A numerical study of Markov decision process algorithms for multi-component replacement problems

Andersen, Jesper Fink; Andersen, Anders Reenberg; Kulahci, Murat; Nielsen, Bo Friis

Published in:
European Journal of Operational Research

Link to article, DOI:
10.1016/j.ejor.2021.07.007

Publication date:
2022

Document Version
Peer reviewed version

A numerical study of Markov decision process algorithms for multi-component replacement problems*

Jesper Fink Andersen[a,*], Anders Reenberg Andersen[a], Murat Kulahci[a,b], Bo Friis Nielsen[a]

[a] *Technical University of Denmark, Department of Applied Mathematics and Computer Science, Anker Engelunds Vej 1, 2800 Kgs. Lyngby, Denmark*

[b] *Luleå University of Technology, Department of Business Administration, Technology and Social Sciences, Luleå, Sweden*

Abstract

We present a unified modeling framework for Time-Based Maintenance (TBM) and Condition-Based Maintenance (CBM) for optimization of replacements in multi-component systems. The considered system has a K-out-of-N reliability structure, and components deteriorate according to a multivariate gamma process with Lévy copula dependence. The TBM and CBM models are formulated as Markov Decision Processes (MDPs), and optimal policies are found using dynamic programming. Solving the CBM model requires that the continuous deterioration process is discretized. We therefore investigate the discretization level required for obtaining a near-optimal policy. Our results indicate that a coarser discretization level than previously suggested in the literature is adequate, indicating that dynamic programming is a feasible approach for optimization in multi-component systems. We further demonstrate this through empirical results for the size limit of the MDP models when solved with an optimized implementation of modified policy iteration. The TBM model can generally be solved with more components than the CBM model, since the former has a sparser state transition structure. In the special case of independent component deterioration, transition probabilities can be calculated efficiently at runtime. This reduces the memory requirements substantially. For this case, we also achieved a tenfold speedup when using ten processors in a parallel implementation of algorithm. Altogether, our results show that the computational requirements for systems with independent component deterioration increase at a slower rate than for systems with stochastic dependence.

Keywords: Maintenance, Dynamic Programming, Multi-component system, Markov decision process, Numerical study

1. Introduction

In traditional maintenance models, the decision to maintain a system is often based on its age at the time of the decision, also known as Time-Based Maintenance (TBM). Records of failure times of similar systems make it possible to estimate a lifetime distribution that describes the uncertainty in the time to failure within that population of systems. An alternative approach is Condition-Based Maintenance (CBM), where information about the physical condition of the system is utilized for maintenance decisions.

For both TBM and CBM, the methods for optimizing the maintenance policy can be classified using the scheme presented in Nicolai & Dekker (2008). Here, the authors divide optimization procedures into policy optimization, exact-, and heuristic methods. They also classify according to the planning horizon, which can be either infinite or finite. We consider exact methods and an infinite horizon in this study. Policy optimization is the most common approach for infinite planning horizons, where the system dynamics are assumed stationary. A parametrized policy, for instance a variation of a control-limit policy, is considered and its parameters are optimized by deriving an analytical expression for the long-run cost per time unit using renewal theory (Castanier et al., 2005; Abdel-Hameed, 1987; Zhang et al., 2020). The restriction to a specific type of policy facilitates the analyses, but the solution is generally not guaranteed to be globally optimal.

In both TBM and CBM, a decision is made towards maintaining the system based on new evidence (data) collected from the system. This makes Markov Decision Processes (MDPs), a modeling framework for sequential decision making, a natural candidate for TBM and CBM models. The methods for optimizing the policy in an MDP can be either Dynamic Programming (DP), which is an exact optimization method, or reinforcement learning and approximate DP, which both belong to the category of heuristic optimization methods. DP algorithms are, under very mild assumptions, guaranteed to find globally optimal solutions in finite time. However, the required computational effort grows exponentially with the dimension of the state space in the MDP. This is known as *the curse of dimensionality in dynamic programming*.

Several MDP models have already been proposed for single-component systems, and because

⋆ *Declarations of interest*: none
*Corresponding author
Email addresses: jfan@dtu.dk (Jesper Fink Andersen), arean@dtu.dk (Anders Reenberg Andersen), muku@dtu.dk (Murat Kulahci), bfni@dtu.dk (Bo Friis Nielsen)

dimensionality is not an issue, these are solved using DP (Elwany et al., 2011; Chen et al., 2015; Neves et al., 2011). In a review on CBM optimization by Alaswad & Xiang (2017) the authors emphasize that as modern industrial systems rarely consist of a single maintainable component, there is an increasing need for multi-component models. In a more recent review on maintenance optimization (de Jonge & Scarf, 2020), it is pointed out that there has been a shift towards heuristic policies and approximate methodologies for multi-component systems. Indeed, reinforcement learning algorithms, even though they are not guaranteed to converge to a globally optimal policy, have been used on both multi-component TBM problems (Xia et al., 2008) and CBM problems (Andriotis & Papakonstantinou, 2019) of massive size.

In this study, we investigate the largest problem size, or correspondingly the maximum number of components, for which it is computationally feasible to obtain a globally optimal solution with DP. We focus on models with replacement as the only maintenance action, fixed regular inspection intervals, and full system observability. In accordance with the motivation for practical use of MDPs given in a recent book (Boucherie & van Dijk, 2017), we give the following two reasons for investigating this:

- Firstly, even though policy optimization and heuristics might scale well for a given multi-component maintenance problem, they should be validated by comparison with optimal solutions to instances of the problem that are ideally as large as possible. This is the only way of strengthening the belief that the heuristics perform well for even bigger problems.

- Secondly, the computational power of a present-day CPU combined with an optimized implementation may be efficient enough for solving real-life multi-component problems of moderate size.

In this study we consider a multi-component system, and formulate two MDP models for the replacement problem for CBM and TBM. The system is general enough that the two resulting models closely resemble most of the MDP models considered in other papers. We solve both MDPs for a varying number of components using DP. We do not discuss the cost performance of the resulting TBM and CBM optimal policies, but consider how the computational requirements for solving the MDPs scale with the number of components. Since the structure of the two MDPs are largely different, the answer is not the same in the two cases.

Single-component CBM studies using MDPs, as proposed in Derman (1963); Kurt & Kharoufeh (2010); Neves et al. (2011), model component deterioration on a discrete set of states $S' = \{0, \ldots, D\}$ with 0 being as good as new and D being a failed component. In TBM modeling, these states represent the age of a component (Dekker et al., 1996; Barreto et al., 2014). When formulating an MDP for a system with N components, a natural choice for the state space, S, would be $S = (S')^N$, which has size $|S| = (D+1)^N$. The numerical examples presented in most multi-component CBM studies are limited to a few components, so that general conclusions from identified structures in the optimal policy and sensitivity analyses are easier to convey. For instance, in Sun et al. (2018) the authors consider a system with $N = 3$ and $D = 19$ as their largest example, which results in $|S| = (19+1)^3 = 8000$. A larger example is found in Jiang & Powell (2015) where a finite-horizon CBM replacement problem with $N = 7$ components, $D = 10$, and a state space size of $|S| = (10+1)^7 \approx 1.9 \times 10^7$ is solved to optimality. Concerning multi-component TBM models using MDP, the largest example we found is in Barreto et al. (2014), where an infinite-horizon TBM replacement problem with up to $N = 5$ components and $|S| \approx 1.6 \times 10^5$ is solved to optimality. The reported computation times in these studies are heavily dependent on the algorithm implementation and the hardware that is used. Their purpose is to act as a baseline for comparison with novel heuristic algorithms, and as such we cannot use them as guidelines for a general size limit of the multi-component maintenance problems. The implementation details are an often overlooked aspect in studies using DP, but they have a crucial impact on the performance. For example, our implementation is able to solve numerical examples from Olde Keizer et al. (2016) and Barreto et al. (2014) orders of magnitude faster, than the times reported in the studies.

Maintenance models using MDP often start by assuming a discrete state space, because this is a requirement of DP algorithms. In practice, the condition monitoring procedure in a CBM application includes processing raw data signals of physical variables such as vibration or temperature. In this case, the deterioration process of the components is most naturally modeled on a continuous state-space. In accordance, we assume the underlying deterioration of the components follow a gamma process in the system we consider. We then explore the effects of discretization, that is, number of intermediate states between as-good-as-new and failure. The level of discretization greatly affects the number of components we can handle in a multi-component system. Nonetheless, we have not found any studies that go into detail in this aspect. Our results indicate

that a coarser discretization level than previously suggested is adequate.

The potential for optimization in an implementation of DP depends on the system characteristics, for instance how we compute and store the transition probabilities in the MDP. In this context, it is important whether the deterioration processes of the system components are dependent, also known as stochastic dependency. In the system we consider, the dependency between the gamma process of each component is modeled using a Lévy Copula. We illustrate how the level of dependency affects the computational requirements for solving the MDP.

In summation, the contributions of our paper are the following:

- We propose a unifying modeling framework for TBM and CBM in multi-component systems.
- We provide realistic limits to the size of multi-component replacement problems, for which they can still be solved to optimality using commonly accessible computer resources.
- We empirically demonstrate the different computational limitations when solving large CBM problems compared to large TBM problems.
- We show that a relatively coarse choice of discretization level is sufficient to solve multi-component CBM problems with continuous-state deterioration processes, such that a near-optimal solution is obtained.
- We show how stochastic dependency among components affects the computational requirements.

The rest of the paper is organized as follows: In Section 2, we present the system and formulate the MDP models for CBM and TBM. In Section 3, we summarize the DP algorithms we use to solve the MDPs, and relevant implementation details. Section 4 contains the results of the numerical experiments, and in Section 5 we provide a conclusion to the study.

2. Problem description

We consider a multi-component system for which we formulate two replacement models for CBM and TBM, both using an MDP. In the models, we assume (a) fixed inspection intervals, (b) full system observability, and (c) perfect maintenance actions, i.e., replacements.

The purpose of this study is to investigate the computational requirements for solving the two models, and we note that relaxing any of the three assumptions makes this task more difficult. Before we proceed to the formulation of the models, we therefore provide some examples of how assumptions (a)-(c) can be relaxed and how this affects the resulting optimization problem.

(a) The inspection frequency, for revealing the system condition, is sometimes assumed to be a decision variable. An example of a policy optimization approach is the delay-time model with delayed postponement in van Oosterom et al. (2014). Here, the policy is parametrized by a threshold for the age at which we inspect a component, and if it is found to be close to failing, a second parameter determines the time until it is replaced. One way to include irregular inspections in an MDP formulation is to consider a very short interval between decision epochs and include "inspection" and "no inspection" as possible actions (Andriotis & Papakonstantinou, 2019). The MDP version then considers a more flexible policy space than the policy optimization version, since an inspection or replacement may be triggered by any combination of the time since the last inspection and the condition found at that inspection. In that sense, this MDP formulation can also be seen as a hybrid of TBM and CBM. However, an obvious caveat is that the state space in this MDP has a higher dimension than a pure CBM or TBM model, since it requires two variables per system component, namely age and condition.

(b) The issue of partial information is due to inspections not being perfect. This can be dealt with by formulating the optimization problem as a partially observable MDP. Except for very small problems, these models can only be solved approximately either using point-based algorithms (Nguyen et al., 2019; Pineau et al., 2003), or a policy optimization approach as in Naderkhani ZG & Makis (2015).

(c) Imperfect maintenance can be incorporated by including actions in the MDP that improve the condition or reduce the age of components, but not all the way to the as-good-as-new state. The result is a larger action space and, if the effect of the imperfect maintenance is random, more nonzero transition probabilities in the MDP. Both of these factors make for a more difficult optimization problem.

2.1. System description

We consider a multi-component system with N deteriorating components with a K-out-of-N reliability structure, meaning that the system is functioning as long as K components are

functioning. We choose this reliability structure as it is quite general and it includes the special cases of series systems ($K = N$) and parallel systems ($K = 1$). We note that other reliability structures, such as series-parallel systems, can be modeled by appropriately changing the reward function in Section 2.3. Examples of other studies that use MDP and the K-out-of-N structure are Sun et al. (2018); Olde Keizer et al. (2016); Andriotis & Papakonstantinou (2019).

The components are subject to deterioration, and their joint condition is described by a multivariate stochastic process $\{\mathbf{X}_t\}_{t\in[0,\infty)} = \{(X_t^1, \ldots, X_t^N)\}_{t\in[0,\infty)}$, with X_t^i being the condition of component i at time t. We assume $X_0^i = 0$ and that component i fails when X_t^i reaches a failure threshold L, which is assumed to be the same for all components without loss of generality for the process we describe in Section 2.2. The components are stochastically dependent, meaning that the marginal processes X_t^i are mutually dependent. This is relevant for systems where the deterioration of components are affected by the same external factors in the operating environment, e.g., weather condition.

In the CBM model, we assume that we observe the process, i.e., the condition of each component. In the TBM model, we only observe whether components are functioning or not, and replacement decisions are based on the ages of the components. We assume that replacements can be carried out at regularly spaced maintenance windows, and that the time required to replace a component is negligible. If component i is replaced before it fails, we incur a preventive replacement cost, c_p^i, and if the replacement happens after the failure we incur a corrective replacement cost, c_c^i. Furthermore, we assume there is a setup cost, c_s, if at least one component is replaced in a given maintenance window and a system failure cost, c_f, if less than K components are functioning at the time of replacement.

2.2. Deterioration process

In practice, CBM often involves monitoring physical variables of the components, hence it is natural to model the deterioration as a continuous-state stochastic process. The choice of stochastic process is often dictated by the nature of the physical deterioration process, whether it is corrosion, shock damage, crack growth, and the kind of data that is collected, e.g. temperature, vibration magnitude, or geometry. In several CBM studies, MDPs are used in conjunction with such deterioration processes. For example, Brownian motion with drift (Sun et al., 2018), geometric Brownian motion (Elwany et al., 2011), inverse Gaussian process (Chen et al., 2015), and the gamma process

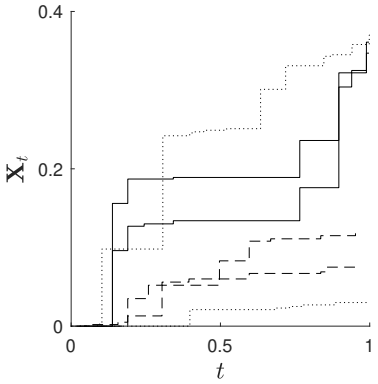

Figure 1: Realizations of $\mathbf{X}_t$ for $N = 2$, $\alpha_1 = \alpha_2 = 7/4$, $\beta_1 = \beta_2 = 15/2$ and $\theta = 0.2$ (dotted), $\theta = 1$ (dashed), $\theta = 3$ (solid).

(Nguyen et al., 2019; Andriotis & Papakonstantinou, 2019).

In this paper, we assume that $\mathbf{X}_t$ is a multivariate Lévy process, which has time-homogeneous and independent increments $\mathbf{X}_t - \mathbf{X}_s$, $0 \leq s < t$. We assume for each $i = 1, \ldots, N$ that the marginal process is a gamma process, that is, $X_t^i - X_s^i \sim \text{Gamma}((t-s)\alpha_i, \beta_i)$, where α_i and β_i are the shape and rate parameter of component i. The dependence between the components is described via a Clayton-Lévy copula function. This copula has one parameter, $\theta > 0$, which dictates the dependency of jump sizes, where larger values increases the tendency to observe simultaneous large jumps in each component. Figure 1 shows simulated realizations of $\mathbf{X}_t$ with two components for different values of θ. We will also consider the special case of independent components, which we will abbreviate as the $\theta = 0$ case. We will not go into details about Lévy Copulas but refer the reader to Grothe & Hofert (2015), which presents the simulation algorithm we use in Appendix A, or Shi et al. (2020); Li et al. (2016); Jiang et al. (2019), where gamma processes and the Clayton-Lévy copula are used in the context of deterioration modeling.

2.3. MDP formulation

In this section we formulate an MDP for both the CBM and the TBM version of the replacement problem. In both models we consider the maintenance windows, which in MDP terminology are called decision epochs, to be spaced with unit distance, $\tau \in \{0, 1, \ldots\}$. At each epoch, the system occupies one of a finite set of states $S = \{0, \ldots, D\}^N$, where an element $\mathbf{s} = (s_1, \ldots, s_N)$ describes the state of the system. In both the CBM and the TBM model, the state of component i is a number $s_i \in \{0, \ldots, D\}$, where $s_i = 0$ corresponds to a new component and $s_i = D$ is a failed

component. The information in s_i is different for the two models though. In the CBM model, s_i is a discretized version of the condition, X_τ^i, and in the TBM model s_i is the age of the component, i.e., the number of epochs since the last replacement.

At each decision epoch we choose an action from a finite set $A = \{0,1\}^N$. When an element a_i of an action $\mathbf{a} \in A$ equals 1 (0), this corresponds to replacing (not replacing) component i. Depending on the current state $\mathbf{s} \in S$ and action $\mathbf{a} \in A$ we receive a reward $r(\mathbf{s},\mathbf{a})$ and the system transitions to a new state $\mathbf{s}' \in S$ with probability $p(\mathbf{s}'|\mathbf{s},\mathbf{a})$. Together S, A , $p(\cdot|\cdot,\cdot)$, and $r(\cdot,\cdot)$ define an MDP. Actions are chosen according to a policy, $\pi \in \Pi$, where Π is the set of all mappings from S to A. Solving the MDP means to identify an optimal policy, $\pi^* \in \Pi$, that maximizes the reward received over an infinite horizon. The optimal policy π^* minimizes the long-run maintenance cost, which we formulate as a maximization problem where all rewards in the MDP are negative.

The reward function is the same for both the TBM and the CBM model and it is defined by

$$r(\mathbf{s},\mathbf{a}) = \sum_{i=1}^{N} a_i \left(c_p^i \mathbf{1}_{s_i<D} + c_c^i \mathbf{1}_{s_i=D}\right) + c_s \left(1 - \prod_{i=1}^{N}(1-a_i)\right) + c_f \mathbf{1}_{|\{i:s_i<D\}|<K}, \tag{1}$$

where $\mathbf{1}_A$ is the indicator function for event A. The first term of Equation (1), is the replacement costs. The second and third terms account for the setup cost and system failure cost, respectively.

In Sections 2.4 and 2.5 we describe, for each of the two models, how we obtain transition probabilities $p(\mathbf{s}'|\mathbf{s},\mathbf{a})$ from the underlying deterioration process $\mathbf{X}_t$. Since the replacement of a component is assumed to be instantaneous, the replacement action $\mathbf{a}$ instantly moves the system from state $\mathbf{s}$ to a *post-decision* state, $((1-a_1)s_1,\ldots,(1-a_N)s_N) \in S$, that is, the same state only with zero entries for the replaced components. The transition to state $\mathbf{s}'$ then occurs with probability $p(\mathbf{s}'|((1-a_1)s_1,\ldots,(1-a_N)s_N),\mathbf{0})$, where $\mathbf{0} \in A$ is the action of not replacing any components. Let $q(\mathbf{s}'|\mathbf{s}) = p(\mathbf{s}'|\mathbf{s},\mathbf{0})$. It now suffices to specify $q(\mathbf{s}'|\mathbf{s})$ for all $\mathbf{s},\mathbf{s}' \in S$, which is determined from the evolution of $\mathbf{X}_t$.

The process $\mathbf{X}_t$ describes the condition of the unmaintained system, where no replacements are performed. The condition of the maintained system, here denoted $\mathbf{Y}_t = (Y_t^1,\ldots,Y_t^N)$ is the process, with the same increments $\mathbf{Y}_{t_2} - \mathbf{Y}_{t_1} = \mathbf{X}_{t_2} - \mathbf{X}_{t_1}$, $0 \le t_1 < t_2 < \infty$, when no replacements are done in the interval $[t_1,t_2]$, and for which a replacement, say of component i at time t, leads to $Y_t^i = 0$. The time of replacements depends on the policy, π being used. Specifically, at epoch τ, the true system condition is mapped to an element of the discrete MDP state space, $\mathbf{Z}_\tau = (Z_\tau^1,\ldots,Z_\tau^N)$, and $\mathbf{Z}_\tau$ is then mapped to a replacement action via π. The mapping to $\mathbf{Z}_\tau$ is defined differently

for the CBM model and the TBM model. In Section 2.6, we discuss the implications of using $\mathbf{Z}_\tau$ to make decisions.

2.4. CBM model

For the CBM model, a state $\mathbf{s} \in S$ is the condition information from the process $\mathbf{Y}_t$, only in a discretized form. We could formulate the MDP using the same continuous state space as the process $\mathbf{Y}_t$. Such a model can be useful if the objective is to prove that the optimal policy has a specific structure, which is done in Elwany et al. (2011); Chen et al. (2015); Sun et al. (2018); Özekici (1988). Identifying the optimal policy in an MDP with a continuous state-space is equivalent to solving a nonlinear functional equation, namely the Bellman equation. But, even with a characterization of the optimal policy, it is generally impossible to solve this equation analytically (Özekici, 1988). The MDP must therefore be solved using the iterative algorithms in Section 3, but in order to do so, the state space must first be discretized and transition probabilities between the new discrete states must be approximated. Besides the references already mentioned in this paragraph, Andriotis & Papakonstantinou (2019); Nguyen et al. (2019); Olde Keizer et al. (2016) also discretize a continuous deterioration process in order solve an MDP.

Recall that L denotes the failure limit for each of the marginal deterioration processes. We discretize the interval $[0, L)$ into D equally sized intervals $I_k = [kL/D, (k+1)L/D)$, $k = 0, \ldots, D-1$, and let $I_D = [L, \infty)$. We then form a mapping of the the true condition of component i at epoch τ, Y_τ^i, to a discrete state $s_i \in \{0, \ldots, D\}$, by letting $Z_\tau^i = s_i$, if $Y_\tau^i \in I_{s_i}$. Thus, $\mathbf{Y}_\tau \in \mathbf{I_s}$ where $\mathbf{I_s} = I_{s_1} \times \cdots \times I_{s_N}$ is mapped to $\mathbf{Z}_\tau = \mathbf{s}$. In the MDP, we approximate the transition probabilities, $q(\mathbf{s}'|\mathbf{s})$, between the discrete states $\mathbf{s}, \mathbf{s}' \in S$ by

$$q(\mathbf{s}'|\mathbf{s}) = P(\mathbf{X}_{\tau+1} \in \mathbf{I}_{\mathbf{s}'} | X_\tau^i = s_i L/D, i = 1, \ldots, N), \tag{2}$$

i.e., we assume component i is at the left endpoint of interval I_{s_i} at epoch τ.

In the case of independent component deterioration ($\theta = 0$), the probability in Equation (2) factorizes. Let $q_i(s'|s) = P(X_{\tau+1}^i \in I_{s'} | X_\tau^i = sL/D)$, $s, s' \in \{0, \ldots, D\}$, i.e. the probability that component i moves from discrete state s to s'. Then

$$q(\mathbf{s}'|\mathbf{s}) = \prod_{i=1}^{N} q_i(s_i'|s_i). \tag{3}$$

We can calculate $q_i(s_i'|s_i)$ from the distribution function of the marginal process X_t^i. When $\theta > 0$ there is, however, no known analytical expression for the distribution of $\mathbf{X}_t$, so we resort to Monte-Carlo simulation to estimate the probabilities in Equation (2). The details of this procedure are described in Appendix A.1.

2.5. TBM model

A number of studies consider multi-component TBM models similar to the MDP we formulate in this section, namely Haurie & l'Ecuyer (1982); Dekker et al. (1996); Sun et al. (2007); Xia et al. (2008); Barreto et al. (2014). In a TBM model, the element s_i of a state $\mathbf{s} \in S$ represents the age of component i. At each transition, a component can either fail and transition to state D, or age by one time unit, thereby transitioning to state $s_i + 1$. In the mentioned references, the failure probability is assumed to be a known function of the component age. In our model, we construct the failure probabilities from the underlying unobserved deterioration process. That is, $Z_\tau^i = s_i$ if $Y_\tau^i < L$ and the last replacement was at epoch $\tau - s_i$. If the component has failed, $Y_\tau^i \geq L$, or the last replacement was at least D epochs ago, then $Z_\tau^i = D$. Therefore, D is also a truncation point for the maximum age of a component, because the algorithms we describe in Section 3 only work for finite state spaces. We want to set D at a level high enough that the components are very unlikely to reach that age. On the other hand, we do not want the state space to be larger than necessary, so D is set individually for each component, D_i, $i \in \{1, \ldots, N\}$, such that $P(X_{D_i}^i < L)$ is close to zero. The state space of the TBM MDP is therefore $S = \{0, \ldots, D_1\} \times \cdots \times \{0, \ldots, D_N\}$.

We first look at the transition probabilities of a system with independent component deterioration ($\theta = 0$) as these are easier to express. Let $q_i(s'|s)$ denote the transition probability of a single component as in Section 2.4. For the TBM model we define it as

$$q_i(s'|s) = \begin{cases} P(X_{s+1}^i \geq L | X_s^i < L) & s' = D_i, s < D_i - 1 \\ P(X_{s+1}^i < L | X_s^i < L) & s' = s + 1 < D_i \\ 1 & D_i - 1 \leq s \leq s' = D_i \\ 0 & \text{else.} \end{cases} \tag{4}$$

The first line is the probability of a functioning component of age s failing, the second line is the probability of not failing. If component i has age $s = D_i - 1$ or is failed, $s = D_i$, it will be in state D_i at the next epoch with certainty, which is the content of the third line. For $\theta = 0$, the joint transition probability $q(\mathbf{s}'|\mathbf{s})$ can again be calculated using Equation (3).

Now consider $\theta > 0$ and that the system is in (post-decision) state $\mathbf{s} \in S$. We define $F_{\mathbf{s}}, E_{\mathbf{s}} \subseteq \{0, \ldots, N\}$ by $F_{\mathbf{s}} = \{i : s_i = D_i\}$ and $E_{\mathbf{s}} = \{i : s_i = D_i - 1\}$, i.e., the set of failed and almost failed components, respectively. The only possible transitions are to the states $\mathbf{s}' \in S$ for which $F_{\mathbf{s}} \cup E_{\mathbf{s}} \subseteq F_{\mathbf{s}'}$ and $s'_i = s_i + 1$ for $i \in F_{\mathbf{s}'}^{\mathsf{c}}$, where A^{c} denotes the set complement of A. In other words, each working component in state $\mathbf{s}$ has the possibility of failing, and there is one state, $\mathbf{s}'$, for each combination of possible failures. We approximate this transition probability with

$$q(\mathbf{s}'|\mathbf{s}) = P\left(\left(\bigcap_{i \in F_{\mathbf{s}'} \setminus (F_{\mathbf{s}} \cup E_{\mathbf{s}})} Y_{\tau+1}^i \geq L\right) \cap \left(\bigcap_{i \in F_{\mathbf{s}'}^{\mathsf{c}}} Y_{\tau+1}^i < L\right) \,\middle|\, \bigcap_{i \in (F_{\mathbf{s}} \cup E_{\mathbf{s}})^{\mathsf{c}}} Y_{\tau}^i < L\right), \tag{5}$$

where the policy that decides replacement times in $\mathbf{Y}_t$ are described together with the Monte Carlo estimation procedure in Appendix A.2. The set $F_{\mathbf{s}'} \setminus (F_{\mathbf{s}} \cup E_{\mathbf{s}})$ in Equation (5) are the functioning components that fail by exceeding the limit L in the transition to $\mathbf{s}'$. The appearance of this set is because we do not require the components in $F_{\mathbf{s}} \cup E_{\mathbf{s}}$ to exceed the failure limit L as they are by definition already certain to be in the failed state after the transition. The set $F_{\mathbf{s}'}^{\mathsf{c}}$ are the components that are still functioning in state $\mathbf{s}'$. In Section 2.6 we elaborate on why Equation (5) is only an approximation and the implications for the policy we obtain by solving the MDP.

2.6. Markov properties of induced stochastic processes

In an MDP, any policy, $\pi : S \to A$, induces a Markov chain, $\mathbf{S}_\tau$, $\tau = 0, 1 \ldots$, on the set of states S, where $\mathbf{S}_\tau$ is the state at epoch τ. Specifically, $\mathbf{S}_\tau$ has the Markov property since $P(\mathbf{S}_{\tau+1} = \mathbf{s}'|\mathbf{S}_\tau = \mathbf{s}) = p(\mathbf{s}'|\mathbf{s}, \pi(\mathbf{s}))$, which does not depend on $\mathbf{S}_{\tau-1}, \ldots, \mathbf{S}_0$. Recall that $\mathbf{Z}_\tau$ is the stochastic process obtained by mapping the true system condition, $\mathbf{Y}_\tau$, to the MDP state space S. The transition probabilities, $p(\mathbf{s}'|\mathbf{s}, \pi(\mathbf{s}))$ defined in Equations (2) and (5), are approximations of the probabilities $P(\mathbf{Z}_{\tau+1} = \mathbf{s}'|\mathbf{Z}_\tau = \mathbf{s})$, $\mathbf{s}, \mathbf{s}' \in S$; thus, the processes $\mathbf{Z}_\tau$ and $\mathbf{S}_\tau$ have different properties. This is demonstrated in Sections 2.6.1 and 2.6.2 through examples showing that $\mathbf{Z}_\tau$ is not necessarily Markovian.

2.6.1. TBM

Consider the case of a two-component system where the policy is to never replace any components. Suppose we have observed the following: At the beginning of the previous epoch, both components were new, $\mathbf{Z}_{\tau-1} = (0, 0)$, and at the current epoch the first component has failed while the second has aged by one, $\mathbf{Z}_\tau = (D_1, 1)$. The failure is caused by a large increment,

$Y_\tau^1 - Y_{\tau-1}^1$, in the underlying deterioration process. Since this is correlated with $Y_\tau^2 - Y_{\tau-1}^2$, knowing $\mathbf{Z}_{\tau-1}$ provides additional information about how likely the second component is to fail before the next epoch, $\tau + 1$, so it is possible that $P(\mathbf{Z}_{\tau+1} = (D_1, D_2)|\mathbf{Z}_\tau = (D_1, 1))$ and $P(\mathbf{Z}_{\tau+1} = (D_1, D_2)|\mathbf{Z}_\tau = (D_1, 1), \mathbf{Z}_{\tau-1} = (0, 0))$ are not equal.

The implication is that in a multi-component system with dependent deterioration increments among components, the globally optimal TBM policy is a history-dependent policy (Puterman, 2005). This means that actions are chosen based on the entire history of observed states, and finding the optimal policy within this class is generally computationally intractable. The MDP we propose in Section 2.5 is therefore only an approximate model. Nonetheless, we include it in the numerical study for two reasons: First, the policy obtained from solving the MDP is at least as good as any heuristic policy that is contained in the set of Markovian policies, e.g., age-based and block-replacement policies. Secondly, from a computational aspect the case with dependent components is interesting as it represents the situation that all transition probabilities in the MDP have to be estimated in advance, which requires additional memory. Finally, it is important to notice that when components deteriorate independently, the issues regarding the Markov property disappear, since the MDP transition probabilities in Equation (4) are not approximations as in Equation (5) but exact.

2.6.2. CBM

In the CBM case, it is possible to construct a two-component example similar to that in Section 2.6.1, to show that the process $\mathbf{Z}_\tau$ is non-Markovian. However, this is also true even in a single-component system. Suppose we observe $\mathbf{Z}_{\tau-2} = 0$, $\mathbf{Z}_{\tau-1} = 0$, and $\mathbf{Z}_\tau = 0$, which from the definitions of $\mathbf{Z}_\tau$ and S imply that $Y_0^1, Y_1^1, Y_2^1 \in [0, L/D)$. Knowing that Y_τ^1 spent two periods in $[0, L/D)$ provides additional information about whether it will stay in this interval, so in general $P(\mathbf{Z}_{\tau+1} = 0|\mathbf{Z}_\tau = 0)$ and $P(\mathbf{Z}_{\tau+1} = 0|\mathbf{Z}_\tau = \mathbf{Z}_{\tau-1} = \mathbf{Z}_{\tau-2} = 0)$ are not equal. As the example illustrates, the non-Markovian behavior of the observed MDP states, $\mathbf{Z}_\tau$, stems from the discretization of the continuous state space. As we increase the number intervals, D, the error introduced by the discretization diminishes, hence the policy we obtain from the MDP will approach a globally optimal CBM policy.

3. Solution methods

Our overall goal is to solve as large instances of the MDPs formulated in Section 2 as possible. To solve the MDPs, we use iterative DP algorithms. There are two different avenues of optimization for these algorithms: Lowering the required number of iterations, and optimizing the speed of the calculations within each iteration. The latter mainly revolves around how transition probabilities are handled in the implementation. Details regarding this are provided in Section 3.7.

The former can be achieved by choosing the best algorithm configuration from a toolbox of methods. These are presented in Sections 3.1-3.6. The three main algorithms, Value Iteration (VI), Policy Iteration (PI), and Modified Policy Iteration (MPI) are briefly reviewed in Sections 3.1-3.3. It is generally impossible to know in advance, which of the three algorithms is better for a particular MDP. Complexity results for MPD algorithms are surveyed in Littman et al. (1995), however, the authors conclude with the statement that these results are of marginal use to practitioners. In practice, the algorithms are usually much faster than their theoretical worst-case run times (Sutton & Barto, 2018). The empirical approach we employ is therefore justified. Besides iterative algorithms, optimal policies can also be obtained via linear programming, but this method becomes impractical already at much smaller problem sizes (Sutton & Barto, 2018), and is therefore not included.

We solve the infinite-horizon version of the problems using the expected total discounted reward optimality criterion. That is, from the set of all mappings from S to A, Π, we seek to find the optimal policy, $\pi^* \in \Pi$, satisfying $v_{\pi^*}(\mathbf{s}) = \max_{\pi \in \Pi} v_\pi(\mathbf{s})\ \forall \mathbf{s} \in S$, where $v_\pi : S \to \mathbb{R}$ is called the value function and where $v_\pi(\mathbf{s})$ is the expected total discounted reward when starting in state $\mathbf{s} \in S$ and following policy $\pi \in \Pi$. The optimal policy is found by solving the Bellman equations,

$$v(\mathbf{s}) = \max_{\mathbf{a} \in A} \left\{ r(\mathbf{s}, \mathbf{a}) + \gamma \sum_{\mathbf{s}' \in S} p(\mathbf{s}'|\mathbf{s}, \mathbf{a}) v(\mathbf{s}') \right\} \quad \forall \mathbf{s} \in S, \tag{6}$$

where $0 \leq \gamma < 1$ is the discount factor.

3.1. Value Iteration

The standard value iteration algorithm is an iterative algorithm that in each iteration, n, updates an estimate of the value function, $v_n \in \mathbb{R}^{|S|}$. Starting with an arbitrary v_0 the updates

are computed by

$$v_{n+1}(\mathbf{s}) = \max_{\mathbf{a}\in A} \left\{ r(\mathbf{s},\mathbf{a}) + \gamma \sum_{\mathbf{s}'\in S} p(\mathbf{s}'|\mathbf{s},\mathbf{a}) v_n(\mathbf{s}') \right\} \quad \forall \mathbf{s} \in S. \tag{7}$$

When Equation (7) is used repeatedly $v_n(\mathbf{s}) \to v_{\pi^*}(\mathbf{s})$ as $n \to \infty$, see Theorem 6.3.1 in Puterman (2005). When the value function converges, the final policy is obtained from the maximizing actions in the last use of Equation (7). We elaborate more on convergence in Section 3.5.

3.2. Policy Iteration

PI is based on two fundamental steps: *Policy evaluation* in which the value function of the current best policy is computed, and *Policy improvement*, in which the policy is improved based on the recently computed value function.

Let π_n denote the policy in iteration n, and let v_n denote the value function that corresponds to policy π_n. The purpose of the first step of PI is to derive v_n by solving the linear system $(I - \gamma P_{\pi_n})v_n = r_{\pi_n}$. Here, $r_{\pi_n} \in \mathbb{R}^{|S|}$ is the reward vector with elements $r(\mathbf{s}, \pi_n(\mathbf{s}))$ and $P_{\pi_n} \in \mathbb{R}^{|S|\times|S|}$ is the transition probability matrix with elements $p(\mathbf{s}'|\mathbf{s}, \pi_n(\mathbf{s}))$, $\mathbf{s}, \mathbf{s}' \in S$. Subsequently, the second step of PI finds an improved policy by applying Equation (7) and assigning the maximizing actions to π_{n+1}. The algorithm is initiated with an arbitrary $\pi_0 \in \Pi$ and continues until an improving policy can no longer be obtained — i.e., when $\pi_{n+1} = \pi_n$. Because the problem instances we wish to solve are quite large, we solve the linear system $(I - \gamma P_{\pi_n})v_n = r_{\pi_n}$ using iterative methods. Letting $\hat{v}_0$ be the value function after using Equation (7) in the improvement step we then produce iterates $\hat{v}_k$ that approach v_n as $k \to \infty$ in the following manner,

$$\hat{v}_{k+1}(\mathbf{s}) = r(\mathbf{s}, \pi_n(\mathbf{s})) + \gamma \sum_{\mathbf{s}'\in S} p(\mathbf{s}'|\mathbf{s}, \pi_n(\mathbf{s}))\hat{v}_k(\mathbf{s}') \quad \forall \mathbf{s} \in S. \tag{8}$$

In practice, the required number of policy improvement steps can be quite low, and if this is the case, PI is often superior to VI. However, for some MDPs it may be inefficient to perform an exact policy evaluation at each iteration, in particular if the number of actions is low, because the VI update in Equation (7) is then not much more expensive than the policy evaluation step in Equation (8). The next algorithm can be seen as a combination of VI and PI and attempts to incorporate the advantages of both algorithms.

3.3. Modified Policy Iteration

We do not require an exact estimate of the value function v_n to find a better policy in the improvement step of PI. If we terminate the policy evaluation procedure in Equation (8) prematurely, we get the MPI algorithm (Puterman, 2005). The evaluation step is now referred to as partial evaluation, and in our implementation we stop at $k = m$ or when the sequence $\hat{v}_k$ converges to v_n, whichever comes first. The optimal iteration limit, $m \in \mathbb{N}$, is determined experimentally. The MPI algorithm terminates when the value function converges upon being updated to $\hat{v}_0$ in the improvement step.

3.4. Update schemes

The updates of the value function in Equations (7) and (8) will be referred to as standard (STD) in Section 4. Besides this, we test two additional update schemes, namely the Gauss-Seidel (GS) method and Successive Over-Relaxation (SOR). Both schemes speed up the convergence of the algorithms by inserting updated values $\hat{v}_{k+1}(\mathbf{s})$ ($v_{n+1}(\mathbf{s})$) into Equation (8) (Equation (7)) as soon as they become available. The SOR method involves a relaxation parameter, $1 \leq \omega < 2$, and the best value must be determined experimentally.

3.5. Stopping criteria

All three algorithms that were presented in Sections 3.1-3.3 require a definition of convergence to be able to stop. In this study, we test two different methods: The supremum norm, $\|v_{n+1} - v_n\| = \max_{\mathbf{s}\in S} |v_{n+1}(\mathbf{s}) - v_n(\mathbf{s})|$, and the span seminorm, $sp(v_{n+1} - v_n) = \max_{\mathbf{s}\in S}\{v_{n+1}(\mathbf{s}) - v_n(\mathbf{s})\} - \min_{\mathbf{s}\in S}\{v_{n+1}(\mathbf{s}) - v_n(\mathbf{s})\}$. Let $\epsilon > 0$ denote a tolerance parameter. When using the supremum norm, we stop the algorithm when $\|v_{n+1} - v_n\| < \epsilon(1-\gamma)/2\gamma$ and for the span seminorm we stop when $sp(v_{n+1} - v_n) < \epsilon(1-\gamma)/\gamma$. This ensures that $\|v_{\pi_{n+1}} - v_{\pi^*}\| < \epsilon$, see Theorem 6.3.1 and Proposition 6.6.5 in Puterman (2005). The span criterion is more sensitive and often terminates the algorithm much earlier. However, it does not apply to the GS and SOR update, which is why we consider both criteria.

3.6. Initializations

The initialization v_0, and π_0 in PI and MPI, affects the number of iterations that are required for algorithms to terminate. In order to assess the effect of the initialization, we consider three

different strategies. First, an initialization above the optimal value function, $v_{\pi^*} \leq v_0$, which is simply a zero initialization $v_0(\mathbf{s}) = 0$ and $\pi_0(\mathbf{s}) = \mathbf{0}$ for all $\mathbf{s} \in S$. Secondly, an initialization below the optimal value function, $v_{\pi^*} \geq v_0$, which is formed by setting $\pi_0(\mathbf{s}) = \text{argmax}_{\mathbf{a}}\{r(\mathbf{s}, \mathbf{a})\}$ and $v_0(\mathbf{s}) = r(\mathbf{s}, \pi_0(\mathbf{s})) + \gamma(1-\gamma)^{-1} \min_{\mathbf{s}}\{r(\mathbf{s}, \pi_0(\mathbf{s}))\}$ for all $\mathbf{s} \in S$. Thirdly, we also use a random initialization where $\pi_0(\mathbf{s})$ is sampled randomly from A and $v_0(\mathbf{s})$ is sampled randomly between the upper and lower bound from the other two initializations.

3.6.1. Multigrid algorithm

In the CBM model we also consider a multigrid algorithm similar to that in Chow & Tsitsiklis (1991), where the MDP is solved multiple times for successively finer discretization levels. We first solve the MDP with a small value of D (coarse grid), and use the resulting value function, v^D, and policy, π^D, to initialize the algorithm for solving the MDP with a $2D$ discretization (finer grid), that is, where the length of each interval $I_k \subset [0, L)$ defined in Section 2.4 has been halved. More formally, just as we defined S and the regions $\mathbf{I}_{\mathbf{s}} \subset [0, L)^N$ for $\mathbf{s} \in S$ for the discretization level D, we let S' and $\mathbf{I}'_{\mathbf{s}'}$ for $\mathbf{s}' \in S'$, be defined correspondingly for the discretization level 2D. Then for each $\mathbf{s}' \in S'$ there is exactly one state $\mathbf{s} \in S$ such that $\mathbf{I}'_{\mathbf{s}'} \subset \mathbf{I}_{\mathbf{s}}$, and we initialize the MDP for 2D discretization with $v_0(\mathbf{s}') = v^D(\mathbf{s})$ and $\pi_0(\mathbf{s}') = \pi^D(\mathbf{s})$. After solving this MDP, the whole process is repeated until a policy for a sufficiently fine discretization has been obtained.

3.7. Implementation details

In this section we describe different strategies for handling transition probabilities in the algorithm implementation, as this is the main performance bottleneck when solving large MDPs.

3.7.1. Store in memory

The most efficient method in terms of speed is to calculate all transition probabilities $q(\mathbf{s}'|\mathbf{s})$, for all $\mathbf{s}', \mathbf{s} \in S$ where $q(\mathbf{s}'|\mathbf{s}) > 0$, and store them in memory before we run the algorithm. For the TBM model the memory requirement for this is in the order of $|S|2^N$ because it requires storing a probability for each combination of failures of the N components for each state $\mathbf{s} \in S$. For the CBM model the number of nonzero transition probabilities is in the order of $|S|^2$. However, since the deterioration process $\mathbf{X}_t$ is assumed to have stationary increments, we can choose to only store $|S|$ probabilities at the expense of additional computations. For $\mathbf{u} \in S$, $q(\mathbf{u}|\mathbf{0}) = P(\mathbf{X}_1 \in \mathbf{I}_{\mathbf{u}})$ is the

probability that each component advances u_i discrete states in one transition. We define the set

$$U_{\mathbf{s},\mathbf{s}'} = \{\mathbf{u} \in S : u_i + s_i \geq D \text{ if } s_i \leq s'_i = D \text{ and } u_i + s_i = s'_i \text{ if } s_i \leq s'_i < D\}, \tag{9}$$

which are the possible increments that moves us from state $\mathbf{s}$ to $\mathbf{s}'$. Note that if $s'_i = D$, then any increment $u_i \in \{0, \ldots, D\}$ such that $s_i + u_i \geq D$ will result in $s'_i = D$. The probability $q(\mathbf{s}'|\mathbf{s})$ can now be calculated at runtime as the sum

$$q(\mathbf{s}'|\mathbf{s}) = \sum_{\mathbf{u} \in U_{\mathbf{s},\mathbf{s}'}} q(\mathbf{u}|\mathbf{0}), \tag{10}$$

provided we calculate and store $q(\mathbf{s}|\mathbf{0})$ for all $\mathbf{s} \in S$ before running the algorithm. The simulation procedure for estimating $q(\mathbf{s}|\mathbf{0})$ is provided in Appendix A.1.

3.7.2. Calculate at runtime

At some point, when the number of components is large enough, storing all probabilities is impossible. Whether or not we can still solve the MDP depends on how fast we can compute the transition probabilities as they appear in the algorithm at runtime. For the case of dependent components, we would have to do this with Monte-Carlo estimation, which is extremely slow. In the case of independent components it can be done quite efficiently using Equation (3), and it only requires storing $q_i(s'_i|s_i)$ for each $s', s \in \{0, \ldots, D\}$ and $i \in \{1, \ldots, N\}$ which is $(D+1)^2 N$ probabilities.

When updating the value function at a system state $\mathbf{s} \in S$, as in e.g. Equation (7), we need the transition probabilities for all states $\mathbf{s}, \mathbf{s}' \in S$ for which $q(\mathbf{s}'|\mathbf{s}) > 0$. Suppose $q(\mathbf{s}'|\mathbf{s}) > 0$, $q(\mathbf{s}''|\mathbf{s}) > 0$, and that $s'_j \neq s''_j$ for some j but $s'_i = s''_i$ for $i \neq j$. If we have already calculated $q(\mathbf{s}'|\mathbf{s})$ then we have $q(\mathbf{s}''|\mathbf{s}) = q(\mathbf{s}'|\mathbf{s}) q_j(s''_j|s_j)/q_j(s'_j|s_j)$, which is faster than calculating Equation (3) from scratch. In our implementation, we use a lexicographical ordering of the states in S to systematically go through all possible transitions knowing which components are identical in $\mathbf{s}'$ and $\mathbf{s}''$, and thereby avoid a substantial number of unnecessary operations.

4. Numerical study

The difficulty of solving the TBM and CBM models described in Section 2 might depend on the parameters of the system. For each number of components, N, we therefore consider three levels of component dependency: Independence ($\theta = 0$), weak dependence ($\theta = 0.2$), and strong

dependence ($\theta = 3.0$). For each combination of θ and N we then generate 30 system instances with varying deterioration and cost parameters chosen as follows: K is uniformly distributed on $\{1, \ldots, N\}$, $L = 1$, $\alpha_i \sim \mathcal{U}(13/8, 15/8)$, $\beta_i \sim \mathcal{U}(25/4, 35/4)$, $c_s = -25 - 5(N - K)$, $c_f = -500 - 500(N - K)$, $c_p^i \sim \mathcal{N}(-6\beta_i/\alpha_i, 5^2)$, $c_c^i \sim \mathcal{N}(-12\beta_i/\alpha_i, 5^2)$, where $\mathcal{U}(a, b)$, and $\mathcal{N}(\mu, \sigma^2)$ denote the uniform distribution on the interval $[a, b]$ and the Normal distribution with mean μ and variance σ^2, respectively. We found these parameter ranges, by experimenting with different configurations for $N = 2$ and $N = 3$, and inspecting the resulting optimal policy. If the parameters are not balanced, the resulting optimal policy might be trivial, e.g. $\pi^*(\mathbf{s}) = \mathbf{0}$ for all $\mathbf{s} \in S$, which can be identified very quickly, and we want to avoid these uninteresting cases.

For each value of N, we now have 90 MDPs for CBM and 90 MDPs for TBM. The transition probabilities in the CBM models are estimated from 10^9 realizations of $\mathbf{X}_1$, and in the TBM models we estimate from 10^8 trajectories of $\mathbf{X}_t$ as described in Appendix A. In the TBM models, the age truncation, D_i, for component i, is set as the lowest integer such that $P(X_{D_i}^i < L) < 10^{-6}$. For the chosen distributions of α_i and β_i, this results in D_i being in the range of 12 to 17.

All numerical experiments were performed on a Huawei XH620 V3 server node, which has two Intel Xeon Processor 2660v3 with ten 2.60GHz cores each (we only utilize one core unless explicitly mentioned) and 128GB RAM. All the algorithms were implemented in C++.

4.1. Algorithm comparison

In this section we test each configuration of the solution methods presented in Sections 3.1-3.6. The objective is to identify a configuration that is consistently fast across all system instances.

We test all feasible configurations of algorithm (VI, PI, MPI), value function update (STD, GS, SOR), stopping criterion (supremum, span), MPI iteration limit $m \in \{10, 20, \ldots, 100\}$, SOR relaxation $\omega \in \{1.0, 1.1, \ldots, 1.9\}$, and initialization (above, below, and one random per MDP). The number of feasible combinations is 468 for the TBM model, and 624 for the CBM model because the CBM model also includes the multigrid initialization method. The discount factor and the tolerance parameter are set to $\gamma = 0.99$ and $\epsilon = 0.001$, respectively.

Testing all configurations is too time consuming for the MDPs where $|S|$ is large. Initially we therefore focus on a set of smaller MDPs, where the fastest algorithm configuration for each MDP uses between 10 seconds and 250 seconds to solve it. In the TBM case, we have 51 MDPs with $N = 4$ and $|S|$ between 57344 and 143640. In the CBM case, we have 318 MDPs with combinations

of $N \in \{2, 3, 4, 5\}$ and $D \in \{4, 6, 8, 12, 16, 24\}$ resulting in $|S|$ between 2197 and 15625.

Table 1 shows the best performing configurations, and demonstrates that the fastest configuration is not the same for all MDPs. In the TBM case, no configuration is the fastest in more than 7 out of the 51 MDPs. Furthermore, any one configuration is on average at least 30% slower when comparing its runtime to the lowest runtime among all configurations. We also note that among the fastest configurations for each MDP, the initialization methods "below", "above", and "random" were present 68%, 28%, and 4% of the time, respectively. Furthermore, for the MPI-m-STD-span-(...) configurations the average increase in runtime from the worst initialization to the best initialization increases with m, from 9% at $m = 20$ to 21% at $m = 90$. Taking all of this into account, we choose the **MPI-60-STD-span-below** configuration when solving larger system instances in the Section 4.3.

In the CBM case, we pick the **MPI-20-STD-span-multigrid** configuration for solving large MDPs. For all combinations of MDP and algorithm configuration, the multigrid initialization is the fastest 74% of the time. In these 74% of the combinations, the second fastest initialization is 60% slower on average. In comparison, when the multigrid initialization is not the fastest, it is 17% slower on average. In 7 out of the 318 CBM MDPs a configuration using the SOR update scheme was the fastest. In these 7 cases, the MPI-20-STD-span-multigrid configuration is 78% slower on average, so for a given set of system parameters there is a small chance that an SOR configuration will be most effective. However, in many of the MDPs the SOR configurations do not converge, and when they do, they are between 8 and 16 times slower than the fastest configuration.

4.2. CBM discretization

When we constructed the MDP for the CBM case in Section 2.4, we discretized the continuous deterioration process. Let π_D^* denote the policy we obtain from solving the CBM MDP with D discretization intervals. This policy is optimal w.r.t. the discretized deterioration process, and by choosing a large enough D, π_D^* will also be near-optimal w.r.t. the original continuous-state process. In this section, MDPs are solved using a discount factor $\gamma = 0.99$. A low tolerance, ϵ, is redundant if we cannot solve the problem for a sufficiently large value of D, hence we use a less strict tolerance $\epsilon = 1$.

As a way of assessing how close π_D^* is to being globally optimal, we look at the total discounted reward obtained from simulating the maintained systems, $\mathbf{Y}_t$, when actions are chosen

TBM: 51 MDPs	Configuration	relative runtime	#fastest
Top 3 relative runtime	MPI-80-STD-span-above	+30.0%	0
	MPI-80-STD-span-below	+30.1%	1
	MPI-90-STD-span-below	+30.2%	3
Top 3 #fastest	MPI-40-STD-span-below	+39.0%	7
	MPI-100-SOR-1.0-sup-below	+54.0%	7
	MPI-60-STD-span-below	**+32.0%**	**4**
CBM: 318 MDPs			
Top 3 relative runtime	MPI-30-STD-span-multigrid	+16.0%	35
	MPI-20-STD-span-multigrid	**+16.3%**	**42**
	MPI-40-STD-span-multigrid	+17.3%	15
Top 3 #fastest	MPI-10-STD-span-multigrid	+19.0%	186
	MPI-20-STD-span-multigrid	**+16.3%**	**42**
	MPI-30-STD-span-multigrid	+16.0%	35

Table 1: Best performing algorithm configurations. The abbreviation in the configuration column denotes *algorithm-m-update scheme-stopping criterion-initialization*. The column *relative runtime* is a measure for whether the configuration works well across different system parameter settings. Specifically it is the configuration runtime relative to the lowest runtime among all configurations and then averaged over all the solved MDPs. The column *#fastest* is the number of MDPs for which the configuration had the lowest runtime.

according to $\pi_D^*(\mathbf{Z}_\tau)$. This is shown in Figure 2 for $N = 4$ and different discretization levels $D \in \{2, 3, 4, 6, 8, 12, 16, 24\}$. These particular values of D are used as they appear when starting the multigrid procedure with either $D = 2$ or $D = 3$. For each value of D and system instance $j = 1, \ldots, 90$, we let $\bar{v}_j^D$ denote an estimate of the expected total discounted reward when starting with all new components, i.e., in state $\mathbf{Y}_0 = \mathbf{0}$. We calculate $\bar{v}_j^D$ as the average of 10000 realizations of 1000 time steps of the maintained system. We choose this simulation length because after time 1000 all remaining rewards in an infinitely long horizon account for only $\gamma^{1000}/(1-\gamma) \approx 0.5\%$ of the total. The 10000 realizations result in standard errors of $\bar{v}_j^D$ between 3.4 and 19.6 where the 95th percentile is 11.3.

As Figure 2 indicates, in all the 90 instances of 4-component systems the performance of π_D^* does not improve beyond $D = 16$. For a similar plot of the $N = 5$ systems, we are not able to conclude the same, since the largest value of D we are able to solve in less than 100 hours is $D = 12$. Instead, we compare the incremental improvement of $\bar{v}_j^D$ for different N. Define $\delta^{D \to D'}$, $D < D'$, as the mean relative increase in the estimated total discounted reward when using policy

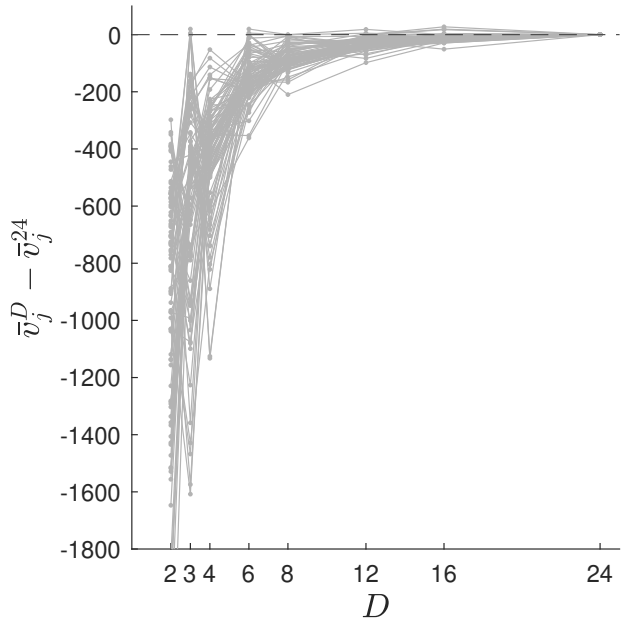

Figure 2: Estimated expected total discounted reward for all $j = 1, \ldots, 90$ system instances with $N = 4$ components for different discretization levels, D. The solid lines are the estimates $\bar{v}_j^D$ shifted by $\bar{v}_j^{24}$ in order to show simultaneously for all j how the total reward increases when increasing D.

$\pi^*_{D'}$ instead of policy π^*_D, that is

$$\delta^{D \to D'} = \frac{1}{90} \sum_{j=1}^{90} \frac{\bar{v}_j^D - \bar{v}_j^{D'}}{\bar{v}_j^D}. \tag{11}$$

Table 2 shows the rate at which $\delta^{D \to D'}$ tends towards zero for different values of N. Seeing as $\delta^{12 \to 16}$ is well below one percent for $N \leq 4$, the expected total discounted reward from using π^*_D is already close to the asymptote value when $D = 12$. Furthermore, when solving $N = 2$ and $N = 3$ with $D = 64$ we get an estimated $\delta^{12 \to 64}$ of 0.48%(0.28%) and 0.75%(0.19%), respectively. This is a strong indication that the performance improvement beyond $D = 12$ is very small. A reasonable conjecture could be that a higher number of components requires a finer discretization. However, this does not appear to be the case, since for all values of N the incremental improvements decay at the same rate up to $D = 12$.

Olde Keizer et al. (2016) presents a two-component example with $D = 48$, Elwany et al. (2011) uses $D = 20$ for a single-component system, Andriotis & Papakonstantinou (2019) uses $D = 24$ for a 25-component system (solved approximately), and in Sun et al. (2018) the authors suggest using a D such that the interval length is in the same order of magnitude as the precision of

N	2	3	4	5
$\delta^{2\to3}$	2.11%(7.77%)	4.82%(8.27%)	5.28%(9.63%)	6.63%(8.37%)
$\delta^{3\to4}$	2.44%(8.33%)	1.70%(8.06%)	1.67%(8.53%)	1.43%(7.07%)
$\delta^{4\to6}$	2.63%(3.80%)	4.11%(3.87%)	5.04%(4.46%)	5.10%(3.83%)
$\delta^{6\to8}$	1.45%(2.27%)	1.18%(1.67%)	1.17%(1.60%)	1.52%(1.45%)
$\delta^{8\to12}$	0.97%(1.37%)	0.87%(0.78%)	0.97%(0.80%)	1.04%(0.72%)
$\delta^{12\to16}$	0.15%(0.46%)	0.31%(0.46%)	0.36%(0.32%)	-
$\delta^{16\to24}$	0.19%(0.25%)	0.23%(0.33%)	0.20%(0.21%)	-

Table 2: The mean relative increase in the estimated total discounted reward for increasing discretization levels. Standard deviations are shown in parentheses.

the sensors that measure the degradation level. Undoubtedly, the coarsest discretization level for which a near-optimal policy can be obtained depends on which deterioration process is assumed. The gamma process is arguably the most commonly used deterioration process in maintenance optimization literature. As our results indicate, for this deterioration process, setting D lower than in the aforementioned studies is adequate. Indeed, by replicating the gamma process example from Olde Keizer et al. (2016) with different values of D, we find that $D = 9$ results in a policy with the same performance as when $D = 48$.

4.3. Runtime and memory

In this section we solve the models with as many components as possible based on the hardware we use. The TBM MDPs are solved with the MPI-60-STD-span-below configuration and $\epsilon = 0.001$. The CBM MDPs are solved using the MPI-20-STD-span-multigrid configuration with $\epsilon = 1$ and $D = 12$. The results are summarized in Table 3, which shows the runtime and memory usage for the different transition probability storage methods in Section 3.7.

In all cases where it is possible to store all nonzero $q(\mathbf{s}'|\mathbf{s})$, $\mathbf{s}', \mathbf{s} \in S$, we can solve the MDP within a relatively short amount of time. The TBM models are faster to solve than the CBM models because the number of nonzero probabilities out of all the $|S|^2$ possible combinations of $\mathbf{s}'$ and $\mathbf{s}$ is smaller in the TBM models, as shown in Table 4. It is worth noting that when storing all probabilities, the CBM MDPs for which components are strongly dependent ($\theta = 3$) have a lower runtime and use less memory compared to $\theta = 0.2$ and $\theta = 0$. The reason is that deterioration increments, where some components have large jumps and others do not, are very unlikely and

		N			
TBM		4	5	6	7
All θ	store $q(\mathbf{s}'\|\mathbf{s})$	5sec(3) 21MB(2)	3min(1.5) 394MB(58)	1.6hr(0.8) 9GB(2)	- $>$ 128GB
$\theta = 0$	calculate $q(\mathbf{s}'\|\mathbf{s})$	10sec(5) 8MB(0.4)	7min(4) 67MB(10)	4.5hr(2.4) 1GB(0.1)	*47hr(2) *4GB(0.3)
CBM					
$\theta = 3$	store $q(\mathbf{s}'\|\mathbf{s})$	30sec(10) 700MB(80)	1hr(0.5) 15GB(1)	- $>$ 128GB	- $>$ 128GB
	store $q(\mathbf{s}'\|\mathbf{0})$	8min(2) 8MB(0)	57hr(21) 61MB(0)	$>$ 1week* -	- -
$\theta = 0.2$	store $q(\mathbf{s}'\|\mathbf{s})$	2min(0.3) 2GB(0)	9hr(3) 80GB(0.2)	- $>$ 128GB	- $>$ 128GB
	store $q(\mathbf{s}'\|\mathbf{0})$	8min(3) 8MB(0)	60hr(19) 61MB(0)	$>$ 1week* -	- -
$\theta = 0$	store $q(\mathbf{s}'\|\mathbf{s})$	2min(0.3) 2GB(0.3)	9hr(2) 80GB(0.1)	- $>$ 128GB	- $>$ 128GB
	calculate $q(\mathbf{s}'\|\mathbf{s})$	4min(1) 8MB(0)	26hr(4) 55MB(0)	$>$ 1week* -	- -

Table 3: Runtime and used memory, for each transition probability storage method from Section 3.7. Each cell is the mean over the 30 different MDPs for either $\theta = 0$, $\theta = 0.2$, or $\theta = 3$, except the first TBM row which includes all 90 MDPs. Parentheses denote standard deviations. (*Parallel implementation using 10 cores.)

therefore estimated to have zero probability. However, the benefit in terms of memory is limited since storing all nonzero $q(\mathbf{s}'|\mathbf{s})$ is still only feasible up to five components.

In the case of independent component deterioration ($\theta = 0$), calculating the probabilities at runtime is at most 3 times slower than storing all probabilities. Considering that adding one component generally increases the runtime by a factor of 40 for TBM and 300 for CBM, this is a relatively small difference. Furthermore, the low memory requirements of calculating probabilities at runtime enable us to attempt solving MDPs with more components. Our serial implementation is too slow for $N = 6$ in CBM and $N = 7$ in TBM. Table 3 includes the runtimes of a parallel implementation that divides Equations (7) and (8) between 10 CPU cores when these are calculated for each $\mathbf{s} \in S$. This implementation solves the TBM MDPs with $\theta = 0$ and $N = 6$, in an average

	N			
TBM	4	5	6	7
$\lvert S\rvert$	$5 \cdot 10^4$	$9 \cdot 10^5$	$1 \cdot 10^7$	$2 \cdot 10^8$
Nonzero	$1 \cdot 10^{-4}$	$2 \cdot 10^{-5}$	$2 \cdot 10^{-6}$	$4 \cdot 10^{-7}$
CBM				
$\lvert S\rvert$	$3 \cdot 10^4$	$4 \cdot 10^5$	$5 \cdot 10^6$	$6 \cdot 10^7$
Nonzero ($\theta < 3$)	0.08	0.05	0.02	
Nonzero ($\theta = 3$)	0.03	0.01		

Table 4: Mean state space sizes $|S|$ and the mean fraction of transition probabilities that are nonzero, i.e., $|\{(\mathbf{s}', \mathbf{s}) : \mathbf{s}', \mathbf{s} \in S, q(\mathbf{s}', \mathbf{s}) > 0\}|/|S|^2$. The missing entries take more than one week to calculate.

of 34 minutes, which is an eightfold speedup compared to the 4.5 hour runtime of the serial version. For the CBM MDPs where $\theta = 0$ and $N = 5$, the 10 cores use 2.6 hours on average, which is a tenfold speedup. Even so, we where not able to solve the $N = 6$ CBM instance within a one-week time limit. It is possible that instances with more components can be solved with more computer resources, given that the parallel algorithm scales linearly up to 10 cores. At some number of components it does, however, become difficult to even store the solution to the optimization problem. For instance, the 128GB available memory allows us to store a value function in an MDP with $|S| = 3.4 \cdot 10^{10}$ using single-precision floating point numbers. This corresponds to 9 components for the values of D and D_i we use in our numerical experiments.

Finally we note that our parallel implementation does not scale as well for the CBM MDPs where $\theta = 0.2$ and $\theta = 3$. It solves the $N = 5$ instances in an average of 11 hours, which is only a 5-fold speedup. Combined with the fact that the TBM models, where $\theta > 0$, require a lot of memory, we consider it infeasible to solve larger models with copula dependence among components. There are, however, other ways of modeling stochastic dependence, that allows for efficient runtime calculations of transition probabilities. In the TBM model in Barreto et al. (2014) and the CBM model in Olde Keizer et al. (2018), stochastic dependence is modeled such that the failure rate and deterioration increments are conditionally independent given the current state of the system. Hence, transition probabilities can be written on the form

$$q(\mathbf{s}'|\mathbf{s}) = \prod_{i=1}^{N} q_i(s_i'|\mathbf{s}). \tag{12}$$

This equation resembles Equation (3), and we can calculate transition probabilities at runtime

using a procedure similar to the one in Section 3.7.2.

5. Conclusion

In this paper we consider the optimization of component replacements in multi-component systems. We find that other works in maintenance literature assume at least one of the following: discrete system dynamics, single-component system, or a heuristic optimization approach. We present a unified view of TBM and CBM in a general setting where all three of these assumptions are relaxed. We do this by formulating MDP models for TBM and CBM based on the same multi-component system where the component deterioration process is continuous. We use DP to compute optimal policies in the MDPs, which are also optimal with respect to the controlled system when deterioration increments are independent among the components.

For the CBM model, discretization of the continuous deterioration process must be employed in order to use DP. The performance of the resulting policies do not improve when the component condition is discretized into more than twelve levels, which is fewer than suggested in previous studies. According to our results, this number is not sensitive to the system parameters or the number of components. This is an important finding, as it indicates that DP is a feasible solution approach in systems with several components.

We investigate the computational limitations of the proposed TBM and CBM models. An efficient implementation of DP algorithms allows us to solve instances that have 200 million states and 400 thousand states for the TBM and CBM model, respectively. The different size limits are a consequence of the number of possible transitions between states, which is inherently different for the two maintenance approaches. Therefore, multi-component CBM problems are generally harder to solve to optimality than multi-component TBM problems. Furthermore, the limiting factor for the TBM model investigated here is the memory requirement of storing the MDP transition probabilities. However, for the special case of independent component deterioration, the transition probabilities can be calculated efficiently at runtime, allowing for larger instances to be solved via parallelization of the DP algorithm.

Heuristic optimization methods are necessary when dealing with industrial systems composed of dozens of components or assets. However, we argue that the MDPs presented in this study provide a more general and flexible class of policies, and though the optimal policy is difficult to

obtain, they serve an important purpose of validating heuristics. Being able to solve large instances to optimality can strengthen statements on how the performance of the heuristics scale with the size of the problem.

Acknowledgments

We would like to express our sincere gratitude to the three anonymous referees for their comments, questions, and suggestions, which have led to major improvements of the content and readability of this paper.

References

Abdel-Hameed, M. (1987). Inspection and maintenance policies of devices subject to deterioration. *Advances in Applied Probability*, *19*, 917–931. doi:10.2307/1427108.

Alaswad, S., & Xiang, Y. (2017). A review on condition-based maintenance optimization models for stochastically deteriorating system. *Reliability Engineering & System Safety*, *157*, 54–63. doi:10.1016/j.ress.2016.08.009.

Andriotis, C. P., & Papakonstantinou, K. G. (2019). Managing engineering systems with large state and action spaces through deep reinforcement learning. *Reliability Engineering & System Safety*, *191*, 106483. doi:10.1016/j.ress.2019.04.036.

Barreto, A. M. S., Pineau, J., & Precup, D. (2014). Policy iteration based on stochastic factorization. *Journal of Artificial Intelligence Research*, *50*, 763–803. doi:10.1613/jair.430.

Boucherie, R. J., & van Dijk, N. M. (Eds.) (2017). *Markov Decision Processes in Practice*. Springer.

Castanier, B., Grall, A., & Bérenguer, C. (2005). A condition-based maintenance policy with non-periodic inspections for a two-unit series system. *Reliability Engineering & System Safety*, *87*, 109–120. doi:10.1016/j.ress.2004.04.013.

Chen, N., Ye, Z.-S., Xiang, Y., & Zhang, L. (2015). Condition-based maintenance using the inverse gaussian degradation model. *European Journal of Operational Research*, *243*, 190–199. doi:10.1016/j.ejor.2014.11.029.

Chow, C.-S., & Tsitsiklis, J. N. (1991). An optimal one-way multigrid algorithm for discrete-time stochastic control. *IEEE transactions on automatic control*, *36*, 898–914.

Dekker, R., Wildeman, R., & van Egmond, R. (1996). Joint replacement in an operational planning phase. *European Journal of Operational Research*, *91*, 74–88. doi:10.1016/0377-2217(94)00363-7.

Derman, C. (1963). On optimal replacement rules when changes of state are markovian. In R. Bellman (Ed.), *Mathematical Optimization Techniques* (pp. 201–210). University of California Press Berkeley, CA.

Elwany, A. H., Gebraeel, N. Z., & Maillart, L. M. (2011). Structured replacement policies for components with complex degradation processes and dedicated sensors. *Operations research*, *59*, 684–695. doi:10.1287/opre.1110.0912.

Grothe, O., & Hofert, M. (2015). Construction and sampling of archimedean and nested archimedean lévy copulas. *Journal of Multivariate Analysis*, *138*, 182–198.

Haurie, A., & l'Ecuyer, P. (1982). A stochastic control approach to group preventive replacement in a multicomponent system. *IEEE Transactions on Automatic Control*, *27*, 387–393. doi:10.1109/TAC.1982.1102919.

Jiang, D. R., & Powell, W. B. (2015). An approximate dynamic programming algorithm for monotone value functions. *Operations Research*, *63*, 1489–1511. doi:10.1287/opre.2015.1425.

Jiang, W., Hong, H., & Ren, J. (2019). Estimation of model parameters of dependent processes constructed using lévy copulas. *Communications in Statistics - Simulation and Computation*, (pp. 1–17). doi:10.1080/03610918.2019.1565581.

de Jonge, B., & Scarf, P. A. (2020). A review on maintenance optimization. *European Journal of Operational Research*, *285*, 805–824. doi:10.1016/j.ejor.2019.09.047.

Kurt, M., & Kharoufeh, J. P. (2010). Monotone optimal replacement policies for a markovian deteriorating system in a controllable environment. *Operations Research Letters*, *38*, 273–279. doi:10.1016/j.orl.2010.03.001.

Li, H., Deloux, E., & Dieulle, L. (2016). A condition-based maintenance policy for multi-component systems with lévy copulas dependence. *Reliability Engineering & System Safety*, *149*, 44–55. doi:https://doi.org/10.1016/j.ress.2015.12.011.

Littman, M. L., Dean, T. L., & Kaelbling, L. P. (1995). On the complexity of markov decision processes. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI-95), Montreal, Québec, Canada*.

Naderkhani ZG, F., & Makis, V. (2015). Optimal condition-based maintenance policy for a partially observable system with two sampling intervals. *International Journal of Advanced Manufacturing Technology*, *78*, 795–805. doi:10.1007/s00170-014-6651-4.

Neves, M. L., Santiago, L. P., & Maia, C. A. (2011). A condition-based maintenance policy and input parameters estimation for deteriorating systems under periodic inspection. *Computers & Industrial Engineering*, *61*, 503–511. doi:10.1016/j.cie.2011.04.005.

Nguyen, K. T., Do, P., Huynh, K. T., Bérenguer, C., & Grall, A. (2019). Joint optimization of monitoring quality and replacement decisions in condition-based maintenance. *Reliability Engineering & System Safety*, *189*, 177 – 195. doi:https://doi.org/10.1016/j.ress.2019.04.034.

Nicolai, R. P., & Dekker, R. (2008). Optimal maintenance of multi-component systems: a review. In K. A. H. Kobbacy, & D. P. Murthy (Eds.), *Complex system maintenance handbook* chapter 11. (pp. 263–286). Springer.

Olde Keizer, M. C. A., Teunter, R. H., & Veldman, J. (2016). Clustering condition-based maintenance for systems with redundancy and economic dependencies. *European Journal of Operational Research*, *251*, 531–540. doi:10.1016/j.ejor.2015.11.008.

Olde Keizer, M. C. A., Teunter, R. H., Veldman, J., & Babai, M. Z. (2018). Condition-based maintenance for systems with economic dependence and load sharing. *International Journal of Production Economics*, *195*, 319–327.

Özekici, S. (1988). Optimal periodic replacement of multicomponent reliability systems. *Operations Research*, *36*, 542–552.

Pineau, J., Gordon, G., & Thrun, S. (2003). Point-based value iteration: An anytime algorithm for pomdps. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence* IJCAI'03 (p. 1025–1030). San

Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Puterman, M. L. (2005). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. (2nd ed.). Wiley. doi:10.1002/9780470316887.

Shi, Y., Feng, Q., Shu, Y., & Xiang, Y. (2020). Multi-dimensional lévy processes with lévy copulas for multiple dependent degradation processes in lifetime analysis. *Quality Engineering*, *32*, 434–448. doi:10.1080/08982112.2020.1757704.

Sun, Q., Ye, Z.-S., & Chen, N. (2018). Optimal inspection and replacement policies for multi-unit systems subject to degradation. *IEEE Transactions on Reliability*, *67*, 401–413. doi:10.1109/TR.2017.2778283.

Sun, T., Zhao, Q., & Luh, P. B. (2007). Incremental value iteration for time-aggregated markov-decision processes. *IEEE Transactions on Automatic Control*, *52*, 2177–2182. doi:10.1109/TAC.2007.908359.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. (2nd ed.). MIT press.

van Oosterom, C., Elwany, A., Çelebi, D., & van Houtum, G. (2014). Optimal policies for a delay time model with postponed replacement. *European Journal of Operational Research*, *232*, 186–197. doi:https://doi.org/10.1016/j.ejor.2013.06.038.

Xia, L., Zhao, Q., & Jia, Q.-S. (2008). A structure property of optimal policies for maintenance problems with safety-critical components. *IEEE Transactions on Automation Science and Engineering*, *5*, 519–531. doi:10.1109/TASE.2007.910763.

Zhang, N., Fouladirad, M., Barros, A., & Zhang, J. (2020). Condition-based maintenance for a k-out-of-n deteriorating system under periodic inspection with failure dependence. *European Journal of Operational Research*, *287*, 159–167. doi:10.1016/j.ejor.2020.04.041.

Appendices

A. Monte Carlo estimation

In both models, we use Algorithm 4.2 from Grothe & Hofert (2015) to sample from the distribution of the one-epoch increments, $\mathbf{X}_1$. The algorithm requires a truncation parameter, K, for the number of jumps to include from an infinite sum of process jumps. For $\theta = 0.2$ we use $K = 4000$ and for $\theta = 3$ we use $K = 30$. These values were determined using the method from Section 5.1 in Grothe & Hofert (2015).

A.1. CBM

When $\theta > 0$, the transition probabilities $q(\mathbf{s}'|\mathbf{s})$, $\mathbf{s}', \mathbf{s} \in S$ are calculated via Equation (10), so we only have to estimate $q(\mathbf{s}|\mathbf{0})$. We do this by sampling M realizations of $\mathbf{X}_1$, denoted $\mathbf{x}_1^k$, $k = 1, \ldots, M$, and then form the estimates as

$$q(\mathbf{s}|\mathbf{0}) = \frac{|\{k : \mathbf{x}_1^k \in \mathbf{I}_\mathbf{s}\}|}{M}, \quad \mathbf{s} \in S. \tag{13}$$

A.2. TBM

When $\theta > 0$, we estimate $q(\mathbf{s}'|\mathbf{s})$ given by Equation (5). This equation is the probability of a specific combination of component failures, conditioned on these components not having failed at the ages given in the vector $\mathbf{s} \in S$. In the following procedure, we therefore account for all the likely combinations of component ages, and for each age combination, then account for each combination of failures.

First, we simulate M trajectories of $\mathbf{X}_\tau$, denoted $\mathbf{x}_\tau^k = (x_\tau^{1,k}, \ldots, x_\tau^{N,k})$, $k = 1, \ldots, M$, $\tau = 0, \ldots, D_{max}$, where $D_{max} = \max_i\{D_i\}$. For each k and τ, we then consider all possible combinations of component ages up to age τ, via all the possible replacement times of each component. With the restriction that each component is replaced exactly once, there are $(\tau+1)^N$ combinations for N components. Let $\mathbf{r} = (r_1, \ldots, r_N) \in \{0, \ldots, \tau\}^N$ be a vector denoting the replacement times of each component. For each k, τ, and $\mathbf{r} \in \{0, \ldots, \tau\}^N$ we can then construct a realization of $\mathbf{Y}_\tau$ were each component i was last replaced $\tau - r_i$ epochs ago. We define these by

$$\mathbf{y}_{\tau,\mathbf{r}}^k = (y_{\tau,\mathbf{r}}^{1,k}, \ldots, y_{\tau,\mathbf{r}}^{N,k}) = (x_\tau^{1,k} - x_{r_1}^{1,k}, \ldots, x_\tau^{N,k} - x_{r_N}^{N,k}), \tag{14}$$

that is, the k'th simulated trajectory at time τ if we replaced components at the times given in $\mathbf{r} \in \{0, \ldots, \tau\}^N$.

Now, let $V_\mathbf{s}$ be the set of tuples, $(k, \tau, \mathbf{r})$, where $\mathbf{y}_{\tau,\mathbf{r}}^k$ corresponds to the MDP being in state $\mathbf{s} \in S$. Furthermore, we let $W_{\mathbf{s},\mathbf{s}'}$ be the set of tuples, $(k, \tau, \mathbf{r})$, where $\mathbf{y}_{\tau,\mathbf{r}}^k$ and $\mathbf{y}_{\tau+1,\mathbf{r}}^k$ corresponds to the MDP being in state $\mathbf{s} \in S$ at time τ and state $\mathbf{s}' \in S$ at time $\tau + 1$. The estimates $q(\mathbf{s}'|\mathbf{s})$ are now calculated as

$$q(\mathbf{s}'|\mathbf{s}) = \frac{|W_{\mathbf{s},\mathbf{s}'}|}{|V_\mathbf{s}|}, \quad \mathbf{s}, \mathbf{s}' \in S. \tag{15}$$

The sets $V_\mathbf{s}$ and $W_{\mathbf{s},\mathbf{s}'}$ are formally defined as

$$\begin{aligned} V_\mathbf{s} = \big\{&(k, \tau, \mathbf{r}) : k \in \{1, \ldots, M\}, \tau \in \{0, \ldots, D_{max}\}, \mathbf{r} \in \{0, \ldots, \tau\}^N, \\ &s_i = \tau - r_i \text{ and } y_{\tau,\mathbf{r}}^{i,k} < L \text{ for } s_i < D_i, \\ &y_{\tau,\mathbf{r}}^{i,k} \geq L \text{ for } s_i = D_i\big\}, \quad \mathbf{s} \in S, \end{aligned} \tag{16}$$

and

$$\begin{aligned} W_{\mathbf{s},\mathbf{s}'} = \big\{&(k, \tau, \mathbf{r}) : (k, \tau, \mathbf{r}) \in V_\mathbf{s}, s_i' = \tau + 1 - r_i \text{ and } y_{\tau+1,\mathbf{r}}^{i,k} < L \text{ for } s_i' < D_i, \\ &y_{\tau+1,\mathbf{r}}^{i,k} \geq L \text{ for } s_i' = D_i\big\}, \quad \mathbf{s}, \mathbf{s}' \in S. \end{aligned} \tag{17}$$