how to evaluate 0 as these actions are not saved in the replay memory. In the summer months, the p-ARP guideline for the given state is 9. The DQN visibly keeps this guideline in mind, as 9 has q-values estimated slightly above 0. The agent decides to schedule additional maintenance for the wind turbines at age 10, whereas the p-ARP would not. Action 31 is also judged rather positively by the agent, however, always strictly worse than 15 for this state. This is surprising, considering the rest of the wind turbines have age 1. Interestingly, the estimated q-values between 9 and 15 do not increase gradually, but rather are 0. Perhaps, this is because, at the later stages of training, the agent has reaffirmed that 15 is the best option for similar states and no longer considers lower numbers of PMs at all. As the statistics in 5.1.2 confirm, the model rather aggressively schedules maintenance and sometimes schedules large operations to share the set-up costs by grouping the PMs.

The number of scheduled PMs is not influenced by the month for this problem instance. Perhaps, the grouping of PMs leads to cost savings that make the time-varying costs less relevant for decision making. What is more interesting is that the estimated q-values for months at the beginning of the year are higher than for months towards the end of the year. Possibly, this can be explained by the parameter settings of $\alpha = 36$ and $\beta = 2$. This leads to an average wind turbine lifetime of 32 months. If 15 PMs are performed in January, all turbines will have age 1 in the next state. As a result, the next expected failure is in August two years later, a much cheaper month to perform PM or CM in than earlier or later on in the year.

Let us now have a closer look at a problem instance where the DQN does not perform as well. More specifically, let us investigate the problem for 3 wind turbines and $c_{set-up} = 30$. Figure 8 shows the estimated q-values for each action with on the x-axis the action, i.e. the number of PMs, and on the y-axis the month. As the figure illustrates, the DQN learned to not maintain, even for states where the wind turbine ages far surpass the critical maintenance age from p-ARP. Again, recall that 0-actions were not saved to the replay memory, making the evaluation of this action unstable. However, the figure clearly illustrates that little was learned regarding the relationship between age and maintenance.

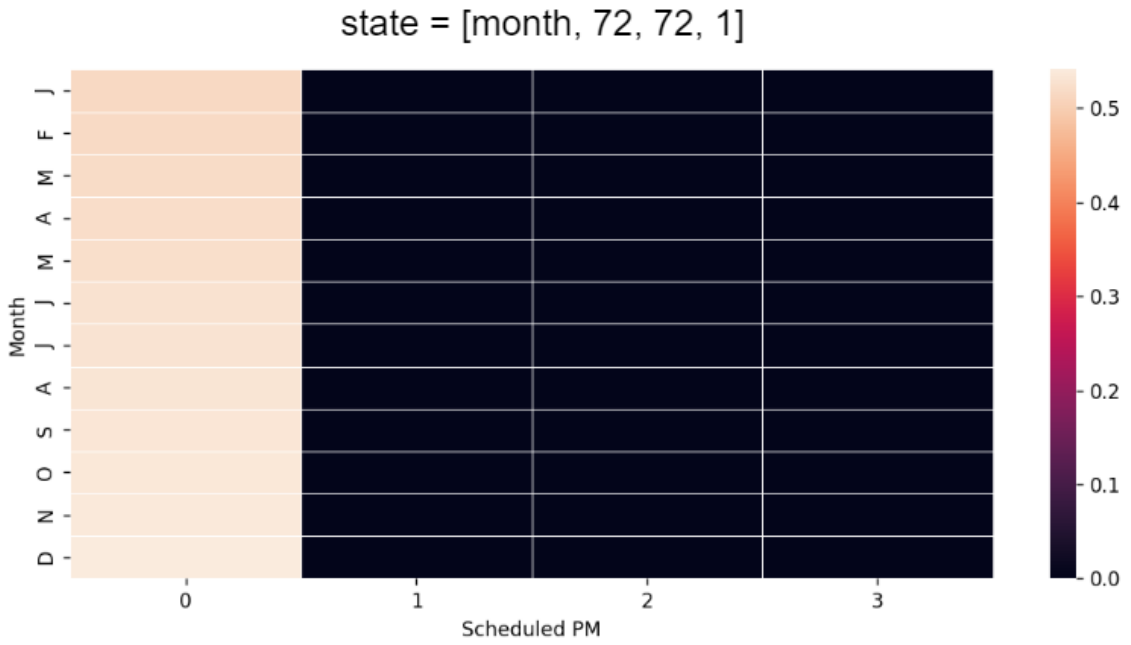

Figure 8. Output of the policy network for various months, represented as a heat map of the estimated q-values for each possible action. For this instance, there are 3 wind turbines and $c_{set-up} = 30$. The other parameters are $\Delta = 20\%$, $\bar{c}_{cm} = 50$, $\bar{c}_{pm} = 10$, $\alpha = 36$, $\beta = 2$.

6 Conclusion and Discussion

In this paper, we have inspected the option of using Deep Q-Learning for scheduling preventive maintenance for large-scale wind farms with set-up costs. A custom environment was built to train the DQN and measure its performance against single wind turbine strategies, applied to each wind turbine individually. After enhancing the DQN with knowledge from these strategies, and implementing a decaying

learning rate, simulations of various problem instances were run. Mainly the decaying learning rate for the DQN seemed to drastically improve its performance; without it, the agent suffered from catastrophic forgetting. This means that the "good" memories were slowly forgotten as new, sub-optimal memories update the weights incorrectly and flooded the replay memory. Without the decaying learning rate, the agent did not accurately learn a good policy during its exploration phase, leading to worse performance when the agent started to rely more on its own "optimal" policy. The behavior of the agent was further analyzed using various statistics as a result of the simulation. The results paint a two-faced picture: the performance of the Deep Q-Learning agent is notably reliant on the problem setting. The favorable scenarios are characterized by large set-up costs and a large number of wind turbines. In these scenarios, cost savings of up to 50% were found. In less favorable scenarios, the costs of the agent were in-between the costs of p-ARP and not scheduling preventive maintenance at all.

This two-faced picture is not strange, on the contrary, it was to be expected. When set-up costs are low, there is little benefit to be gained from combining PMs, making p-ARP when individually applied to each wind turbine rather strong. Moreover, p-ARP and MBRP only perform maintenance in one or two months, making random grouping of PMs occur frequently. However, in instances with many wind turbines and high set-up costs, the model strongly outperforms the benchmark for various reasons. The DQN is able to exploit the cost structure by implementing the following strategical elements: maintenance is done much more frequently and at a younger age. The grouping of multiple PMs leads to a lower cost per PM, which makes maintaining more frequently at younger ages favorable. Moreover, the agent occasionally scheduled a large number of PMs in which almost the entire wind farm was maintained. In fact, by using these strategies, the DQN was able to reduce costs by over 50% for specific problem instances. Although the cost savings were significant, the agent was not able to accurately reflect the structure of the marginal costs of preventive maintenance.

Reflecting critically on the presented research, the journey is far from over. The model required various iterations before it provided decent results. In the end, Deep Q-Learning is not a one-size-fits-all technique; it needs to be appropriately adapted to fit the setting. One of the main downsides of DQL is the number of hyperparameters that need to be tuned. In fact, little effort was made to tune these parameters due to the limited time and the required computational power. However, hypertuning is mainly of interest for commercial purposes and further improving performance. Instead, future research could spend their resources on the following topics. Multi-objective Deep Q-Learning could be considered for this setting. Currently, the maintenance costs are being minimized by the agent. However, in reality, maintenance is only part of the bigger picture; manufacturing, staff management, inventories, etc. are all shaping elements of the wind turbine cost structure. Mossalam et al. (2016) propose a Deep Reinforcement Learning method to solve multi-objective, high dimensionality problems. Additionally, an effort could be made to reduce the credit assignment problem, i.e. the issue of solving problems with long-term reward dependencies. In this setting, the "benefit" from maintaining is the evasion of corrective maintenance. However, in DQL, the network is trained using the current state, action, next state, and direct reward. This leaves little room for long-term benefits to be understood by the model.

After all, DQL shows promising results in specific problem instances. However, its power certainly is limited by the setting. After various iterations, the model finally yielded great results, albeit for specific cases. This research shows that a thorough understanding of the problem is required to get decent results,

and even then, they are not guaranteed as some problems are simply not suited for the method due to long-term reward dependencies. That is why, in the end, we would call this project a moderate success.

References

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*.

Arvesen, A., Birkeland, C., & Hertwich, E. G. (2013). The importance of ships and spare parts in lcas of offshore wind power. *Environmental science & technology*, *47*(6), 2948–2956.

Association, E. W. E. et al. (2011). *Eu energy policy to 2050*. EWEA.

Barlow, R., & Proschan, F. (1965). *Mathematical theory of reliability*. Wiley.

Ciang, C. C., Lee, J.-R., & Bang, H.-J. (2008). Structural health monitoring for a wind turbine system: A review of damage detection methods. *Measurement Science and Technology*, *19*(12), 122001. https://doi.org/10.1088/0957-0233/19/12/122001

Coelingh, J., van Wijk, A., & Holtslag, A. (1996). Analysis of wind speed observations over the north sea. *Journal of Wind Engineering and Industrial Aerodynamics*, *61*(1), 51–69. https://doi.org/https://doi.org/10.1016/0167-6105(96)00043-8

Degrave, J., Hermans, M., Dambre, J., et al. (2019). A differentiable physics engine for deep learning in robotics. *Frontiers in neurorobotics*, 6.

Elia, A., Taylor, M., Ó Gallachóir, B., & Rogan, F. (2020). Wind turbine cost reduction: A detailed bottom-up analysis of innovation drivers. *Energy Policy*, *147*, 111912. https://doi.org/https://doi.org/10.1016/j.enpol.2020.111912

Fan, J., Wang, Z., Xie, Y., & Yang, Z. (2020). A theoretical analysis of deep q-learning. *Learning for Dynamics and Control*, 486–489.

Gao, Z., Gao, Y., Hu, Y., Jiang, Z., & Su, J. (2020). Application of deep q-network in portfolio management. *2020 5th IEEE International Conference on Big Data Analytics (ICBDA)*, 268–275.

Gurobi Optimization, LLC. (2022). Gurobi Optimizer Reference Manual. https://www.gurobi.com

Hariharan, N., & Paavai, A. G. (n.d.). A brief study of deep reinforcement learning with epsilon-greedy exploration. *International Journal of Computing and Digital Systems*, *11*(1), 541+. https://doi.org/https://dx.doi.org/10.12785/ijcds/110144

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., et al. (2018). Deep q-learning from demonstrations. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1).

Huang, J., Chang, Q., & Arinez, J. (2020). Deep reinforcement learning based preventive maintenance policy for serial production lines. *Expert Systems with Applications*, *160*, 113701.

Jang, B., Kim, M., Harerimana, G., & Kim, J. W. (2019). Q-learning algorithms: A comprehensive classification and applications. *IEEE Access*, *7*, 133653–133667. https://doi.org/10.1109/ACCESS.2019.2941229

Kansky, K., Silver, T., Mély, D. A., Eldawy, M., Lázaro-Gredilla, M., Lou, X., Dorfman, N., Sidor, S., Phoenix, S., & George, D. (2017). Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. *International conference on machine learning*, 1809–1818.

Karyotakis, A. (2011). On the optimisation of operation and maintenance strategies for offshore wind farms.

Mossalam, H., Assael, Y. M., Roijers, D. M., & Whiteson, S. (2016). Multi-objective deep reinforcement learning. *CoRR*, *abs/1610.02707*. http://arxiv.org/abs/1610.02707

Nakagawa, T. (1984). A summary of discrete replacement policies. *European Journal of Operational Research*, *17*(3), 382–392. https://doi.org/https://doi.org/10.1016/0377-2217(84)90134-6

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in pytorch.

Ren, Z., Verma, A. S., Li, Y., Teuwen, J. J., & Jiang, Z. (2021). Offshore wind turbine operations and maintenance: A state-of-the-art review. *Renewable and Sustainable Energy Reviews*, *144*, 110886. https://doi.org/https://doi.org/10.1016/j.rser.2021.110886

Rocchetta, R., Bellani, L., Compare, M., Zio, E., & Patelli, E. (2019). A reinforcement learning framework for optimal operation and maintenance of power grids. *Applied energy*, *241*, 291–301.

Röckmann, C., Lagerveld, S., & Stavenuiter, J. (2017). Operation and maintenance costs of offshore wind farms and potential multi-use platforms in the dutch north sea. In B. H. Buck & R. Langan (Eds.). Springer. https://doi.org/10.1007/978-3-319-51159-7_4

Ross, S. (1970). *Applied probability models with optimization applications.* Holden-Day.

Schouten, T. N., Dekker, R., Hekimoğlu, M., & Eruguz, A. S. (2022). Maintenance optimization for a single wind turbine component under time-varying costs. *European Journal of Operational Research*, *300*(3), 979–991.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, *362*(6419), 1140–1144.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction.* MIT Press.

van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *30*(1). https://doi.org/10.1609/aaai.v30i1.10295

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning*, *8*(3), 279–292.

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

Appendix A Optimal p-ARP for various set-up costs

Table 5 shows the time-varying costs over the year, as well as the optimal p-ARP policy for various levels of transportation costs. As the table clearly illustrates, maintenance costs are lowest in July. This reflects within the p-ARP outcome as well. For $c_{set-up} = 5, 10, 20, 30$ the critical maintenance age is lowest in July. For $c_{set-up} = 50$, PM is only performed in August. Likely, this is due to the expensive upcoming winter months: it is more favorable to do PM in August than in July because then the wind turbine is young and renewed for the winter period, in which a failure would lead to expensive CM.

Moreover, one can note that the critical maintenance ages increase as the set-up costs increase. Intuitively, this makes a lot of sense: higher set-up costs mean higher marginal maintenance costs. As a result, it is less attractive to do maintenance, and postponing it to a later age reaps more benefits.

Table 5. *period Age Replacement Policies throughout the year, for parameters* $\Delta = 20\%$, $\bar{c}_{cm} = 50$, $\bar{c}_{pm} = 10$, $\alpha = 36$, $\beta = 2$, $M = 72$

	J	F	M	A	M	J	J	A	S	O	N	D
Time-Varying Costs												
c_{cm}	60.00	58.66	55.00	50.00	45.00	41.34	40.00	41.34	45.00	50.00	55.00	58.66
c_{pm}	12.00	11.73	11.00	10.00	9.00	8.27	8.00	8.27	9.00	10.00	11.00	11.73
c_{set-up}												
5	M	M	M	M	M	M	24	16	M	M	M	M
10	M	M	M	M	M	M	26	20	M	M	M	M
20	M	M	M	M	M	M	37	27	M	M	M	M
30	M	M	M	M	M	M	42	33	M	M	M	M
50	M	M	M	M	M	M	M	44	M	M	M	M

Appendix B Optimal MBRP for various levels of set-up costs

Table **??** provides the optimal Period-dependent MBRP for various levels of set-up costs. Interestingly, the MBRP simplifies to a p-ARP policy that only maintains in August. For $c_{set-up} = 50$, the policy is equal to p-ARP, which explains the equal results in Table 3. For the computation of this policy, policies that repeat every $m = 3$ years are considered. For more information on this, see the sections 4.2 and 4.3 in the original paper by Schouten et al. (2022).

Table 6. *The optimal Period-dependent Modified Block Replacement Policy for various levels of* c_{set-up} *for parameters* $\Delta = 20\%$, $\bar{c}_{cm} = 50$, $\bar{c}_{pm} = 10$, $\alpha = 36$, $\beta = 2$, $M = 72$, $m = 3$

c_{set-up}	Maintenance Months	Critical Maintenance Ages
5	8, 20, 32	16, 16, 16
10	8, 20, 32	20, 20, 20
20	8, 20, 32	27, 27, 27
30	8, 20, 32	33, 33, 33
50	8, 20, 32	44, 44, 44

Appendix C Custom Environment

To train the agent, a custom environment is set up from scratch. An instance of this environment has the following variables:

1. alpha: scale parameter of Weibull distribution
2. beta: shape parameter of Weibull distribution
3. N: the number of periods in one year
4. cost: total cost of the episode
5. size: number of wind turbines
6. device: GPU used to train the model
7. data: Pandas DataFrame object containing the current age, time of next failure, and lifetime vector for each wind turbine

Besides that, the environment contains the following functions relevant to simulating the setting:

1. reset(): sets cost to 0, time to 1, and generates a new data set
2. sort_data(): sorts the dataframe based on the wind turbine ages, oldest to youngest
3. close(): returns the total costs of the episode
4. num_actions_available(): returns the number of available actions
5. take_action(preventive_maintenance): executes the previously scheduled preventive maintenance.
6. breakdowns(): realizes failure of wind turbines
7. get_state(): returns current state
8. get_expected_state(): returns state at the start of next period

Appendix D Psuedo Code Training Loop

Algorithm 1 Psuedo Code for Training Loop Deep Q-Network

Result: Trained Policy Network
episode = 0
costs = []
$\epsilon_{episode}$ = 1
policy_net = DQN
target_net = policy_net
target_net.eval()
while *While episode < max_episodes* **do**
    environment.reset()
    if *episode mod C == 0* **then**
        update weights of target_net
    end
    t = 1
    cost = 0
    pm_{t+1} = 0
    expected_state = environment.get_expected_state()
    while *t < steps_per_episode* **do**
        environment.breakdown()
        reward = environment.take_action(pm_{t+1}, t)*
        current_state = environment.get_state()
        if pm_{t+1} ! = 0 **then**
            e_t = (expected_state, pm_{t+1}, current_state, reward)
            add e_t to replay memory
        end
        expected_state = current_state
        pm_{t+1} = policy_net(expected_state)
        t = t + 1
        policy_net ← train on replay memory
    end
    update $\epsilon_{episode}$ costs.append(cost)
    if $\epsilon_{episode} < \epsilon_{end}$ **then**
        save model
        break
    end
end

* random action with probability ϵ_t, best-known action with probability $1 - \epsilon_t$

Appendix E Hyperparameters

Table 7 presents the hyperparameters used to run the various problem instances. Note that, in all cases except one, the hyperparameters are set the same for all levels of c_{set-up}, the exception being for 64 wind turbines and $c_{set-up} = 50$. As a sizeable amount of problem instances needed to be run, and limited computation power and time were available, the hypertuning efforts were restricted. No grid search was performed, instead, we experimented with various hyperparameters for various problem instances and selected the ones presented below.

Table 7. *Hyperparameters for the different number of turbines n.*

	hyperparameter					
	Policy Network				Reinforcement Learning	
n	hidden layers	neurons per layer	learning rate	decay	target update step	epsilon decay
1	1	[6]	0.001	0.9	5	0.003
3	2	[12, 8]	0.001	0.95	5	0.002
12	2	[24, 16]	0.001	0.9	5	0.003
32	2	[60, 50]	0.001	0.95	5	0.001
64	3	[80, 100, 75]	0.001	0.9*	5	0.001

Note. * For $c_{set-up} = 50$ a decay of 0.86 was used. A decay of 0.9 did not resolve the issue mentioned in section 4.4.3.