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

Multi-component analysis of wind turbine maintenance with time-varying costs

Ruben van Gelder (572708)

Abstract

In this research we examine the optimization of wind turbine maintenance with multiple components under time-varying costs. Based on models by Schouten et al. (2022) we find that including time variant maintenance costs can lead to cost reductions of up to 28%. We formulate two exact models and two heuristics to look at the optimization of scheduling multiple components at the same time. These models result in more cost-efficient schedules that plan preventive maintenance for different components in the same period. We also include the option to delay corrective maintenance to potentially save even more on setup costs. We find that this inclusion leads to significant cost reductions of up to 18%.

Supervisor:	Rommert Dekker
Date final version:	2nd July 2023

Contents

1 Problem statement

Renewable energy has become a hot topic in politics over the past decade. The climate crisis has the potential to have long lasting effects on our planet and future generations. Recently the first Dutch minister of Climate and Energy announced a climate package of around 28 billion euros to go towards the reduction of carbon emissions and an increase in the production of green energy. One of the main pillars of renewable energy are wind turbines, which generate energy via the speed of the wind. Hartman (2022) says that wind turbines have steadily been increasing in size and power capacity over the last 20 to 30 years. Turbines that are currently being constructed are around 66% larger in height than in 1998-1999 and have had a power capacity increase of more than 300% since then.

Due to the increase in utility of wind turbines, the efficiency of generating wind energy has become an increasingly popular research topic. Generating energy in the most cost-effective way is of critical importance to minimize the (carbon) costs of this green energy source. Because the general public is not pleased with a wind turbine in their backyard, wind farms are often placed offshore. Röckmann et al. (2017) argues that these wind farms are better at generating electricity due to the higher wind speeds. They also do not interfere with their natural surroundings like onshore wind farms usually do. However, offshore wind farms have higher technical risks and are more difficult to maintain. Röckmann et al. (2017) argues that around 25% to 30% of the total life cycle costs for offshore wind turbines hides in the maintenance costs. This means that optimization of the maintenance policies can result in more efficient green energy, which indirectly lowers the energy bill for all households that use wind energy.

Röckmann et al. (2017) also describe the difficulties in offshore wind turbine maintenance. They say that during the winter wind speed is higher than during the summer and waves are bigger. This means that during the winter time there are far less opportunities to perform maintenance, which results in a lot of scheduling issues. When maintenance is performed, the wind turbine has to be shut down and thus it cannot generate any energy in this time frame. This means that the higher wind speed also leads to a higher opportunity cost in the winter months. So it seems beneficial to perform preventive maintenance during the summer time and to account for time-varying maintenance costs when modelling its scheduling problem. These costs are split up in transportation costs, the costs to maintain a component and the costs of shutting down the wind turbine during the maintenance operation. Transportation costs and component maintenance costs are roughly the same during the year, because no mayor differences exist for them between the seasons. However, the opportunity costs of shutting down the wind turbine vary a lot by the season, as KNMI (2023) shows that the average wind speed in the Netherlands over the past 30 years is around 45% higher in the winter than in the summer months. That is why it is important to incorporate a time variance in the costs of the offshore wind turbine maintenance scheduling problem.

Shafiee (2015) describes that maintenance is either performed preventively or when a component has broken down. Röckmann et al. (2017) argues that preventive maintenance (PM) has significantly lower costs than corrective maintenance (CM), because during the lead time for CM the wind turbine does not work, which results in large additional missed production. To distinguish which maintenance schedules are the most efficient with time-varying costs we have to look at different policies and measure which of them is the most cost-effective. These policies decide when to perform CM or PM and are traditionally time-, age- or condition-based. To measure which policy provides the most cost-effective schedule under time-varying main-

tenance costs Schouten et al. (2022) suggests using an even playing field, where we first only look at a single component that needs some maintenance schedule. This leads to the following research question: **'Which maintenance scheduling policy provides the most cost-effective single-component schedule with time-varying maintenance costs?'**

When looking at a wind farm we look at a group of multiple wind turbines with multiple components that evidently all need maintenance at some point. Dalgic et al. (2015) suggests that around 73% of all maintenance costs lay in the transportation of vessels and equipment. This means that minimizing the number of trips made to wind farms is of critical importance to reduce the total maintenance costs. When we schedule multiple components to be maintained at the same time, we only need to pay the transportation costs once, which means we can potentially save on costs using this method. To research this we formulate the following research question: **'Which maintenance scheduling policy provides the most cost-effective multi-component schedule with time-varying maintenance costs?'**

Intuitively it seems logical to always immediately repair a broken component, to get the wind turbine up and running again. However, when PM is planned to be performed for another component shortly after this component breaks down, it might be beneficial to postpone CM to save on transportation costs. This opportunity arises when the costs of missing production due to a broken component are outweighed by the transportation costs to the offshore wind turbine. Because transportation costs make up such a large part of the overall costs of the maintenance this poses a realistic scenario where costs might be saved. This leads to our last research question: **'Does the inclusion of delaying CM to the multi-component maintenance scheduling problem lead to a reduction in the maintenance costs?'**

This thesis is structured in the following way. Firstly, in Section 2 we provide a short literary review of earlier works that this research is based on. In Section 3 we provide the methodology that is used to find answers to our research questions. This section is divided in two main parts: Section 3.1 describes the models to look at single component models, whilst Section 3.2 shows the formulations and heuristics to consider for the multi-component models. In Section 4 we present the results for respectively the single- and multi-component models and share our insights regarding these results. Lastly, we provide the conclusion of our research in Section 5.

2 Literature review

The literary focus of this research has in a large part already been done by Schouten et al. (2022), which this paper partly reproduces. To give an insight into the main findings we firstly summarize their literature review, after which we look at literature that belongs to multi-component maintenance.

Barlow & Proschan (1996) introduced the most prominent maintenance policies called the age replacement policy (ARP) and the block replacement policy (BRP). They respectively perform preventive maintenance when a critical component age has been reached or when a pre-specified time has passed. Ross (2013) shows that under the assumption of constant costs and average conditions the ARP is optimal. All maintenance policies considered have a deterioration process that is stochastic by nature, which means the common condition-based policy is not considered. The only extra policy considered is the modified block replacement

policy (MBRP), which is the same as the block replacement policy, but with this policy PM is not performed if CM has been done shortly before its scheduled date.

Schouten et al. (2022) also mention that the literature regarding time-varying cost in wind turbine maintenance optimization is really limited, which means we mainly look for literature regarding multiple-component maintenance optimization. Archibald & Dekker (1996) provide multi-component framework for the MBRP with constant costs and find that it performs better than the single-component MBRP. Perez et al. (2015) also show in their research that including multiple components in the optimization via the condition based policy allows significantly more efficient maintenance schedules, by reducing the number of trips made to the offshore wind farms. Dekker et al. (1997) gives a review on the literature of multiple component maintenance optimization with only an economic dependence. This can be applied in our case, because the transportation cost plus missed production cost from maintenance is the only deficit considered in our problem, which means we only have an economic dependence.

The published literature combining time-varying costs and multiple-component maintenance is very limited. The master thesis of Schouten et al. (2019) provides a multi-component extension to the models they suggest in Schouten et al. (2022). They provide a multi-component framework for the ARP, BRP and MBRP policies with time-varying costs. They show that including time-variance in these multi-component models also leads to cost reductions in the maintenance schedules. Moreover, they show that the LP- and MILP-formulations blow up in computation time when we consider more than three components. This means they resort to heuristics for the block-based policies to create maintenance schedules with more than three components.

3 Methodology

In this section we propose the methodology to approach the problems presented in Section 1. In Section 3.1 we discuss the approach to reproduce the research by Schouten et al. (2022). This is done by solving the single-component maintenance problem with an LP formulation for the ARP policy and MIP formulations for the BRP and MBRP policies. In Section 3.2 we discuss our proposed methodologies to approach the multi-component maintenance scheduling problem. We formulate an LP formulation for the ARP multi-component model, which allows for delay of corrective maintenance, based upon the model by Schouten et al. (2019). We also provide two heuristics to solve the multi-component setting with the BRP and MBRP policies and compare them accordingly.

3.1 Single component maintenance

To establish the structure of the problem it is important to state the assumptions Schouten et al. (2022) makes to formulate this problem. They consider a single component in a wind turbine that will keep functioning until it breaks. Because a broken component leads to missed production, the breakage of the component has a monetary incentive to be avoided. To avoid this problem, preventive maintenance (PM) is performed at pre-planned times, which are decided with the maintenance policy. When a component does break, corrective maintenance (CM) is performed instantly. Both types of maintenance replace the current component, albeit not broken, with a new undamaged component. We discretize time into N periods, which can be anything from weeks, months to quarters. In some cases we allow the total period cycle to extend over m years. The component has a stochastic lifetime denoted by X, where $P(X = k) > 0 \; \forall k \in \bar{\mathbb{N}} \backslash \{\infty\}$, with $\bar{\mathbb{N}} = \{0, 1, ..., \infty\}$.

The lifetime distribution is assumed to be known and the failure rate is non-decreasing with component age. Furthermore, the costs for both types of the maintenance are dependent on the period, which follow a cyclical pattern according to the seasonal pattern of nature. This means that costs are usually lowest during the summer, because the average wind speed is lower than during other parts of the year.

3.1.1 Markov Decision Process

To assess the optimization of the maintenance of a single component under time-varying costs, we formulate a model structure that allows the usage of all three proposed policies. Schouten et al. (2022) suggests to use a Markov Decision Process, which comprises of a set of states, an action space, a transition probability matrix and a cost parameter. All states contain a particular period $i_0 \in I_0 = \{1, 2, ..., Nm\}$ ($I_0 \subseteq \mathbb{N}$) and its respective component age $i_1 \in I_1 = \{1, 2, ..., M\}$ ($I_1 \subseteq \mathbb{N}$), where M is the maximum component age. A component cannot exceed this constant maximum age, so it gets replaced immediately when it reaches this age. The state space is thus defined as $I = I_0 \; x \; I_1$. The action space A consists of two possible actions for the current period: we do not maintain the component ($a = 0$) or we do maintain the component ($a = 1$). In most states either actions can happen, except for the broken state, in which $i_1 = 0$ or $i_1 = M$. When the component falters it automatically gets component age 0 and needs to be replaced instantly. This means we always perform CM in the broken state, whilst we always perform PM in the state where the maximum component age has been reached, which leads to the following action space:

$$A(i_0, i_1) = \begin{cases} \{1\} & \text{, if } i_1 \in \{0, M\} \\ \{0, 1\}, & \text{otherwise.} \end{cases} \tag{1}$$

-

To transition between the states in the Markov chain it is necessary to formulate a transition probability matrix. This probability is dependent on whether we perform planned maintenance or the component breaks down and we perform corrective maintenance. In all cases the system can only transition between states that are one period apart and can only transition forward in time, which means it can only transition from a state with $i_0 = j$ to a state with $i_0 = j + 1$. As discussed earlier the component age is lowered to 0 in case of breakage. When we perform preventive maintenance the system can transition to a state with either $i_1 = 0$ or $i_1 = 1$. If the component would break down within the same period that PM was performed the system would take component age 0, if this does not occur the component age naturally goes to 1. If the component does not break down nor PM is planned (i.e. $a = 0$) the component age will simply increase by 1. This leads to the transition probability $\pi_{(i_0,i_1)(j_0,j_1)}(a)$, which is the probability to transition from state $i = (i_0, i_1)$ to state $j = (j_0, j_1)$ under action a. This leads to the following values for $\pi_{i,j}(a)$:

$$\pi_{(i_0,i_1)(j_0,j_1)}(0) = \begin{cases} 1 - p_{i_1} & \text{, if } j_0 = i_0 + 1 \text{ (mod } N), \; j_1 = i_1 + 1, \; i_1 \notin \{0, M\}, \\ p_{i_1} & \text{, if } j_0 = i_0 + 1 \text{ (mod } N), \; j_1 = 0, \; i_1 \notin \{0, M\}, \\ 0 & \text{, otherwise,} \end{cases} \tag{2}$$

$$\pi_{(i_0,i_1)(j_0,j_1)}(1) = \begin{cases} 1 - p_1 & \text{, if } j_0 = i_0 + 1 \text{ (mod } N), \; j_1 = 1, \\ p_1 & \text{, if } j_0 = i_0 + 1 \text{ (mod } N), \; j_1 = 0, \\ 0 & \text{, otherwise,} \end{cases} \tag{3}$$

where $p_x = P(X = x|X \geq x)$, which denotes the failure probability at component age x. Mod is the modulo operator to ensure transitions from period N to period 1 are also included.

The framework now only needs a cost parameter for every state and action combination. Because we only look at a single component we can choose to exclude explicit fixed transportation costs, as they can be included in the price of both types of maintenance. We only have costs if we perform maintenance (i.e. $a = 1$). Furthermore we have to differentiate between costs when performing PM or CM, as CM ensues higher costs due to its unplanned nature. The PM and CM costs are dependent on what period in the year we look at, which is why we denote them respectively by $c_p(i_0)$ and $c_f(i_0)$. This leads to the following cost parameter:

$$c_i(a) = \begin{cases} c_f(i_0) & \text{, if } a = 1 \ \& \ i_1 = 0 \\ c_p(i_0) & \text{, if } a = 1 \ \& \ i_1 \neq 0 \\ 0 & \text{, if } a = 0. \end{cases} \tag{4}$$

3.1.2 Age Replacement Policy

Now that the framework for the Markov Decision Process is constructed, we can apply particular policies on it. The first of which is the Age Replacement Policy (ARP), where we replace the component once it has reached its critical component age. When the component is replaced its component age will reset back again to 0. Ross (2013) already showed that the ARP policy under constant costs is the optimal policy. Under time-varying we do not want to look at a constant critical maintenance age, because we want to favor doing the maintenance during time periods where the costs are low. Schouten et al. (2022) introduces the following rendition of the ARP policy, which is time dependent:

Definition 1: p-ARP. A periodic Age Replacement Policy (p-ARP) is an adaptation of the ARP where the critical maintenance age $t(i_0) \in \mathbb{N}$ is dependent on the period $i_0 \in I_0$. This means that for each period the component has a critical maintenance age $t(i_0)$ at which or above at which PM will be performed.

This critical maintenance age $t(i_0)$ can become ∞, which means PM is never performed for this $i_0 \in I_0$. When we have at least one $t(i_0) < \infty$ we have a finite p-ARP problem, otherwise we resort to a policy where only CM is performed. Schouten et al. (2022) also provides three theorems that support the p-ARP policy. They describe that an optimal policy under time-varying costs exists and this is a p-ARP solution. Furthermore, they provide that under certain conditions a finite optimal p-ARP solution exists. For details we refer to Schouten et al. (2022) page 4.

To model this Schouten et al. (2022) suggests to use an LP formulation based on the formulation introduced by Tijms (2003). To make this LP formulation we introduce decision variable $x_{i,a}$, which represents the long run fraction of situations where we find the system in state i perform action a. Furthermore, we need to split the objective in states where PM is performed and states where CM is performed, to allocate

the correct cost parameter. We do so by defining at $I^b = I_0 \, x \, \{0\}$, i.e. at the states with a broken component. This leads to the following formulation:

$$\text{min.} \quad \sum_{i \in I \setminus I^b} c_p(i_0) x_{i,1} + \sum_{i \in I^b} c_f(i_0) x_{i,1} \tag{5}$$

$$\text{s.t.} \quad \sum_{a \in A(i)} x_{i,a} - \sum_{j \in I} \sum_{a \in A(j)} \pi_{ji}(a) x_{j,a} = 0 \qquad \forall i = (i_0, i_1) \ \in I \tag{6}$$

$$\sum_{i_1 \in I_1)} \sum_{a \in A(i_0, i_1)} x_{i_0, i_1, a} = \frac{1}{Nm} \qquad \forall i_0 \ \in I_0 \tag{7}$$

$$x_{i,a} \geq 0 \qquad \forall i = (i_0, i_1) \ \in I, \ \forall a \in A(i) \tag{8}$$

The objective of this LP formulation (5) minimizes the long-run maintenance costs of the problem. Constraint 6 ensures that the in- and outflow for each state is equal, whilst Constraint 7 ensures that each the long run probability to be in a particular period $i_0 \in I_0$ is the same for all periods. Important to note is that in the p-ARP problem we usually look at cases where $m = 1$, but Constraint (7) is formulated in this way for completeness. Schouten et al. (2022) provides a method to retrieve an optimal strategy for each period, which shows for each period if maintenance should be performed.

3.1.3 Block Replacement Policy

The block replacement policy performs PM after a fixed time span since the last PM. CM is still performed when the component breaks down, but this does not affect the PM schedule. These time intervals need not be the same for every interval, but do follow a pattern, which repeats itself every m years.

Definition 2: p-BRP. A periodic Block Replacement Policy is a policy where the component is preventively maintained in all periods $T_1, T_2, \ldots, T_n < mN$ for some $n \in \mathbb{N}$, since the the start of the period cycle.

So, these periods $T_1, T_2, \ldots, T_n$ do not have to have the same period difference between them, but should be the same for each period cycle. So the first PM can be 6 periods after start, the next can be 8 periods later and the last can be 4 periods after that. However, when starting a new period cycle this same pattern is followed again. To allow this behaviour in the system we look at cases where $m > 1$, instead of $m = 1$.

Schouten et al. (2022) provides two theorems that, similarly to the ARP, provide a guarantee for an optimal p-BRP under certain circumstance and for more details we refer to Schouten et al. (2022) page 5. To formulate the mixed integer linear programming (MILP) problem for the p-BRP we use the formulation for the ARP problem as a base and add constraints to get a correct formulation. Schouten et al. (2022) suggests to introduce decision variables y_{i_0}, which represent the decision to maintain preventively in period i_0. We thus define it as the following:

$$y_{i_0} = \begin{cases} 1, & \text{if we perform PM in period } i_0, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

The addition of the following constraints to the ARP problem forms the MILP formulation of the BRP problem:

$$\text{s.t. } x_{i,0} + y_{i_0} \leq 1 \qquad \forall i = (i_0, i_1) \in I : i_1 > 0 \tag{10}$$

$$x_{i,1} - y_{i_0} \leq 0 \qquad \forall i = (i_0, i_1) \in I : i_1 > 0 \tag{11}$$

$$y_{i_0} \in \{0, 1\} \qquad \forall i_0 \in I_0 \tag{12}$$

Constraint (10) ensures that it is impossible to not take action (i.e. $a = 0$) in periods i_0, where $y_{i_0} = 1$. Whilst, Constraint (11) ensures that we never take action in periods i_0, where $y_{i_0} = 0$. The combination of these constraints guarantees we perform PM in the periods that are allowed and thus that y_{i_0} is modeled as it is designed.

3.1.4 Modified Block Replacement Policy

The p-BRP introduced in Section 3.1.3 has one obvious flaw that is directly tied to its biggest benefit: PM is completely independent from component age. This means that PM can be performed on components that have just been installed via CM, which is suboptimal. Schouten et al. (2022) suggests to use a Modified Block Replacement Policy (MBRP), that only performs PM when the component age is larger than some time-varying threshold. We formally define it in the following way:

Definition 3: p-MBRP. A periodic Modified Block Replacement Policy is a policy where the component is preventively maintained in periods $T_1, T_2, \ldots, T_n < mN$ for some $n \in \mathbb{N}$, since the the start of the period cycle. However, this is under the condition that for situation k a critical maintenance age $t_{(k)}$ is reached, where $t_{(k)} \leq T_k - T_{k-1}$ for $k = 2, 3, \ldots, mN$ and $t_{(1)} \leq T_1 - T_{Nm} + mN$.

To formulate the MILP for the p-MBRP problem we adapt the notation for the critical maintenance age slightly. For each period, we denote a set critical maintenance age t_k for $k \in I_0$. This means that if $T_k = i_0$, we get that $t_{i_0} = t_{(k)}$, which denotes the minimum age at which we perform PM in period i_0. If PM is not planned in a period the critical maintenance age is set to an arbitrary number bigger than M. Furthermore, Schouten et al. (2019) provides a theorem that the optimal p-MBRP has a finite cycle, finite critical maintenance ages and finite block times, which ensures a usable result. Like in Section 3.1.3 we use the p-ARP from Section 3.1.2 as a base for the formulation of the p-MBRP problem. We again add decision variables y_{i_0} introduced in Section 3.1.3. On top of that we add extra decision variables z_i, which represent the decision to maintain the component for a certain age in a certain period.

$$z_i = \begin{cases} 1, & \text{if we maintain at component age } i_1 \in I_1 \text{in period } i_0 \in I_0 \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

This leads to the addition of the following constraints to form the formulation of the p-MBRP problem:

$$
\begin{aligned}
\text{s.t. } & z_{i_0,i_1} - y_{i_0} \leq 0 && \forall i_0 \in I_0,\ \forall i_1 \in I_1 && (14)\\
& z_{i_0,i_1} - z_{i_0,j_1} \leq 0 && \forall i \in I,\ \forall j_1 \in I_1 : i_1 < j_1 && (15)\\
& t_{i_0} + j_0 y_{j_0} + mN y_{j_0} \leq mN + i_0 && \forall i_0, j_0 \in I_0 : j_0 < i_0 && (16)\\
& t_{i_0} + j_0 y_{j_0} \leq mN + i_0 && \forall i_0, j_0 \in I_0 : j_0 > i_0 && (17)\\
& M y_{i_0} - M z_{i_0,i_1} - t_{i_0} \leq M - 1 - i_1 && \forall i \in I && (18)\\
& M z_{i_0,i_1} + t_{i_0} \leq M + i_1 && \forall i \in I && (19)\\
& x_{i,0} + z_i \leq 1 && \forall i = (i_0, i_1) \in I : i_1 > 0 && (20)\\
& x_{i,1} - z_i \leq 0 && \forall i = (i_0, i_1) \in I : i_1 > 0 && (21)\\
& z_i \in \{0, 1\} && \forall i \in I && (22)\\
& y_{i_0} \in \{0, 1\} && \forall i_0 \in I_0 && (23)\\
& t_{i_0} \in \bar{\mathbb{N}} && \forall i_0 \in I_0 && (24)
\end{aligned}
$$

Constraint (14) ensures that y_{i_0} behaves as it is defined by enforcing the component to be maintained for all component ages bigger or equal to the critical maintenance age. Constraints (16) and (17) makes sure that the critical maintenance age condition as defined in **Definition 3** is upheld. This is done by allowing t_{i_0} to take all values when PM is not performed in period i_0, whilst restricting it to its defined condition when PM is performed, i.e. $y_{i_0} = 1$. Lastly, Constraints (18) and (19) ensure that the interaction between the y_{i_0} and z_{i_0} is taken care of by using the maximum age M in clever way. Furthermore, the formulation by Schouten et al. (2019) actually results in a p-ARP formulation most of the time, because the interaction between the $x_{i,a}$ decision variables and the other decision variables is not considered. That is why we add Constraints (20) and (21, which serve the same purpose as Constraints (10) and (11) in the formulation of the p-BRP model in Section 3.1.3). They ensure that PM is always and only performed ($x_{i,1} \geq 0$) in the states i for which $z_i = 1$.

3.2 Multiple component maintenance

To extend the framework of our analysis and include grouping of components and delaying CM to save on transportation costs we have to look at scheduling multiple components at the same time. Schouten et al. (2019) provides a structure to tackle this problem, which extends the MDP formulated in Section 3.1.1. In Section 3.2.1 we explain how we extend the p-ARP problem to include multiple components. Because this formulation blows up for more than three components, Schouten et al. (2019) suggests to use heuristics to extend the block-based policies, which is explained in Sections 3.2.3 and 3.2.4.

3.2.1 Multi-ARP

To extend the single-component analysis of the p-ARP problem to a multi-component analysis Schouten et al. (2019) suggests to change the MDP constructed in Section 3.1.1 and change the problem formulation to allow for multiple components. To incorporate n ($n \geq 1$) components in the MDP we increase the state space to $I = I_0 \times I_1 \times ... \times I_n$. In this case $I_0 \subseteq \mathbb{N}$ is the period the system resides in and $I_j \subseteq \mathbb{Z}$ is the age of component $j = 1, \ldots, n$.

To allow the delay of CM, we extend the state space of I_j $\forall j = 1, \ldots, n$ to all integers, so the component ages can take negative values. If component j is in the broken state $i_j = 0$, it is not necessary anymore to immediately repair it. This means that when the system progresses by one period, the component age of j is lowered by one and becomes $i_j = -1$. This process goes on until CM is performed on component j, after which its component age can go to 0 or 1, depending on if component j breaks down within the period it is repaired in. We choose to allow the component to take a negative component age for two reasons. The first of which is to track for how many periods a component has been broken in this system, which is important to assess the usefulness of delaying CM. Secondly, it allows us to deal with the difference between a component that gets repaired immediately (i.e. $i_j = 0$) and a component that stays broken for a number of periods (i.e. $i_j < 0$). This difference is important because we assume that we cannot combine CM with PM when the component is repaired immediately, while we do allow this for components that have been broken for at least one period. This means we have a difference in the cost parameter for these two cases, which is shown shortly after this. Intuitively it seems logical to just allow the component to stay at the same component age when it is broken and not repaired immediately. However, the two distinctions mentioned cannot be made in this scenario unless we create extra decision variables, which is not preferable.

To extend the action space $A(i_0, i_1)$ from Section 3.1.1 we need to add the other components to it, to retrieve action space $A(i_0, i_1, \ldots, i_n)$. Important to note is that the necessity to immediately take action when a component is broken is not applicable in this case, because waiting for PM on another component can be beneficial. This means that the action space is forced to perform CM in case all considered components are broken. We also enforce the system to repair a broken component if it has reached some maximum broken age MB. This means the system is forced to take action when a component j has reached component age $i_j = -MB$ or $i_j = M$. Because delaying CM is only beneficial for the number periods where the transportation costs outweigh the total missed production costs we can set $MB = \lceil \frac{c_s}{c_w(i_{low})} \rceil$, where $c_w(i_{low})$ is the cost of missed production in the cheapest period i_{low}. This leads to the action space for component j, shown in Formula 25. The total action space is comprised of the combination of the individual action spaces with the same period i_0, unless all components are broken, in which case the system would perform CM for all of them.

$$A^j(i_0, i_j) = \begin{cases} \{1\} & \text{, if } i_j \in \{-MB, M\} \\ \{0, 1\} & \text{, otherwise.} \end{cases} \tag{25}$$

The probability for component j to fail depends on the the age of component and the particular failure distribution component j has. We use a discretized Weibull failure distribution for each component, where differences are caused by differences in the shape and scale parameters. This means that the transition probability to move towards the failed state (i.e. $i_j = 0$) is similar to that in Section 3.1.1. However the transition probability to move forward one period while decreasing component age i_j by one period is now possible, under the condition that the component is broken, i.e. $i_j \leq 0$. We introduce the individual transition probability $\pi^k_{(i_0,i_k)(j_0,j_k)}(a_k)$, which represents the probability to transition from period i_0 to j_0 with the age of component k transitioning from i_k to j_k under action a_k. Because the individual transition probabilities are independent from the other components we multiply these probabilities to get the total transition probability $\pi_{(i_0,i_1,\ldots,i_n)(j_0,j_1,\ldots,j_n)}(A(i_0, i_1, \ldots, i_n)$. The individual transition probability for component k is the following,

where $p^k_{i_k}$ denotes the failure probability for component k at component age i_k:

$$\pi^k_{(i_0,i_k)(j_0,j_k)}(0) = \begin{cases} 1 & \text{, if } j_0 = i_0 + 1 \pmod N,\ i_k \in \{-MB+1, -MB+2, \dots, 0\},\ j_k = i_k - 1, \\ 1 - p^k_{i_k} & \text{, if } j_0 = i_0 + 1 \pmod N,\ j_k = i_k + 1,\ i_k \notin \{-MB, -MB+1, \dots, 0, M\}, \\ p^k_{i_k} & \text{, if } j_0 = i_0 + 1 \pmod N,\ j_k = 0,\ i_k \notin \{-MB, -MB+1, \dots, 0, M\}, \\ 0 & \text{, otherwise,} \end{cases} \tag{26}$$

$$\pi^k_{(i_0,i_k)(j_0,j_k)}(1) = \begin{cases} 1 - p^k_1 & \text{, if } j_0 = i_0 + 1 \pmod N,\ j_k = 1, \\ p^k_1 & \text{, if } j_0 = i_0 + 1 \pmod N,\ j_k = 0, \\ 0 & \text{, otherwise,} \end{cases} \tag{27}$$

The individual cost parameters stay largely the same, but get the addition of a cost $c_w(i_0)$, which represents the cost of lost production of a broken wind turbine due to component j not being repaired for a whole period. We furthermore differentiate the costs of PM and CM of different components by adding a component index to their costs. Schouten et al. (2019) suggests that transportation costs are always paid if we perform CM immediately after a component breaks. This is because CM is performed on unplanned moments within a period, so we do not group it with other maintenance in this period. However, as will be shown in Section 4.2.2, the results gathered by Schouten et al. (2019) actually do not correctly implement this assumption. That is why we allow for CM to be grouped other maintenance and thus add fixed costs once if any maintenance is performed in period i_0. The individual cost parameter for component j thus becomes:

$$c^j_{(i_0,i_j)}(a) = \begin{cases} 0 & \text{, if } a = 0, i_j > 0, \\ c_w(i_0) & \text{, if } a = 0, i_j \leq 0, \\ c^j_p(i_0) & \text{, if } a = 1, i_j \neq 0, \\ c^j_f(i_0) & \text{, if } a = 1, i_j \leq 0. \end{cases} \tag{28}$$

To get the total cost for a state $c_{i,a}$ we add all individual costs together, that share the same period but have their own component age and action combination. We add transportation costs c_s to the total, if we perform any type of maintenance in period i_0. This means that if we perform either PM and/or CM we add the transportation costs to the total costs. This is because we assume that all maintenance in a period is performed in the same maintenance session and thus we only need to go to wind turbines once. Using this way, we guarantee that the fixed costs are only added when performing at least one form of maintenance.

With this model structure we can formulate the multi-component p-ARP problem. Schouten et al. (2019) does this in a similar way as is done in Section 3.1.2. This formulation does not include delay of CM and does not allow the component ages to be negative. When comparing the model with and without delay we use the formulation by Schouten et al. (2019) to gather results for the problem without delay. We adjust this formulation slightly to incorporate delay of CM. We again introduce decision variables $x_{i,a}$, which represents

the long-run probability to be in system state i and perform action a. This leads to the following formulation:

$$\text{min.} \quad \sum_{i \in I} \sum_{a \in A} c_{i,a} x_{i,a} \tag{29}$$

$$\text{s.t.} \quad \sum_{a \in A(i)} x_{i,a} - \sum_{j \in I} \sum_{a \in A(j)} \pi_{ji}(a) x_{j,a} = 0 \qquad \forall i = (i_0, i_1, \ldots, i_n) \ \in I \tag{30}$$

$$x_{i,a} = 0 \qquad \forall k \leq n, \ i \in I, \ a \in A: \ i_k \in \{-MB, M\}, \ a_k = 0 \tag{31}$$

$$\sum_{i_1 \in I_1)} \sum_{a \in A(i_0, i_1)} x_{i_0, i_1, a} = \frac{1}{mN} \qquad \forall i_0 \ \in I_0 \tag{32}$$

$$x_{i,a} \geq 0 \qquad \forall i = (i_0, i_1) \ \in I, \ \forall a \in A(i) \tag{33}$$

This formulation has largely the same interpretation as the one in Section 3.1.2. Only Constraint (31) is added to ensure that the system always performs PM (or CM) when one of the components reaches the maximum or minimum age.

3.2.2 Multi-BRP

Schouten et al. (2019) provides a MILP formulation for the the multi-component block-based policies. They show that this formulation blows up in computation time when looking at three or more components, which makes it unsuitable for usage in real life situations. We introduce heuristics to circumvent this problem in Sections 3.2.3 and 3.2.4. However, in small scale problems we want to compare these heuristics to the exact solution to get a grasp of the usefulness of these heuristics. That is why we also look at the MILP formulation for the multi-BRP problem. We introduce decision variable $y^k_{i_0}$, which describes if we perform PM in period i_0 for component k.

$$y^k_{i_0} = \begin{cases} 1, & \text{if we perform PM in period } i_0 \text{ for component } k, \\ 0, & \text{otherwise.} \end{cases} \tag{34}$$

This problem uses the exact same framework as the multi-ARP formulation and only needs two additional constraints, similarly to the single component model from Section 3.1.3. This means that delaying CM is also an option for the exact multi-BRP problem. We add the following constraints on top of multi-ARP formulation from Section 3.2.1 to get the multi-BRP formulation:

$$x_{i,a} + y^k_{i_0} \leq 1 \qquad \forall k \leq n, \ i \in I, \ a \in A : i_k > 0, \ a_k = 0 \tag{35}$$

$$x_{i,a} - y^k_{i_0} \leq 0 \qquad \forall k \leq n, \ i \in I, \ a \in A : i_k > 0, \ a_k = 1 \tag{36}$$

$$y^k_{i_0} \in \{0, 1\} \qquad \forall k \leq n, \ i_0 \in I_0 \tag{37}$$

Constraints (35) and (36) fulfill similar purposes as Constraints (10) and (11) from Section 3.1.3. They ensure that the $y^k_{i_0}$ decision variables enforce the system to only perform PM on component k in the periods $i_0 \in I_0$ that are allowed by the $y^k_{i_0}$ decision variables.

3.2.3 Sequential optimisation heuristic for block-based policy

Schouten et al. (2019) provides a MILP formulation for the the multi-component block-based policies. They show that this formulation blows up in computation time when looking at three or more components, which makes it unsuitable for usage in real life situations. Because block-based policies base their decision to perform PM completely independent of component age, they are suitable for heuristics Here we describe the sequential optimisation algorithm and in Section 3.2.4 we describe the genetic algorithm, which are both based on the heuristics proposed by Schouten et al. (2019). We compare these heuristics in terms of performance and adaptability to determine which is the most useful. While both are possible to create for both the BRP and MBRP, Schouten et al. (2019) argues that the heuristics systematically overestimate the costs for the MBRP problem. This can only be solved by implementing a simulation for the MBRP process. Because this takes a long time for little gain, we only look at the heuristics for the BRP problem.

The sequential optimisation algorithm is a simple algorithm to find a maintenance schedule. The algorithm essentially optimizes a single component and adds the result to the maintenance schedule. It does this repeatedly for each component in the given component order until all components are scheduled. We update the cost parameter for a period where PM is planned for the first time by reducing it with the transportation costs for the next components. In this way we avoid paying fixed transportation costs for PM in a period where PM is already planned for another component. However, Schouten et al. (2019) shows that the algorithm has no guarantee to be near the optimal solution, which seems logical due to the pre-planned order of the components that need to optimized. This means that periods where PM is performed for the first few components are favored, while those might not have been chosen if the order of the components is changed. We can add delay in this algorithm by allowing the components to take on negative ages, as discussed in Section 3.2.1. If the gain of not paying fixed costs outweighs the costs of losing production, CM can potentially be delayed. This needs some change to the structure of the 'normal' p-(M)BRP structure discussed in Sections 3.1.3 and 3.1.4, but these are straightforward versions of the ones presented in Section 3.2.1. We provide a pseudocode for the algorithm to make it easy to understand:

Algorithm 1 Sequential Optimisation

Require: List of components L ordered via order method OM
  List of periods where PM is performed: $P \leftarrow \emptyset$
  Cost parameter $c_{i_0,i_1}(a)$, including fixed cost for transportation c_s
  Total cost $c_t \leftarrow 0$
  for L **do**
    Solve single component p-(M)BRP for current component l and remove l from L
    Retrieve all periods in which PM is performed for component l
    Add solution cost to the total cost c_t
    for Periods in which PM is performed for component l **do**
      if Current period $k \notin P$ **then**
        $c_{k,i_1}(1) = c_{k,i_1}(1) - c_s$
        add k to P
      end if
    end for
  end for
  Return c_t

The order methods Schouten et al. (2019) suggests are based on characteristics of the components. These order methods do not guarantee better results, but varying the orders might provide a lower total cost, due to difference in order. They suggest ordering the components in one of three following ways, where we start with the component with the characteristic on the left descend the list to the component with the characteristic on the right:

- (SF) Highest maintenance frequency $\longrightarrow$ lowest maintenance frequency
- (SR) Lowest maintenance frequency $\longrightarrow$ highest maintenance frequency
- (SC) Most expensive component maintenance $\longrightarrow$ cheapest component maintenance

3.2.4 Genetic heuristic for block-based policy

The genetic algorithm is a heuristic that is used to perform a smart search through all combinations of periods in which PM is performed. The algorithm performs this smart search by optimizing the maintenance schedule for each component according to the formulations in Section 3.1.3 and 3.1.4. However, it only allows PM to be performed in certain periods of the cycle by adding an additional constraint. To do so, we define a chromosome containing $l = Nm$ genes. Each gene represents a period in the system, where we either allow PM in this period or not. If we do allow for PM in the period gene i_0 represents, we call gene i_0 an active gene. We do this by assigning the value of 1 to gene i_0, if we allow PM to be performed in period i_0, and 0 otherwise. We introduce $g \in \{0,1\}^l$, with the following definition:

$$g_{i_0} = \begin{cases} 1, & \text{if we allow PM in period } i_0 \in I_0 \\ 0, & \text{otherwise} \end{cases} \tag{38}$$

Furthermore, Schouten et al. (2019) defines a formula that calculates the costs for each chromosome. This is done by determining the maintenance costs without setup costs c_k for each component $k \in K$, as is done when solving the MILP described in Sections 3.1.3 and 3.1.4. After which we add the setup costs c_s for each period, where PM can be performed. Schouten et al. (2019) also adds an inflation factor $e(g)$ that depends on the failure probability $b^k_{i_0}$, which represents the probability for component $k = 1, \ldots, n$ to fail in period i_0. This leads to the following formulas:

$$c(g) = \sum_{k \in K} c_k + c_s \sum_{i_0=1}^{l} g_{i_0} + e(g) \tag{39}$$

$$e(g) = \frac{c_s}{m} \sum_{i_0 \in I_0} g_{i_0}((1 - b^1_{i_0}) \ldots (1 - b^n_{i_0}) - 1 + b^1_{i_0} + \cdots + b^n_{i_0} \tag{40}$$

$$b^k_{i_0} = Nm \sum_{a \in A^k} \sum_{i_j \in \{i_1, \ldots, i_n\}} x_{i_0, i_j, a} p^k_{i_j} \tag{41}$$

Schouten et al. (2019) does not describe how to find the failure probability per period, only that it should be used in the formula. So we decided to use Formula 41, which is based on the solution of the single component problems. It calculates the chance to be in a particular component age and multiplies this chance with the chance to fail at this component age. By adding all separate values for the same period together we get the inflation factor for one period. Thus, we per-multiply this large sum with Nm to account the inflation factor for the complete time period.

With these parameters defined we can construct the algorithm as is done in Algorithm 2. In this algorithm we frequently solve the p-(M)BRP problem formulated in Sections 3.1.3 and 3.1.4. However, in this algorithm the following constraint is added, to enforce that PM is only performed in the periods where this is allowed.

$$y_{i_0} = 0 \qquad \forall i_0 : g_{i_0} = 0. \tag{42}$$

Algorithm 2 Genetic Algorithm

Require: List with initial starting population G, which contains a group of starting chromosomes
List of components K
while Improvement in last 3 iterations **do**
for $g \in G$ **do**
Solve single component p-(M)BRP with additional constraint 42 for each component $k \in K$, where we allow PM in the periods according to g.
end for
Pick the fittest $\alpha^{fit} > 0$ chromosomes from the solutions. Fitness is determined by the formula $f(g) = \frac{1}{c(g)}$, i.e. the fittest chromosomes are the cheapest ones.
Create child chromosomes from the fittest α^{fit} chromosomes via cross-over and mutation, explained in more detail in Schouten et al. (2019).
Empty G and add the n fittest children chromosomes to form the new starting population.
end while
Create set N_g as the set of neighbours of the 10 fittest chromosomes.
for $g \in N_g$ **do**
Solve single component p-(M)BRP with additional constraint 42 for each component $k \in K$, where we allow PM in the periods according to g.
end for
Return Fittest chromosome of original 10 and neighbours

We create children of chromosomes in the same way as is done by Schouten et al. (2019). We combine the genes of two parents via cross-over to create two children and mutate each gene of the children with mutation probability $\rho = \frac{0.1}{Nm}$. We also determine the starting population via the same method as Schouten et al. (2019). This is done by performing the single component optimization for each component and creating chromosomes with the minimum and maximum number of required active genes spread evenly through the chromosome.

To perform the local search Schouten et al. (2019) defines the set N_g, which contains all neighbours of a chromosome g like in Formula 43. The neighbours of a chromosome consist of all combinations that are possible to make with their active genes and their neighbours, while maintaining the same number of active genes in the total time period. The neighbours of a gene i_0 are simply the genes that represents the period before or after the period gene i_0 represents. So if we consider a setting with $N = 12$, $m = 1$ with the genes that represent periods 2 and 7 active, the set of neighbours consist of all chromosomes that have a combination of a gene from $\{1, 2, 3\}$ and a gene from $\{6, 7, 8\}$ as their active genes. If more than three genes in a chromosome are equal to 1, we limit the number of neighbours by only considering neighbours where one gene is shifted by a period or where all genes are shifted by one period in the same direction.

$$N_g = \{g' : \forall i \in \{1, \ldots, l\} \text{ with } g_i = 1 \; \exists! \; j \in \{1, \ldots, l\} \text{ with } |(i - j) \bmod l \leq 1\} \tag{43}$$

4 Results

In this section we present the results gathered in this research. To gather the results we implement the methodology explained in Section 3 in Java using the CPLEX solver. The code that is used to gather these results is available as a separate download link with the research paper. In Section 4.1 we present the results for the single component models and our insights regarding these findings. In Section 4.2 we present our findings regarding the multi-component models for either two, three or four components scheduled at the same time.

To conduct these experiments we need to establish at what rate and costs the system transitions to the following state. As mentioned in Section 3.1.2 we assume the the component lifetime X follows a discretized Weibull function, which is dependent on the component age of the component. The appropriate CDF for this function is the following: $F(x) = 1 - \exp{(\frac{x}{\alpha})^{\beta}}$, where $\alpha > 0$ is the scale parameter and β is the shape parameter of the CDF. Furthermore, we calculate the failure probability per component age via $P(X = x|X \geq x) = \frac{F(x)-F(x-1)}{1-F(x-1)}$ for $x \in \bar{\mathbb{N}}$. We also need a cost function that follows a cyclical pattern, which roughly follows the seasonal patterns. Due to the wind patterns we need the highest point to be in the winter periods and the lowest point in the summer periods. Because we choose $N = 12$, January is the month with index 1. To accomplish that January is the most expensive month and July (the 7th month of the year) the cheapest month of the year, Schouten et al. (2019) suggests to use the following formulas to calculate the time-varying PM, CM and delay costs:

$$c_p(i_0) = \bar{c_p} + \Delta \cos\left(\frac{2\pi i_0}{N} - \frac{2\pi}{12}\right), \tag{44}$$

$$c_f(i_0) = \bar{c_f} + \Delta \cos\left(\frac{2\pi i_0}{N} - \frac{2\pi}{12}\right), \tag{45}$$

$$c_w(i_0) = \bar{c_w} + \Delta \cos\left(\frac{2\pi i_0}{N} - \frac{2\pi}{12}\right), \tag{46}$$

where the index p, f and w respectively represent the preventive, corrective and delay parameters. Furthermore, $\bar{c_l}$ are the average costs of index $l = p, f, w$, while Δ is the fractional difference in costs between the peak months and the average months. For all experiments we use that $\bar{c_p} = c_s = 5$, because this creates an even playing field to test the influence of other parameter decisions on the model. We choose the PM costs to be equal to the fixed transportation costs, because Dalgic et al. (2015) argues that a sizeable portion of the maintenance costs lay in the transportation costs. So, we assume for our computational experiments that half of all preventive maintenance costs are caused by the transportation costs.

When we include the delay of CM in a model we use delay cost $\bar{c_w} = 4$. We choose this value, because we intuitively know that the delaying costs need to be lower than the fixed costs that can be saved, otherwise delaying CM is never performed. But, delaying CM comes at the cost of missing production for a whole period, which means that it does amount to a significant cost barrier. That is why we choose the average to be 80% of the fixed transportation costs. Note that the delay cost is dependent on the season and can thus become bigger than the fixed costs in the periods where missing production comes at higher cost. We also use $\alpha^{fit} = 15$ for the genetic algorithm unless stated otherwise. This parameter is kept at such a level where we have enough chromosomes for the starting population, but do not blow up the computation time.

4.1 Single component analysis

We examine the performance of the single component models from Section 3.1 by running the p-ARP, p-BRP and p-MBRP models with the same parameters. We perform the same experiment as Schouten et al. (2022) to show our models work the same as theirs. They assume that the CM costs c_f are 5 times higher than the PM costs, which means $c_f = 50$ and $c_p = 10$. Furthermore, they choose to run two configurations, where one has the combination of $\alpha = 1$, $\beta = 2$ and $m = 1$, while the other combination looks at a longer time duration by looking at $\alpha = 3$, $\beta = 2$ and $m = 3$. The results are presented in Figure 1.

Table 1: Single component yearly costs in \$ with $c_p = 10$, $c_f = 50$, where both the combinations $\alpha =$ 1 years, $\beta = 2$, $m = 1$ and $\alpha = 3$ years, $\beta = 2$, $m = 3$ are considered. Δ represents the percentage wise difference in costs between the peaks of the season and the average costs.

	p-ARP		**p-BRP**			**p-MBRP**		
Δ	Costs	Savings %	Costs	Savings %	PM months	Costs	Savings %	PM months
0%	40.098		41.501		5, 11	40.311		4, 10
10%	40.035	0.16	41.420	0.20	6, 11	40.263	0.12	6, 11
20%	39.701	0.99	40.933	1.37	6, 11	39.855	1.13	6, 11
30%	39.224	2.18	40.361	2.75	6, 10	39.338	2.41	6, 10
40%	38.461	4.08	39.439	4.97	6, 10	38.556	4.35	6, 10
50%	37.635	6.14	38.466	7.31	7, 10	37.773	6.29	6, 10
alpha = 1, m = 1								
0%	13.530		14.173		1, 19	13.622		11, 29
10%	13.252	2.06	13.828	2.44	6, 21	13.338	2.08	18, 33
20%	12.707	6.09	13.135	7.32	7, 19, 31	12.707	6.72	7, 19 , 31
30%	11.779	12.94	12.114	14.53	7, 19, 31	11.779	13.53	7, 19 , 31
40%	10.844	19.85	11.093	21.73	7, 19, 31	10.844	20.39	7, 19 , 31
50%	9.900	26.83	10.072	28.93	7, 19, 31	9.900	27.32	7, 19 , 31
alpha = 3, m = 3								

As expected we see the same results in 1 as presented in Schouten et al. (2022). We see that the costs decrease when the difference in costs between the seasons becomes larger, which means the time-variance in the costs provides an opportunity to save on costs. We also see that the savings become even larger when we look at components with a lower maintenance frequency, because we see bigger savings percentages for the results where multiple years are considered rather than a single year. This is because the system can perform PM in the cheapest month for each year, as we see for the p-BRP and p-MBRP solutions with $\Delta \geq 20\%$. This means that components with a higher average lifetime benefit more from adding time-variant costs to their scheduling problem. Furthermore, we see that the p-ARP is best performing model in all cases, as suggested by the theorems formulated by Schouten et al. (2019). The costs of the p-MBRP solutions are quite close to those of the p-ARP, while they both outperform the p-BRP by quite a margin. Although the computation times are not mentioned in this table, none exceeded 3 seconds and thus the single component models solve rather quickly.

4.2 Double component analysis

Here we present the findings of the multi-component models introduced in Section 3.2. For all these experiments we use the same failure probability and cost parameter as used for the single component problem. It is important to note that the moments when PM are performed can become very complex when looking at more than one component. That is why our multi-component model results mainly focus on the cost advantages that may be gained by including certain variables in the model and we do not provide elaborate insights in the strategy to be used by maintenance teams when they find themselves in a particular situation. To properly dissect the multi-component models and their advantages and disadvantages we create three different main settings.

All three of these settings examine the effects of the time-variant costs on multi-component models, to assess the importance of adding this variance in the model. In Section 4.2.1 we schedule two identical components using all multi-component models presented. In this setting we also look at the influence of delaying CM and assess the preciseness of the two heuristics. In Section **??** we examine the same variables but now in a setting with two different components, to also view the interaction between components that do not naturally follow the same lifetime cycle. In Section 4.2.2 we again view the models with two identical components, but view if the the assumption of Schouten et al. (2019) to not allow for grouping of CM is influential to the costs of the solution. In Section 4.2.3 we compare the models for the double component problem based on the experiments performed in the sections prior to it. Lastly, we view a setting where we look at more than 2 components in Section 4.3. In this setting we can look at the Sequential Optimisation algorithm and Genetic algorithm, to find solutions for the presented problem. We look at both a three and four component problem and see how the heuristics fare compared to each other in terms of computation time and gained results.

4.2.1 Double component experiments

The first step to examine the effects of scheduling multiple components at the same time is by looking at the gains of scheduling two identical components together. These components naturally follow the same lifetime cycle and thus would be scheduled to be repaired on the same moments via the single component BRP model. Similarly we can argue that when the components are identical their critical maintenance ages are similar, which would result in similar maintenance patterns. To test this we use the same average PM, average delay and fixed costs as for all other experiments (i.e. $\bar{c_p} = c_s = 5$, $\bar{c_w} = 4$), with $\alpha = 1$ year, $\beta = 2$, $m = 1$ for all models. To get a broader view of the impact of all variables we look at two different values for $\bar{c_f}$. The first average CM cost is $\bar{c_f} = 15$, which would only maintain once a year in the single component model, while the second value $\bar{c_f} = 45$ results in two scheduled maintenance moments in the single component model, as we have seen in Table 1.

The results of all presented models (including the delay models) in these settings are presented in Table 2. The first thing we notice is that, as is the case for the single component problem, the costs of the optimal solutions decrease when the variance in costs over time increases, with the biggest difference amounting to around 12 %. As expected the optimal BRP solution in the case of $c_f = 15$ is to schedule maintenance once a year in July. Both components get maintained in this month and thus setup costs are saved once. When looking at the case of $c_f = 45$ we always maintain twice a year. When $\Delta = 10\%$ we maintain January and July, while we maintain in July and December for $\Delta \geq 20\%$. We also see that the heuristics find the optimal solution for all cases and thus follow the same schedule.

Table 2: Double identical component yearly costs in $ with $c_s = c_p = 5,\ \alpha = 1$ years, $\beta = 2,\ m = 1$ as its parameters and a c_f that varies. Δ represents the percentage wise difference in costs between the peaks of the season and the average costs.
* Schedules are optimal even though costs are lower than exact model

	ARP		**ARP Delay**		**BRP**		**BRP Delay**		**Genetic**	**SF/SR/SC**	
Δ	**Costs**	**Δ %**	**Costs**	**Delay %**	**Costs**	**Δ %**	**Costs**	**Delay %**	**Costs**	**Costs**	**c_f**
0%	37.879		37.870	0.03	42.645		41.135	3.54	42.603*	42.643	
10%	37.761	0.31	37.722	0.10	41.600	2.45	40.072	3.67	41.472*	41.598	
20%	37.480	1.05	37.396	0.23	40.555	4.90	38.797	4.34	40.341*	40.553	
30%	37.070	2.14	36.604	1.26	39.510	7.35	37.313	5.56	39.210*	39.508	15
40%	36.533	3.55	35.195	3.66	38.465	9.80	35.613	7.42	38.079*	38.463	
50%	35.902	5.22	33.361	7.08	37.421	12.25	33.721	9.89	36.948*	37.418	
CPU	7 min		9 min		5 min		10 min		6 min	1 sec	
0%	70.184		70.087	0.14	73.046		70.962	2.85	73.003*	73.043	
10%	70.020	0.23	69.221	1.14	73.046	0.00	70.017	4.15	73.003*	73.043	
20%	69.805	0.54	66.899	4.16	72.530	0.71	67.725	6.62	72.364*	72.528	
30%	69.473	1.01	63.856	8.08	71.866	1.62	64.881	9.72	71.638*	71.863	45
40%	68.977	1.72	60.322	12.55	71.202	2.52	61.209	14.03	70.913*	71.199	
50%	68.140	2.91	56.162	17.58	69.971	4.21	57.301	18.11	69.742*	69.969	
CPU	7 min		9 min		5 min		12 min		5 min	<1 sec	

We also want to look at two different components, to see how these can synergize without having the exact same characteristics. To do this we look at two different components that get paired with the component from the first experiment with $c_f = 15$, which get maintained once a year in the single component solution. This means we want to find two components that get maintained more times in a year, to see how these different characteristics interact with each other. For the first experiment we choose to use same component as the second experiment, i.e. the component with $c_f = 45$, which gets maintained twice a year in its single component solution. The second experiment is executed with a component that has CM costs and a different scale parameter of the lifetime distribution. We choose to use a component with $\alpha = \frac{2}{3}$ years and $c_f = 35$, which is maintained three times a year in the single component model. All parameters not mentioned are the same for the components from the last experiment.

The results of the first experiment are shown in Table 3. We immediately see similar results in terms of time-variant cost differences and the delay cost reductions as the second experiment in Table 2. We delve more into these factors during the model comparison in Section 4.2.3. The BRP solution presented in Table 3 always performs PM twice for the component with $c_f = 45$, while it only performs PM twice on the other component for $\Delta \leq 10\%$. When $\Delta \geq 20\%$ the component with $c_f = 15$ gets maintained in the cheapest month that the other component also gets maintained in, which is August. The other maintenance takes place in either December or January depending on the Δ.

Table 3: Double component yearly costs in $ with $c_s = c_p = 5,\ \alpha = 1$ years, $\beta = 2,\ m = 1$ for both components with $c_f^1 = 15$ and $c_f^2 = 45$. Δ represents the percentage wise difference in costs between the peaks of the season and the average costs.
* Schedules are optimal even though costs are lower than exact model

	ARP	**ARP Delay**		**BRP**	**BRP Delay**		**SF/SR**	**SC**	**Genetic**
Δ	**Costs**	**Costs**	**Delay %**	**Costs**	**Costs**	**Delay %**	**Costs**	**Costs**	**Costs**
0%	55.830	55.778	0.09	59.358	57.428	3.25	59.355	59.355	59.315*
10%	55.802	55.019	1.40	59.358	56.887	4.16	59.355	59.355	59.315*
20%	55.527	53.428	3.78	58.896	55.458	5.84	58.926	58.926	58.885*
30%	55.011	51.329	6.69	58.105	53.427	8.05	58.155	58.155	58.339
40%	54.356	48.863	10.11	57.314	50.819	11.33	57.383	57.383	57.794
50%	53.653	45.981	14.30	56.213	47.375	15.72	56.383	56.383	56.140
CPU	4 min	9 min		6 min	11 min		<1 sec	<1 sec	11 min

The results for the second experiment are shown in Table 4 and they also follow similar patterns regarding the cost differences caused by time-variant costs and the addition of the delay of CM. When looking at the BRP solution we find that PM is performed three times for the component with $c_f = 35$ for $\Delta = 10, 20\%$, while the component with $c_f = 15$ gets maintained in the cheapest month of the three, i.e. August. For $\Delta = 0\%$ both components get maintained twice in the same period, while for $\Delta \geq 30\%$ we see that the component with $c_f = 35$ is only maintained twice and the other component again joins the PM in August. So we clearly see that scheduling maintenance for two components can lead to a massive decrease in the costs, because the schedules of the two components collide together to ensure that only the setup costs that are necessary are paid.

Table 4: Double component yearly costs in $ with $c_s = c_p = 5,\ \beta = 2,\ m = 1$ for both components with $c_f^1 = 15,\ \alpha^1 = 1$ years and $c_f^2 = 35,\ \alpha^2 = \frac{2}{3}$ years. Δ represents the percentage wise difference in costs between the peaks of the season and the average costs.
* Schedules are optimal even though costs are lower than exact model

	ARP	**ARP Delay**		**BRP**	**BRP Delay**		**SR / SC**	**SF**	**Genetic**
Δ	**Costs**	**Costs**	**Delay %**	**Costs**	**Costs**	**Delay %**	**Costs**	**Costs**	**Costs**
0%	67.606	66.395	1.79	72.395	68.769	5.01	72.639	73.265	72.314*
10%	67.581	65.442	3.17	72.179	67.369	6.66	72.117	72.742	72.223
20%	67.310	63.564	5.56	71.657	65.417	8.71	71.761	71.916	71.613*
30%	66.880	61.212	8.48	70.800	62.953	11.08	70.938	71.030	70.769
40%	66.256	58.527	11.66	69.616	60.024	13.78	69.573	69.800	69.837
50%	65.486	55.453	15.32	68.376	56.602	17.22	68.333	68.560	68.672
CPU	3 min	9 min		5 min	12 min		<1 sec	<1 sec	8 min

4.2.2 Grouping maintenance costs

In Section 3.2.1 we described that Schouten et al. (2019) assumes that corrective maintenance costs always include the fixed transportation costs. This means that when we perform preventive maintenance in a period we do not group this with corrective maintenance that needs to be performed on a broken component. However, during the result gathering we found the same results as Schouten et al. (2019), while we did group

CM and PM in the same period. This should not be the case, as the costs are lower for each state with our assumption. To find out if this difference in assumption indeed does not make a difference, we also ran experiments where CM and PM can not be grouped, as suggested by Schouten et al. (2019). For a fair comparison, we use the same parameters as done in Section 4.2.1 and the results are shown in Table 5.

Table 5: Double component model yearly costs in $ with $c_f = c_p = 5$, $\alpha = 1$ year and $\beta = 2$ under the assumption that we cannot group CM with PM to save on fixed costs. The percentage increase is compared to the model we use with the same parameters as displayed in Table 1.

	ARP				**BRP**			
Δ	**Costs**	**Increase %**	**Costs**	**Increase %**	**Costs**	**Increase %**	**Costs**	**Increase %**
0%	40.43	6.72	72.53	3.34	43.48	1.96	74.27	1.68
10%	40.21	6.49	72.50	3.54	42.43	2.00	74.27	1.68
20%	39.67	5.84	72.38	3.69	41.39	2.06	73.74	1.67
30%	39.00	5.20	72.06	3.72	40.34	2.11	73.08	1.68
40%	38.26	4.74	71.43	3.55	39.30	2.17	72.40	1.69
50%	37.48	4.39	70.47	3.42	38.25	2.23	71.14	1.67
$\mathbf{c_f}$	15		45		15		45	
CPU	4 min		6 min		5 min		4 min	

Even though the results concerning when to perform PM are the same regardless of the assumption, we immediately see the difference in total cost values between the two. With a cost increase ranging between 1.67% and 6.72%, it becomes clear that the assumption of grouping CM with PM provides a significant difference in the costs presented. The schedules created can differ quite significantly with both the possibility to schedule PM more often and less often, because both offer a benefit when allowing for the delay of CM. We either have more opportunities to delay CM, or we have to pay less guaranteed fixed costs, so the exact reasons for difference in schedules are quite complex. Furthermore, we see that this increase is smaller for the BRP than the ARP, which means that the BRP is less reliant on corrective maintenance operations than the ARP policy. We also see that the cost increase diminishes when the CM costs increase, which is most likely due to the proportional impact of the fixed costs being smaller when the CM costs are higher. This means avoiding the need for CM gets a higher priority above the opportunity to group CM with PM, which results in a smaller difference between the costs of the two assumptions.

4.2.3 Model comparisons

We have seen in the experiments so far that the just like the single component model, the ARP formulation always outperforms the BRP formulation. We also see that the inclusion of the possibility to delay CM can lead to huge gains reaching up to 18% and always provides a positive impact on the total costs, as long as $\bar{c_w} < c_s$. Interestingly, the inclusion of delaying CM seems to have a higher average cost decrease for the BRP problem than for the ARP problem. This can be explained by the planned nature of the BRP. Because the schedule of the BRP is always adhered to, the components have a bigger chance to stay in sync, which presents more opportunities for a component to break down just before PM is performed for another component. The influence of including delay also becomes bigger when Δ increases, because this presents more opportunities to cheaply postpone CM by one or two periods to save on fixed costs. The last difference we notice is between the component combinations that need to be maintained once a ear and the ones that

need to be maintained more often. We see that including delay has a stronger effect the more PM we perform in a period cycle, which makes sense as we have more periods to delay to and thus more opportunities to save on costs. The computation time of the models with delay are about twice as high as the computation time of the models without delay, due to increase in state space.

When looking at the heuristics we see that both find the optimal solutions for at least 75 % of all settings. The Genetic algorithm has a systematic underestimation of the costs, because even though it selects the correct chromosome the costs appear to be smaller than the optimal costs. This is likely caused due to a faulty inflation factor, because the one in our algorithm never returns values higher than 0.03. The biggest difference between an optimal cost value and an optimal genetic value is $ 473, for the BRP model with $\Delta = 50\%$, $c_f = 15$ and identical components. To account for this underestimation we add $ 500 to all solutions in Section 4.3 when we look at models with more than two components. Furthermore, we see no real big difference between the values gathered via the different order methods and thus can not say that one of these is decisively better than the other. The computation time of the Genetic algorithm is actually quite high and exceeds the exact problem formulations in half the cases. The computation time of the Sequential Optimisation algorithm is extremely fast, due to its simple nature.

4.3 Triple and quadruple component analysis

The big advantage of using heuristics to find solutions when scheduling maintenance is that we can examine a lot more components at the same time. Now that their precision has been shown by comparing them to the exact double component models, we can look at the actual usefulness of these heuristics. We do so by examining how the Sequential Optimisation Algorithm and the Genetic Algorithm perform when optimizing for three or four different components. We do so by looking at two combinations of components show in Table 6.

Table 6: Component parameters for respectively the triple component and quadruple component experiments.

Comp no.	$\bar{\mathbf{c_p}}$	$\bar{\mathbf{c_f}}$	α	β
1	5	15	12	2
2	5	45	12	2
3	5	35	8	2

Comp no.	$\bar{\mathbf{c_p}}$	$\bar{\mathbf{c_f}}$	α	β
1	5	15	12	3
2	5	25	24	3
3	5	25	24	2
4	5	45	36	2

The components from the three component configuration are the ones that were present in the experiment from Section 4.2.1 with two different components. We have chosen them again, because they separately they would be maintained once, twice or three times a year. So seeing this interaction could prove to be interesting. The four components are chosen with bigger differences between their characteristics. They start with cheaper but more volatile components and gradually become more expensive to perform CM on, but also less volatile. We chose this configuration, because usually cheaper components can break more easily and are less optimized, while more expensive components are usually designed to last longer and have less volatile lifetime spans.

For the triple component models we looked at both $m = 1$ and $m = 3$, while we examine $m = 3$ for the quadruple component models. However, these results proved to be the same, so we used the lowest m per model to get the lowest computation times. The results regarding the triple and quadruple component models

are shown in Table 7. Note that the costs for the Genetic algorithm are actually an upperbound for the costs. This is because the actual solution costs have been increased by $ 500, due to the underestimation of the costs found in the previous sections. For the triple component model we find that the genetic algorithm find the best solution for $\Delta \geq 20\%$, while the Sequential Optimisation algorithm with order method SR performs the best for the other values of Δ. we also see that the SR order method outperforms the Sf and Sc order methods in this case. The cost values do not differ massively across the models, deviating by a maximum of around $ 2000, which amounts to a 2 % difference. We do see some interesting behaviour for the SR order method, where the costs are lower for $\Delta \leq 10$ than for $\Delta = 20, 30\%$ for the first time in any model. This can be explained by the order of the components causing a period to be chosen for the first or even second component, which does not provide a cost decrease sharp enough for the last component to also maintain in this period. This leads to extra fixed costs, which causes the increase in price.

Table 7: Yearly costs for the triple and quadruple component models with the parameters described in Table 6.

	Triple component costs				**Quadruple component costs**			
Δ	**SR**	**SF**	**SC**	**Genetic**	**SR**	**SF**	**SC**	**Genetic**
0%	102.232	103.855	103.855	102.692	56.433	54.816	54.816	62.706
10%	101.709	103.855	103.855	102.126	51.658	51.622	51.622	60.664
20%	103.005	103.005	103.005	101.561	47.648	49.430	49.430	58.621
30%	101.849	101.849	101.849	100.995	45.355	47.498	47.498	56.579
40%	100.361	100.694	100.694	99.963	43.059	43.056	43.056	54.536
50%	98.537	98.980	98.980	97.923	40.759	40.759	40.759	52.494
CPU	1 sec	<1 sec	<1 sec	27 min	38 sec	33 sec	32 sec	>2 hours

For the quadruple component problem we immediately notice that the Genetic algorithm performs way worse than the Sequential Optimisation models. No clear explanation can be found for this, partly due to the high computation time of the Genetic algorithm not allowing for tests to be performed quickly and efficiently. We find that the SF and SC order methods provide the lowest cost solutions for all values of Δ. Furthermore, we see that the Sequential Optimisation model outperforms the Genetic algorithm by a landslide regarding the computation times. The Genetic algorithm takes more than two whole hours to solve the four component model and takes half an hour to solve the three component problem.

This experiment shows the downsides of our Genetic algorithm. Even though it finds the optimal solution with more reliability than the Sequential Optimisation algorithm in the double component models, it blows up in computation item and does not guarantee better results for more than three components. This means that the Genetic algorithm in its current state finds itself in an awkward position where it performs better than the Sequential Optimisation algorithm for the models that we can actually compute exactly. However, it performs worse than the Sequential Optimisation algorithm in models that can not be computed exactly.

5 Conclusion

To goal of this research is to optimize the strategic maintenance scheduling problem for wind turbines. We do so by incorporating maintenance costs that vary over time and are at their lowest during the summer, to simulate the variance of the wind speed during the year. We aim to optimize this maintenance problem via one of three policies: the Age Replacement Policy (ARP), the Block Replacement Policy (BRP) or the Modified Block Replacement Policy. The first chooses when to perform preventive maintenance (PM) based on the age of our component of interest in the wind turbine, while the others base it on the time that has passed since the last PM. We respectively make an LP and two MILP formulations, based on the ones created by Schouten et al. (2022).

To extend this problem we also look at models that can solve multiple components at the same time to save on setup costs. We introduce two MILP formulations for both the multiple component ARP and BRP problems. We also introduce two heuristics to avoid potential lengthy computation time problems: the Sequential Optimisation algorithm and the Genetic algorithm. For the exact formulations of the ARP and BRP problems we also introduce a way to delay corrective maintenance (CM) until the next PM is performed, to avoid paying for fixed costs.

This selection of models provide a plethora of variability in the results of the maintenance scheduling problem. Similarly as Schouten et al. (2019) we find that the sole inclusion of time variant maintenance costs can lead to a reduction of up to 28% in maintenance costs for slowly degrading components, while we see a maximum cost reduction of 12% for components that need to be maintained at least once a year. We also find that scheduling multiple components at the same time leads to numerous possibilities to perform maintenance in the same period and thus save on setup costs. This is the case when looking at two of the same component, but also when examining two components that do not share many characteristics. Also including the possibility to delay CM in the double component models proved to be fruitful as this can lead to cost reductions up to 18%.

Both heuristics proved to be near optimal when looking at only two components. While the Sequential Optimisation algorithm has a really small computation time, the Genetic algorithm performed as slow or even slower than the MILP models. We also found that the Genetic algorithm explodes even further in computation time when looking at more than three components, similarly to the exact models. It also performed extremely poor when optimizing for four components, because it gathered optimal costs that where 20 - 30 % higher than the costs for the Sequential Optimisation algorithm. We thus conclude that in its current state the Genetic algorithm is not that useful, because it functions as a worse variant of the exact MILP formulations. However, the Sequential Optimisation algorithm provides a fast and decently reliable solution with the potential to optimize dozens of components at the same time. So we highly recommend this heuristic to be used when optimizing for a large number of components.

For further research we recommend to take a deeper look at the Genetic algorithm. Schouten et al. (2019) already showed its potential, but by improving on its computation speed and providing the correct inflation factor this can be a very useful heuristic. We also recommend to examine an extension of the heuristics to allow for the delay of CM. We could not do this due to time constraints, but the inclusion of the delay of CM seems relatively simple to implement. Because we found that there is a big gain in including this possibility, we recommend that this option is investigated further.

References

Archibald, Y. & Dekker, R. (1996). Modified block-replacement for multiple-component systems. *IEEE transactions on reliability*, *45*(1), 75–83.

Barlow, R. E. & Proschan, F. (1996). *Mathematical theory of reliability.* SIAM.

Dalgic, Y., Lazakis, I., Dinwoodie, I., McMillan, D. & Revie, M. (2015). Advanced logistics planning for offshore wind farm operation and maintenance activities. *Ocean Engineering*, *101*, 211–226.

Dekker, R., Wildeman, R. E. & Van der Duyn Schouten, F. A. (1997). A review of multi-component maintenance models with economic dependence. *Mathematical methods of operations research*, *45*, 411–435.

Hartman, L. (2022, Aug). *Wind turbines: The bigger, the better.* Retrieved from `https://www.energy.gov/eere/articles/wind-turbines-bigger-better`

KNMI. (2023). *Climate of the netherlands.* Retrieved from `https://www.knmi.nl/klimaat`

Perez, E., Ntaimo, L. & Ding, Y. (2015). Multi-component wind turbine modeling and simulation for wind farm operations and maintenance. *Simulation*, *91*(4), 360–382.

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. *Aquaculture Perspective of Multi-Use Sites in the Open Ocean: The Untapped Potential for Marine Resources in the Anthropocene*, 97–113.

Ross, S. M. (2013). *Applied probability models with optimization applications.* Courier Corporation.

Schouten, T., 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.

Schouten, T., Rommert, D. & Eruguz, A. S. (2019). *Optimal maintenance policies for wind turbines under time-varying costs* (MSc). Master's thesis. Rotterdam, the Netherlands: Erasmus Universiteit.

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

Tijms, H. C. (2003). *A first course in stochastic models.* John Wiley and sons.

Code explanation

All codes to gather results in this research are programmed in Java and are provided with both Javadocs and comments to make them more understandable. There are five classes that have a main method and are used to run the codes to get the results from Section 4. These are named the following and we providea short description:

- **MainARP**: This class is used to run the single component p-ARP formulation from Section 3.1.2. It also includes codes to get all states, to get the cost parameter and to get the transition probability matrix that is used for all single component models.
- **MainBRP**: This class is used to run the single component p-BRP formulation from Section 3.1.3.
- **MainMBRP**: This class is used to run the single component p-MBRP formulation from Section 3.1.4.
- **MultiARP**: This class is used to run the multi-component ARP and BRP formulations from Sections 3.2.1 and 3.2.2.
- **Sequential**: This class is used to run both the heuristics from Sections 3.2.3 and 3.2.4. It contains all the codes to run these algorithms.

To recreate the runs that are used in this research all three single component models (i.e. MainARP, MainBRP and MainMBRP) need to be run once with the combination of $alpha = 12$, $m = 1$ and once with $alpha = 36$, $m = 3$.

The following runs are all in MUltiARP. If a parameter is not mentioned it does not need to be altered. This goes for: int $M = 50$, int $N = 12$, int $minAge = 5$, double[] $avgCostPM = \{5, 5\}$, double $avgCostSetup = 5$, double $avgCostDelay = 4$ and double[] $beta = \{2, 2\}$ We need to run the system with the following configurations for both boolean BRP = true (we use the BRP) and BRP = false (we use the ARP) and for both boolean $delay$ = true and $delay$ = false:

- Separately run $assumption = true$ or $assumption = false$ for double[] $avgCostCM = \{15, 15\}$ or double[] $avgCostCM = \{45, 45\}$. Thus 4 combinations all with $\alpha = \{12, 12\}$, with the two options for BRP and $delay$ we get to 16 combinations. After this we always assume $assumption = false$.
- $avgCostCM = \{15, 45\}$ and $\alpha = \{12, 12\}$, i.e 4 combinations.
- $avgCostCM = \{15, 35\}$ and $\alpha = \{12, 8\}$, i.e. 4 combinations.

The last runs are all done in Sequential. For the both heuristics simply comment out 5 of the 6 combinations of components (from either line 83 to 102 or line 118 to 137) and leave the one that needs to be optimized. By doing this for all 6 component combinations we can find all values necessary for the algorithm we are interested in. This is done for both $m = 1$ and $m = 3$, all other parameters can stay untouched.