Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis Econometrics and Operational Research

Three State Markov Transition Approach for the p-ARP Maintenance Model

Bram Vader (534794)

Supervisor:	Prof. dr. ir. R. Dekker
Second assessor:	dr. R. Spliet
Date final version:	19th July 2023

Abstract

In this thesis, we extend the model described by Schouten et al. (2022), to further optimize maintenance for wind turbines considering time-varying cost. Our method extends the used Markov model using a three state transition approach. Daily wind velocities are used to take the large variability of the weather into account. We estimate the power output curve of a single wind turbine, to calculate the daily wind power output. For every week, we divide the daily power distribution in three states: low, average and high wind conditions. This classification is then used to estimate transition probabilities, which are necessary for the linear programming (LP) formulation to determine the optimal maintenance policy. We show that there is a total cost decrease using our policy.

Contents

1 Introduction

Wind energy is becoming a more important source of renewable energy as countries are working hard to mitigate climate change. Wind energy shows great potential due to its relatively low environmental footprint. The global increase in demand in green energy has led to an upturn in the deployment of wind turbines. For example, the Netherlands must produce at least 35 TeraWatt-Hour (TWh) of renewable energy by 2030 (Rijksoverheid (2020a)). Wind energy contributes significantly to achieve this goal. By 2050, all energy used in the Netherlands must be generated by renewable sources (Rijksoverheid (2020b)). Land to build new wind turbines on is exhaustive, and therefore the Netherlands, among others, is increasing their wind energy generation from sea. By 2030, the Netherlands plan to have wind farms that produce 21 Giga-Watt of wind energy. These wind farms then supply 16% of the total used energy.

As the offshore wind industry advances, wind turbine technology will evolve. Wind turbines are getting bigger, higher and generating more power than before and wind farms will move further into sea. This leads to the importance of optimal maintenance planning. The goal is to find the best approach to distribute preventive maintenance and corrective maintenance to minimize the total costs over time. Due to the increasing distance of the offshore wind farms, it is preferred that the number of maintenance visits is minimized, while keeping the probability of the turbine shutting down due to deterioration low, where the more costly corrective maintenance is needed. Planning maintenance in periods where the average wind speeds are high results in a bigger loss in revenue compared to periods where the average wind speeds are low. Therefore it is crucial to decide when and what type of maintenance to perform.

In Schouten et al. (2022), long-run cost optimal age-based policies (ARP) and block-based maintenance policies (BRP) are investigated. These policies, the ARP, BRP and modified BRP, are extended to be able to include time-varying costs. By discretizing time, the optimal ARP is found using a linear programming (LP) formulation. This thesis provides an addition to the research of Schouten et al. (2022).

The cost parameters are calculated by estimating the power output of the wind turbines. Using historic wind velocity data, the daily power output is approximated. By taking the averages over all years and within the week results in the average historic power output for each week of the year. Furthermore, a cosine function is fitted to the resulting weekly averages. This method is very practical, but can be less realistic in terms of the weather variability. By using the wind data in the way it is described in Schouten et al. (2022), the variability in the wind velocities are possibly neglected. This is concerning because a wind turbine's power production is a nonlinear function of wind speed. Therefore the aim is to further improve the maintenance decision-making process by introducing a more realistic method in modelling the weather conditions.

More specifically, it is investigated whether the used Markov Decision Process (MDP) could be further improved. In Schouten et al. (2022), the MDP consists of two states, namely the period when maintenance can be conducted, and the age of the wind turbine component. As the weather is a fast-changing phenomena, a more specific, and therefore more realistic model could improve the decision-making process. Therefore, we add a third state variable accounting for the weather variability to the existing two state Markov decision process. Historic daily wind data are gathered to estimate weekly power output distributions. These are used to calculate new transition probabilities for the extended three state Markov Decision model. Subsequently, the main research question is as follows: Is it possible to further optimize the decision-making process to decrease the expected costs for a single wind turbine component under time-varying costs by accounting for the high variability of the weather conditions? So in other words: *What is the impact of weather variability on maintenance costs for a single wind turbine component?* In this thesis we will focus on extending the Age Replacement Policy (ARP).

2 Literature Review

In almost every sector maintenance is necessary to optimize production. To optimize production it is important that the right maintenance decisions are made. Subsequently, the topic of maintenance optimization is extensively researched. Investigating the cost structure implies that maintenance costs are one of the most important factors, as stated in Dinwoodie & Mcmillan (2014). Offshore operating and maintenance costs imply 23% of the total investment cost of a single wind turbine, compared to 5% of their onshore counterpart.

2.1 Structuring wind parks

The development of a wind park is a challenging and complex assignment, as optimization is needed in every domain to keep the price of the generated power low. First, we look at the wind farm layout problem. This consists of the optimal placement of the wind turbines with respect to interference due to close proximity to other wind turbines, resulting in reduced air velocity. Fischetti & Monaci (2016) introduced proximity search heuristics for optimal layout of wind parks, that combines ad-hoc heuristics and mixed-integer linear programming. A mixed-integer model is used, which focused on turbine distance constraints and on interference of the nearby wind turbines.

Next, cable routing is the problem that arises when modelling wind parks. To solve this problem, Fischetti & Pisinger (2018) introduced a Mixed Integer Linear Programming (MILP) model adhering that all turbines connect to one (or more) offshore substation(s) while being limited by cable capacities, no-cross restrictions, connection-limits at the substation, and obstacles at the site.

Lastly, the design of the wind turbine itself is also important in optimizing power output. Negm & Maalawi (2000) described several optimization models for the design of a typical wind turbine tower structure. The resulting optimization problem is then formulated as a nonlinear programming problem, and solved using the interior penalty function technique. The model succeeded in computing solutions that significantly improved the design with respect to system performance compared to baseline designs.

2.2 Maintenance optimization

Maintenance is typically categorized as corrective maintenance and proactive maintenance as stated in Shafiee (2015). Corrective maintenance is a failure based maintenance where maintenance is only executed when the failure of a component already has taken place. This type of maintenance is favorable if the downtime is low, such that the system is running back at normal circumstances in minimal time. However due to the marine environment of the offshore wind parks, the reduced availability implies that this type of maintenance, on itself, is not the most optimal choice. Proactive maintenance implies that the state of the system is periodically inspected. Noticing or repairing a minor deterioration could save the system from completely failing, and resulting in much greater costs.

Gonzalo et al. (2022) proposed a novel optimization problem, and is solved for real case studies by using Genetic Algorithms and Particle Swarm Optimization algorithms to minimize operational costs and maximize performance of the wind turbines. Wang & Deng (2022) proposed an optimization model for offshore wind turbines maintenance under the multiple constraints of the strategy time window. This model accounts for several restrictions concerning maintenance urgency. Aafif et al. (2022) investigates two models specifically for the wind turbine gearbox. It takes the gearbox' temperature into account, and decides its maintenance policy on the number of times the temperature of the gearbox exceeds a threshold. Concluding, there has been done significant amount of research. However, a notable deficiency in these studies is the absence of highly changing wind conditions in their models.

3 Problem statement

In this section the extended model is introduced, which adjusts the original Markov Chain defined in Schouten et al. (2022). Subsequently, we impose a new approach of determining the costs relevant to the downtime of a wind turbine due to maintenance. We focus on a single component system, where the lifetime of the component is modelled by a Weibull distribution. In the optimization model, we minimize long run costs by using the expected costs in each state. The goal is to develop a more specific approach that decides the optimal period to conduct preventive maintenance.

3.1 Markov Chain

We begin by defining the Markov Decision Process. The Markov Chain is created with a three dimensional state space, $\mathcal{I} = \mathcal{I}_1 \times \mathcal{I}_2 \times \mathcal{I}_3$, with $i_1 \in \mathcal{I}_1, i_2 \in \mathcal{I}_2, i_3 \in \mathcal{I}_3$, which differs from Schouten et al. (2022), where only $\mathcal{I}_1$ and $\mathcal{I}_2$ are used. We define $\mathcal{I}_1 \subseteq \mathbb{N}^+$, where $\mathbb{N}^+$ is the set of positive integers. $\mathcal{I}_1$ is defined as the set of periods in a year. In this model, we discretize the time into weeks. This results in $\mathcal{I}_1 = \{1 \dots 52\}$. Next, the age of the component is defined as $\mathcal{I}_2 \subseteq \bar{\mathbb{N}}$, where $\bar{\mathbb{N}}$ is the set of non-negative integers. Furthermore, $\mathcal{I}_3$ is defined as the wind conditions, where we differentiate the wind in to three states: low, average and high wind condition. In every time period i_1, with component age i_2, the wind is classified in one of the three states defined below.

$$\mathcal{I}_3 = \begin{cases} 0, & \text{Low wind conditions} \\ 1, & \text{Average wind conditions} \\ 2, & \text{High wind conditions} \end{cases} \tag{1}$$

As the addition of the new state variable i_3 does not affect the possible actions one could take when conducting maintenance, the action space is unchanged. We therefore define it the same way as stated in Schouten et al. (2022). There are two possible actions: replacing the component ($a = 1$) or doing nothing ($a = 0$). This results in the following definition:

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

3.2 Transition probabilities

Now that the possible states in the model have been defined, the transition probabilities are constructed. The transitions regarding the wind are independent of the age of the component, and are dependent of the period of the year. Furthermore, the transition is independent of the wind condition of the previous period. Let $\pi_{(i_1,i_2,i_3)(j_1,j_2,j_3)}(a)$ be the transition probability from state (i_1, i_2, i_3) to (j_1, j_2, j_3), under action $a \in \mathcal{A}$

$$\pi_{(i_1,i_2,i_3)(j_1,j_2,j_3)}(0) = \begin{cases} 1 - \tilde{p}_{i_2,i_3} & \text{for } j_1 = i_1 + 1 \text{ (mod N)}, \; j_2 = i_2 + 1, \;\; i_2 \notin \{0, M\} \\ \tilde{p}_{i_2,i_3} & \text{for } j_1 = i_1 + 1 \text{ (mod N)}, \; j_2 = 0, \;\; i_2 \notin \{0, M\} \\ 0, & \text{else} \end{cases} \tag{3}$$

$$\pi_{(i_1,i_2,i_3)(j_1,j_2,j_3)}(1) = \begin{cases} 1 - \tilde{p}_{1,i_3} & \text{for } j_1 = i_1 + 1 \text{ (mod N)}, \; j_2 = 1 \\ \tilde{p}_{1,i_3}, & \text{for } j_1 = i_1 + 1 \text{ (mod N)}, \; j_2 = 0, \\ 0, & \text{else} \end{cases} \tag{4}$$

Where $\tilde{p}_{i_2,i_3}$ is defined the product of $P(\mathcal{I}_3 = i_3 | \mathcal{I}_1 = i_1)$ and $P(X = i_2 | X \geqslant i_2)$, where the first term is the probability that the weather is in state i_3 given period i_1, and the second term is the failure probability of the component of age i_2.

3.3 Lifetime distribution

As frequently used in failure analysis, the Weibull distribution is used for modelling the lifetime of the single component. The Weibull distribution has the property that the failure rate is easily specified. Here, a Weibull distribution with scale parameter $\beta > 0$ and shape parameter $\alpha > 0$ is used. As the model is defined in a way that we discretize time, a discrete Weibull distribution has to be used. The CDF is defined as $F(x) = 1 - \exp\left\{-(\frac{x}{\alpha})\right\}^{\beta}$ for $x > 0$, and for every non-negative x, the probability of failure is then calculated as derived in Schouten et al. (2022).

$$p_x = \frac{P(X = x)}{P(X \geqslant x)} = \frac{\exp\left\{-(\frac{x-1}{\alpha})\right\}^{\beta} - \exp\left\{-(\frac{x}{\alpha})\right\}^{\beta}}{\exp\left\{-(\frac{x-1}{\alpha})\right\}^{\beta}} \tag{5}$$

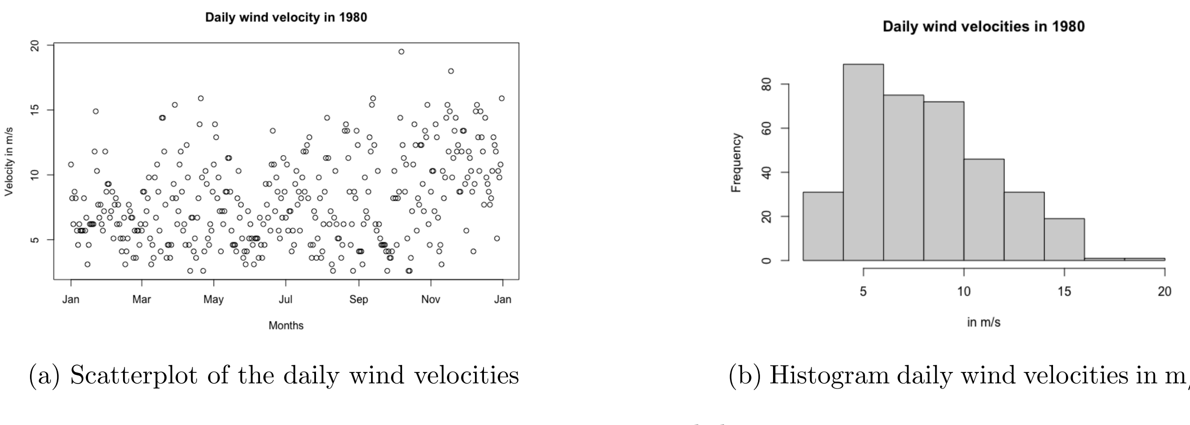
3.4 Costs

A large proportion of the maintenance costs is due to the fact that the turbine is not operating. As the wind turbine is getting serviced, the wind turbine is not able to produce electricity that can be sold. We define the costs as the loss of production in a certain period, in which we take the wind conditions in to account. In periods of high wind, the cost of maintenance is higher than in periods where there is less wind. Therefore, the costs of maintenance depends heavily on the period in which maintenance is conducted. In Schouten et al. (2022), it is stated that it takes around ten days to conduct preventive maintenance, and that corrective maintenance on average is almost four times the duration of preventive maintenance. We take this into account when calculating the costs. One side note, in this thesis we define the costs by only taking the loss of production into account. Other costs related to maintenance are not included, as well as the time to conduct the maintenance.

4 Data

4.1 Wind- and turbine data

In this section we discuss the data that is used for this thesis. As in Schouten et al. (2022), the daily weather data of the coastal city of Ijmuiden are used, provided by the KNMI (2023). The dataset consists of several variables regarding the wind, rainfall and air pressure. In this research, we use the daily wind velocities. The dataset originally consisted of 19117 observations, starting on the 1st of January 1971, and ending on the 4th of May 2023. Due to some missing observations, incompleteness of the year 2023 and computational ease we discard the observations after the 31st of December 2022. So the resulting dataset consists of 18849 observations, which is 52 years of daily data. The values of the daily averages in the original dataset are measured in 0.1 m/s, which are converted to m/s. Next, a brief look in to the data for the year 1980 is provided.



(a) Scatterplot of the daily wind velocities (b) Histogram daily wind velocities in m/s

Figure 1: 1980 wind data

As expected, there is a large variation in the daily wind velocities within a year. Figure 1b shows the distribution of the wind speeds in 1980. The highest frequency occurs at around $5m/s$, while average velocities of $15m/s$ occur around 4 times less. Based on the scatter plot in Figure 1a, the data is not showing any kind of pattern, since the variability of the data is simply too high. Hence to analyze the data for our calculations in Section 6.1, weekly averages over the 52 years are used.

The turbine power output with respect to the velocity is used to make an estimation of the power output curve. We use data from the RVO (2023), which contains the output data from a VESTAS V164-10.0MW wind turbine.

5 Methodology

This section provides a detailed explanation of the methods used to realize the maintenance policies. First of all, the wind data has to be transformed into the estimated power output of a wind turbine. Next we define distributions of the daily power output for each week of the year. Lastly, the transition probabilities are calculated and the distributions are divided into the three states required for the Markov Decision Model.

The first thing to notice is that we have to make adjustments to the wind data. We need to consider that the height at which the wind data is observed, differs from the height of the wind turbine's rotor. Schouten et al. (2022) describes how the wind velocity changes when the altitude increases. The daily wind data is measured at a height of 10m, while the height of a wind turbine rotor is at a height of 138m. Following the logarithmic wind profile formulas in Schouten et al. (2022), the wind velocity at rotor height increases by a factor of 1.181.

5.1 Power function estimation

To be able to compute the weekly power output distributions, we need to define the function that maps the wind velocity to the generated power of the wind turbine. To estimate this function, RVO (2023) provides the data for constructing the power curve of a Vestas V164-10.0MW wind turbine. Using the statistical programming software R, we fit a model to the data. We observe that the Power Curve consists of three separate functions that we can model, consisting of the quadratic part, and two linear parts. For the quadratic part, we fit a quadratic regression model using the *lm* function in R. This results in a continuous function $\hat{P}(v)$ that gives the estimated power output in kW for every wind velocity in the domain of the turbines output. As the function is non-linear, we can not simply take the average wind velocity and then compute the average power output. Hence we have to be careful in averaging the wind velocities, as this will result in a different answer than if we reversed the order of operations.

5.2 Obtaining the Power Distributions for each week

As we now have estimated the Power Curve of the wind turbine, the curve is used to calculate the corresponding daily output levels. The goal is to obtain 52 power distributions, one for each week within the year. We calculate this by collecting all the observations which fall within that week. As there are 7 days in a week, this results in 7 observations in each year, obtaining

7×52 years $= 364$ observations for each week over 52 years. We now have gathered 52 weekly distributions consisting of 364 observations.

Alternatively, one could do this in a different manner. For example by examining the total power output within a week, since the model makes maintenance decisions on a weekly basis. For each week in a year, the total output is calculated by summing over the power output per day within that week. Due to the number of weeks in a year, this results in 52 distributions. Also, each distribution contains 52 observations, as we have 52 years of data.

Both methods can be applied for the calculation of the transition probabilities. In Section 6 we will compare both methods.

5.3 Calculating the transition probabilities

Next, we want to calculate the transition probabilities based on the weekly distributions. The goal is to divide the daily power output into three states, for every $i_3 \in \mathcal{I}_3$. For every week, we look at the power output distribution. We divide that distribution into three states based on the mean of that week. As the distributions differ greatly each week because of the wind variability, it is not fair to apply one single condition that differentiates one state from another. For each week, we calculate the mean of the power distribution. We define the three states as follows: first, let the set $\mathcal{D}_{i_1}$ be the set containing all the observations d_{i_1} in week i_1. Based on the mean of that week, we define $i_3 = 1$ if an element d_{i_1} is contained in the interval of 20% under and 20% above the mean. So $d_{i_1} \in (0.8\bar{P}_{i_1},\ 1.2\bar{P}_{i_1})$. Where $\bar{P}_{i_1}$ is the average power output in week i_1. For $i_3 = 0$ and $i_3 = 2$, the intervals are defined as $[0,\ 0.8\bar{P}_{i_1})$ and $(1.2\bar{P}_{i_1},\ \bar{P}_{max}]$ respectively. The 20%-rule of dividing the distribution in to three classes is arbitrary, and chosen for simplicity. We also examined using the median instead of the mean for this purpose. However, we found that due to the large amount of observations near the cut-off point of the wind turbines power output, the median would be such that almost all of the probability was in state $i_3 = 2$.

$$i_3 = \begin{cases} 0 & \text{if } d_{i_1} \in [0,\ 0.8\bar{P}_{i_1}) \\ 1 & \text{if } d_{i_1} \in (0.8\bar{P}_{i_1},\ 1.2\bar{P}_{i_1}) \\ 2 & \text{if } d_{i_1} \in (1.2\bar{P}_{i_1},\ \bar{P}_{max}] \end{cases} \tag{6}$$

Next, we determine the frequency f_{i_1,i_3} for all observations d_{i_1} for which the power output is in state i_3 for a given week i_1. We calculate the proportions which are equivalent to the transition probabilities. $p_{i_1,i_3} = \frac{f_{i_1,i_3}}{n_{i_1}}$, where n_{i_1} is the total observations in week i_1.

However, it may be possible that the probability of being in state i_3 depends on the week before. If there is a dependence between two consecutive weeks, the transition probabilities are not valid. We use the Pearson Correlation test to examine if two consecutive weeks are significantly correlated with each other. When this is not the case, then we can assume that the calculated probabilities are valid, and use them in the Linear Program.

5.4 Markov Decision Process Mixed Integer Program Approach

The same approach is used as in Schouten et al. (2022), where the long run probabilities are calculated using a Mixed Integer Programming formulation. In this thesis, the same formulation is used, but some minor changes have been made to accommodate for the new approach. The objective and constraints have the same interpretation as in Schouten et al. (2022). The third set of constraints ensures that maintenance in a certain period can be done in only one of the three states.

$$\begin{aligned}
\min \quad & \sum_{i=(i_1,i_2,i_3)\in\mathcal{I}\setminus\mathcal{I}^b} c_p(i_1,i_3)x_{i,1} + \sum_{i=(i_1,i_2,i_3)\in\mathcal{I}^b} c_f(i_1,i_3)x_{i,1} \\
\text{s.t.} \quad & \sum_{a\in\mathcal{A}(i)} x_{i,a} - \sum_{j\in\mathcal{I}}\sum_{a\in\mathcal{A}_j} \pi_{ji}(a)x_{j,a} = 0 \qquad \forall i = (i_1,i_2,i_3)\in\mathcal{I} \\
& \sum_{i_2\in\mathcal{I}_2}\sum_{a\in\mathcal{A}(i_1,i_2)} x_{i_1,i_2,i_3,a} = \frac{1}{N} \quad \forall i_1\in\mathcal{I}_1, i_3\in\mathcal{I}_3 \\
& \sum_{i_3\in\mathcal{I}_3} x_{i_1,i_2,i_3,1} = \min\{\sum_{i_3\in\mathcal{I}_3} x_{i_1,i_2,i_3,1}\} \quad \forall i_1\in\mathcal{I}_1, i_2\in\mathcal{I}_2 \\
& x_{i,a} \geqslant 0, \quad \forall i = (i_1,i_2,i_3)\in\mathcal{I}, a\in A(i) \\
& x_{i0} = 0 \quad \forall i = (i_1,i_2,i_3)\in\mathcal{I} : i_1\in\{0,M\}
\end{aligned} \tag{7}$$

As the third set of constraints is not linear, we will have to transform these constraints. $\forall i_1\in\mathcal{I}_1, i_2\in\mathcal{I}_2$, define $\sum_{i_3\in\mathcal{I}_3} x_{i_1,i_2,i_3,1} = \min\{\sum_{i_3\in\mathcal{I}_3} x_{i_1,i_2,i_3,1}\}$ as the function we want to make linear. We introduce a new variable z to represent the minimal value. We define M as a large positive value, and finally α, β, $\gamma \in \mathbb{B}$

$$\begin{aligned}
& z \leqslant x_{i_1,i_2,0,1} \\
& z \leqslant x_{i_1,i_2,1,1} \\
& z \leqslant x_{i_1,i_2,2,1} \\
& z \geqslant x_{i_1,i_2,0,1} - M(1-\alpha) \\
& z \geqslant x_{i_1,i_2,1,1} - M(1-\beta) \\
& z \geqslant x_{i_1,i_2,2,1} - M(1-\gamma) \\
& \alpha + \beta + \gamma = 1 \\
& z \in \mathbb{N} \\
& \alpha, \beta, \gamma \in \mathbb{B}
\end{aligned} \tag{8}$$

The terms $M(1-\alpha)$, $M(1-\beta)$, and $M(1-\gamma)$ penalize the selection of $x_{i_1,i_2,0,1}$, $x_{i_1,i_2,1,1}$, $x_{i_1,i_2,2,1}$ respectively, when they are not the minimum. The last constraint ensures that only one variable can be equal to 1, while the others are forced to be 0. This represents the selection of the minimum.

5.5 Costs

To calculate the costs, we differ from the model in Schouten et al. (2022), as this method takes the weather variability into account in a more specific manner. We model the costs by examining the loss of production when maintenance is executed. This depends on the period of the year, and the state of the weather. To calculate the costs, we therefore look at the distribution of the estimated power output in that week. The loss of production is defined for every i_1 and every i_3. There are other factors which will increase the total costs such as replacement parts, transportation and costs of the maintenance crew. In this research we will only take the costs of loss of production into account.

For every week i_1 and every wind state i_3, we select the observations that are included in the interval. This results in a distribution of daily power output for week $i_1 \in \mathcal{I}_1$ and state $i_3 \in \mathcal{I}_3$. Next, we take the average power output as the amount of power the turbine will not generate as it is being maintained.

We model the decisions on a weekly basis and taking the average preventive maintenance duration into account. We assume that the preventive maintenance consists of one week, that is seven days. We therefore have to multiply the average output by seven days to get the weekly output. Assuming the wind turbine operates 24 hours a day and a price of €0.06/kWh, the guaranteed price in a 2018 German project, see Ten Brinck (2018), we have to multiply the costs by $24 * 7 * 0.06$ to get to the costs in euros.

$$CostsPM(i_1, i_3) = P_{el} c^{w}_{i_1,i_3} \tag{9}$$

Where P_{el} is the electricity cost, and $c^{w}_{i_1,i_3}$ is the average cost of maintaining calculated as described above. As mentioned in the paper by Schouten et al. (2022), on average the duration of the corrective maintenance is about four times as long. Hence for simplicity, we take for corrective maintenance four times the cost of preventive maintenance.

6 Results

In this section we present the results of the analysis and optimization. We will discuss the estimated power curve, the weekly distributions and the costs. Furthermore we discuss the optimization of the main model, followed by a comparison with the model from Schouten et al. (2022).

6.1 Data Analysis

6.1.1 Power Curve

To estimate the Power curve, we use the data that relates the wind velocities in m/s to the power output in kW.

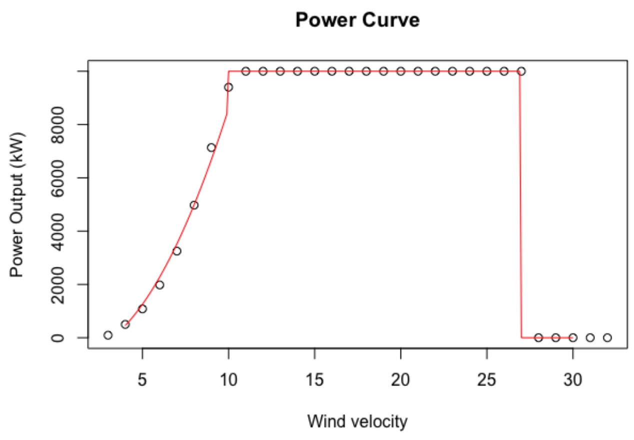


Figure 2: Actual and estimated Power Curve of the VESTAS V164-10.0MW turbine

In Figure 2, the first thing one may notice is a steep increase in marginal output until the maximum output of 10kW is reached at a velocity of $11m/s$. We also observe that the turbine stops operating at velocities greater than 28 m/s. This is due to safety reasons. We estimate the first part of the curve with quadratic regression, using the first nine observations. This results in the following function $\hat{P}(v) = 111.46v^2 - 203.46v - 511.86$ for $v \geqslant 3$ and $v \leqslant 11$, with $R^2 = 0.9859$. The estimated function is shown as the red line. For $v > 11$ the function is constant. The resulting function is then as follows:

$$Power(v) \begin{cases} \hat{P}(v) & \text{if } v \geqslant 3 \text{ and } v \leqslant 11 \\ 10.000 & \text{if } v > 11 \text{ and } v < 28 \\ 0 & \text{if } v \leqslant 3 \text{ and } v \geqslant 28 \end{cases} \tag{10}$$

6.1.2 Weekly distributions and transition probabilities

There are multiple ways to determine the weekly power distributions of which two were described in Section 5.3. In one method, the distributions are based on the daily power output, while in the other method the distributions are derived from the total weekly power output. We first discuss the method where the total weekly output is used. Figure 3 below gives an insight in the pattern of the total wind output in a week. For each week, the total weekly output averaged over 52 years is shown. We observe that during wintertime, the average total weekly output is higher than during summertime.

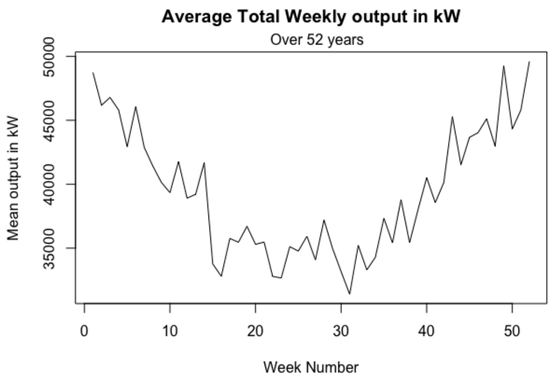
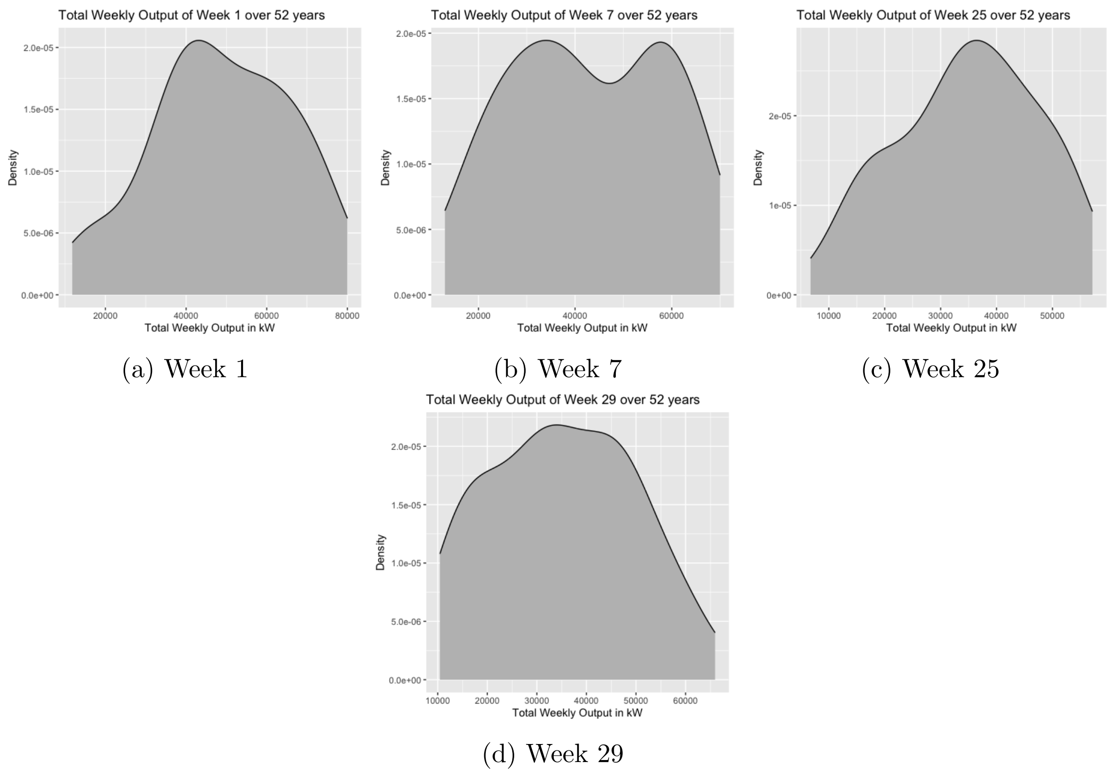


Figure 3: The averaged total weekly output over all weeks.

When examining the distributions of the total weekly output of a certain week, we find that in the weeks where we would expect lots of high total output, that this is not the case. For example in wintertime, as shown in 4a and 4b we observe that the majority is not at the higher end of the distribution. The same happens for the weeks during summertime, where you would expect low total output, as shown in Figure 4c and 4d. The majority of the observations is not on the lower end of the distribution.



(a) Week 1 (b) Week 7 (c) Week 25

(d) Week 29

Figure 4: Distribution of the total weekly output over the years

One explanation for this behavior is that the intra-week variance of the daily output is high. Due to large differences in wind output for each day of the week, it is unlikely that the weekly total output over the years will follow the described pattern. For example, we examine this for week 20 in the year 2000, where the large differences in daily power output results in high variance (See Table 1).

Date	15-05	16-05	17-05	18-05	19-05	20-05	21-05
Daily output in kW	95	1916	10000	10000	10000	3482	819.9
Variance:	21.349.540						

Table 1: Daily output for week 20 in year 2000

The high variance is also noticeable over all the years. We take a look at the year 2000 again, but now over all weeks. In Figure 5 it is can be seen that large variances of the intra-week daily output exist, and may influence the usability of total weekly power output for the distributions.

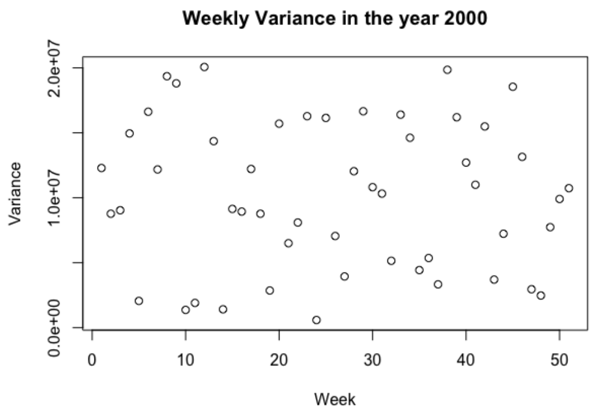


Figure 5: Variances of the daily output in a week in the year 2000.

These distributions resulted in transition probabilities that did not exhibit the pattern of more wind output in wintertime and lower output in summertime. As shown in Figure 6, the probabilities are roughly constant over time. In the light of our model, these transition probabilities may lose their meaning as they fail to capture the behavior of the power output over time.

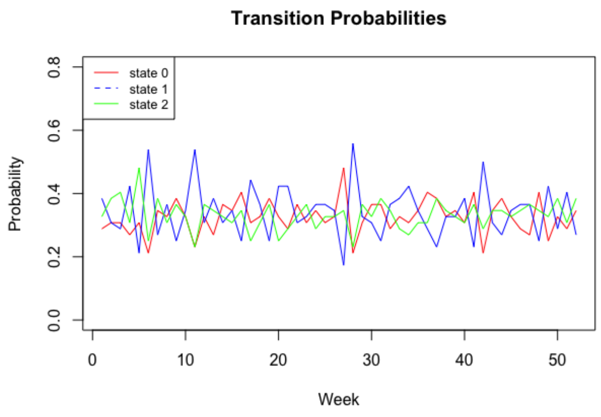
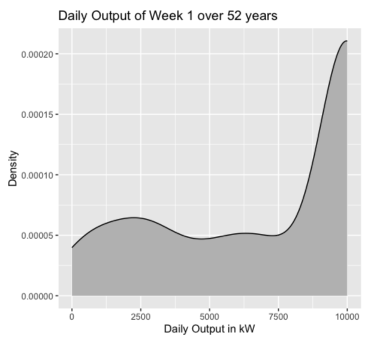


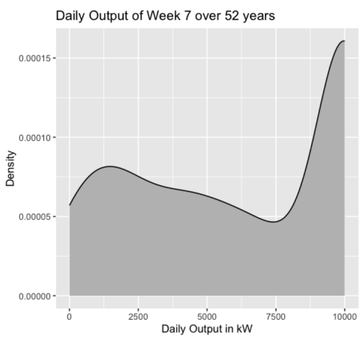
Figure 6: Transition probabilities for every state, using the weekly total output

Hence, we will continue with the weekly distributions where the daily output levels are used.

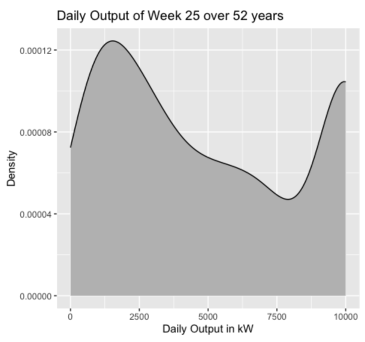
In Figure 7 we show the same weeks as in Figure 4. We see that in Figure 7a and 7b the majority of the density is in the upper end of the distribution, and for Figure 7c and 7d in the lower end of the distribution. This method will result in transition probabilities that will follow the pattern of the power output over time.



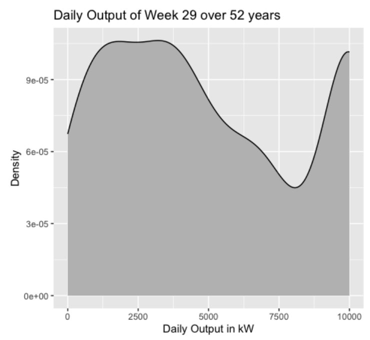
(a) Week 1



(b) Week 7



(c) Week 25



(d) Week 29

Figure 7: Distribution of the total weekly output over the years

In Figure 7a and 7b, the densities represent the 1st and the 7th week respectively. The x-axis is defined as the output in kW. The figure shows that the density is left skewed, as the majority of the observations are on the upper end on the domain of the possible output values. This indicates that during week one and seven the power output is high. In comparison to 7c and 7d, we observe that the majority of the observations are located at the lower end on the domain, therefore the output generated by the turbine is much lower. The remainder of the distributions are found in Appendix 9.1, where the differences between the distributions are visible. We can clearly see differences in distribution per week, hence it could be useful to differentiate the costs on a weekly basis.

Next, we show how the transition probabilities are calculated for a given week. We use the method which classifies three states, as explained in Section 5.3. For example in Figure 8, the power distribution for week 22 is shown. The distribution of week 22 has an average output of 4742.64 kW.

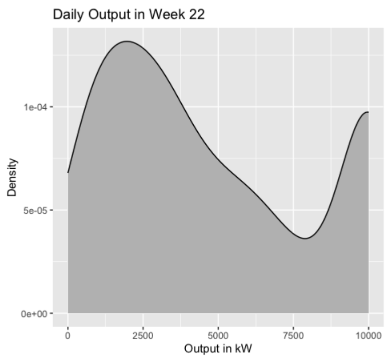
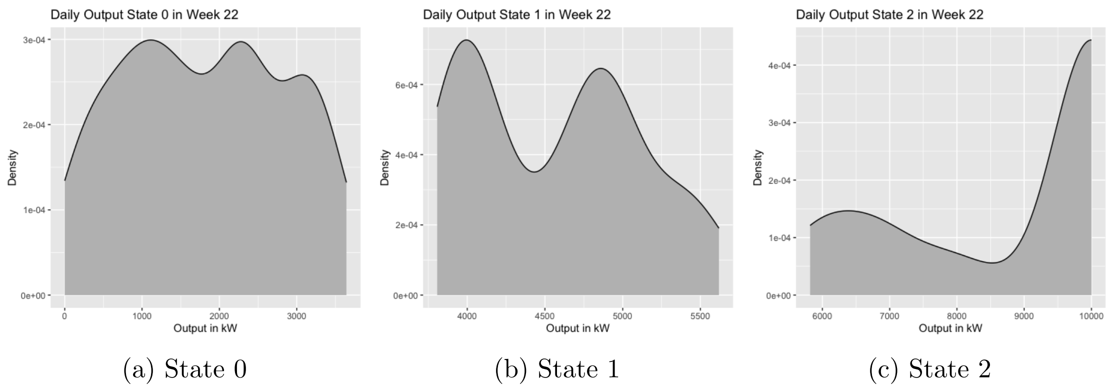


Figure 8: Distributions of the daily wind output for week 22

This results in the following intervals for i_3:

$$i_3 = \begin{cases} 0 & \text{if } d_{i_1} \in [0,\ 3794.11) \\ 1 & \text{if } d_{i_1} \in (3794.11,\ 5691.17) \\ 2 & \text{if } d_{i_1} \in (5691.17,\ 10000] \end{cases} \tag{11}$$

Which then results in the following distributions per state as shown in Figure 9.



(a) State 0 (b) State 1 (c) State 2

Figure 9: Distributions of daily wind output for week 22

Next, the transition probabilities for Week 22 are calculated. There are 364 observations resulting in the distribution as shown in Figure 8. Recall that the probabilities p_{i_1,i_3} are calculated as follows: $p_{i_1,i_3} = \frac{f_{i_1,i_3}}{n_{i_1}}$, where f_{i_1,i_3} is the frequency for which the power output is in state i_3 for a given week i_1, and n_{i_1,i_3} is the total observations in week i_1. So, $p_{22,0} = \frac{180}{364} = 0.49$, $p_{22,1} = \frac{51}{364} = 0.14$ and $p_{22,2} = \frac{133}{364} = 0.37$.

Subsequently, we present the transition probabilities for all weeks, since these are necessary for the Markov transition model. In Figure 10 the transition probabilities are shown. The probability being in state $i_3 = 0$ is the highest in week 13 up to week 40. For $i_3 = 2$, this pattern is as expected as it is during wintertime. State $i_3 = 1$ stays about constant over the weeks. See the Appendix for the actual transition probabilities.

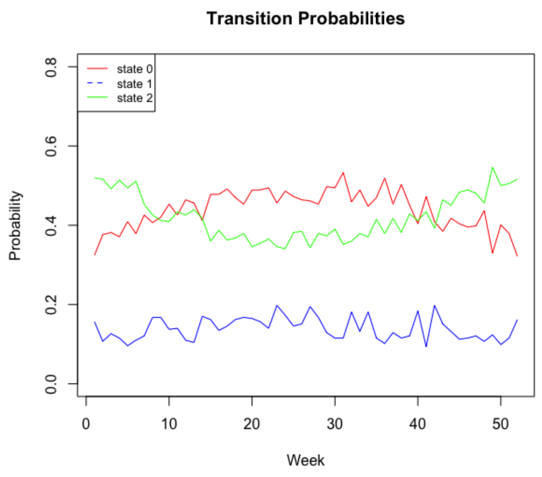


Figure 10: Transition probabilities in state i_3 for every week of the year

Lastly, the Pearson correlation for the independence between two subsequent weeks resulted for 77% of the time in insignificant correlation. As the majority of the weeks are independent, we therefore assume overall independence between any two subsequent weeks.

6.1.3 Costs

The costs per week per state when preventive maintenance is performed are presented in Figure 11 below. It shows that the costs of conducting preventive maintenance in state 2 are the highest for all weeks. This is as expected since the wind power output during wintertime is greater than during summertime.

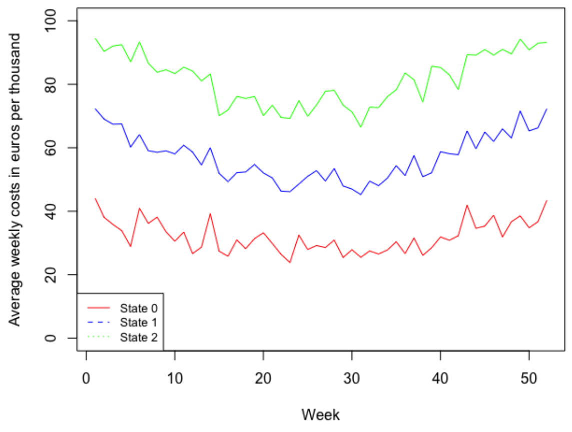


Figure 11: Total weekly costs for all states for every week of the year

6.2 Optimization

In this section we present the results of the optimization. We first summarize the results obtained from the replication. Next, the results from the extended p-ARP model are compared with the original p-ARP model of Schouten et al. (2022).

6.2.1 Optimization results

First, we present the results obtained from the replication. Using the p-ARP model, we let $\bar{c}_f = 50$, $\bar{c}_p = 10$, the scale parameter $\alpha = 52$ (weeks), shape $\beta = 2$ and $M = 52$. We use $\Delta = 0\%$, which refers to the constant cost case. Using these parameters, the resulting costs are €39092. These costs differ from the €40098 obtained in Schouten et al. (2022). The maintenance policy follow Schouten et al. (2022), which is the same across every period, in this case every week. The optimal maintenance is after 27 weeks, or equivalently after 6 months.

In the extended model, instead of using cost-parameters we directly used the weather data for determining the costs and maintenance policy. We used scale parameter $\alpha = 52$ (weeks), shape $\beta = 2$ and $M = 52$. The model described by Equation 7 first resulted in an infeasible solution. The problem arises from the third set of constraints, which ensures that in a period i_1, maintenance is performed in only one state $i_3 \in \mathcal{I}_3$. After removing those constraints, the model resulted in a yearly cost of €561340. The optimal maintenance policy depends on the week of the year. It is optimal to perform preventive maintenance as described in Table 2 below. For example, in week 5, preventive maintenance should be done if the age of the component is at least 28 weeks. Due to the violation of the third set of constraints when maintenance is performed, the long-run probability $x_{i_1,i_2,i_3,1} > 0 \; \forall i_3 \in \mathcal{I}_3$, for i_1 & i_2. This implies that the yearly costs are on average three times as high.

6.2.2 Comparison with the model of Schouten et al. (2022)

As explained in Section 5, instead of using cost-parameters we directly used the weather data for determining the costs and maintenance policy. Hence, the results of the extended model can not be compared directly to the constant cost model in Schouten et al. (2022).

To solve this problem, the same dataset is used in both models, as well as the same parameters for the lifetime distribution. The only difference is the way the data is handled to provide the expected costs based on the average output used for the three state approach.

To be able to compare the yearly costs, we divide the costs of the extended p-ARP model by three, resulting in yearly maintenance costs of €187113. This is due to the violation of the third set of constraints.

The optimization with the model of Schouten et al. (2022) results in the total maintenance costs of €187598. Maintenance policy is as shown in Table 2 below. We observe that the extended p-ARP model performs less maintenance with respect to the standard p-ARP model. Also, the period in which maintenance will be conducted differs between the two models. Next, the cost difference between the two models is negligible. Due to the more specific approach in the cost procedure described in Section 5.5, the small difference in costs can indicate that the model of Schouten et al. (2022) already gives an accurate reflection of the total yearly maintenance costs.

Both models appear to be effective in minimizing the total yearly maintenance costs, but the adjusted p-ARP model takes weather fluctuations more into account than Schouten et al. (2022). As the goal is to make the mathematical models as close to reality as possible, the extended model p-ARP could be preferred. Also, a difference was found in maintenance policy. This might be due to the transition probabilities being defined as the product of the probability of failure and the probability of observing average, low or high wind velocities.

extended p-ARP								
Week	5	15	16	31	38	42		
Age	28	33	24	28	19	20		
p-ARP								
Week	10	15	16	31	33	38	41	48
Age	35	33	24	30	31	18	20	23

Table 2: Optimal critical maintenance ages

7 Discussion

The aim of this research was to implement a more specific model to better incorporate the weather conditions with respect to the p-ARP model described in Schouten et al. (2022). In this paper we introduced a three state Markov transition model, where it was investigated if the thee state model performs better than the original model. Data on a Vestas V164-10.0MW wind turbine was used for the power curve. Next, we used a different method for the analysis of the daily wind data to obtain the induced costs when preventive maintenance in a certain period is conducted. We had to estimate the transition probabilities to incorporate a third state, which takes the weather variability into account.

Each model executed on the same dataset resulted in different maintenance policies. The total costs were lower in the extended model. This can be explained due to the fact that the model has more accurate estimates of the average wind velocities. Hence, the policies resulting from the model may be more realistic.

The resulting maintenance policy consists of critical maintenance ages for certain weeks in a year. We found that it is optimal to conduct maintenance in weeks 5, 15, 16, 31, 38 and 42, with component ages 28, 33, 24, 28, 19 and 20 respectively. The optimal replacement ages can differ from eachother in the weeks of the year. The decision regarding the age at which maintenance is performed on the component depends on the week. The model resulted in yearly total maintenance costs of €187113.

Despite the results from our study, several limitations should be discussed. First of all, the cost specification is general and only concerns the costs due to the loss of production. This could be improved by including the costs of maintenance, namely replacement parts, labour and other related costs. Second, due to the infeasibility of the third set of constraints, we had to provide a sub-optimal solution by dividing the total maintenance costs as described in Section 6.2.2. Lastly, the chosen percentage above and below the mean for classifying states $\mathcal{I}_3$ is arbitrary. Therefore, it is possible that alternative approaches for dividing a distribution are more appropriate.

8 Conclusion

In this paper, the aim was to improve the model for optimal maintenance policies for wind turbines under time-varying costs. We extended the age replacement policy model by a three state Markov decision process which takes the the high variability of the wind velocities into account. We examined if it was possible to improve the maintenance decision making process by modelling the behavior of the weather in a more accurate manner. We found that there are at least two possible ways to handle the wind data, namely by creating wind-power output distributions using daily or total weekly wind output levels. We found that due to the high variance of the wind data, the second method failed to show relevant patterns that are required for the Markov model. Also, we found that the extended p-ARP model performs less maintenance compared to the standard p-ARP model, which resulted in lower costs. However, the cost difference is negligible. Furthermore, the period in which maintenance is conducted also differs between the two models.

Future research could focus on the restriction that results in an infeasible solution to improve the minimization of the total maintenance costs. Additionally, one could look at other implementations of modelling the lifetime of components. For example by including sensory input data to gain more insight or by including a model for the deterioration of the wind turbine components. Also, the model can be incorporated in different fields or industries where decisions regarding maintenance depend on the weather conditions, for example in aviation.

However, the weather is and will be hard to predict, especially for long-term predictions. The high variance of the wind output in a week results in an even harder decision regarding maintenance. Data from the past gives an indication what the best moment could be to conduct maintenance. Nevertheless, it is not guaranteed that history will repeat itself.

References

Aafif, Y., Chelbi, A., Mifdal, L., Dellagi, S. & Majdouline, I. (2022). Optimal preventive maintenance strategies for a wind turbine gearbox. *Energy Reports*, *8*, 803–814.

Dinwoodie, I. & Mcmillan, D. (2014). Operation and maintenance of offshore wind farms. *Eng. Technol. Ref*, *1*(1).

Fischetti, M. & Monaci, M. (2016). Proximity search heuristics for wind farm optimal layout. *Journal of Heuristics*, *22*, 459–474.

Fischetti, M. & Pisinger, D. (2018). Optimizing wind farm cable routing considering power losses. *European Journal of Operational Research*, *270*(3), 917–930.

Gonzalo, A. P., Benmessaoud, T., Entezami, M. & Márquez, F. P. G. (2022). Optimal maintenance management of offshore wind turbines by minimizing the costs. *Sustainable Energy Technologies and Assessments*, *52*, 102230.

KNMI. (2023). *Royal netherlands meteorological institute.* Retrieved 7-05-2023, from `https://www.knmi.nl/nederland-nu/klimatologie/daggegevens`

Negm, H. M. & Maalawi, K. Y. (2000). Structural design optimization of wind turbine towers. *Computers & Structures*, *74*(6), 649–666.

Rijksoverheid. (2020a). *Windenergie op land.* Retrieved 7-05-2023, from `https://www.rijksoverheid.nl/onderwerpen/duurzame-energie/windenergie-op-land`

Rijksoverheid. (2020b). *Windenergie op zee.* Retrieved 7-05-2023, from `https://www.rijksoverheid.nl/onderwerpen/duurzame-energie/windenergie-op-zee`

RVO. (2023). *Rijksdienst voor ondernemend nederland.* Retrieved 21-06-2023, from `https://www.rvo.nl/sites/default/files/2021/02/MER-HKW-VI-Bijlagen20-10-11.pdf`

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.

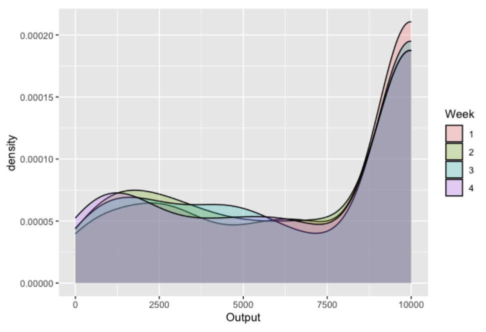
Shafiee, M. (2015). Maintenance logistics organization for offshore wind energy: Current progress and future perspectives. *Renewable energy*, *77*, 182–193.

Ten Brinck, T. (2018). *Opnieuw wind op zee zonder subsidie in duitse tender.* Retrieved 17-07-2023, from `http://www.wattisduurzaam.nl/9119/featured/nieuw-record-offshore-wind-zakt-zakt-naar-6-cen`

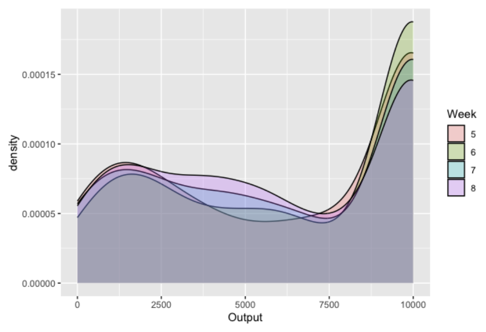
Wang, Y. & Deng, Q. (2022). Optimization of maintenance scheme for offshore wind turbines considering time windows based on hybrid ant colony algorithm. *Ocean Engineering*, *263*, 112357.

9 Appendix

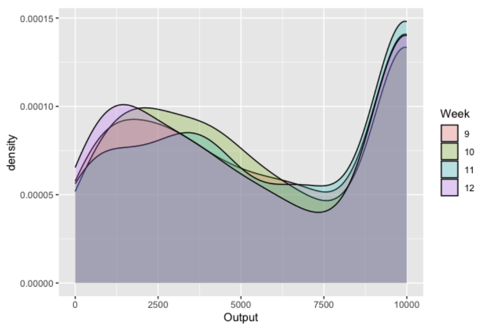
9.1 Result Tables



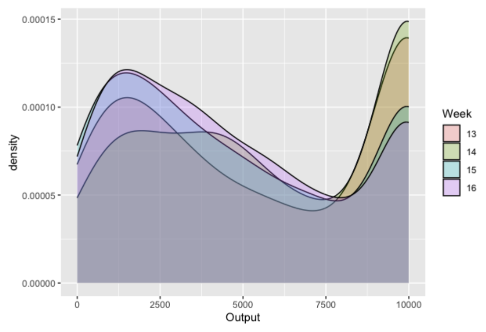
(a) Week 1-4



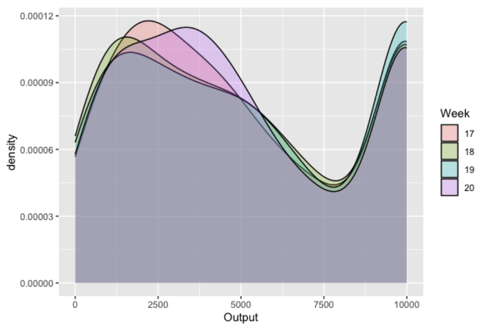
(b) Week 5-8



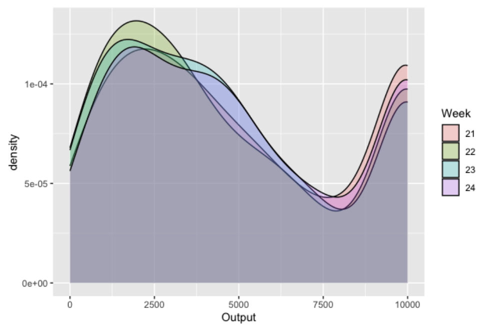
(c) Week 9-12



(d) Week 13-16

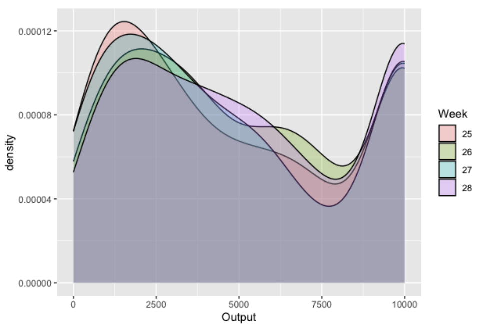


(e) Week 17-20

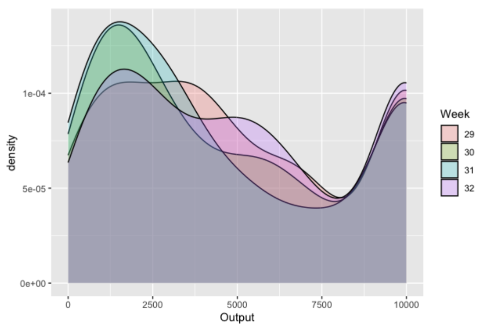


(f) Week 21-24

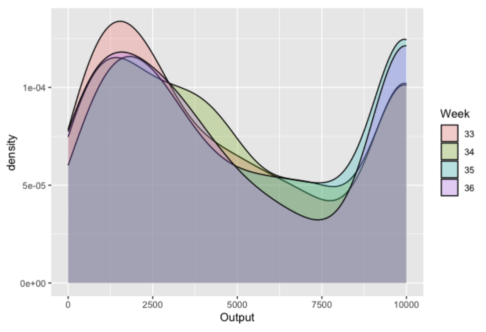
Figure 12: Weekly distributions



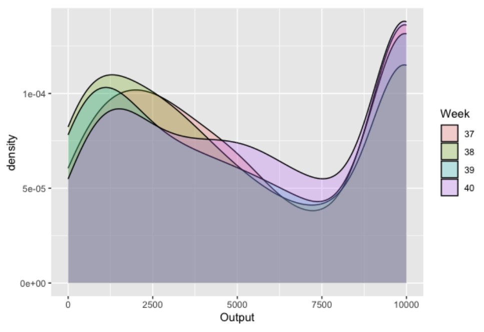
(a) Week 25-28



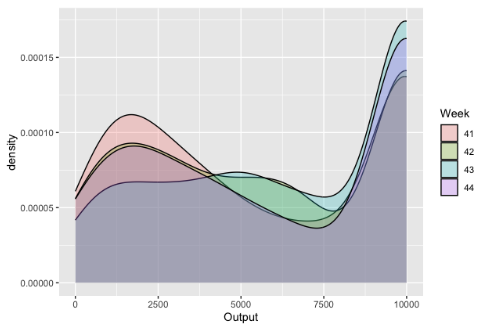
(b) Week 29-32



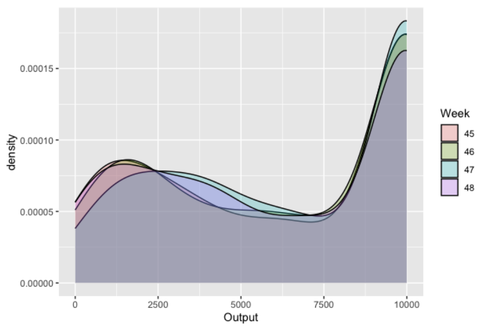
(c) Week 33-36



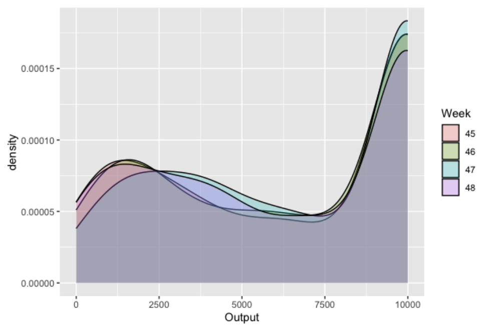
(d) Week 37-40



(e) $y = 3\sin x$



(f) $y = 5/x$



(g) $y = 5/x$

Figure 13: Weekly distributions

Week	p_0	p_1	p_2
1	0.324	0.157	0.519
2	0.376	0.107	0.516
3	0.382	0.126	0.492
4	0.371	0.115	0.514
5	0.409	0.096	0.495
6	0.379	0.11	0.511
7	0.426	0.121	0.453
8	0.407	0.168	0.426
9	0.42	0.168	0.412
10	0.453	0.137	0.409
11	0.426	0.14	0.434
12	0.464	0.11	0.426
13	0.456	0.104	0.44
14	0.412	0.17	0.418
15	0.478	0.162	0.36
16	0.478	0.135	0.387
17	0.492	0.146	0.363
18	0.47	0.162	0.368
19	0.453	0.168	0.379
20	0.489	0.165	0.346
21	0.489	0.157	0.354
22	0.495	0.14	0.365
23	0.456	0.198	0.346
24	0.486	0.173	0.341
25	0.473	0.146	0.382
26	0.464	0.151	0.385
27	0.462	0.195	0.343
28	0.453	0.168	0.379
29	0.497	0.129	0.374
30	0.495	0.115	0.39
31	0.533	0.115	0.352
32	0.459	0.181	0.36
33	0.489	0.132	0.379
34	0.448	0.181	0.371
35	0.47	0.115	0.415
36	0.519	0.102	0.379
37	0.453	0.129	0.418
38	0.503	0.115	0.382
39	0.451	0.121	0.429
40	0.404	0.184	0.412
41	0.473	0.093	0.434
42	0.409	0.198	0.393
43	0.385	0.151	0.464
44	0.418	0.132	0.451
45	0.404	0.113	0.484
46	0.396	0.115	0.489

Week	State 0	State 1	State 2
1	44.09	72.33	94.5
2	38.09	69.04	90.34
3	35.85	67.43	92.03
4	33.89	67.51	92.47
5	28.89	60.18	87.05
6	40.94	64.16	93.36
7	36.15	59.07	86.64
8	38.12	58.58	83.77
9	33.53	59.02	84.57
10	30.53	58.06	83.37
11	33.42	60.79	85.39
12	26.65	58.66	84.17
13	28.65	54.56	81.03
14	39.21	60.02	83.23
15	27.41	51.93	70.08
16	25.81	49.34	71.9
17	30.96	52.13	76.18
18	28.2	52.38	75.53
19	31.38	54.71	76.16
20	33.21	52.07	70.11
21	29.85	50.5	73.39
22	26.4	46.33	69.55
23	23.84	46.16	69.23
24	32.47	48.54	74.85
25	27.94	50.98	69.9
26	29.19	52.81	73.49
27	28.57	49.51	77.77
28	30.91	53.46	78.11
29	25.41	47.96	73.39
30	27.85	46.98	71.31
31	25.5	45.24	66.46
32	27.48	49.49	72.8
33	26.52	48.05	72.62
34	27.77	50.4	76.12
35	30.39	54.37	78.27
36	26.64	51.23	83.57
37	31.56	57.53	81.39
38	26.1	50.86	74.39
39	28.48	52.13	85.69
40	31.9	58.75	85.28
41	30.84	58.12	82.95
42	32.27	57.81	78.32
43	41.89	65.3	89.34
44	34.6	59.66	89.18
45	35.33	64.95	90.93
46	38.74	61.98	89.2

Week	p-value	Week	p-value
1	0.022	27	0.846
2	0.024	28	0.040
3	0.451	29	0.283
4	0.003	30	0.969
5	0.388	31	0.000
6	0.588	32	0.220
7	0.056	33	0.283
8	0.010	34	0.201
9	0.134	35	0.757
10	0.060	36	0.580
11	0.059	37	0.232
12	0.282	38	0.101
13	0.526	39	0.715
14	0.036	40	0.367
15	0.818	41	0.955
16	0.628	42	0.716
17	0.839	43	0.013
18	0.239	44	0.126
19	0.878	45	0.785
20	0.041	46	0.002
21	0.721	47	0.217
22	0.706	48	0.000
23	0.106	49	0.011
24	0.644	50	0.874
25	0.572	51	0.267
26	0.314		

Table 4: P-values from the Pearson Correlation test.

9.2 Programming code

9.2.1 Replication code

```
    import com.google.common.collect.Sets;
import ilog.concert.*;
import ilog.cplex.IloCplex;
import org.apache.commons.math3.distribution.WeibullDistribution;

import java.io.FileNotFoundException;
import java.util.*;

public class pARP {

    public static double delta = 0.0;
   public static double cp_hat = 10;
   public static double cf_hat = 50;
   public static int N; // periods
   public static int M; // max age


    public static void solvepARP(WeibullDistribution weib, int periods,
       int maxAge) throws IloException, FileNotFoundException {
        IloCplex cplex = new IloCplex();
        // VARIABLE DEFINITIONS

       Double[][] costmatrix = ReaderClassARP.getCostMatrix("/Users/
          bramvader/Documents/Econometrie/B3/Scriptie/src/
          weekycoststate_mainmodel.csv");
        IloNumVar[][][] x_matrix = new IloNumVar[periods+1][maxAge
           +1][2];
        for (int i = 1; i <= periods; i++){
            for (int j = 0; j <= maxAge; j++) {
                for (int k = 0; k <= 1; k++) {
                    x_matrix[i][j][k] = cplex.numVar(0, Double.
                        POSITIVE_INFINITY, "x_matrix(" + i + "," + j + ",
                         " + k + ")");
                }
            }
        }

        Double[][][][][] pi_matrix = new Double[periods+1][maxAge+1][
           periods+1][maxAge+1][2];
        double p_1 = Distribution.hazfunc(1, weib);
        double oneMinusP_1 = (1.0 - p_1);

        // a = 1 case
        for (int i = 1; i <= periods; i++) {
            for (int j = 0; j<= maxAge; j++) {
                for (int k = 1; k <= periods; k++) {
                    for (int l = 0; l <= maxAge; l++) {
                         if ((k == i+1 && l == 0) || (k == (i+1) %
                            periods && l == 0))
                         {
                             pi_matrix[i][j][k][l][1] = p_1;
                         }
                         else if ((k == i+1 && l == 1) || (k == (i+1) %
                            periods && l == 1))
                         {
                             pi_matrix[i][j][k][l][1] = oneMinusP_1;
                         }
                         else pi_matrix[i][j][k][l][1] = 0.0;
                    }
```

```
                }
            }
        }
        // a = 0 case
        for (int i = 1; i <= periods; i++) {
            for (int j = 0; j <= maxAge; j++) {
                for (int k = 1; k <= periods; k++) {
                    for (int l = 0; l <= maxAge; l++) {
                        if (j != 0 && j!= maxAge) {
                            double p_j = Distribution.hazfunc(j, weib);
                            double oneminusp_j = (1.0-p_j);
                            if ((k == i+1 && l == 0) || (k == (i+1) %
                                periods && l == 0)) {
                                pi_matrix[i][j][k][l][0] = p_j;
                            }
                            else if ((k == i+1 && l == j+1) || (k == (i
                                +1) % periods && l == j+1)) {
                                pi_matrix[i][j][k][l][0] = oneminusp_j;
                            }
                        }
                        else pi_matrix[i][j][k][l][0] = 0.0;
                    }
                }
            }
        }

        for (int i = 1; i <= periods; i++) {
            for (int j = 0; j <= maxAge; j++) {
                for (int k = 1; k <= periods; k++) {
                    for (int l = 0; l <= maxAge; l++) {
                        for (int m = 0; m <= 1; m++){
                            if (pi_matrix[i][j][k][l][m] == null) {
                                pi_matrix[i][j][k][l][m] = 0.0;
                            }
                        }
                    }
                }
            }
        }

        // SET DEFINITIONS
        Set<Integer> I_1 = new HashSet<>();
        for (int i = 1; i <= periods; i++) {
            I_1.add(i);
        }

        Set<Integer> I_2 = new HashSet<>();
        for (int i = 0; i <= maxAge; i++) {
            I_2.add(i);
        }

        Set<List<Integer>> I_12 = Sets.cartesianProduct(I_1, I_2);
        Set<Integer> zero = new HashSet<>();
        zero.add(0);
        Set<List<Integer>> I_b = Sets.cartesianProduct(I_1, zero);
        Set<List<Integer>> I_noZero = Sets.difference(I_12, I_b);


        IloNumExpr costsPM = cplex.constant(0);
        IloNumExpr costsCM = cplex.constant(0);

        for (List<Integer> element : I_noZero){
            costsPM = cplex.sum
                    (costsPM, cplex.prod(cplex.constant(costmatrix[
                       element.get(0)][1]), x_matrix[element.get(0)][
                       element.get(1)][1]));
                    }

        for (List<Integer> element : I_b){
            costsCM = cplex.sum
                    (costsCM, cplex.prod(cplex.constant(4*costmatrix[
                       element.get(0)][1]), x_matrix[element.get(0)][
                       element.get(1)][1]));
                 }

        IloNumExpr objective = cplex.sum(costsPM, costsCM);
        cplex.addMinimize(objective);

        // Restrictions:
        for (int i = 1; i <= periods; i++) {
            IloNumExpr sumX = cplex.constant(0);
            for (int j = 0; j <= maxAge; j++) {
                for (int k = 0; k <=1; k++) {
                    sumX = cplex.sum(sumX, x_matrix[i][j][k]);
                }
            }
            cplex.addEq(sumX,cplex.constant(1.0/periods));
        }

        for (int i = 1; i <= periods; i++) {
            for (int j = 0; j <= maxAge; j++) {
                if (j == 0 || j == maxAge) {
                    cplex.addEq(x_matrix[i][j][0], 0);
                }
            }
        }

        for (int i = 1; i <= periods; i++)
        {
            for (int j = 0; j <= maxAge; j++) {
                IloNumExpr sumX = cplex.constant(0);
                IloNumExpr sumPiX = cplex.constant(0);

                // X variable
                if (action(i, j).size() != 1) {
                    sumX = cplex.sum(sumX, x_matrix[i][j][0]);
                }
                sumX = cplex.sum(sumX, x_matrix[i][j][1]);

                // X-PI variable: voor elke mogelijke staat (j1,j2)
                   bepaal de x variablen wrt acties,
                for (List<Integer> element : I_12) {
                    if (action(element.get(0), element.get(1)).size() !=
                         1) {
                        sumPiX = cplex.sum(sumPiX, cplex.prod(pi_matrix[
                            element.get(0)][element.get(1)][i][j][0],
                                x_matrix[element.get(0)][element.get(1)
                                    ][0]));
                    }
                    sumPiX = cplex.sum(sumPiX, cplex.prod(pi_matrix[
                        element.get(0)][element.get(1)][i][j][1],
                            x_matrix[element.get(0)][element.get(1)][1])
                                );
                }
                IloNumExpr lhs = cplex.sum(sumX, cplex.prod(-1, sumPiX))
                    ;
                cplex.addEq(lhs, 0);
            }
        }


        cplex.solve();

        Set<List<Integer>> I_LP = new HashSet<>();

        for (int i = 1; i <= periods; i++) {
            for (int j = 0; j <= maxAge; j++) {
                if (action(i, j).size() != 1) {
                    if (cplex.getValue(x_matrix[i][j][0]) > 0)
                    {
                        ArrayList<Integer> indices = new ArrayList<>();
                        indices.add(i);
                        indices.add(j);
                        I_LP.add(new ArrayList<>(indices));
                    }
                }
                if (cplex.getValue(x_matrix[i][j][1]) > 0)
                {
                    ArrayList<Integer> indices = new ArrayList<>();
                    indices.add(i);
                    indices.add(j);
                    I_LP.add(new ArrayList<>(indices));
                }
            }
        }

        Set<List<Integer>> I_leftover = Sets.difference(I_12, I_LP);
        Set<List<Integer>> I_LPtemp = new HashSet<>();

        for (List<Integer> leftover : I_leftover)
            {
                for (List<Integer> lp : I_LP){
                    if (action(leftover.get(0), leftover.get(1)).size()
                       != 1)
                    {
                        if (pi_matrix[leftover.get(0)][leftover.get(1)][
                           lp.get(0)][lp.get(1)][0] > 0)
                        {
                            ArrayList<Integer> temp = new ArrayList<>();
                            temp.add(leftover.get(0));
                            temp.add(leftover.get(1));
                            temp.add(0);
                            I_LPtemp.add(temp);


                        }
                    }
                    if (pi_matrix[leftover.get(0)][leftover.get(1)][lp.
                       get(0)][lp.get(1)][1] > 0)
                    {
                    //    if (leftover.get(0) ==
                     //   {
                        ArrayList<Integer> temp = new ArrayList<>();
                        temp.add(leftover.get(0));
                        temp.add(leftover.get(1));
                        temp.add(1);
                        I_LPtemp.add(temp);
                     //   }
                    }
                }
            }
        I_LP.addAll(I_LPtemp);
        cplex.exportModel("ARP.lp");

        if (cplex.getStatus() == IloCplex.Status.Optimal) {
            System.out.println("Found optimal Solution!");
            System.out.println("Long-run total average cost = " + N*
                cplex.getObjValue());
            System.out.println("Long-run probabilities: ");

           for (int i = 1; i <= periods; i++) {
               for (int j = 0; j <= maxAge; j++) {
                   for (int k = 0; k <=1; k++){
                       if (k == 1)
                       System.out.println("x(" + i + "," + j + "," + k +
                           ") = "  + cplex.getValue(x_matrix[i][j][k]));
                   }
               }
           }
        }

        else
        {
            System.out.println("No optimal solution found");
        }
        cplex.close();
    }

    public static double costsPM(int period)
    {
        double result =  Math.cos(((2.0*Math.PI)*(period-1))/N);
        return cp_hat*(1+(delta*result));
    }

    public static double costsCM(int period)
    {
        double result = Math.cos(((2.0*Math.PI)*(period-1))/N);
        return cf_hat*(1+(delta*result));
    }

    public static ArrayList<Integer> action(int period, int age) {
        ArrayList<Integer> ints = new ArrayList<>();
        int a = 0;
        int b = 1;

        if (age != 0 && age != M) {
            ints.add(a);
            ints.add(b);
        } else {
            ints.add(a);
```

```java
        }
        return ints;
    }

    public static int numberOfActions(int period, int age) {
        int number;
        if (action(period, age).size() == 1) {
            number = 1;
        } else {
            number = 2;
        }
        return number;
    }

    public static void main(String[] args) {

            N = 52;
            M = 52;
            int shape = 2; // beta 2
            int scale = N; // alpha 52
            WeibullDistribution weib = new WeibullDistribution(shape,
                scale);

            try {
                solvepARP(weib, N, M);
            } catch (IloException e) {
                System.out.println("A Cplex exception occured: " + e.
                    getMessage());
                e.printStackTrace();
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            }

    }

}
```

9.2.2 Extended p-ARP model

```java
import com.google.common.collect.Sets;
import ilog.concert.*;
import ilog.cplex.IloCplex;
import org.apache.commons.math3.distribution.WeibullDistribution;

import java.io.FileNotFoundException;
import java.util.*;

public class pARP_ext {
    public static int N; // periods
    public static int M; // max age
    public static Double [][] costmatrix;

    public static void solvepARP_ext(WeibullDistribution weib, int periods, int maxAge) throws IloException, FileNotFoundException {
        IloCplex cplex = new IloCplex();
        // VARIABLE DEFINITIONS
        IloNumVar[][][][] x_matrix = new IloNumVar[periods+1][maxAge+1][3][2];
        for (int i = 1; i <= periods; i++){
            for (int j = 0; j <= maxAge; j++) {
                // w is index voor de weer-state
                for (int w = 0; w <= 2; w++){
                    for (int k = 0; k <= 1; k++) {
                        x_matrix[i][j][w][k] = cplex.numVar(0, Double.POSITIVE_INFINITY, "x_matrix(" + i + ", " + j + ", "+ w + ", " + k + ")");
                    }
                }
            }
        }

        // (i,j,w)  -> (k,l,w^) onder actie {1}
        Double[][][][][][][] pi_matrix = new Double[periods+1][maxAge+1][3][periods+1][maxAge+1][3][2];
        Double[][] w_matrix = CSVReader.calcProbMatrix("/Users/bramvader/Documents/Econometrie/B3/Scriptie/src/probabilities.csv");
        Double[][] costmatrix = CSVReader.getCostMatrix("/Users/bramvader/Documents/Econometrie/B3/Scriptie/src/weekycoststates.csv");
        double p_1 = Distribution.hazfunc(1, weib);

        double oneMinusP_1 = (1.0 - p_1);

        // a = 1 case
        for (int i = 1; i <= periods; i++) {
            for (int j = 0; j<= maxAge; j++) {
                for (int w1 = 0; w1<=2; w1++){
                for (int k = 1; k <= periods; k++) {
                    for (int l = 0; l <= maxAge; l++) {
                        for (int w2 = 0; w2 <= 2; w2++)
                        {
                            if ((k == i+1 && l == 0) || (k == (i+1) % periods && l == 0))
                            {

                                pi_matrix[i][j][w1][k][l][w2][1] =  p_1
```

```
                                    * w_matrix[i][w2];
                            }
                            else if ((k == i+1 && l == 1) || (k == (i+1)
                                % periods && l == 1))
                            {

                                pi_matrix[i][j][w1][k][l][w2][1] =
                                    oneMinusP_1 * w_matrix[i][w2];
                            }

                            else pi_matrix[i][j][w1][k][l][w2][1] = 0.0;
                        }
                    }
                }
            }
        }
        }
        // a = 0 case
        for (int i = 1; i <= periods; i++) {
            for (int j = 0; j <= maxAge; j++) {
                for (int w1 = 0; w1 <= 2; w1++) {
                    for (int k = 1; k <= periods; k++) {
                        for (int l = 0; l <= maxAge; l++) {
                            for (int w2 = 0; w2 <= 2; w2++) {
                                if (j != 0 && j != maxAge) {
                                    double p_j = Distribution.hazfunc(j,
                                        weib);
                                    double oneminusp_j = (1.0 - p_j);
                                    if ((k == i + 1 && l == 0) || (k ==
                                        (i + 1) % periods && l == 0)) {
                                        pi_matrix[i][j][w1][k][l][w2][0]
                                            = p_j * w_matrix[i][w2];
                                    } else if ((k == i + 1 && l == j +
                                        1) || (k == (i + 1) % periods &&
                                        l == j + 1)) {
                                        pi_matrix[i][j][w1][k][l][w2][0]
                                            = oneminusp_j * w_matrix[i][
                                            w2];
                                    }
                                } else pi_matrix[i][j][w1][k][l][w2][0]
                                    = 0.0;
                            }
                        }
                    }
                }
            }
        }
        for (int i = 1; i <= periods; i++) {
            for (int j = 0; j <= maxAge; j++) {
                for (int w1 = 0; w1 <= 2; w1++) {
                    for (int k = 1; k <= periods; k++) {
                        for (int l = 0; l <= maxAge; l++) {
                            for (int w2 = 0; w2 <= 2; w2++) {
                                for (int m = 0; m <= 1; m++) {
                                    if (pi_matrix[i][j][w1][k][l][w2][m]
                                        == null) {
                                        pi_matrix[i][j][w1][k][l][w2][m]
                                            = 0.0;
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }

        // SET DEFINITIONS
        Set<Integer> I_1 = new HashSet<>();
        for (int i = 1; i <= periods; i++) {
            I_1.add(i);
        }

        Set<Integer> I_2 = new HashSet<>();
        for (int i = 0; i <= maxAge; i++) {
            I_2.add(i);
        }

        Set<Integer> I_3 = new HashSet<>();
        for (int i = 0; i <= 2; i++)
        {
            I_3.add(i);
        }

        Set<List<Integer>> I_123 = Sets.cartesianProduct(I_1, I_2, I_3);
        Set<Integer> zero = new HashSet<>();
        zero.add(0);
        Set<List<Integer>> I_b = Sets.cartesianProduct(I_1, zero, I_3);
        Set<List<Integer>> I_noZero = Sets.difference(I_123, I_b);


        IloNumExpr costsPM = cplex.constant(0);
        IloNumExpr costsCM = cplex.constant(0);

        for (List<Integer> element : I_noZero){
            for (int i = 0; i <= 2; i++){
                costsPM = cplex.sum
                    (costsPM, cplex.prod(cplex.constant(costsPM(
                        costmatrix, element.get(0), element.get(2))),
                        x_matrix[element.get(0)][element.get(1)][i][1]));
            }
        }

        for (List<Integer> element : I_b) {
            for (int i = 0; i <= 2; i++) {
                costsCM = cplex.sum
                        (costsCM, cplex.prod(cplex.constant(costsCM(
                            costmatrix, element.get(0), element.get(2))),
                             x_matrix[element.get(0)][element.get(1)][i
                            ][1]));
            }
        }
        IloNumExpr objective = cplex.sum(costsPM, costsCM);
        cplex.addMinimize(objective);

        // Restrictions:
        for (int i = 1; i <= periods; i++) {
            IloNumExpr sumX = cplex.constant(0);
            for (int j = 0; j <= maxAge; j++) {
                for (int k = 0; k <=1; k++) {
                    for (int w = 0; w <= 2; w++)
                  sumX = cplex.sum(sumX, x_matrix[i][j][w][k]);
                }
            }
            cplex.addEq(sumX,cplex.constant(1.0/periods));
        }

        for (int i = 1; i <= periods; i++) {
            for (int j = 0; j <= maxAge; j++) {
                for (int w = 0; w<=2; w++){
                    if (j == 0 || j == maxAge) {
                  cplex.addEq(x_matrix[i][j][w][0], 0);
                    }
                }
            }
        }


        for (int i = 1; i <= periods; i++) {
            for (int j = 0; j <= maxAge; j++) {
                IloNumExpr xvars = cplex.sum(x_matrix[i][j][0][1],
                    x_matrix[i][j][1][1], x_matrix[i][j][2][1]);
                IloNumExpr minimum = cplex.min(x_matrix[i][j][0][1],
                    x_matrix[i][j][1][1]);
                IloNumExpr minimum2 = cplex.min(minimum,x_matrix[i][j][2][1]);
                cplex.addEq(xvars, minimum2);
            }
        }

        for (int i = 1; i <= periods; i++)
        {
            for (int j = 0; j <= maxAge; j++) {
                for (int w1 = 0; w1 <= 2; w1++) {
                        IloNumExpr sumX = cplex.constant(0);
                        IloNumExpr sumPiX = cplex.constant(0);
                        // X variable
                        if (action(i, j).size() != 1) {
                            sumX = cplex.sum(sumX, x_matrix[i][j][w1][0]);
                        }
                        sumX = cplex.sum(sumX, x_matrix[i][j][w1][1]);

                        // X-PI variable: voor elke mogelijke staat (j1, j2) bepaal de x variablen wrt acties,
                        for (List<Integer> element : I_123) {

                            if (action(element.get(0), element.get(1)).size() != 1) {
                                sumPiX = cplex.sum(sumPiX,
                                        cplex.prod(pi_matrix[element.get(0)][element.get(1)][element.get(2)][i][j][w1][0], x_matrix[element.get(0)][element.get(1)][element.get(2)][0]));
                            }
                            sumPiX = cplex.sum(sumPiX,
                                    cplex.prod(pi_matrix[element.get(0)][element.get(1)][element.get(2)][i][j][w1][1], x_matrix[element.get(0)][element.get(1)][element.get(2)][1]));
                        }
                        IloNumExpr lhs = cplex.sum(sumX, cplex.prod(-1, sumPiX));
                        cplex.addEq(lhs, 0);

                }
            }
        }

```

```java

        cplex.solve();

        cplex.exportModel("ARP_ext.lp");

        if (cplex.getStatus() == IloCplex.Status.Optimal) {
            System.out.println("Found optimal Solution!");
            System.out.println("Long-run average cost = " + cplex.getObjValue());
            System.out.println("Long-run probabilities: ");

            for (int i = 1; i <= periods; i++) {
                for (int j = 0; j <= maxAge; j++) {
                    for (int w1 = 0; w1 <= 2; w1++){
                        for (int k = 0; k <=1; k++){
                            if (k == 1)
                            System.out.println("x(" + i + "," + j + ","+ w1 + ","+ k + ") = "  + cplex.getValue(x_matrix[i][j][w1][k]));
                        }
                    }
                }
            }

        } else
        {
            System.out.println("No optimal solution found");
        }
        cplex.close();
    }


    public static double costsPM(Double [][] costs, int period, int state)
    {

        double dailycost = costs[period][state];
        // calculate costs per week for PM:
        // 7 days * 24h * kWh price
        double result = dailycost*7*24*0.06;
        return result;
    }

    public static double costsCM(Double [][] costs, int period, int state)
    {
        double dailycost = costs[period][state];
        // calculate costs per week for CM:
```

```java
        // 7 days * 24h * kWh price
        // Account for longer duration: 4 weeks

        double result = dailycost*7*24*0.06*4;
        return result;
    }

    public static ArrayList<Integer> action(int period, int age) {
        ArrayList<Integer> ints = new ArrayList<>();
        int a = 0;
        int b = 1;

        if (age != 0 && age != M) {
            ints.add(a);
            ints.add(b);
        } else {
            ints.add(a);
        }
        return ints;
    }


    public static void main(String[] args)
    {
        N = 12;
        M = 12;
        double shape = 2.0;
        double scale = N;  // 1
        WeibullDistribution weib = new WeibullDistribution(shape,scale);



        try {
            solvepARP_ext(weib, N,M);
        } catch (IloException e)
        {
            System.out.println("A Cplex exception occured: " + e.
               getMessage()); e.printStackTrace();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }


}
```

9.2.3 Helper classes

```java

import org.apache.commons.math3.distribution.WeibullDistribution;

public class Distribution {

    public static void main(String[] args) {
        int shape = 3; // beta 2
        int scale = 80; // alpha 52

        System.out.println("Weibull distribution with scale " + scale +
            " And shape " + shape);
        WeibullDistribution weib = new WeibullDistribution(shape, scale)
            ;
        System.out.println(weib.getNumericalMean());



    }
    // Failure probability at age x
    public static double hazfunc(int x, WeibullDistribution weib){

            // geeft objective lager maar juiste policy
        double exp = Math.exp(-Math.pow(((x-1)/weib.getScale()),weib.
            getShape()));
        double exp2 = Math.exp(-Math.pow(((x)/weib.getScale()),weib.
            getShape()));

        // geeft output dichter bij de objective maar policy schuift op
       // double exp = Math.exp(-Math.pow(((x)/weib.getScale()),weib.
           getShape()));
       // double exp2 = Math.exp(-Math.pow(((x+1)/weib.getScale()),weib.
           getShape()));

        return (exp- exp2)/exp;
    }


}

import java.io.*;
import java.util.Scanner;

public class CSVReader {

    public static void main(String[] args) throws Exception
        {
            calcProbMatrix("/Users/bramvader/Desktop/probabilities.csv")
                ;
        }




    public static Double[][] calcProbMatrix(String pathname) throws FileNotFoundException {
        Double[][] transProbs = new Double[53][3];

        Scanner sc = new Scanner(new File(pathname));
        //parsing a CSV file into the constructor of Scanner class
        sc.useDelimiter(",");
        //setting comma as delimiter pattern
        for (int i = 1; i<= 52; i++)
        {
            for (int j = 0; j <= 2;j++)
                transProbs[i][j] = Double.valueOf(sc.next());
        }
        sc.close();
        return transProbs;
    }

    public static Double[][] getCostMatrix(String pathname) throws FileNotFoundException {
        Double[][] costmatrix = new Double[53][3];

        Scanner sc = new Scanner(new File(pathname));
        //parsing a CSV file into the constructor of Scanner class
        sc.useDelimiter(",");
        //setting comma as delimiter pattern
        for (int i = 1; i<= 52; i++)
        {
            for (int j = 0; j <= 2;j++)
                costmatrix[i][j] = Double.valueOf(sc.next());
        }
        sc.close();
        return costmatrix;
    }
}
```