Aalto University
School of Science
Master's Programme in Mathematics and Operations Research

Jussi Leppinen

A Dynamic Optimization Model for Maintenance Scheduling of a Multi-Component System

Master's Thesis
Espoo, May 25, 2020

Supervisor: Prof. Antti Punkka
Advisors: Prof. Antti Punkka
Research Prof. Tommi Ekholm

Aalto University
School of Science
Master's Programme in Mathematics and Operations Research

ABSTRACT OF
MASTER'S THESIS

Author	Jussi Leppinen		
Title	A Dynamic Optimization Model for Maintenance Scheduling of a Multi-Component System		
Major	Systems and Operations Research	**Code**	SCI3055
Supervisor	Prof. Antti Punkka		
Advisors	Prof. Antti Punkka Research Prof. Tommi Ekholm		
Date	May 25, 2020	**Pages**	vi + 68

Technical systems consist of many components, which need maintenance for the system to operate reliably. The maintenance of components usually involves different costs. In addition, economic and structural dependencies can affect the total maintenance costs and the optimal maintenance decisions. Previously efficient maintenance scheduling policies for such systems have been solved with simulation or with rolling horizon approaches that take the reliability of the system poorly into account.

This thesis develops a solution method for optimal maintenance scheduling of a multi-component system with economic and structural dependencies. These dependencies are modeled with a directed graph. We assume pre-defined maintenance instances where we can only replace components. The failures of components are modeled with probability distributions. Every component is critical. These assumptions lead to a discrete time Markov decision process where the state of the system depends on the ages of the components and the failure state of the system. We emphasize system reliability by setting a reliability threshold. The reliability threshold, structural dependencies and the need to replace failed components define feasible maintenance action portfolios. We then apply policy-iteration algorithm to find the cost optimal maintenance portfolio for every state of the system.

When the model is applied to maintenance scheduling problems of different sizes, we notice that solution times of the algorithm depend heavily on the size of the state space. Also, other model parameters, like the set-up cost, can have an impact on the computation time of the policy iteration algorithm. We also apply the model to a case example with four components and show how consideration of different maintenance intervals and reliability thresholds provides valuable decision support for maintenance planning and how our model can lead to cost savings compared to simple heuristic opportunistic policies.

Keywords	Maintenance scheduling, multi-component system, maintenance portfolio, Markov decision process, policy-iteration
Language	English

Aalto-yliopisto
Perustieteiden korkeakoulu
Matematiikan ja operaatiotutkimuksen maisteriohjelma

DIPLOMITYÖN TIIVISTELMÄ

Tekijä	Jussi Leppinen		
Työn nimi	Monikomponenttijärjestelmän huollon ajoittaminen dynaamisella optimointimallilla		
Pääaine	Systeemi- ja operaatiotutkimus	**Koodi**	SCI3055
Valvoja	Prof. Antti Punkka		
Ohjaajat	Prof. Antti Punkka Tutkimusprof. Tommi Ekholm		
Päiväys	25. toukokuuta 2020	**Sivumäärä**	vi + 68

Tekniset järjestelmät koostuvat usein monista komponenteista, jotka tarvitsevat huoltoa, että järjestelmä voi toimia luotettavasti. Komponenttien huoltoon liittyy usein erilaisia kustannuksia. Lisäksi taloudelliset ja rakenteelliset riippuvuudet voivat vaikuttaa huoltokustannuksiin. Aikaisemmin tämäntyylisten järjestelmien huollonajoitusstrategiat on ratkaistu simuloimalla tai liikkuvan horisontin lähestymistavoilla. Ne ottavat jäjestelmän luotettavuuden yleensä huonosti huomioon.

Tässä opinnäytetyössä kehitetään malli huollon aikatauluttamiseksi monikomponenttijärjestelmälle, jolla on taloudellisilla ja rakenteellisilla riippuvuuksia. Näitä riippuvuuksia mallinnetaan suunnatulla graafilla. Malliin oletetaan ennalta määritellyt huoltoajankohdat, joissa vain komponenttien uusiminen on mahdollista. Komponenttien vikaantumisaikoja mallinnetaan todennäköisyysjakaumilla. Jokainen komponentti on kriittinen. Oletusten pohjalta järjestelmää mallinnetaan diskreetin ajan Markovin päätösprosessina, jossa järjestelmän tila riippuu komponenttien i'istä ja järjestelmän vikatilasta. Järjestelmän luotettavuuden merkitystä korostetaan luotettavuuskynnyksellä. Luotettavuuskynnys, rakenteelliset riippuvuudet ja tarve uusia vikaantuneet komponentit rajaavat käypiä huoltotoimenpideportfolioita, joista kustannustehokkain vaihtoehto jokaiselle tilalle etsitään käyttäen ohjauksen iterointialgoritmia (engl. policy iteration).

Mallia sovellettiin erikokoisiin huollonajoitusongelmiin. Huomattiin, että algoritmin ratkaisuajat riippuvat suuresti tila-avaruuden koosta. Myös muut mallin parametrit, kuten huollon aloituskustannukset, voivat vaikuttaa algoritmin laskenta-aikaan. Mallia sovellettiin myös esimerkkijärjestelmään, jossa on neljä komponenttia. Erilaisten huoltovälien ja luotettavuuskynnysten huomioon ottaminen tarjoavat arvokasta tukea huoltotoimenpiteiden päätöksentekoon. Lisäksi malli tarjoaa kustannussäästöjä verrattuna yksinkertaisiin opportunistisen huollon käytäntöihin.

Asiasanat	Ylläpidon ajoittaminen, monikomponenttijärjestelmä, huoltoportfolio, Markov päätösprosessi, ohjauksen iterointi
Kieli	Englanti

Acknowledgements

I would like to thank my supervisor Antti Punkka for the useful comments, remarks and engagement through the learning process of this master thesis. Furthermore I would like to thank my other advisor Tommi Ekholm for the support and comments on the way. Also, I like to thank the representatives of VR, who have shared their time to discuss about the developed model. Last, I want to thank about the Systems Analysis Laboratory of Aalto University and the possibility to work there while doing the thesis.

Espoo, May 25, 2020

Jussi Leppinen

Contents

Chapter 1

Introduction

Technical systems consist typically of many components. For example, a car with a combustion engine has pistons, cylinders, valves, tyres, breaks, lights, windows and many more components. These multi-component systems should operate reliably. The car should not break down during high speeds which can cause a serious accident. However, in the long run, components wear out causing the reliability of the system to decrease. For example, tyres wear off when driving which makes them more prone to burst suddenly. In the end, the multi-component system needs maintenance to stay in an operating condition.

Maintenance usually involves different kinds of costs, including fixed costs for maintenance set-up, component specific costs, shutdown costs and downtime costs. Components can also have economic dependencies which affect total maintenance costs. In addition, components can have structural dependencies, which either limit the number of feasible maintenance alternatives, or affect the performance of the system for example via reliability. Lastly, components can have stochastic dependencies which means that the condition of one component can affect the condition of another one for example when it fails. As a result, maintenance scheduling of a multi-component system is not an easy task.

Maintenance scheduling should focus on the whole system as well as on the individual components and their interactions. The key question is how to schedule maintenance in the long run when the maintenance decisions influence the state of the system and its future wear-off. The schedule should keep costs low and reliability high, in other words balance between under- and over-service. Under-service makes the maintained system more prone to

failures and over-service consumes unnecessary resources.

One approach to maintenance scheduling of a multi-component system is to minimize maintenance costs subject to some chosen measures of system reliability (Wang, 2002). In practice, maintenance scheduling is often based on first solving the optimal maintenance activities of single components and then grouping them in an efficient way by taking advantage of the system's cost structure (e.g. Wildeman et al., 1997; Van Horenbeek and Pintelon, 2013; Vu et al., 2014). This grouping is based on economic dependencies. Structural dependencies are mainly included from a performance perspective, meaning that maintaining a group of components may require shutting down the system which increases maintenance costs (e.g. Van Horenbeek and Pintelon, 2013; Nguyen et al., 2015). Also the reliability of the system is considered poorly. The maintenance scheduling aims to minimize maintenance costs caused by failures rather than minimize the number of failures itself. This is good from the economical point of view but it may harm the reputation of the company.

In reality, there can also exist structural dependencies, which limit the number of possible maintenance alternatives. These should be taken into account when scheduling maintenance activities. Geng et al. (2015) include these kind of structural dependencies in their model by making them cause extra maintenance costs. Nguyen et al. (2015) consider the number of failures by introducing a reliability threshold to ensure the operation of the system with some probability until next possible maintenance instance. Both of these models are solved with simulation which means other methods are needed to expand the knowledge.

One interesting direction is a Markov decision process where analytical methods can be used to find optimal solutions. Policy-iteration has been part of the maintenance literature for example with single components (e.g. Chan and Asgarpoor, 2006) and special multi-component systems, like two component series system (van der Duyn Schouten and Vanneste, 1990). However, it is a potential approach to use also with more complex systems.

This thesis develops an optimal maintenance scheduling method for a multi-component system with economic and structural dependencies. These dependencies are modeled as a directed graph which allows to model economic dependencies in more detail than many previous models. The components wear out in time and fail more likely at an older age. The failures of components are modeled with probability distributions. Every component is critical meaning if one component fails the whole system fails. The risk of system failure is controlled with a reliability threshold chosen by the decision maker.

The only possible maintenance action is to replace a component into a new one which takes a negligible amount of time. Broken components are always replaced but it is also possible to replace older, working components. The replacements of components are only possible at pre-defined maintenance instances.

The system and the maintenance decisions are presented as a Markov decision process. At every maintenance instance we know how old the current components are and which component has failed. Based on this state of the system, we choose a maintenance action portfolio, which tells which components are replaced, among the feasible maintenance portfolios which take structural dependencies into account, replace broken components and keep the reliability of the system above the threshold. The objective is, for every state, to find a maintenance portfolio which minimizes the total maintenance costs in the long run. This optimal maintenance policy is found with dynamic programming, more precisely with policy-iteration.

The rest of the thesis is organized as follows. Chapter 2 introduces the key concepts used in the maintenance scheduling literature and presents different maintenance models related to these concepts. Chapter 3 presents the maintenance scheduling model, assumptions and methods for solving the model. Chapter 4 demonstrates the convergence of the solution method in different situations. In Chapter 5 we solve a maintenance scheduling problem for a case example and compare the model's results to a simpler maintenance policy. Chapter 6 discusses the model's strengths and weaknesses. In Chapter 7 we draw conclusions and suggest further development ideas.

Chapter 2

Background

Maintenance scheduling research uses mostly model based, computational approaches. The models are typically case-specific, but they often have common structures. These structures are presented in this chapter. The focus here is on a system which consists of many components. The condition of the component can be modeled as a binary or a multi-state variable, meaning that the component can either work or not (e.g. Vu et al., 2014), or it can have multiple working states (e.g. Chan and Asgarpoor, 2006).

2.1 Multi-component systems

The maintenance scheduling of a multi-component system is more complex problem than of a single component system due to two primary reasons: system structure and dependencies between components. First, the system structure means that the configuration of components can be any mixture of basic connections (e.g. series, parallel). These connections affect for example the reliability of the system (Nguyen et al., 2015). Second, Thomas (1986) presents three general categories for dependencies between components: economic, structural and stochastic dependence.

Economic dependence can be positive or negative (Nicolai and Dekker, 2008). Positive economic dependence implies that maintaining a group of components simultaneously is cheaper than maintaining them separately. This can happen if the maintenance set-up costs are relatively high compared to component specific maintenance costs. In contrast, negative economic dependence exists if joint maintenance of components is more expensive than maintaining them separately. This can happen for example if maintaining a

single component is possible without shutting down the system. Economic dependence is common in most continuously operating systems like aircrafts, ships, power plants, telecommunication systems, chemical processing facilities and mass production lines (Wang, 2002).

Structural dependence means that some components cannot be maintained independently, but always at the same time (Nicolai and Dekker, 2008). In practice this means that some operating components have to be dismantled or even repaired before other components can be maintained.

Stochastic dependence implies that the state of one component can influence the state of some other component. According to Nicolai and Dekker (2008) this is usually modeled as a failure dependence. In practice, failure dependence between two components can be presented so that the failure of one component can cause the failure of the other component with a certain probability, or it can cause a shock which alters the failure rate of the other component.

These categories have been developed further to meet the modern developments in the maintenance literature. Keizer et al. (2017) present the fourth category, *resource dependence*, where components have e.g. shared spares, tools or maintenance workers. They also distinguish structural dependence between technical and performance structural dependencies. The technical part refers to the previous definition of structural dependency and the performance part refers to different component configurations like series, parallel and k-out-of-N which have an effect on the reliability of the system.

2.2 Deteriorating systems

Rausand and Høyland (1994) present the concepts needed to describe the the deterioration of the system. The *reliability* of a system is defined as its ability to perform the required function under prevailing operational conditions and for a stated period of time. The improvement of system reliability is one of the primary objectives of maintenance. Many 'high-risk' industries (e.g. aviation, defence, nuclear power) have implemented a reliability-centered maintenance methodology. *Failure* is defined as the termination of the system's ability to perform the required function. The failure of the system can be represented with a binary number which tells does the system operate (1) or not (0).

Time to failure is the time elapsing from starting the component as new until it fails for the first time. Because time to failure is subject to randomness, it is natural to model it as a random variable. For modeling purposes it

is customary to assume a suitable *probability distribution* (e.g. exponential, gamma, Weibull) to model the time to failure. These distributions define failure probabilities of components for different times. If the system structure is known it is possible to define the failure probability of the system from these distributions. Then the mathematical definition for reliability is the probability of system not failing until certain age. *Failure rate* represents the probability of how likely a component of certain age will fail per unit of time. This can be seen as a conditional time to failure. Many maintenance scheduling models assume an increasing failure rate with respect to time. This corresponds to the fact that wear out of components makes components and system more subtle to failures. *Residual life* tells the expected remaining life of the component of certain age.

2.3 Maintenance costs

One of the key objectives for maintenance scheduling is to minimize maintenance cost of the system. The cost structure of a multi-component system can be quite complex. Common costs to be modeled include (e.g. Nguyen et al., 2015):

- *Set-up cost* is a fixed cost which is paid every time the system is maintained (Wildeman et al., 1997). The set-up cost is a system-dependent cost and covers all the maintenance preparations.
- *Component specific cost* is the cost of maintaining a specific component of the system. A component can have both a corrective maintenance cost, which is paid if the component has failed before maintenance, and a preventive maintenance cost, which is paid if the component works when maintained. These can be different for a component, as repairing an already broken component can be more expensive than preventive maintenance.
- *Shutdown cost* models the cost of shutting down the system for maintenance. Shutdown can be planned or unplanned because of a sudden failure. Usually an unplanned shutdown cost is more expensive than a planned one.
- *Downtime cost* differs from shutdown cots in that it also takes into account the duration of maintenance when modeling production losses. Do Van et al. (2013) use the term unavailability cost.

Less common maintenance costs are for example disassembly cost (Dao and

Zuo, 2017) when the disassembly work is separated from other maintenance activities and its duration is taken into account. Another one is component specific inspection cost (e.g. Castanier et al., 2005), when the condition of the component is inspected. This allows knowing the state of wear for the component and to help decide whether or not to maintain it.

2.4 Maintenance scheduling policies

Maintenance policies are rules for scheduling maintenance of different components in different scenarios. These policies tell which maintenance actions should we do for the components. Usually actions are decided upon a continuous time line (e.g. Wildeman et al., 1997) or during discrete maintenance instances (e.g. Nguyen et al., 2015).

Possible maintenance actions can be component or system-specific. Moghaddam and Usher (2011) assume three types of actions. The first possibility is to do nothing. In that case the components continue to age normally from their current state. Second, components can be partially maintained. This places the state of the component to somewhere between the states 'good-as-new' and 'bad-as-old'. The third possibility is to do renewal which replaces the component by a new one and also restores the state of the component to 'good as new'. If all components of the system are replaced, it restores the state of the system also to 'good as new'. Actions two and three are also known as *imperfect* and *perfect maintenance actions*, respectively (Do et al., 2015a).

Maintenance scheduling policies can be divided into different categories. Bevilacqua and Braglia (2000) present five alternatives:

1. *Corrective maintenance* is applied to a system only after it has failed. Corrective maintenance can be based on either imperfect or perfect maintenance actions (e.g. Van Horenbeek and Pintelon, 2013).

2. *Preventive maintenance* is based on information about the system's reliability. This makes it possible to analyze the behaviour of system and determine a series of actions to maintain the system in best possible ways.

3. *Opportunistic maintenance* is based on using upcoming maintenance operations as opportunities to maintain also something else. For example, if a system fails due to one component, and another component is close to its preventive maintenance threshold, it can be cheaper in

the long run to maintain both components at the same time.

4. *Condition-based maintenance* is based on monitoring the system and making decisions for preventive maintenance based on the observed data.

5. *Predictive maintenance* takes the idea of condition-based maintenance further and predicts failures based on the observed data. Maintenance scheduling decisions are based on that prediction.

The relationship between maintenance policies and actions can be clarified with examples such as age replacement policy and block replacement policy. Ab-Samat and Kamaruddin (2014) define the age replacement policy as follows: a component is replaced if it has reached a certain age without failures or when it fails, whichever happens first. Thus, this policy can be categorized to be preventive maintenance. In block replacement policy some components form a block and when one of those components needs replacement, other members of the block are replaced as well. Therefore, block replacement policy can be categorized to be a combination of preventive and opportunistic maintenance.

According to Ab-Samat and Kamaruddin (2014), the origins of opportunistic maintenance policy can be attributed to age and block replacement policies. They also present some benefits and drawbacks of the opportunistic maintenance policy. The benefits are that the policy reduces failures, lowers maintenance costs and increases lifetime of the system. The drawbacks are the difficulties of determining when to maintain the system, how to avoid under and over maintenance, and how to make sure there are enough spare parts and work force available if a sudden opportunity arises.

Other examples are minimal repair policies and cannibalization (Cho and Parlar, 1991). Minimal repair policies usually follow a periodic replacement policy where, instead of replacing a component upon failure, it is minimally repaired to gain cost advantage. The periodic replacement is interpreted as preventive maintenance. Cannibalization is a rule of interchanging working and failed components among systems to maximize the number of operating systems. It is the only replacement policy which is possible when new spare parts are unavailable.

2.5 Maintenance scheduling criteria

The optimal maintenance scheduling policy of the multi-component system is based on different criteria. Wang (2002) presents four possibilities:

1. Minimize system long-term average maintenance costs.
2. Maximize the system reliability measures.
3. Minimize system long-term average maintenance costs subject to the system reliability measures.
4. Maximize the system reliability measures subject to maintenance cost requirements.

Wang (2002) also points out two relevant remarks. First, many models find the optimal maintenance scheduling by minimizing system maintenance cost rate and not taking reliability performance into account. As a result, the policy causes the system reliability to be unacceptable in practise. Second, maintenance models should not assume negligible maintenance durations because it gives unrealistic reliability measures.

Opportunistic maintenance policies can answer to the scheduling criteria presented by Wang (2002). The opportunistic maintenance policies of the multi-component systems are usually based on different age thresholds values. The basic idea is that, in the case of failure, other than broken components are also maintained, if their ages exceed the threshold. Wang (2002) presents different practical approaches how to create these thresholds. According to Geng et al. (2015) these policies were long based only on economic dependence. Geng et al. (2015) present a model considering both economic and structural dependence where structural dependence affects both the maintenance costs and maintenance durations and where the goal is to find opportunistic maintenance age thresholds which minimize total maintenance costs. They use Monte Carlo simulation to find these thresholds.

In later studies, the expected maintenance costs have been minimized as such or under different constraints (see Geng et al., 2015). For example, Nguyen et al. (2015) apply a reliability constraint and Do et al. (2015b) an availability constraint. Reliability is usually based on a failure rates of the components. These can be for example a Weibull distribution (e.g. Do et al., 2015b) or a homogeneous gamma stochastic process (e.g. Nguyen et al., 2015). The use of availability constraints, on the other hand, requires that maintenance durations of system and its components are included in the model. It is also possible to swap the objectives and constraints of the model, and for example

maximize system availability subject to a maintenance budget constraint. Dao and Zuo (2017) maximize the reliability of the system under budget and time constraints. They also model structural dependencies with a directed graph which tells how the system must be dismantled if something needs to be maintained. However, their approach considers only one maintenance instance at a time.

In addition to cost minimization related objectives, some multi-objective approaches have been presented. Jiang and Ji (2002) provide a multi-attribute value model to evaluate the performance of an age replacement policy. They consider four attributes, cost, availability, reliability and lifetime and develop performance measures to obtain the overall performance. Moghaddam and Usher (2011) use multi-objective optimization to minimize maintenance costs and maximize system reliability. Bevilacqua and Braglia (2000) use the Analytic Hierarchy Process to select the best possible maintenance strategy.

Cho and Parlar (1991) point out that it is possible to consider models with incomplete information. Either the lifetime distribution, current state or the cost structure of the system can be unknown.

2.6 Solutions for scheduling models

This section explains how some the scheduling models of multi-component systems are solved in practice. The focus is on grouping and dynamic programming. When it comes to grouping maintenance activities, the rolling horizon approach has received much attention in the area of maintenance research. Markov decision processes are widely used for maintenance scheduling from the area of dynamic programming,.

2.6.1 Maintenance action grouping

Doing more than one maintenance action at the same time is called grouping. Grouping takes advantage of opportunistic maintenance policy and positive economic dependencies between components to obtain cost efficient maintenance plans. In practise this means that the cost savings define whether a maintenance of one components serves a good enough opportunity to maintain other components as well.

According to Chalabi et al. (2016) three different types of maintenance groupings are studied in the literature: long-term (static), medium-term (dynamic) and short-term (opportunistic) grouping. Pargar et al. (2017) explain these

as follows. Static grouping is based on a stable situation with static grouping rules over the planning horizon. The rules are based on predefined preventive maintenance policies of components. In dynamic grouping, medium-term information about the components' residual life can be used to adapt maintenance scheduling. It is possible to dynamically group planned preventive maintenance activities with each other and with planned corrective maintenance activities. In opportunistic grouping, short term information (e.g. failures) are taken into account which makes it possible to group preventive maintenance activities with unplanned corrective maintenance activities.

Rolling horizon is the main approach used in the dynamic maintenance grouping models (Pargar et al., 2017). Rolling horizon means that we look at the maintenance of the system a finite horizon into the future. During that horizon we plan future maintenance decisions in the most effective way. Then the system is maintained according to these plans until the situation is changed so that new planning horizon is needed. The system rolls on according to these horizons.

Wildeman et al. (1997) present the first rolling horizon approach to find optimal maintenance planning in terms of maintenance costs. This approach has five phases. First the decomposition phase creates an individual maintenance rule for each component. This rule tells the age at which a component should be maintained if minimizing long term average maintenance costs with an infinite planning horizon. In phase two, a penalty function is defined for each component. It tells the expected additional cost if a component is not maintained according to rule defined in phase one. This additional cost can be associated with wasting residual life if the component is maintained early, or adding risk of failure if the component is maintained late (Bouvard et al., 2011). In the tentative planning phase, a finite planning horizon is assumed and the necessary maintenance actions (based on the current ages of components and results of phase one) are located within the horizon. In the maintenance activities grouping phase, the previously defined maintenance actions are grouped by moving maintenance actions in time within the planning horizon. The optimal grouping structure maximizes maintenance cost savings which result from reduced number of set-up costs minus penalty costs (phase two). Phase five is the rolling horizon step where the created maintenance plan is executed. If the state of the system evolves unexpectedly or a new planning period is needed, the process continues again from phase three.

The rolling horizon approach of Wildeman et al. (1997) has been developed further by minimizing maintenance costs under different extensions of the

model. Bouvard et al. (2011) use inspection dates to take component wear out into account when deciding whether to advance or delay maintenance activities when also guaranteeing a minimal operation time between consecutive maintenance instances. Do Van et al. (2013) expand the rolling horizon model by considering the duration of maintenance activities and possibility of maintaining same component multiple times during the planning horizon. They also take into account available short-term information by modeling upcoming inactivity periods as maintenance opportunities. They notice that grouping increases cost savings the more the bigger is the set-up cost.

Van Horenbeek and Pintelon (2013) incorporate predictive information in the form of remaining useful life, imperfect maintenance and maintenance downtimes as extensions into the rolling horizon model. They also model partial dependencies by including a dependence parameter which scales the set-up cost savings when grouping maintenance activities. Do et al. (2015b) model rolling horizon maintenance with availability constraint by bounding the duration of maintenance actions. They also constrain the number of allowed maintenance teams, which affect the duration of maintenance operations. Vu et al. (2014) expand the system structure from series to series-parallel. The maintenance optimization of this more complex system leads to an NP-hard problem which is too complicated for the dynamic grouping algorithm, wherefore they use a genetic algorithm to solve the model.

When using finite horizon and rolling horizon maintenance models, Wildeman et al. (1997) say that short-term horizons should include some kind of residual value of the system. In other words, the state of the system at the end of the planning horizon should be taken into account when deciding between different maintenance schedules. Dekker et al. (1996) give examples of how different definitions of residual values affect the maintenance scheduling policies. Wildeman et al. (1997) have taken the residual values into account because the penalty functions give indicators how short-term decisions influence future cost.

2.6.2 Dynamic programming

Dynamic programming (Bellman et al., 1957) is a mathematical optimization method which simplifies a complicated problem by breaking it down to into simpler sub-problems in a recursive manner. Dynamic programming problems are sequential decision problems where decisions change the state of the system, hence affecting future decision (Powell, 2007). The goal is to find a sequence of decisions which optimize some chosen quantity, for example minimize costs.

Dynamic programming problems are usually easy to formulate but hard to solve. The system has states and state transition probabilities which depend on the current state and the decision made in that state. But if the number of states or decisions is large, the number of different scenarios of the system grows very quickly. This is known as the curse of dimensionality, which makes the finding of an exact solution computationally heavy or even impossible. To get rid of this problem it is possible to use approximative methods, a field known as approximated dynamic programming (Powell, 2007). These methods include for example value-iteration and policy-iteration.

Approximated dynamic programming has been used to solve maintenance scheduling problems. Love et al. (1982) use Howard's (1960) policy-iteration algorithm to determine repair limits of vehicles according to their condition. The repair limit problem is modeled as a Markov decision process. They show that the condition based policy works much better than a simple age or milage based policy. Similarly, van der Duyn Schouten and Vanneste (1990) consider a cost optimal maintenance policy of a two component series system. They say it is possible to solve the maintenance policy of the system by assuming a Markov decision process which is solved with value or policy-iteration. However these approaches have two drawbacks. Either these methods are time consuming for bigger problems or the obtained results, state specific policies, are hard to interpret. Thus, they present a (n, N)-policy with is an approximated version and gives a close to optimal average maintenance cost. In (n, N)-policy a component is replaced if it has failed or reached the age N and, if something is replaced, other components, whose age is at least n, are opportunistically replaced as well.

Chan and Asgarpoor (2006) use policy-iteration to find a cost optimal maintenance policy for a component that can fail randomly or due to deterioration. The component is modeled with a Markov decision process in continuous time and the objective is to maximize earnings which are dependent on the state of the component. Kyriakidis and Dimitrakos (2006) introduce a Markov decision model for the optimal preventive maintenance of an installation that supplies a raw material to a production unit. The installation deteriorates and there is a buffer between the installation and the production unit to cope with unexpected failures of the installation, which may cause interruptions of the production process. They use techniques from Markov decision theory, based on policy-iteration, and show that, for fixed age of the installation and fixed buffer level, the policy that minimizes the expected long run average cost per unit time is of control limit type.

2.7 Conclusions of the chapter

The maintenance scheduling of a multi-component system is not easy. As we have noticed different scheduling models introduce many aspects to consider. Do we have a continuous time line or discrete maintenance instances? How do we model components and which dependencies are included? What is the cost structure of the system? What are the objectives and constraints when optimizing maintenance scheduling? The chosen approach should answer these questions.

It also seems that it is important to minimize maintenance costs subject to the system reliability measures. The rolling horizon approach (Wildeman et al., 1997) is based on first minimizing the expected maintenance costs of a single component. However, this approach does not limit the reliability of the system and thus the operation of the system can become uncertain. This is sometimes unacceptable in practise. However, the ideas of grouping and using opportunities wisely seem important for good maintenance scheduling practises. It also seems that dynamic programming can be used to find good ways to group maintenance operations. As a result, it seems that there are good approaches for maintenance scheduling but what is missing is a model that combines them effectively.

Next we develop a new maintenance scheduling model. We move the focus from component level to system level by applying the Markov decision process and dynamic programming instead of the rolling horizon approach. We also define a reliability threshold similar to Nguyen et al. (2015). As a result, our model will consist of states where the reliability of the system is always high enough. From this model, we will get maintenance recommendations for every state of the system and we can prove that these recommendations are cost optimal. This is a big advantage when we compare this method to simulation.

In addition, we make planning of maintenance activities easier in practise by using discrete maintenance instances where we can only replace components. Also, it should be reasonable to assume that time between consecutive maintenance instances is much longer than replacement durations, so it not necessary to model shutdown costs. Therefore, we only include the set-up cost and component specific costs in the model.

Chapter 3

Optimization model for maintenance scheduling

This chapter explains the model developed in the thesis in detail. First, we present the system as a directed graph and form the maintenance cost structure of the system. Then, we explain how the system evolves over time and how the reliability of the system is defined. We then define maintenance action portfolios and their feasibility. After that, we combine these concepts into a Markov decision process and show how to find optimal maintenance policies using policy-iteration algorithm and discounted policy-iteration algorithm.

3.1 System structure and cost assumptions

The system consists of n components, denoted by $N = \{1, ..., n\}$. For the system to operate, every component must operate. Components fail according to some known probability distributions. We assume that only one component can fail at a time. Additionally, components can only be replaced into new ones. The replacements are carried out during predefined *maintenance instances* at times $t_{k+1} = t_k + \Delta t$, $k \in \mathbb{N}$, with a constant $\Delta t > 0$ known as *maintenance interval*. The maintenance interval can be measured, for example, in time, working hours or distances. If a component fails during (t_k, t_{k+1}) it is replaced into a new one at the next maintenance instance t_{k+1}. It is also possible to replace working components preventively. The time durations of maintenance instances are assumed to be negligible meaning the replacements do not take much time in practice.

The last replacement time of each component is known. Specifically, let $\tau \in \mathbb{R}^n$ represent *a maintenance history* where τ_i is the last replacement time of component i. In addition, *the ages of the components* at maintenance instance t_k are collected in $a_k = t_k - \tau \in \mathbb{R}^n$.

There are no stochastic dependencies between components, which means components wear off and fail independently of each other. However, economic and structural dependencies between components exist. Economic dependencies mean that the replacement costs depend on the combination of replaced components. Structural dependencies mean that maintaining a component can require simultaneous maintenance of some other component.

These dependencies between components are modeled with *a directed graph*, denoted by $G = (V, A)$, where $V = \{0\} \cup N$ is the set of nodes and A is the set of directed arcs (i, j) with the start node i and the end node j. The node 0 is *the root node* of the graph. It represents the start of the maintenance session and includes *a fixed set-up cost* c_0, which is paid every time maintenance actions are carried out. It is noteworthy that a maintenance instance does not incur costs, if no components are replaced. Other nodes represent replacements of the components. *The weight* c_{ij} *of arc* (i, j) is the cost of replacing component j on the condition that component i is also replaced. In practise this means that, for example, if a component is connected to root node, it can be replaced independently of other components but otherwise a component with an arc to the target component has to be replaced also.

Consider the 5-component-system presented in Figure 3.1. We will use this example throughout this chapter. In this case component 2 is not replaceable without also replacing component 1. Furthermore, it is cheaper to replace component 5 if component 4 is also replaced because $c_{45} = 120 < 190 = c_{05}$.

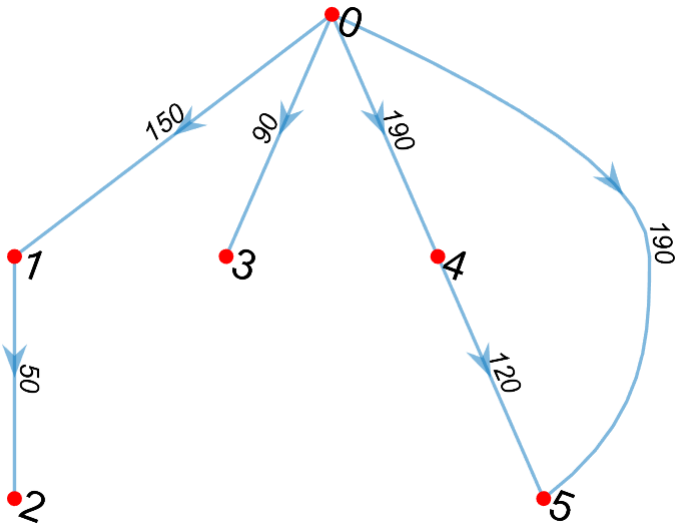


Figure 3.1: Example of a system of five components and corresponding costs of c_{ij}

If component i is replaced due to its failure, then a component-specific *corrective replacement surplus* cost r_i is paid. These surpluses include both a corrective maintenance surplus for the set-up cost $c_0^{CM} \geq 0$ and a component specific surplus costs $c_i^{CM} \geq 0$:

$$r_i = c_0^{CM} + c_i^{CM} \quad \forall i = 1, ..., n. \tag{3.1}$$

The corrective maintenance set-up cost c_0^{CM} can be interpreted for example as an extra cost of using a backup system to cover the demand of the system until the next maintenance instance. The component specific surplus costs c_i^{CM} are extra work resulting from failure. In practise, it can mean that the system must be properly tested after replacing a failed component or that replacing a failed component requires more work than replacing a functional one.

For the system presented in Figure 3.1, we assume set-up cost $c_0 = 60$ and corrective replacement surpluses $r = (120, 90, 85, 70, 90)^T$.

3.2 Modeling the evolution of system

During each maintenance interval Δt the states of the components evolve by either ageing or failing. The age of the component i at maintenance

instance t_k is $(a_k)_i = t_k - \tau_i$. The failure time of each component t_i^f follows its own probability distribution, modeled with probability density function (PDF) $f_i((a_k)_i)$. The cumulative distribution function (CDF) $F_i((a_k)_i) = \int_{-\infty}^{(a_k)_i} f_i(t)\,dt$ tells the probability that the component will fail before the age $(a_k)_i$.

If component i operates at maintenance instance t_k, having age $(a_k)_i = t_k - \tau_i$, then it operates until t_{k+1} with the conditional probability

$$\begin{aligned} P_k^i\left(t_i^f > t_{k+1} | t_i^f > t_k\right) &= \frac{P_k^i\left(t_i^f > t_{k+1}\right)}{P_k^i\left(t_i^f > t_k\right)} = \frac{1 - F_i(t_{k+1} - \tau_i)}{1 - F_i(t_k - \tau_i)} \\ &= \frac{1 - F_i((a_k)_i + \Delta t)}{1 - F_i((a_k)_i)} \\ &:= R_i(t_k). \end{aligned} \tag{3.2}$$

This is *the reliability of component* i at time t_k. Because every component is critical and the failures are independent, we have a series system and the reliability of the system at t_k is

$$R_{sys}(t_k) = \prod_{i=1}^{n} R_i(t_k). \tag{3.3}$$

Consider the 5-component-system in Figure 3.1. For the sake of simplicity we assume that the PDF of failure time is a linearly increasing function for every component and that each component has a maximal age α it cannot surpass which means $F_i(x) = 1$ for $x \geq \alpha$. Figure 3.2 illustrates the CDFs described in Table 3.1.

Table 3.1: Component statistics

Component i	max. age α_i	PDF $f_i(x)$	CDF $F_i(x)$
1	17	$\frac{2x}{17^2}$	$\frac{x^2}{17^2}$
2	33	$\frac{2x}{33^2}$	$\frac{x^2}{33^2}$
3	12	$\frac{2x}{12^2}$	$\frac{x^2}{12^2}$
4	11	$\frac{2x}{11^2}$	$\frac{x^2}{11^2}$
5	16	$\frac{2x}{16^2}$	$\frac{x^2}{16^2}$

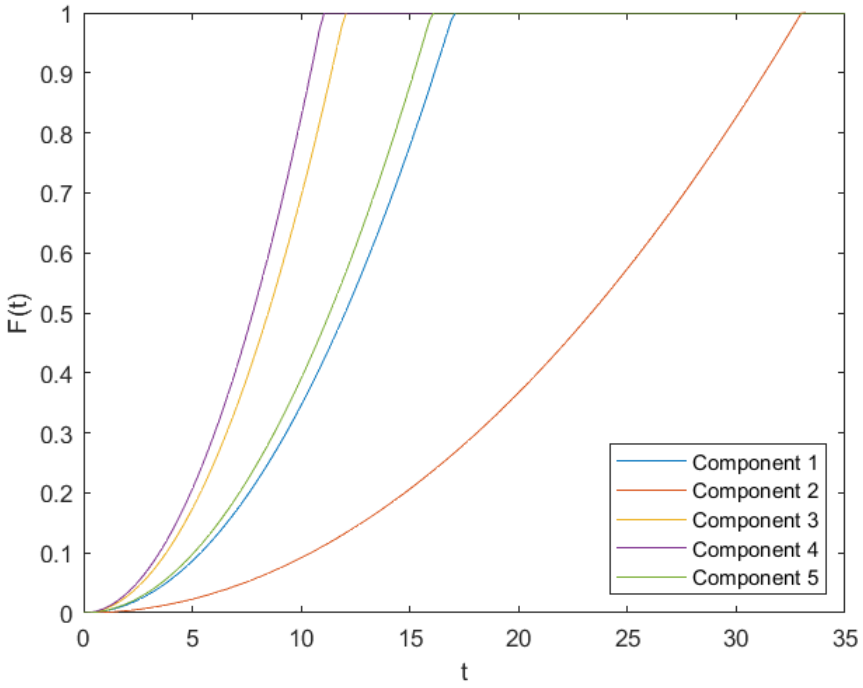


Figure 3.2: CDFs of the five components

The reliability of a component i at age $(a_k)_i$ at t_k can be computed using (3.2). For example, if $t_k = 7$, $\Delta t = 1$ and $\tau_3 = 0$, the age of component 3 is $(a_k)_3 = 7$ and it will work until $t_{k+1} = 8$ with probability

$$P_7^3(t_3^f > 8 | t_3^f > 7) = \frac{1 - \frac{8^2}{12^2}}{1 - \frac{7^2}{12^2}} \approx 0.842.$$

We express the *failure state* of the system at maintenance instance t_k as a binary vector $f_k \in \{0,1\}^n$. If $(f_k)_i = 1$, component i has failed, otherwise not. It is assumed that at most one component fails during each maintenance interval (t_k, t_{k+1}). Thus, either one of the n components fails or none of the components fail. This leads to the constraints

$$\sum_{i=1}^{n} (f_k)_i \leq 1 \quad \forall k \in \mathbb{N}.$$

If all elements of f_k are zero, then none of the components has failed.

A single component operates during (t_k, t_{k+1}) with probability $R_i(t_k)$ calculated with (3.2) so the failure probability is $1 - R_i(t_k)$. Thus, only component

i will fail during (t_k, t_{k+1}) with probability

$$F_i(t_k) := (1 - R_i(t_k)) \prod_{j=1,\dots,n,\ j\neq i} R_j(t_k). \tag{3.4}$$

The probability of no component failures can be calculated with (3.3). If $A =$ 'at most one component fails during (t_k, t_{k+1})', $P(A)$ can be calculated with help of (3.3) and (3.4)

$$P(A) = R_{sys}(t_k) + \sum_{i=1}^{n} F_i(t_k). \tag{3.5}$$

With (3.5) we can formulate the conditional probabilities that describe the evolution of failure state of system under our assumption A:

$$P(i\,\text{fails}|A) = \frac{F_i(t_k)}{P(A)} \tag{3.6}$$

$$P(\text{no failures}|A) = \frac{R_{sys}(t_k)}{P(A)} \tag{3.7}$$

System failure is unwanted. For this reason we set a reliability threshold $\rho \in [0, 1)$ on the system. At every maintenance instance, the system must be maintained so that the reliability of the system remains above the threshold until the next maintenance instance. Mathematically,

$$\frac{R_{sys}(t_k)}{P(A)} \geq \rho \quad \forall k \in \mathbb{N}. \tag{3.8}$$

The reliability threshold is typically set so high that the component failure probabilities have to be kept small. Therefore, simultaneous failure of more than one component is unlikely. As a result, the rare event approximation can be used to justify the assumption that only at most one component can fail during each time step (t_k, t_{k+1}). The probability $P(A)$ tells how likely at most one component fails. This means that the probability $1 - P(A)$ tells the accuracy of the approximation.

3.3 Maintenance action portfolios

At every maintenance instance t_k, a *portfolio* $x_k \subseteq N$ of components to be replaced is chosen. The binary vector $z(x_k) \in \{0, 1\}^n$ tells whether a

component is replaced (1) or not (0). Thus, z is a bijection $z\colon \mathcal{P}(N) \to \{0,1\}^n$ such that $z_i(x_k) = 1$ if $i \in x_k$ and $z_i(x_k) = 0$ if $i \notin x_k$, where $\mathcal{P}$ denotes the power set. For example, if we replace components 1 and 4 from the 5-component-system, then $z(\{1,4\}) = (1,0,0,1,0)^T$.

In general there are $|\mathcal{P}(N)| = 2^n$ possible portfolios that can be chosen at every maintenance instance. However, the system structure, possible component failures and the system reliability threshold constrain the feasibility of the portfolios at a particular maintenance instance. Because a failed component is always replaced we have the requirement

$$z_i(x_k) \geq (f_k)_i \quad \forall i = 1, ..., n. \tag{3.9}$$

The chosen portfolio has to take structural dependencies into account. We define a set

$$\tilde{N} = \{j \in N \,|\, (0,j) \notin A\}$$

to describe the components not adjacent to root node. If these components are maintained there must be a path from root node to these components:

$$\sum_{\{i|(i,j)\in A\}} z_i(x_k) \geq z_j(x_k), \quad \forall j \in \tilde{N} \tag{3.10}$$

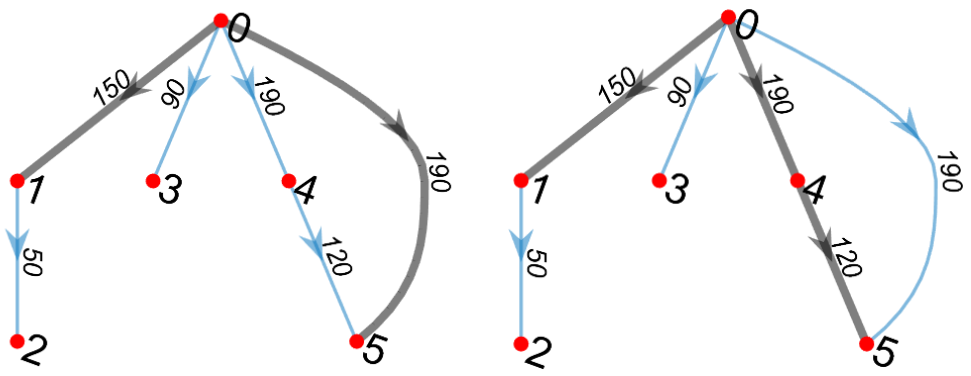
For example in Figure 3.1, component 2 cannot be replaced without replacing component 1 also.

The replacement costs of any x_k are determined from the graph G. This means finding *a tree* which connects all components in x_k to the root node with minimum cost. Such tree is called *a minimum-cost arborescence* and it can be determined using *Edmond's algorithm* (Kleinberg and Tardos, 2006). Every portfolio x_k is a connected subgraph of the graph G and the corresponding cost c_{x_k} is obtained from the minimum-cost arborescence. If $x_k \neq \emptyset$, the set-up cost c_0 is added to the replacement costs:

$$c(x_k) = \begin{cases} 0, & \text{if} \quad x_k = \emptyset \\ c_0 + c_{x_k}, & \text{else.} \end{cases} \tag{3.11}$$

Figure 3.3 shows the minimum cost arborescences of two maintenance action portfolios. The cost of replacing components 1 and 5 is $c(\{1,5\}) = 60 + 150 + 190 = 400$. Adding component 4 to the portfolio increases the cost by 120 to $c(\{1,4,5\}) = 60 + 150 + 190 + 120 = 520$ which is smaller than

$c_{04} = 190$ because replacing component 5 becomes cheaper, due to addition of component 4.



(a) $x_k = \{1, 5\}$ (b) $x_k = \{1, 4, 5\}$

Figure 3.3: Minimum-cost arborescences of the two portfolios are highlighted with thicker, black edges

Finally, the maintenance history τ is updated according to chosen portfolio,

$$\tau_i = \max\{\tau_i,\ z_i(x_k)t_k\} \quad \forall i = 1, ..., n. \tag{3.12}$$

This also updates the ages of components a_k. Thus, both the maintenance history and the ages of components are discretized.

3.4 Markov decision processes

Under the assumptions of the previous sections it is possible to model the maintenance scheduling of the system as a *discrete-time Markov decision process* (MDP; Howard, 1960).

MDPs are extensions of Markov chains. Both have states and state transition probabilities which depend only on the current state of the system, i.e. they obey the Markov property. However, they differ in that in MDPs, it is possible to model actions which affect the state transition probabilities. The current state can restrict the number of allowed actions, which can be taken in it. Furthermore, actions lead to costs. In MDP context, the objective is to find a decision rule which minimizes the *long run average cost per time unit*.

The state of the system is fully defined by the ages of components a_k and the failure state f_k. The state vector s_k at instance t_k can then be written

$$s_k(a_k, f_k) := \begin{bmatrix} a_k^T \\ f_k^T \end{bmatrix} = (a_k^T; f_k^T) \in \mathbb{R}^{2\times n}.$$

The reliability threshold (3.8) constraints the age combinations of the components h in the system. Any age combination can lead to $n+1$ failure states, only the transition probabilities change. Therefore the state space S is finite and of size

$$|S| = h(n+1). \tag{3.13}$$

It is important to remember that it can be possible that $s_{k_1} = s_{k_2}$ even if $k_1 \neq k_2$ because the age combination of components and realization of failure state can be same at different maintenance instances. Therefore we present an alternative notation $\sigma_i \in S$, where $1 \leq i \leq h(n+1)$, to describe the state of the system. In this notation every σ_i is unique.

In the 5-component-system, parameters of components are chosen so that the components wear off quite fast. If we set the reliability threshold to $\rho = 0.9$, the number of possible age combinations is $h = 2597$. This number can be calculated by forming all possible age combinations that fulfil the reliability threshold (3.8). In these age combinations it is also taken into account that component 2 cannot be newer than component 1 because the structure of the system does not allow replacing component 2 without component 1. The size of the state space is then $|S| = 2597(5+1) = 15582$.

The actions of the model are to choose a maintenance action portfolio. In this case, the current state reduces the number of allowed actions.

Definition 1. A portfolio x_k is called a *feasible portfolio*, if it

1. fulfils reliability threshold (3.8)
2. replaces broken components (3.9)
3. satisfies structural dependencies (3.10)

For state σ_i the set of feasible portfolios is X_{σ_i}.

From now on it is assumed that only feasible portfolios are considered, i.e. $x_k \in X_{\sigma_i}$ for all σ_i. For example the 5-component-system does not have states which have $2^5 = 32$ feasible portfolios because structural dependencies (3.10) reduce the number to 24.

After the portfolio is chosen the maintenance history is updated with (3.12). This new state s_k can lead to $n+1$ different states s_{k+1} after Δt depending on the failed component, if any, i.e. f_{k+1}. The transition probabilities for the $k+1$ alternatives can be calculated with (3.6) and (3.7). These probabilities can be presented as a row vector

$$P_{s_k}(x_k) = P_{\sigma_i}(x_k) = \left[\frac{F_1(t_k)}{P(A)}, \ \dots, \frac{F_n(t_k)}{P(A)}, \ \frac{R_{sys}(t_k)}{P(A)}\right] \in \mathbb{R}^{1\times(n+1)}. \tag{3.14}$$

We use the notation $p_{\sigma_i\sigma_j}(x_k)$ for the transition probability from state σ_i to state σ_j when portfolio x_k is chosen. It is notable that these transition probabilities are time-homogeneous which means they are independent of the maintenance instance parameter k.

The cost of action in state σ_i is defined by the cost of the maintenance action portfolio $x_k \in X_{\sigma_i}$ added with the corrective replacement surplus if a component was failed in the state. Using equations (3.1) and (3.11), the cost of action can be written as

$$c_{\sigma_i}(x_k) = c(x_k) + r^T f_k. \tag{3.15}$$

This takes into account the fact that no corrective replacement surplus has to be paid if none of the components fail.

We illustrate these concepts with the 5-component-system. Let the reliability threshold be $\rho = 0.9$, maintenance instance $k = 5$, current time $t_5 = 5 \in \mathbb{R}$, maintenance interval $\Delta t = 1$ and maintenance history $\tau = (4, 2, 3, 2, 4)^T \in \mathbb{R}^5$. Then, the ages of components are $a_5 = t_5 - \tau = (1, 3, 2, 3, 1)^T \in \mathbb{R}^5$. This means that components 1 and 5 were replaced at the last maintenance instance and other components are older. Let us also assume that none of the components has failed, which means $f_5 = (0, 0, 0, 0, 0)^T$. Then the current state is $s_5 = \begin{bmatrix} 1 & 3 & 2 & 3 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$. If we do not replace any components, $x_5 = \emptyset$, the maintenance history stays the same, the age of every component increases with $\Delta t = 1$ and the system ends up in one of the six possible states with probabilities given by (3.14). These states and transition probabilities are given in Table 3.2 and also illustrated in Figure 3.4. From the last row of the table, it is clear that the reliability threshold is not fulfilled, because $0.8829 < \rho$. Thus, $x_5 = \emptyset$ is not a feasible portfolio.

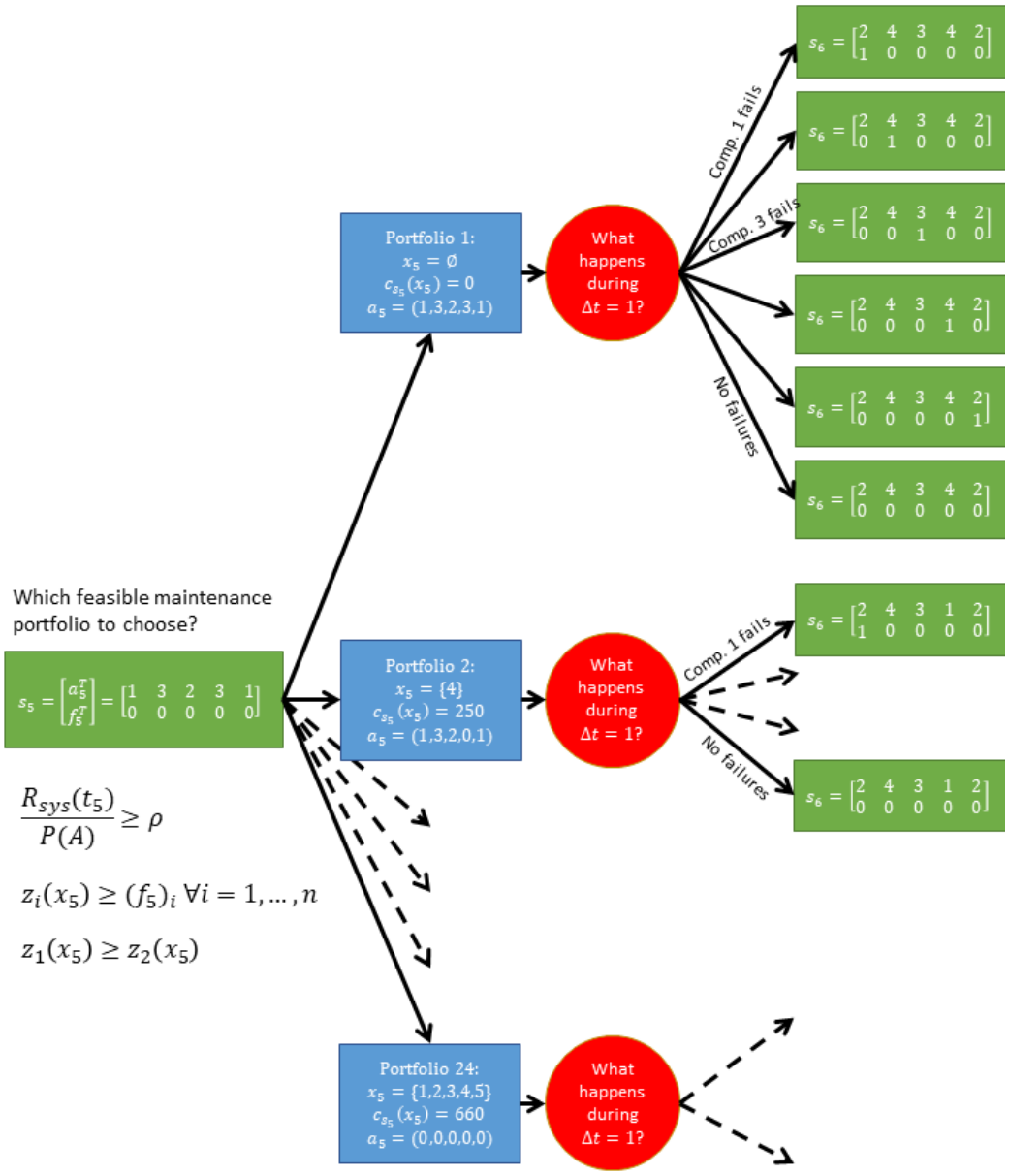


Figure 3.4: Illustration of the MDP

Table 3.2: State transitions from s_5 with $x_5 = \emptyset$

what happens during (t_5, t_6)	state at $k = 6$ $s_6 = (a_6^T; f_6^T)$	Trans. prob. $p_{s_5 s_6}(x_5)$
comp. 1 fails	$(2, 4, 3, 4, 2;\ 1, 0, 0, 0, 0)$	0.0093
comp. 2 fails	$(2, 4, 3, 4, 2;\ 0, 1, 0, 0, 0)$	0.0058
comp. 3 fails	$(2, 4, 3, 4, 2;\ 0, 0, 1, 0, 0)$	0.0327
comp. 4 fails	$(2, 4, 3, 4, 2;\ 0, 0, 0, 1, 0)$	0.0589
comp. 5 fails	$(2, 4, 3, 4, 2;\ 0, 0, 0, 0, 1)$	0.0105
No comp. fails	$(2, 4, 3, 4, 2;\ 0, 0, 0, 0, 0)$	0.8829

Table 3.2 shows that component 4 is more likely to fail than any other component. Thus, one possibility could be to replace it resulting in $x_5 = \{4\}$. In this case, the maintenance history becomes $\tau = (4, 2, 3, 5, 4)^T$. Now, state $s_6 = (2, 4, 3, 1, 2;\ 0, 0, 0, 0, 0)$ is reached with probability $p_{s_5 s_6}(x_5) = 0.9308 > \rho$ which means that x_5 is a feasible portfolio. This decision costs $c_{s_5}(x_5) = c(x_k) = 60 + 190 = 250$. An interesting question is if this is the right decision to make if we want to maintain the system cost-efficiently in the long run.

3.5 Policy-iteration

To help maintenance decision making, we want to find *a stationary policy* for the system. A stationary policy U is a rule which prescribes an action x_k whenever the system is in state σ_i regardless of the current maintenance instance t_k: $U_{\sigma_i} = x_k$. This rule should tell the most cost-efficient way to maintain the system whenever it is in a specific state. A stationary policy for maintenance can be found using a *policy-iteration algorithm*. This policy is optimal in terms of average cost over a very long period of time.

3.5.1 Policy-iteration algorithm

The policy-iteration algorithm is based on the following (Tijms, 1986). First, let $V_n(\sigma_i, U)$ denote the total expected maintenance costs over the n maintenance instances when the initial state is σ_i and policy U is used. Then the long-run average cost $g(U)$ of the policy U is

$$g(U) = \lim_{n \to \infty} \frac{V_n(\sigma_i, U)}{n}. \tag{3.16}$$

Then, the interest lies in the so-called *relative values* v_{σ_i} of the states $\sigma_i \in S$. Relative values indicate how the starting state affects on the total expected costs of a given policy. With the help of (3.16) we can write

$$V_n(\sigma_i, U) \approx ng(U) + v_{\sigma_i}(U) \quad \text{for large } n. \tag{3.17}$$

In state σ_i we choose a maintenance portfolio $x_k = U_{\sigma_i}$ according to the current stationary policy U. This portfolio incurs a cost $c_{\sigma_i}(U_{\sigma_i})$ and transfers the system into state σ_j with probability $p_{\sigma_i\sigma_j}(U_{\sigma_i})$. From this state the total expected costs over the remaining $n-1$ maintenance instances is $V_{n-1}(\sigma_j, U)$. Together these observations form a recursion equation

$$V_n(\sigma_i, U) = c_{\sigma_i}(U_{\sigma_i}) + \sum_{\sigma_j \in S} p_{\sigma_i\sigma_j}(U_{\sigma_i}) V_{n-1}(\sigma_j, U) \tag{3.18}$$

for all $n \geq 1$ and $\sigma_i \in S$. When we substitute (3.17) into (3.18), cancel common terms and use the fact that $\sum_{\sigma_j \in S} p_{\sigma_i\sigma_j}(U) = 1$ we have

$$g(U) + v_{\sigma_i} \approx c_{\sigma_i}(U_{\sigma_i}) + \sum_{\sigma_j \in S} p_{\sigma_i\sigma_j}(U_{\sigma_i}) v_{\sigma_j} \quad \forall \sigma_i \in S. \tag{3.19}$$

These linear value-determination equations (3.19) have a unique solution for average cost $g(U)$ and relative value $v(U)$ when we use a normalization equation: $v_{\sigma_s} = 0$ for an arbitrarily chosen state σ_s. Tijms (1986) proves this in Theorem 3.1 on p. 167.

With the relative values of the policy U it is possible to construct a new policy U' whose average cost is at most as high as of the current policy. Assume that at the first maintenance instance we can choose any feasible maintenance portfolio x_k and thereafter follow the policy U. Then the expected maintenance costs over the first n maintenance instances are $c_{\sigma_i}(x_n) + \sum_{\sigma_j \in S} p_{\sigma_i\sigma_j}(x_k) V_{n-1}(\sigma_j, U)$. Of course, we want to choose the portfolio which minimizes this expression. When n is large, we can use the approximation (3.17) and write

$$\begin{aligned} & c_{\sigma_i}(x_n) + \sum_{\sigma_j \in S} p_{\sigma_i\sigma_j}(x_k) V_{n-1}(\sigma_j, U) \\ \approx & c_{\sigma_i}(x_n) + \sum_{\sigma_j \in S} p_{\sigma_i\sigma_j}(x_k) v_{\sigma_j}(U) + (n-1)g(U). \end{aligned}$$

In the end it is sufficient to choose a portfolio which minimizes the quantity $c_{\sigma_i}(x_n) + \sum_{\sigma_j \in S} p_{\sigma_i\sigma_j}(x_k) v_{\sigma_j}(U)$. Finding a portfolio this way for every

starting state $\sigma_i \in S$ yields an improved policy U' with a lower average cost. Tijms (1986) proves the decrease in the average cost in Theorem 3.2 on p. 170.

To conclude this section, the policy-iteration algorithm has 4 steps (Tijms, 1986, p. 171), which are described below.

1. *Step: Initialization* Choose a stationary policy U.
2. *Step: Value-determination step* For the current policy U, compute the unique solution $\{g(U), v(U)\}$ to the following system of linear equations:

$$v_{\sigma_i} = c_{\sigma_i}(U_{\sigma_i}) - g + \sum_{\sigma_j \in S} p_{\sigma_i \sigma_j}(U_{\sigma_i}) v_{\sigma_j}, \quad \sigma_i \in S \tag{3.20}$$

$$v_{\sigma_s} = 0,$$

   where σ_s is an arbitrarily chosen state.
3. *Step: Policy-improvement step* For each state $\sigma_i \in S$, determine a portfolio x_k yielding the minimum in

$$\min_{x_k \in X_{\sigma_i}} \left\{ c_{\sigma_i}(x_k) - g(U) + \sum_{\sigma_j \in S} p_{\sigma_i \sigma_j}(x_k) v_{\sigma_j}(U) \right\} \tag{3.21}$$

   The new stationary policy U' is obtained by setting $U'_{\sigma_i} = x_k$ for all $\sigma_i \in S$.
4. *Step: Convergence test* If the new policy U' equals U the algorithm is stopped with policy U. Otherwise, set $U = U'$ and go to step 2.

Policy-iteration algorithm converges to the optimum after a finite number of iterations and is empirically found to be a robust algorithm. However, the finite convergence of the algorithm is only warranted when the following assumption holds: For each stationary policy U, there exists a state r, that can be reached from any other state. This state r may depend on U (Tijms, 1986). The validity of this assumption is not easy to show for the model. However, we need to consider the assumption only if the policy-iteration algorithm does not converge.

3.5.2 Policy-iteration algorithm applied to the model

We will next explain how the policy-iteration algorithm can be applied to the 5-component-system. First we form a stationary policy by simply choosing the cheapest maintenance portfolio (according to (3.15)) from feasible

portfolios X_{σ_i} for every state $\sigma_i \in S$. These costs are collected to the vector $c \in \mathbb{R}^{h(n+1)}$.

In the value-determination step, the average cost $g(U) \in \mathbb{R}$ and the values of the states $v(U) \in \mathbb{R}^{h(n+1)}$ are computed from the system of linear equations presented in (3.20). This system can be presented in matrix form:

$$v = c - ge + Pv.$$

Here, we define $e \in \mathbb{R}^{h(n+1)}$ with $e_i = 1$ for all $1 \leq i \leq h(n+1)$ and $P \in \mathbb{R}^{h(n+1)\times h(n+1)}$ as the state transition matrix for policy U where, on every row, only six entries are non-zero. These entries are calculated with (3.14). Following Howard (1960), we choose the last state $\sigma_{h(n+1)}$ to represent the arbitrary state σ_s in the normalization equation. When setting $v_{\sigma_{h(n+1)}} = 0$, the last row reduces to

$$0 = c_{\sigma_{h(n+1)}} - g(U) + \sum_{\sigma_j \in S} p_{\sigma_{h(n+1)}\sigma_j}(U_{\sigma_{h(n+1)}}) v_{\sigma_j}. \tag{3.22}$$

When this row is reduced from other rows, we get a system of linear equations of the form

$$v_{\sigma_i} = c_{\sigma_i}(U_{\sigma_i}) - c_{h(n+1)} + \sum_{\sigma_j \in S} v_{\sigma_j} \left(p_{\sigma_i\sigma_j}(U_{\sigma_i}) - p_{\sigma_{h(n+1)}\sigma_j}(U_{\sigma_{h(n+1)}}) \right)$$

for all $1 \leq i \leq h(n+1) - 1$. Thus, $g(U)$ is eliminated and we have a linear system of the form $\vec{x} = \vec{b} + A\vec{x}$ where $A \in \mathbb{R}^{(h(n+1)-1)\times(h(n+1)-1)}$ because $v_{h(n+1)} = 0$. From this reduced linear system the remaining $h(n+1) - 1$ unknowns can be solved. The system has a unique solution $\vec{x} = (I - A)^{-1}\vec{b}$. After the calculation of $v(U)$, the average cost $g(U)$ can be obtained with (3.22).

In the policy-improvement step for every state $\sigma_i \in S$, the value of (3.21) is calculated for every feasible portfolio $x_k \in X_{\sigma_i}$ using solutions from the previous step. The portfolio producing the lowest value is chosen for state σ_i in the policy U'. In the fourth step, policy U' is compared with the policy set in the initialization test. If there are any differences among states, the first policy is upgraded to U' and the policy-iteration continues from the second step.

In the 5-component-system the policy-iteration algorithm converged during 11th iteration to the optimal policy U'. For the optimal policy, the long-run average maintenance cost is $g(U') = 223.84$. For the initial policy the long-run average maintenance cost is $g(U) = 238.42$. Table 3.3 shows initial

and optimal maintenance portfolios of some randomly chosen states. In some states the initial policy is the optimal one.

Table 3.3: Examples of differences between initial and optimal policies

state σ_i		initial policy $x_k = U_{\sigma_i}$	optimal policy $x_k = U'_{\sigma_i}$
$(8, 12, 1, 1, 1;$	$0, 0, 0, 1, 0)$	$(1, 0, 0, 1, 0)^T$	$(1, 1, 0, 1, 0)^T$
$(1, 4, 1, 4, 2;$	$0, 0, 0, 0, 0)$	$(0, 0, 0, 1, 0)^T$	$(0, 0, 1, 1, 0)^T$
$(3, 16, 2, 1, 3;$	$0, 1, 0, 0, 0)$	$(1, 1, 0, 0, 0)^T$	$(1, 1, 0, 0, 0)^T$
$(2, 2, 3, 3, 1;$	$0, 0, 1, 0, 0)$	$(0, 0, 1, 0, 0)^T$	$(0, 0, 1, 0, 0)^T$
$(1, 6, 1, 3, 4;$	$0, 0, 0, 1, 0)$	$(0, 0, 0, 1, 0)^T$	$(0, 0, 0, 1, 1)^T$
$(5, 15, 1, 2, 1;$	$0, 0, 1, 0, 0)$	$(1, 0, 1, 0, 0)^T$	$(1, 1, 1, 0, 0)^T$
$(3, 9, 1, 1, 3;$	$1, 0, 0, 0, 0)$	$(1, 0, 0, 0, 0)^T$	$(1, 1, 0, 0, 0)^T$
$(4, 6, 1, 1, 6;$	$0, 0, 0, 0, 1)$	$(0, 0, 0, 0, 1)^T$	$(1, 1, 0, 0, 1)^T$

3.5.3 Policy-iteration with discounting

It is customary to discount future cash flows (e.g. Luenberger, 1998). In practise this means that the recursion equation (3.18) is rewritten as

$$V_n(\sigma_i, U) = c_{\sigma_i}(U_{\sigma_i}) + \beta \sum_{\sigma_j \in S} p_{\sigma_i \sigma_j}(U_{\sigma_i}) V_{n-1}(\sigma_j, U) \quad \forall \sigma_i \in S, \tag{3.23}$$

where $0 \leq \beta < 1$ is the discount factor. Now, the future maintenance costs over $n-1$ maintenance instances are discounted. If $\beta = 1$, equations (3.23) and (3.18) are the same.

Policy-iteration algorithm with discounting (Howard, 1960) solves the optimal stationary policy for the Markov decision process with discounting. First we write the equation (3.23) in a simple vector form:

$$v(n+1) = c + \beta P v(n)$$

If we write different $v(i)$:s in explicit form, we get

$$\begin{aligned} v(1) &= c + \beta P v(0) \\ v(2) &= c + \beta P v(1) = c + \beta P c + \beta^2 P^2 v(0) \\ v(3) &= c + \beta P c + \beta^2 P^2 c + \beta^3 P^3 v(0) \\ &\text{etc.} \end{aligned}$$

The general form is thus

$$v(n) = \left[\sum_{j=0}^{n-1}(\beta P)^j\right] c + \beta^n P^n v(0).$$

Since $0 \leq \beta < 1$,

$$\lim_{n\to\infty} v(n) = \sum_{j=0}^{\infty}(\beta P)^j c.$$

Matrix P describes a Markov decision process so it is a state transition probability matrix with eigenvalues less or equal to 1 in magnitude. Thus, matrix βP has eigenvalues strictly less than 1 in magnitude, so we can write $\sum_{j=0}^{\infty}(\beta P)^j = (I - \beta P)^{-1}$ (Howard, 1960) and finally obtain

$$\lim_{n\to\infty} v(n) = v = (I - \beta P)^{-1} c. \tag{3.24}$$

This equation is almost the same as the value determination equation (3.20). Equation (3.24) does not include the long-run average cost $g(U)$ but it is not a meaningful measure in the case of discounting where future costs are less and less meaningful. The component v_i of vector v tells the total expected maintenance cost of the system when it is in state $i \in S$ in the beginning. When vector v is obtained, the policy-improvement step is analogous to the case without discounting.

As a result, we have an algorithm for discounted policy-iteration (Howard, 1960):

1. *Step: Initialization* Choose a stationary policy U.
2. *Step: Value-determination step* For the current policy U, compute the unique solution $v(U)$ to the following system of linear equations:

$$v_{\sigma_i} = c_{\sigma_i}(U_{\sigma_i}) + \beta \sum_{\sigma_j \in S} p_{\sigma_i \sigma_j}(U_{\sigma_i}) v_{\sigma_j}, \quad \sigma_i \in S. \tag{3.25}$$

3. *Step: Policy-improvement step* For each state $\sigma_i \in S$, determine a portfolio x_k yielding the minimum in

$$\min_{x_k \in X_{\sigma_i}} \left\{ c_{\sigma_i}(x_k) + \beta \sum_{\sigma_j \in S} p_{\sigma_i \sigma_j}(x_k) v_{\sigma_j}(U) \right\} \tag{3.26}$$

The new stationary policy U' is obtained by setting $U'_{\sigma_i} = x_k$ for all $\sigma_i \in S$.

4. *Step: Convergence test* If the new policy U' equals U the algorithm is stopped with policy U. Otherwise, set $U = U'$ and go to step 2.

3.5.4 Discounted policy-iteration algorithm applied to the model

We illustrate the use of the discounted policy-iteration algorithm with the 5-component-system. Again we initialize the stationary policy by choosing the cheapest maintenance portfolio from feasible portfolios X_{σ_i} for every state $\sigma_i \in S$. We then determine the values of (3.25) by calculating v from $v = (I - \beta P)^{-1}c$. In the policy-improvement step we calculate the value of (3.26) for every feasible portfolio and choose the one with lowest value. This is the new portfolio for state σ_i and it updates the policy. Last, we compare the new policy with the old one and determine if the algorithm has converged.

Table 3.4 shows some results of the algorithm with three different discount factors. Here the discount factor tells how much we discount costs between two consecutive maintenance instances. The algorithm converged with $\beta = 0.9$ during seven iterations, with $\beta = 0.8$ during six iterations and with $\beta = 0.7$ during five iterations. The states are the same than in Table 3.3. When we compare the cases of maintenance without and with discounting, the optimal policies do not differ much (at least when these states are considered). Also the size of the discount factor does not affect the decisions much but a smaller discount factor favors postponing maintenance decisions.

Table 3.4: Effect of discount factor

state		optimal policies		
σ_i		$\beta = 0.9$	$\beta = 0.8$	$\beta = 0.7$
$(8, 12, 1, 1, 1;$	$0, 0, 0, 1, 0)$	$(1, 1, 0, 1, 0)^T$	$(1, 1, 0, 1, 0)^T$	$(1, 1, 0, 1, 0)^T$
$(1, 4, 1, 4, 2;$	$0, 0, 0, 0, 0)$	$(0, 0, 0, 1, 1)^T$	$(0, 0, 0, 1, 1)^T$	$(0, 0, 0, 1, 1)^T$
$(3, 16, 2, 1, 3;$	$0, 1, 0, 0, 0)$	$(1, 1, 0, 0, 0)^T$	$(1, 1, 0, 0, 0)^T$	$(1, 1, 0, 0, 0)^T$
$(2, 2, 3, 3, 1;$	$0, 0, 1, 0, 0)$	$(0, 0, 1, 0, 0)^T$	$(0, 0, 1, 0, 0)^T$	$(0, 0, 1, 0, 0)^T$
$(1, 6, 1, 3, 4;$	$0, 0, 0, 1, 0)$	$(0, 0, 0, 1, 1)^T$	$(0, 0, 0, 1, 0)^T$	$(0, 0, 0, 1, 0)^T$
$(5, 15, 1, 2, 1;$	$0, 0, 1, 0, 0)$	$(1, 1, 1, 0, 0)^T$	$(1, 1, 1, 0, 0)^T$	$(1, 1, 1, 0, 0)^T$
$(3, 9, 1, 1, 3;$	$1, 0, 0, 0, 0)$	$(1, 1, 0, 0, 0)^T$	$(1, 1, 0, 0, 0)^T$	$(1, 1, 0, 0, 0)^T$
$(4, 6, 1, 1, 6;$	$0, 0, 0, 0, 1)$	$(0, 0, 0, 0, 1)^T$	$(0, 0, 0, 0, 1)^T$	$(0, 0, 0, 0, 1)^T$

Chapter 4

Convergence of the algorithm

When using the policy-iteration algorithm to solve the maintenance scheduling problem, in practise, it is useful to know how effective the method is. One measure for that is the computation time of the algorithm and how changing different parameters affect the convergence of the algorithm. The parameters can be divided into those that affect the size of the state space, and others. This chapter focuses on the solution times when different parameters of the model are changed.

4.1 Size of the state space

The size of the state space is a key factor in time complexity because it defines the size of the probability matrix P which must be inverted during every value determination step. The size depends on the number of components and possible age combinations. The number of possible age combinations h depends on three things:

1. Reliability threshold ρ.
2. Maintenance interval Δt.
3. Failure distributions of components f_i:s.

We look at these parameters separately to see how much they affect the size of the state space and the solution times.

The hardware used is a laptop with 2.40GHz AMD A6-9210 RADEON R4 processor with 5 compute cores, 8Gb of RAM and operating system of Windows 10. The computations are implemented with MATLAB R2019b. The

linear systems (3.20) are solved with conjugate gradients squared method which is built in MATLAB. The tolerance of the method is set to 1×10^{-6} and the maximum number of iterations is set to 500.

Effect of reliability threshold

First we vary the size of the state space by using different reliability thresholds. Other parameters are kept constant. We use the 5-component system from Chapter 3 with the same cost structure, failure time distributions and maintenance interval $\Delta t = 1$. We use both the policy iteration and the discounted policy iteration with discount factor $\beta = 0.95$. The reliability threshold is varied between $0.88 - 0.93$. With a lower threshold the system can withstand older components so the allowed age combinations of components h increases. Then also the size of the state space $|S| = h(n+1)$ (equation (3.13)) increases.

The computation times of the policy iteration algorithm in seconds are given in Table 4.1. T_i is the initialization time. During it the directed graph of the system is formed and the costs of different maintenance portfolios are calculated using Edmond's algorithm. In addition, the allowed age combinations of components are determined and the state space and different actions (maintenance portfolios) are stored in a data matrix. Also the first stationary policy is formed which is the initialization step of the policy iteration algorithm (Section 3.5.1). Next, T_{total} tells the total computation time of all the policy iteration algorithm steps (2)-(4) performed until the convergence is obtained. The row i_c is the iteration counter telling how many times the stationary policy must be updated until convergence. Last, the $T_{\text{avg}} = \frac{T_{\text{total}}}{i_c}$ tells the average time of one iteration.

Table 4.1 shows that the size of the state space grows rapidly when the reliability threshold decreases. The number of iterations remains approximately constant but the solution time of one iteration increases when the state space grows. This is reasonable because systems of linear equations become bigger and more time consuming to compute. It is also interesting that with $\rho = 0.88$ and $|S| = 35088$ MATLAB cannot run the code because the transition matrix of that size is too big compared to available RAM.

Table 4.1: Effects of the size of the state space on computation time without discounting and with discounting ($\beta = 0.95$)

	reliability threshold	ρ	0.93	0.92	0.91	0.90	0.89	0.88
	age combinations	h	481	910	1591	2597	3980	5848
	size of state space	$\|S\|$	2886	5460	9546	15582	23880	35088
no disc.	initialization	T_i	4.3	6.9	9.1	10.8	14.5	
	total time	T_{total}	8.7	29.7	93.7	310.9	1344.1	
	iterations	i_c	9	10	10	11	10	
	average time	T_{avg}	1.0	3.0	9.4	28.3	134.4	
with disc.	initialization	T_i	4.3	6.9	9.1	10.8	14.5	
	total time	T_{total}	5.4	19.3	60.3	184.1	854.8	
	iterations	i_c	6	7	7	7	8	
	average time	T_{avg}	0.9	2.8	8.6	26.3	105.7	

The results of the policy iteration algorithm with discounting are in lower part of Table 4.1. The main difference is that the discounted version needs less iterations to converge. Thus it is the faster version of the policy iteration algorithms.

Effect of number of components

Next we increase the number of components (n) from five to six and seven. The new components do not have structural dependencies between other components. They are only connected to root node with component specific replacement costs $c_{06} = 105$ and $c_{07} = 100$. The corrective replacement cost is 95 for both components. The PDFs of failure time of components are linear and increasing (like in the 5-component-system) with maximal age parameters $\alpha_6 = 15$ and $\alpha_7 = 14$.

The results of the bigger systems are in Table 4.2. We use three different reliability thresholds. One can notice that the number of possible age combinations increases when the number of components increases. The size of the state space grows even faster because the number of possible failures $(n + 1)$ increases. The initialization time T_i increases because the computational complexity of finding all the possible age combinations depends exponentially on n. Also, the average time of one iteration T_{avg} increases.

The solution times increase for example, because the policy improvement step (3.21) takes more time because we have more maintenance portfolios to consider. In the 5-component case the total number of possible portfolios is 24. When the sixth component, independent of other components, is added to the system, it doubles the number of possible maintenance portfolios to 48. The same happens when the seventh component is added to the system, resulting in 96 different portfolios. Of course, not all of the portfolios are feasible for every state.

Table 4.2: Effects of the number of components

number of components	n	6			7		
reliability threshold	ρ	0.93	0.92	0.915	0.93	0.92	0.915
age combinations	h	663	1501	2154	666	1780	2774
size of state space	$\|S\|$	4641	10507	15078	5328	14240	22192
initialization	T_i	29.5	36.9	39.1	324.7	379.5	386.1
total time	T_{total}	18.8	89.1	236.6	35.2	141.3	2656.7
iterations	i_c	8	7	9	10	7	10
average time	T_{avg}	2.4	12.7	26.3	3.5	20.2	265.7

Effect of maintenance interval and failure distributions

The maintenance interval (Δt) has a big effect on the size of the state space. For example, shortening the maintenance interval by 50% in the 5-component system with $\rho = 0.92$ leads to $h_2 = 472518$, making the problem too big to solve. Conversely, doubling the maintenance interval leads to $h_3 = 0$. This means that a brand new system does not operate the first maintenance interval with probability greater or equal to $\rho = 0.92$. The effect of maintenance interval is covered more in Section 5.1.

The failure distributions of components have an impact on the size of the state space. If a failure of a component becomes less likely, the number of allowed age combinations increases. Thus, the state space becomes bigger. The number of different possibilities is limitless so we do not handle it more specifically here.

4.2 Other parameters

Other parameters of the model are for example

- number of arcs in the graph $|A|$,
- set-up cost c_0,
- weights of the arcs i.e. component specific replacement costs c_{ij}:s,
- corrective replacement surpluses r and
- discount factor β.

We can make observations of the convergence of the policy-iteration algorithm by changing these values, for example set-up cost and corrective replacement surpluses.

The effects of the other parameters are more difficult to see because they do not impact the size of the state space and solution times directly. For example, the 5-component-system has a set-up cost $c_0 = 60$ and corrective replacement surpluses $r = (120, 90, 85, 70, 90)^T$. The policy-iteration algorithm is implemented so that changing these does not affect the time duration of one iteration, because the size of the state space stays the same. However, these costs may affect the convergence of the algorithm by changing the number of iterations needed to find the optimal stationary policy. We use the 5-component-system like in the previous section. Now, we fix the reliability threshold to $\rho = 0.92$ and maintenance interval to $\Delta t = 1$, so $|S| = 5460$.

First, we change the set-up cost. Other parameters are kept constant. The results with six different set-up costs are in Table 4.3. From the table we can see that the initialization times and average times do not change. Conversely, the total time decreases when the set-up cost increases. This happens, because the number of iterations decreases. A possible reason for this is that the policy iteration algorithm finds the optimal policy more quickly, when the high set-up cost encourages replacing more components at the same time. Thus, the system is more often in the situation where no components needs replacing to satisfy the reliability threshold. The fact that the average cost grows more moderately than the set-up cost supports this statement. For example, when the set-up cost increases from 200 to 400, the average cost increases only from 371.64 to 490.84 indicating that we want to maintain the system only at every other maintenance instance. The difference is more close to 100 with bigger set-up costs.

Table 4.3: Effects of the set-up cost

set-up cost	c_0	0	60	200	400	600	800
average cost	$g(U)$	202.92	262.84	371.64	490.84	592.30	693.75
initialization	T_i	6.2	5.4	6.7	6.5	5.4	6.2
total time	T_{total}	24.1	27.6	15.0	10.9	8.4	9.0
iterations	i_c	9	10	6	4	3	3
average time	T_{avg}	2.7	2.8	2.5	2.7	2.8	3.0

From Table 4.4 we can see how the recommendations of the optimal policy focus on fewer portfolios, which replace many components, when the set-up cost increases. It is noteworthy that we have chosen the failure distributions and reliability threshold so that it is not feasible to try to maintain the system every third instance. If we have $\sigma_i = (2, 2, 2, 2, 2;\, 0, 0, 0, 0, 0)$ and we do not replace any component, the state $\sigma_j = (3, 3, 3, 3, 3;\, 0, 0, 0, 0, 0)$ is reached with probability $p_{\sigma_i\sigma_j}(\emptyset) = 0.8894 < \rho$.

Table 4.4: The number of states maintained with different portfolios under different set-up costs when looking the 5-component system with $\rho = 0.92$ and $\Delta t = 1$

maintenance	set-up cost					
portfolio	0	60	200	400	600	800
10000	50	50	25	8	0	0
00100	357	357	216	0	0	0
00010	346	313	78	0	0	0
00001	233	207	148	82	0	0
11000	909	921	970	384	0	0
10100	64	61	37	0	0	0
10010	115	48	0	0	0	0
10001	72	72	0	0	0	0
00110	299	171	8	0	0	0
00101	660	610	198	0	0	0
00011	702	623	0	0	0	0
11100	1033	1180	1159	0	0	0
11010	337	125	0	0	0	0
11001	215	215	0	0	0	0
10110	0	23	109	124	124	124
10101	4	0	0	0	0	0
10011	0	0	0	0	0	0
00111	45	242	420	426	426	426
11110	4	219	782	1493	1493	1493
11101	0	0	0	0	0	0
11011	0	0	0	0	0	0
10111	0	4	458	1413	1553	1553
11111	0	4	837	1515	1849	1849
00000	15	15	15	15	15	15

Next, we change the corrective replacement cost. Other parameters are kept constant and the set-up cost is again $c_0 = 60$. We use different multiples of vector $r = (120, 90, 85, 70, 90)^T$ from $0r = (0, 0, 0, 0, 0)^T$ to $6r = (720, 540, 510, 420, 540)^T$. The results are in Table 4.5, which shows that the effect of the surpluses is not as significant as the set-up cost. The number of iterations stays somewhat the same so the total running times do not change much. Also, the average costs grow very slowly. This tells that the role of the corrective replacement surpluses is not significant, especially in this case where failures of the components are unlikely due to the high reliability

threshold.

Table 4.5: Effects of the corrective replacement surplus

corrective surplus	r	$0r$	$2r$	$3r$	$4r$	$5r$	$6r$
average cost	$g(U)$	256.50	269.18	275.53	281.86	288.20	294.53
initialization	T_i	5.2	5.9	5.7	5.9	5.5	5.3
total time	T_{total}	24.0	27.5	27.7	24.5	21.9	22.3
iterations	i_c	9	10	10	9	8	8
average time	T_{avg}	2.7	2.8	2.8	2.7	2.7	2.8

In addition to the set-up cost and corrective replacement surpluses we examine the effect of discount factor. The results are in Table 4.6. There is some decrease in the number of iterations with smaller discount factors but significant reductions happen only with factors which are too small in practise. The significant difference is between $\beta = 1$ and $\beta = 0.99$ but as explained in Sections 3.5.1 and 3.5.3 these algorithms have some differences which affect the solution times.

Table 4.6: Effects of the discount factor

discount factor	β	1	0.99	0.98	0.95	0.90	0.85	0.80
initialization	T_i	6.9	5.4	5.4	6.1	5.5	5.6	5.9
total time	T_{total}	29.7	25.2	19.8	18.9	17.3	14.0	11.4
iterations	i_c	10	9	7	7	7	6	5
average time	T_{avg}	3.0	2.8	2.8	2.7	2.5	2.3	2.3

4.3 Conclusions

Perhaps the most important observation from this chapter is how the size of the state space affects the iteration times. This means that reliability threshold, maintenance interval and number of components are all significant parameters defining the solution time of the algorithm. However, the decision maker should focus on the combined effect of these parameters. If the size of the state space grows too large, the calculations become impossible with standard computers.

Other parameters are less significant because they affect the number of policy iterations, not on the solution times of one iteration. The effect of discount

factor is small, slightly decreasing solution times, when it is kept at realistic values. The set-up cost is more important than the corrective replacement surpluses. We should also remember that it may be possible that these effects do not happen with every system. In general it seems that when a system is modeled as a Markov decision process and solved with policy-iteration, the effects of the size of the state space are similar regardless of a system. In the end, the system defines how fast these changes are.

Chapter 5

Case example

This chapter introduces a real life example of a ground transportation equipment system to be maintained. The system has four different components: engine 1 (E1), engine 2 (E2), chassis (C) and wheels (W). These components deteriorate over time and have structural dependencies. An engine must be dismantled before it can be replaced. In order to replace the chassis, the chassis and both engines must be dismantled first. Finally, in order to replace the wheels, the chassis and the engines must be dismantled.

Economic dependencies of the system are positive. For every maintenance operation there is a fixed set-up cost $c_0 = 388$. Also some dismantling costs can be avoided if multiple components are maintained at the same time. Table 5.1 shows the component specific costs.

Table 5.1: Maintenance costs of different components

		component specific costs		
component	symbols	dismantle	replacement	corrective surplus
engine 1	1, E1	23	393	300
engine 2	2, E2	28	403	300
chassis	3, C	167	413	160
wheels	4, W	0	1000	613

When the structural dependencies are taken into account, the system is presented with a directed graph of Figure 5.1. In addition to the root node and the component nodes, the figure also presents a node named 'DE12'. This is a decision to dismantle both engines which is a prerequisite for replacing either the chassis or the wheels. The weights of the arcs are directly obtained

from Table 5.1. For example, when both engines are dismantled the replacement of wheels costs the wheels replacement cost and the dismantle cost of the chassis. Thus, $c_{\mathrm{DE12,W}} = 1000 + 167 = 1167$.

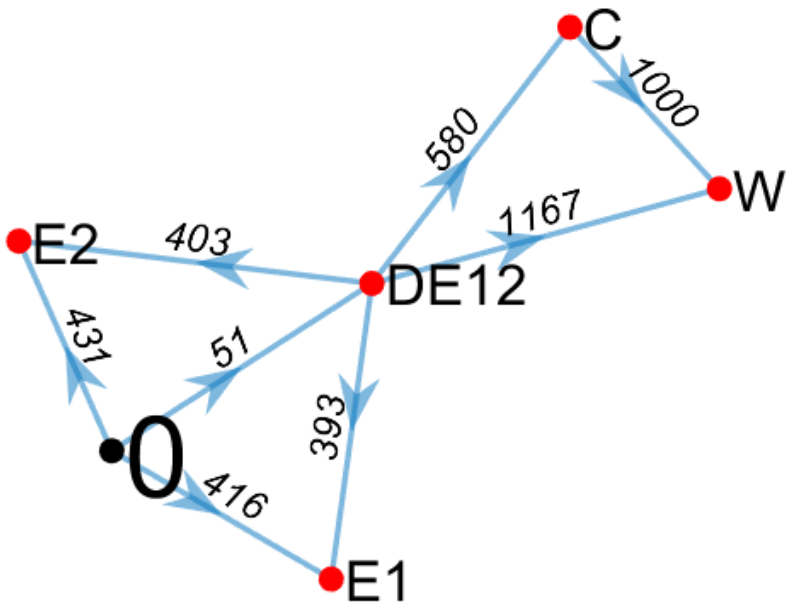


Figure 5.1: Cost structure of the system where the root node is on the left

The components deteriorate when being used. We use Δt to represent distance, because it describes the use of the moving system more reliably in practice. So the maintenance instances, ages of the components and the failure distributions of components are measured with distances.

We choose Weibull distributions to describe the failures of components. The probability density function of a Weibull random variable is

$$f(x, \lambda, k) = \begin{cases} \frac{k}{\lambda} \left(\frac{x}{\lambda}\right)^{k-1} e^{-(\frac{x}{\lambda})^k} & x \geq 0, \\ 0 & x < 0, \end{cases}$$

where $k > 0$ is the shape parameter and $\lambda > 0$ is the scale parameter of the distribution. For the components we use parameters presented in Table 5.2.

Table 5.2: Weibull distribution parameters of components

component	shape k	scale λ
engine 1	5.1	10.8
engine 2	5.1	10.8
chassis	5.5	9.9
wheels	4.0	9.0

Engines 1 and 2 have equal failure distributions. The parameters are scaled so that the unit distance is 100 000km. All the components have shape parameters greater than one so they obey increasing failure rate. The probability density functions are plotted for comparison in Figure 5.2.

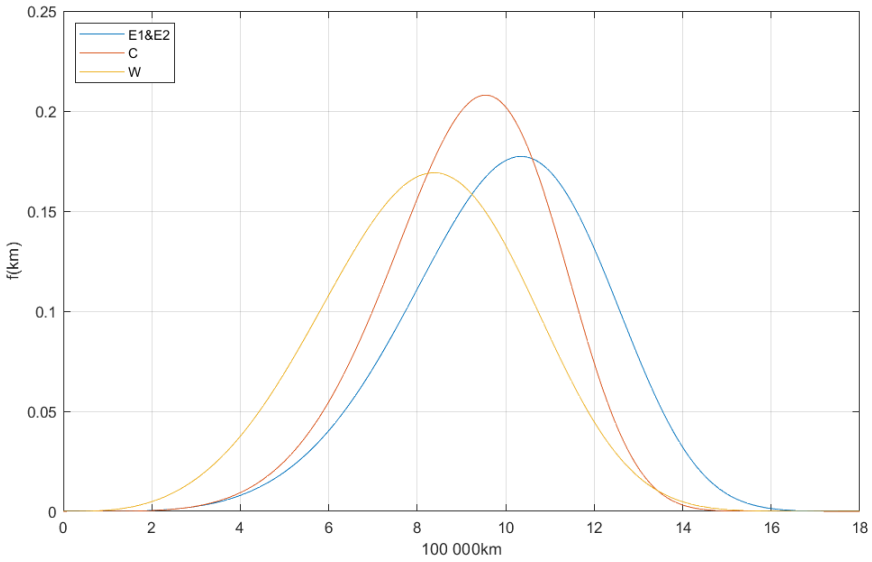


Figure 5.2: Failure probability density as a function of distance driven from last replacement

To understand the problem better we first calculate the size of the state space with different reliability thresholds and maintenance intervals. We use reliability thresholds 0.90–0.95 with a step size 0.01 and distances 50 000km–150 000km with a step size 5000km. These results are in Table 5.3. The table shows that when the maintenance interval is 75tkm or higher we can keep the size or the problem possible to calculate with all the reliability

thresholds. Thus, we choose to examine maintenance intervals of 75tkm, 100tkm, 125tkm and 150tkm with all the six reliability thresholds giving us 24 different scheduling problems to solve. These are bolded in the Table 5.3. We refer to these policies with $U(\Delta t(\text{tkm}), \rho)$.

Table 5.3: Size of the state space with different parameters

maintenance	reliability threshold					
interval	0.95	0.94	0.93	0.92	0.91	0.90
150 000	**375**	**540**	**665**	**685**	**760**	**855**
145 000	490	655	685	795	935	1030
140 000	635	705	850	985	1090	1195
135 000	700	850	1025	1125	1350	1575
130 000	810	1045	1165	1500	1665	1910
125 000	**1055**	**1240**	**1555**	**1820**	**1965**	**2310**
120 000	1265	1620	1900	2195	2485	2765
115 000	1600	1940	2315	2710	3040	3415
110 000	2020	2460	2935	3295	3875	4400
105 000	2575	3125	3680	4305	5040	5470
100 000	**3250**	**4005**	**4870**	**5430**	**6175**	**6905**
95 000	4290	5310	6150	6915	8365	9280
90 000	5590	6690	8245	9475	10610	11820
85 000	7400	9290	10785	12560	14595	16270
80 000	10240	12555	14860	16995	19525	22025
75 000	**14190**	**17270**	**20505**	**23800**	**26545**	**30680**
70 000	20210	24030	29275	33735	38155	42980
65 000	29735	35575	42140	49360	55045	63120
60 000	44130	53030	63695	73335	83365	94485
55 000	68410	82935	98200	113345	130325	146380
50 000	110865	134785	159065	185315	209725	237555

We discount future costs by assuming an annual interest rate of 1%. We assume that the system travels approximately 200tkm during one year. This means that if the maintenance interval is 100tkm, we will maintain the system every six months. Because every maintenance interval maintains the system more than once a year, the corresponding discount factor is calculated with

$$\beta_x = \left(\frac{1}{1+0.01}\right)^{\frac{x}{200}}.$$

The results are $\beta_{75} = 0.9963$, $\beta_{100} = 0.9950$, $\beta_{125} = 0.9938$ and $\beta_{150} = 0.9925$ for the chosen maintenance intervals.

The 24 maintenance scheduling problems were solved using discounted policy-iteration algorithm presented in Section 3.5.3. The algorithm is implemented in MATLAB R2019b, but now we use a desktop with 3.40GHz Intel Xeon CPU E3-1250 v5, 16Gb RAM and Windows 10 as operating system to make the computations faster and bigger problems possible to calculate. The total solving time for the biggest problem $U(75, 0.90)$ is approximately 742 seconds and approximately 0.49 seconds for the smallest problem $U(150, 0.95)$.

5.1 Results

To present the results in a simpler form we introduce a short notation to the binary decision vector $z(x_k) \in \mathbb{R}^n$. The decision vector can be presented as a string of zeros and ones. For example if we replace components 2 and 3 from a 4-component system we have decision vector $(0, 1, 1, 0)^T$ which corresponds to the string 0110. In addition, if we do not replace anything, the decision is 0000 which we will also present as a 0.

5.1.1 Different details

First we examine the effects of different reliability thresholds on the maintenance scheduling policy. Table 5.4 presents maintenance portfolios under two different policies as a function of $(a_k)_{E2}$ (rows) and $(a_k)_W$ (columns), when the age of other components are fixed to $(a_k)_{E1} = (a_k)_C = 75$, the failure state is $f_k = (0, 0, 0, 0)^T$ and the maintenance interval is $\Delta t = 75$tkm for both policies. The table shows how much sooner the components should be replaced with the bigger reliability threshold. Also, both policies recommend not to replace components E1 or C because their age is only 75tkm meaning they have been used only during one maintenance interval.

Table 5.4: Comparing replacement portfolios when changing reliability threshold as a function of $(a_k)_{E2}$ and $(a_k)_W$, when $(a_k)_{E1} = (a_k)_C = 75$ and $f_k = 0$

$(a_k)_2$, engine 2	$(a_k)_4$, wheels 75	150	225	300	375	450	525	600
				$\rho = 0.90$				
75	0	0	0	0	0	0	0001	0001
150	0	0	0	0	0	0	0001	0001
225	0	0	0	0	0	0	0001	0001
300	0	0	0	0	0	0	0	0101
375	0	0	0	0	0	0	0	0101
450	0	0	0	0	0	0	0	0101
525	0	0	0	0	0	0	0	0101
600	0100	0	0	0	0	0	0101	0101
675	0100	0100	0100	0	0	0101	0101	
750	0100	0100	0100	0100	0101	0101		
825	0100	0100	0100	0100				
				$\rho = 0.95$				
75	0	0	0	0	0	0001		
150	0	0	0	0	0	0001		
225	0	0	0	0	0	0001		
300	0	0	0	0	0	0001		
375	0	0	0	0	0	0101		
450	0	0	0	0	0	0101		
525	0	0	0	0	0101	0101		
600	0	0	0	0100	0101			
675	0100	0100	0100	0100				

We can also notice the following: when $\rho = 0.90$ and the system is in state $(75, 600, 75, 75;\ 0, 0, 0, 0)$ the policy suggests replacing component E2 but in state $(75, 675, 75, 300;\ 0, 0, 0, 0)$ the policy does not recommend replacing anything although the components are older and therefore the reliability of the system lower. This difference may result from the idea to keep the age of the components at the same level to make grouping of replacements more appealing. As introduced in Chapter 2, these recommendations are based on an opportunistic maintenance policy where some opportunities are seen better than others. So in this case these heuristics would provide optimal maintenance scheduling decisions.

Next, we review this observation in a different context. We consider the

maintenance scheduling policy $U(100, 0.90)$. Table 5.5 presents maintenance portfolios under different failure states as a function of $(a_k)_{E2}$ (rows) and $(a_k)_W$ (columns), when the age of other components are $(a_k)_{E1} = 200$ and $(a_k)_C = 300$.

Table 5.5: Comparing replacement portfolios when changing failure state as a function of $(a_k)_{E2}$ and $(a_k)_W$, when $(a_k)_{E1} = 200$, $(a_k)_C = 300$ and $\rho = 0.90$

engine	wheels						wheels					
2	100	200	300	400	500	600	100	200	300	400	500	600
	$f_k = (1,0,0,0)^T$						$f_k = (0,1,0,0)^T$					
100	1000	1000	1000	1000	1111	1111	0100	1100	0100	1111	1111	1111
200	1000	1100	1000	1111	1111	1111	0100	1100	0100	1111	1111	1111
300	1000	1100	1000	1111	1111	1111	0100	1100	0100	1111	1111	1111
400	1100	1100	1000	1111	1111		0100	1100	0100	1111	1111	
500	1100	1100	1100	1111	1111		0100	1100	0100	1111	1111	
600	1100	1100	1100	1111	1111		0100	1100	0100	1111	1111	
700	1100	1100	1100	1111			0100	1100	0100	1111		
	$f_k = (0,0,1,0)^T$						$f_k = (0,0,0,1)^T$					
100	0010	0010	0011	0011	0011	0011	0011	0011	0011	0011	0011	0011
200	1010	0010	0011	0011	0011	0011	0011	0011	0011	0011	0011	0011
300	0110	0010	0010	1111	1111	1111	0001	0001	0001	0001	0001	0001
400	0110	0110	0010	1111	1111		1111	1111	1111	1111	1111	
500	0110	0110	1111	1111	1111		1111	1111	1111	1111	1111	
600	0110	0110	1111	1111	1111		1111	1111	1111	1111	1111	
700	0110	0110	1111	1111			1111	1111	1111	1111		
	$f_k = (0,0,0,0)^T$											
100	0	0	0	0	0011	0011						
200	0	0	0	0	0011	0011						
300	0	0	0	0	0001	0001						
400	0	0	0	0	1111							
500	0	0	0	0	1111							
600	0100	0	0	1111	1111							
700	0100	1100	0100	1111								

When the system has not failed the policy suggests replacing components more or less when $(a_k)_{E2} \geq 600$ or $(a_k)_W \geq 500$. On the other hand, if one component has failed, the policy tends to recommend replacing other components as well at the same time. For example when wheels fail ($f_k = (0,0,0,1)^T$) the policy almost always suggest replacing the chassis as well which makes sense when considering the cost structure of the system.

The age of the system $a_k = (200, 300, 300, 300)$ is interesting because at that state the opportunity is not used. The policy only recommends replacing the failed component no matter which one it is. This does not happen with

other age combination presented in Table 5.5.

5.1.2 Simulations

We next consider how these 24 scheduling policies work in practice: More specifically, we examine how much we must actually pay to execute the maintenance policy over a finite time period and how likely are we facing component failures. To answer these question we simulate the system using Monte Carlo method over a time period of 25 years. This corresponds to approximately five million kilometers of traveling time. Now the length of the maintenance interval defines how many maintenance instances the system has during those 25 years. When $\Delta t = 75$tkm, we have 67 instances. When $\Delta t = 100$tkm, we have 50 instances. When $\Delta t = 125$tkm, we have 40 instances. When $\Delta t = 150$tkm, we have 33 instances. This means that the last maintenance instance for policy, which has a maintenance interval of $\Delta = 75$tkm, is t_{27} and so on.

The simulation of one policy goes as the chart in Figure 5.3 presents. First, we define Monte Carlo sample size W which tells how many times we simulate the maintenance period of 25 years. Next, we start from the simulation $w = 1$. We initialize a_0, the ages of the components at maintenance instance t_0. We also initialize the total cost $c_{tot} = 0$ and the total amount of failures during the maintenance period $f_{tot} = (0, 0, 0, 0)^T$. Then, we move to the next maintenance instance and simulate if any component failed from failure probabilities (3.14). Then we know the state of the system s_1 at t_1 and the optimal policy U tells which maintenance portfolio x_1 we choose to maintain the system. The discounted cost of the portfolio is added to the total costs: $c_{tot} = c_{tot} + \beta^k c(x_k)$. Also the information about failures is added to the total failures: $f_{tot} = f_{tot} + f_k$. The ages of the component are updated and we move to the next maintenance instance t_2. The process is repeated until the last maintenance instance $t_{k_{end}}$ is reached and handled. After that we save the results $c_{tot,w}$ and $f_{tot,w}$ from simulation w and start a new simulation until we have simulated the maintenance period W times.

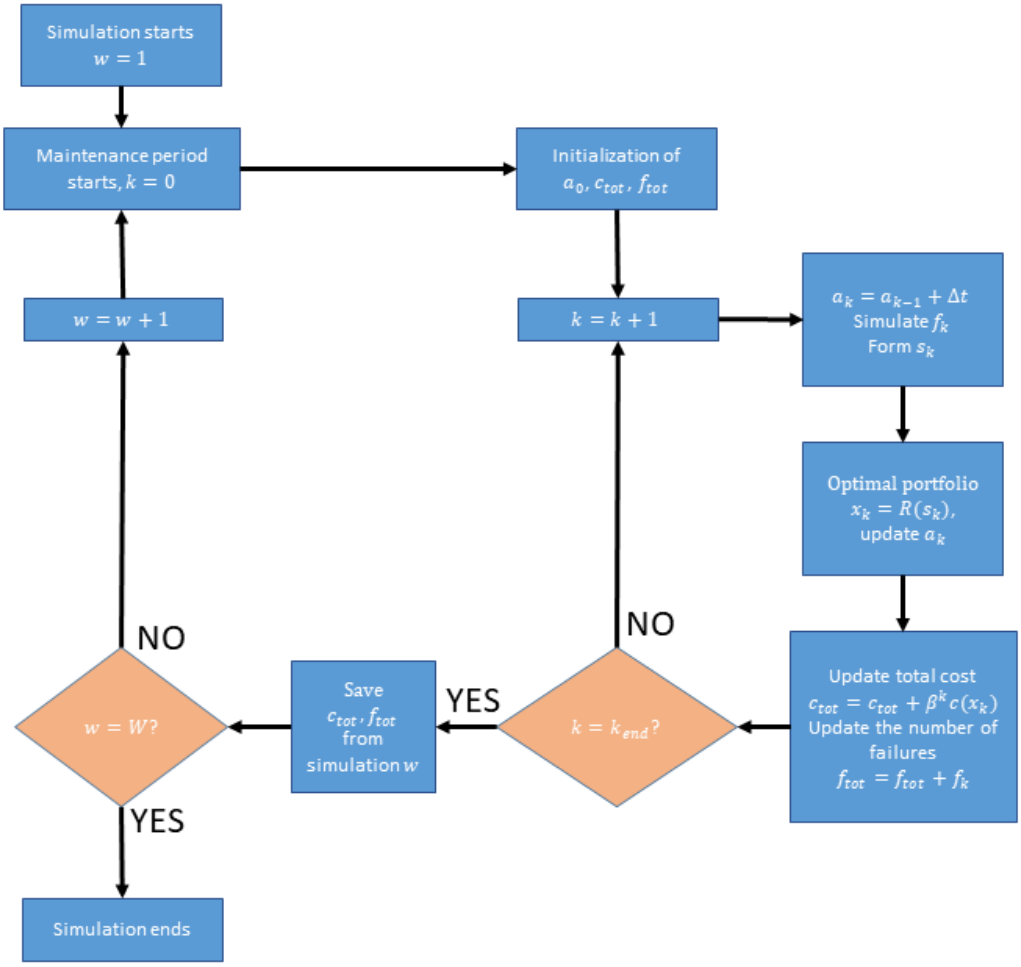


Figure 5.3: Chart on how the simulation proceeds

First, we compare total costs of the 24 policies during the maintenance period with $W = 20000$. We express the cost as cost per thousand kilometers. For one simulation this is calculated from

$$c_w = \frac{c_{tot,w}}{k_{end}\Delta t}.$$

When we calculate the average of the W simulations and do this separately for every policy we get results presented in the Table 5.6.

Table 5.6: Cost of maintenance policy per thousand kilometers during 5 million kilometers (period of 25 years)

reliability	maintenance interval Δt			
threshold ρ	75	100	125	150
0.90	4.67	5.06	5.72	5.57
0.91	5.14	5.46	5.70	5.57
0.92	5.27	5.61	5.81	5.57
0.93	5.47	5.79	6.11	6.36
0.94	5.52	5.97	6.49	6.44
0.95	6.03	6.03	6.49	6.77

Table 5.6 includes interesting results. Mainly the costs increase with a fixed maintenance interval when the reliability threshold increases. This is reasonable because more preventive replacements needs to be done to fulfil the threshold.

The costs increase also when the reliability threshold is fixed and maintenance interval is increased, but here there are also exceptions: With some reliability thresholds, like $\rho = 0.92$, it is cheaper to maintain the system with maintenance interval $\Delta t = 150$tkm than with $\Delta t = 125$tkm. This interval seems to fit better with the failure distributions of components.

Second, we compare the average failures of different components during the maintenance period under different maintenance scheduling policies. We can calculate the average failures of components under a policy with

$$f_{avg} = \frac{1}{W} \sum_{w=1}^{W} f_{tot,w}.$$

When these calculations are done for every scheduling policy we have results presented in the Table 5.7. These results mean that, for example, if we maintain 100 similar ground transportation equipment systems with a policy $U(75, 0.90)$ we will encounter about 22 failures of engine 1 in total during those five million kilometers.

Table 5.7: Average failures of different components during 5 million kilometers (period of 25 years)

reliability	maintenance interval Δt				maintenance interval Δt			
threshold ρ	75	100	125	150	75	100	125	150
	Engine 1				Engine 2			
0.90	0.22	0.18	0.20	0.12	0.21	0.18	0.19	0.12
0.91	0.53	0.48	0.20	0.12	0.63	0.47	0.27	0.12
0.92	0.39	0.31	0.17	0.12	0.52	0.48	0.17	0.11
0.93	0.15	0.17	0.16	0.13	0.16	0.19	0.17	0.12
0.94	0.12	0.20	0.06	0.11	0.12	0.21	0.06	0.11
0.95	0.27	0.08	0.06	0.11	0.32	0.07	0.06	0.11
	Chassis				Wheels			
0.90	0.25	0.22	0.21	0.13	0.99	0.88	0.88	0.65
0.91	0.21	0.17	0.21	0.13	0.85	0.76	0.83	0.64
0.92	0.22	0.18	0.21	0.13	0.87	0.76	0.89	0.64
0.93	0.16	0.21	0.19	0.13	0.73	0.88	0.91	0.66
0.94	0.14	0.20	0.06	0.13	0.66	0.83	0.39	0.66
0.95	0.11	0.08	0.06	0.13	0.53	0.47	0.38	0.66

First, we notice that the average failures tend to decrease when either the maintenance interval or the reliability threshold increases. Both changes increase the number of preventive replacements so this should indeed happen.

Second, we notice that the average failures of engine 1 and 2 seem quite similar. This is reasonable because the engines follow the same probability distributions. There are also differences between engines. Under some policies, for example $U(75, 0.92)$, the engine 1 fails less than engine 2. This may result from the fact that, because we must replace components preventively to fulfil reliability threshold, we more often choose engine 1 because it is a little cheaper to replace.

Third, the chassis fails more often than both engines in more than half of the policies. This is reasonable because the early failures are more likely for chassis than for the engines. Also the number of chassis failures do not change so much than the number of engine failures under different policies.

Fourth, the wheels fail on average more than any other components. This probably results from two reasons. First, wheels have a failure distribution which makes them fail more likely at younger age than any other component. Second, the replacement cost of wheels is quite big due to structural dependencies so we do not want to maintain it often and are thus willing

to take a risk within the reliability threshold. But in the end the wheels need replacement often and therefore the policy utilizes often the positive economic dependence and replaces the chassis at the same time. We can see this for example from Table 5.5. This explains why the number of chassis failures is low overall.

5.1.3 Comparing with opportunistic scheduling policy

Next we find out how the scheduling policy works when it is compared with a simple and heuristic opportunistic policy. The policy is based on optimal maintenance age derived by Wildeman et al. (1997) for Weibull distributed failures. This age is component specific and takes into account only the costs of replacing the component individually. The optimal replacement age for component i is

$$x_i = \sqrt[k_i]{\frac{(c_i^p + c_0)\lambda_i^{k_i}}{c_i^c(k_i - 1)}}, \tag{5.1}$$

where k_i and λ_i are shape and scale parameters of the component, c_0 is the set-up cost, c_i^p is the preventive replacement cost and c_i^c is the corrective replacement cost of the component. From Table 5.1 we can calculate that, for example for wheels $c_W^p = 1218$, because every other component must be dismantled before replacement of wheels is possible, and $c_W^c = c_W^p + r_W = 1831$. Similarly we define the costs for other components and after using equation (5.1) we have the following optimal replacement ages: $x_{E1} = 8.378$, $x_{E2} = 8.374$, $x_C = 7.908$ and $x_W = 6.586$.

We use the optimal replacement ages to define the opportunistic maintenance policy. We set an opportunistic replacement threshold to

$$x_i^{op} = (1 - p)x_i.$$

Here p is a percentage which tells how much earlier a component can be replaced from its replacement age. This percentage is same for all components.

We replace components at a maintenance instance if there is a reason to do so. There are three possible reasons:

1. There are components in the system whose age exceeds x_i.
2. One of the components has failed during the maintenance interval
3. The system does not satisfy the reliability threshold without replacements.

If scenario one above happens, we replace the components whose age exceed x_i and every other components whose age exceed the opportunistic replacement threshold x_i^{op}. If scenario two happens, we replace the failed component and every other component whose age exceed x_i^{op}. If scenario three happens, we replace all the components whose age exceed x_i^{op}.

However, if the replacements above do not fulfil the reliability threshold, we choose to replace a minimal amount of other components. The selection criteria is how close the age of the component is to x_i^{op} (percentage difference). For example, if replacing just engine 1 does not fulfil the reliability threshold and age of the chassis is closest to its x_i^{op}, meaning

$$\frac{a_3}{x_3^{op}} \geq \frac{a_i}{x_i^{op}}$$

for all i not already chosen to be replaced, we check if the replacements of E1 and C are enough for the reliability threshold to be fulfilled. If so, we replace those and simulate the system to the next maintenance instance. If not, we add the next closest component to the maintenance portfolio and check the reliability. This is repeated until the reliability threshold is fulfilled. Other details of the simulation are executed according to the Figure 5.3.

We run the simulation with different values of p and present some results with values $p \in \{0.2, 0.4, 0.6, 0.8\}$. Table 5.8 presents the costs of the opportunistic policies by showing the percentage difference to the results of the Table 5.6.

We notice that the schedule from policy iteration is cheaper than that of opportunistic scheduling policy with all the policies except for one. The cost-based opportunistic replacement thresholds do not take into account economic dependencies and cost savings which result from grouping components to be maintained. These cost savings are bigger when the maintenance interval and reliability threshold are both large, because we have less opportunities to maintain the system and we need to maintain more components to fulfil the reliability threshold.

Table 5.8: The percentage change in the cost of different maintenance scheduling policies per thousand kilometers during 5 million kilometers (period of 25 years) when using a opportunistic policy instead of the one obtained from policy-iteration.

reliability	maintenance interval Δt			
threshold ρ	75	100	125	150
	$p = 0.2$			
0.90	+ 23.0 %	+ 23.7 %	+ 11.0 %	+ 13.9 %
0.91	+ 17.4 %	+ 18.2 %	+ 14.3 %	+ 15.8 %
0.92	+ 16.6 %	+ 19.1 %	+ 13.5 %	+ 17.6 %
0.93	+ 16.3 %	+ 15.6 %	+ 10.6 %	+ 15.8 %
0.94	+ 20.6 %	+ 17.0 %	+ 9.6 %	+ 15.2 %
0.95	+ 15.9 %	+ 20.1 %	+ 13.2 %	+ 17.1 %
	$p = 0.4$			
0.90	+ 3.1 %	+ 8.6 %	+ 8.2 %	+ 13.9 %
0.91	+ 0.5 %	+ 9.8 %	+ 13.9 %	+ 15.8 %
0.92	+ 6.0 %	+ 11.4 %	+ 13.4 %	+ 17.6 %
0.93	+ 4.0 %	+ 14.6 %	+ 10.6 %	+ 15.8 %
0.94	+ 6.7 %	+ 12.9 %	+ 9.6 %	+ 15.1 %
0.95	+ 6.9 %	+ 18.5 %	+ 13.2 %	+ 17.1 %
	$p = 0.6$			
0.90	+ 0.0 %	+ 1.3 %	+ 13.2 %	+ 1.6 %
0.91	+ 7.1 %	+ 10.5 %	+ 13.7 %	+ 1.8 %
0.92	+ 4.9 %	+ 7.7 %	+ 11.4 %	+ 1.8 %
0.93	+ 1.0 %	+ 4.2 %	+ 6.1 %	+ 6.7 %
0.94	+ 0.3 %	+ 1.3 %	+ 0.8 %	+ 6.0 %
0.95	+ 7.3 %	+ 1.7 %	+ 0.8 %	+ 12.7 %
	$p = 0.8$			
0.90	+ 0.5 %	+ 0.0 %	+ 13.5 %	+ 0.1 %
0.91	+ 7.0 %	+ 10.5 %	+ 13.9 %	+ 0.1 %
0.92	+ 4.5 %	+ 7.6 %	+ 11.7 %	+ 0.0 %
0.93	+ 0.5 %	+ 4.2 %	+ 6.3 %	+ 26.7 %
0.94	**− 0.3 %**	+ 1.0 %	+ 0.1 %	+ 25.3 %
0.95	+ 7.2 %	+ 0.2 %	+ 0.0 %	+ 19.1 %

From the Table 5.8 we can notice than when $p = 0.8$, $\rho = 0.94$ and $\Delta t = 75$ it is a little cheaper to use the opportunistic scheduling policy. This is the only situation where opportunistic policy becomes cheaper. One explanation for this is that the policy-iteration algorithm produces an optimal solution over

a long time horizon and it happens in this case that this optimality is not shown during the 67 maintenance instances. If we extend our time horizon to, for example 200 maintenance instances, the opportunistic policy becomes the more expensive one.

It is also noteworthy that when we looked separately at the simulation rounds with no failures during the maintenance period of 25 years, the optimal policy and the opportunistic policy with $p = 0.8$, $\rho = 0.94$ and $\Delta t = 75$ replace components identically. This means that the average costs of the two policies are then also equal. Thus, it is more difficult to see clear differences between these policies.

In addition, we compare the average number of component failures between the opportunistic policy and the optimal policy-iteration policy. Table 5.9 presents the percentage difference between the average number of wheels failures when compared to the optimal policy, the results in lower right part of the Table 5.7. We notice that when p increases, the average number of wheel failures tend to decrease.

We can conclude that by paying more some improvements in the reliability are possible but it also seems that resources are not used efficiently since in some cases they do not lead to any benefits.

Table 5.9: The percentage change in the number of average wheels failures per thousand kilometers during 5 million kilometers (period of 25 years) when using four different opportunistic policy with different maintenance interval and reliability threshold combinations

reliability threshold ρ	maintenance interval Δt 75	100	125	150
	$p = 0.2$			
0.90	30.0 %	5.8 %	−4.5 %	−2.0 %
0.91	44.3 %	11.7 %	0.7 %	2.5 %
0.92	12.8 %	−2.2 %	−6.1 %	2.8 %
0.93	15.2 %	−19.1 %	−13.0 %	−4.8 %
0.94	18.1 %	−14.3 %	−3.4 %	−6.0 %
0.95	19.0 %	−1.1 %	−1.5 %	−69.6 %
	$p = 0.4$			
0.90	9.3 %	−15.2 %	−4.4 %	−1.7 %
0.91	53.4 %	−6.2 %	2.2 %	2.5 %
0.92	13.3 %	−23.3 %	−5.2 %	1.9 %
0.93	31.2 %	−36.4 %	−12.6 %	−4.2 %
0.94	42.5 %	−40.6 %	−2.7 %	−6.7 %
0.95	11.3 %	−2.3 %	−0.3 %	−69.6 %
	$p = 0.6$			
0.90	−1.2 %	−1.3 %	−54.0 %	−0.7 %
0.91	−23.3 %	−37.8 %	−52.6 %	0.8 %
0.92	−23.8 %	−37.0 %	−55.9 %	1.4 %
0.93	−8.3 %	−45.3 %	−55.6 %	−7.1 %
0.94	2.3 %	−42.9 %	−1.8 %	−8.9 %
0.95	−23.8 %	−2.6 %	0.1 %	−70.5 %
	$p = 0.8$			
0.90	0.1 %	−0.3 %	−56.9 %	0.8 %
0.91	−23.3 %	−41.2 %	−54.5 %	2.8 %
0.92	−25.7 %	−39.8 %	−57.0 %	0.5 %
0.93	−11.8 %	−48.2 %	−58.4 %	−70.2 %
0.94	0.0 %	−45.4 %	0.1 %	−71.0 %
0.95	−28.8 %	−2.4 %	−0.7 %	−70.2 %

Chapter 6

Discussion

This thesis developed an optimization model for maintenance scheduling, investigated the convergence of the policy-iteration algorithm and applied the model to a case example. The calculated cases illustrated the strengths and weaknesses of the model in comparison to models presented in scientific literature.

6.1 Strengths of the model

According to Vu et al. (2014), systems have dynamic contexts which are situations where some new information may be available concerning maintenance opportunities, changes in production planning etc. These are divided into three parts: ones affecting the structure of the system, ones affecting the components of the system and ones affecting the environmental context. The developed model can react to changes in all of the three levels. We can update the directed graph to take into account structural changes. We can change the failure distributions to make changes on the component level. We can adjust the maintenance interval to cope with the environmental changes. This highlights flexibility and adaptability as strengths of the model.

The scheduling policies obtained in different examples highlight the sensitivity of the results to parameter changes. For example in Section 4.2, when the set-up cost was high, the optimal scheduling policy was found fast and it had some intuitive properties. Also in Chapter 5, when a component had failed and it had to be replaced, the policy usually recommended replacing other components as well. Because this model is not based on the fact that we first solve optimal replacement times of individual components and then

group these maintenance operations in a cost-optimal way, we can find clever groupings of how to replace components. This grouping is not forced by using artificial penalty functions like in rolling horizon approaches (e.g. Wildeman et al., 1997).

In addition, simulations show that the model focuses on the whole system rather than individual components. These system-level decisions treat components differently. We notice this for example from the number of component failures during simulations. The policy does not try to keep the average number of failures low overall for all the components simultaneously. Instead the decisions are based on economic and structural dependencies. This differs for example from block replacement policies.

The reliability threshold helps to reduce the number of failures and emphasizes the operation of the system. If we have a reliability threshold of $p = 0.90$ for a single system, we can still achieve a very good total reliability for example in situations, where there are parallel systems and only a certain portion of those need to stay operative. Anyway, the threshold ensures that the reliability of the system stays acceptable in practise. Similar reliability measures are not typically applied in the rolling horizon approaches (e.g. Wildeman et al., 1997).

6.2 Weaknesses of the model

The weaknesses of the model are its restrictions. First, components are only allowed to be replaced, not repaired. In practice there can be failures that do not require replacement but a small fix which results into situation that components do not return to 'good as new' state. Instead the resulting state is something between 'good as new' and broken.

Another restriction is the increasing failure rate of components which is needed to make preventive replacements meaningful. We used Weibull distribution but the data from components might not always support the increasing failure rate. If the failure rate is constant, which is true for the exponential distribution, we cannot fulfil the reliability threshold by preventively replacing components. In this case, one way to make the failure probability of a component smaller is to shorten the maintenance interval.

We are also assuming discrete maintenance instances where, in case of failure, the replacements are postponed to the next maintenance interval. However, in reality, many systems can have failures that need maintenance right after the failure happens. Also the assumption that at most one component can

fail at a time might not hold in practise, but the validity of this assumption was discussed in Section 3.2.

Another weakness is the solution method. The convergence time of the policy-iteration algorithm depends heavily on the size of the state space which restricts the size of the system the model can address. Thus, the combination of the parameters discussed in the Section 4.1 must be chosen wisely and as a result, the model is impractical with very complex and large systems. Thus, our approach has some disadvantages when comparing it with rolling horizon approaches and genetic algorithm which have been used to solve bigger problems, like a system with 16 components (Vu et al., 2014).

6.3 Overcoming the weaknesses

There are ways to overcome the weaknesses. If we for example have enough data from the system and from different maintenance operations we could try to model also other maintenance decisions than just replacements. For example, some decisions do not replace the component into new one but reduce its age by some amount. Then we can model new state transitions. We could also form new intermediate states in addition to the current ones but then again this increases the size of the state space which can cause computational difficulties.

In addition, we could consider using modern technologies to monitor the state of the components in real time. It could give realistic information about the current state of the system and reduce the impact of the chosen probability distributions. In advance, it is not certain that the probability distributions would model the individual components of the system well. But by monitoring the system we can update those.

In case of a sudden failure, which requires immediate maintenance, we could move the maintenance instances. Then there are different ways to adapt the maintenance scheduling. At the failure we could maintain the system like it would have been maintained at the next instance. Then, those components, that are not replaced, have an age $a_i \neq k\Delta t$. We could either round up their age, which forgets some of the components residual life, and continue with the policy or we could expand the state space to take into account these intermediate states and recalculate the policy. Those intermediate states vanish when every component is replaced one time after the sudden failure.

In further work it would be interesting to know if there were clever ways to reduce the size of the state space to enable solving bigger problems. In

Table 4.4 there are portfolios which are never chosen. This means that there are states which the system never reaches because maintenance decisions fix the number of possible age combinations. It would be valuable to know if this is a common phenomenon or if this happens with every system and not just with certain assumptions. The next logical question is, how can we identify situations like this without first running the model with all the possible states and after that looking at the results. At least with penalty functions it is possible to create theorems that limit the number of possible groupings (Wildeman et al., 1997).

6.4 Other remarks

One interesting question is how we identify the maintenance scheduling policy which the developed model produces. Clearly this policy does not focus only on corrective replacements. Preventive replacements must be done to fulfil the reliability threshold. In addition, it seems that opportunities are used often. When something breaks, older operative components are replaced as well, like Table 5.5 presents. Thus, we can identify this policy as a opportunistic maintenance policy with a short-term grouping because at every maintenance instance we are prepared for possible failure of the system.

Optimal maintenance scheduling policy creates also other benefits than saving money. Discrete maintenance instances and only allowing replacements should make planning maintenance much easier because we now when and how we can maintain the system. We can also use the maintenance scheduling as a support tool for decision making. For example, we look at the case example from Chapter 5. If we increased the durability of the wheels we could recalculate the optimal maintenance policy and then see how the average maintenance costs change. Then we can use the cost difference to help decide how much we could be willing to pay for the increased durability.

Chapter 7

Conclusions

This thesis developed a maintenance scheduling model for a multi-component system with economic and structural dependencies. The dependencies were presented with a directed graph. We assumed pre-defined maintenance instances where we can only replace components. The failure times of components were modeled with probability distributions. Every component was assumed critical meaning that the system fails if one of the components fails. The system was modeled as a discrete time Markov decision process where the state of the system depends on the ages of the components and the failure state of the system. We emphasized the reliability of the system by setting a reliability threshold which the system had to fulfil at all times. The reliability threshold, structural dependencies and the need to replace failed components defined feasible maintenance action portfolios, the set of components that are replaced.

The key question for solving the maintenance scheduling model was which feasible maintenance action portfolio we should choose when the current state of the system is known. Different options were evaluated based on how well they minimize long term average maintenance costs under a reliability constraint. We also included the possibility to discount future cash flows in the model. In the end, it was possible to solve the optimal maintenance scheduling policy using policy-iteration algorithm. The algorithm was implemented with MATLAB R2019b.

Next, the performance of policy-iteration algorithm was examined in practise. We solved maintenance scheduling models of different sizes and noticed that solution times depend heavily on the size of the state space. We also found out that there are limits to how big problems can be solved using regular

hardware. Also other model parameters, like the set-up cost, can have an impact on the computation time of the policy-iteration algorithm.

The model was applied to a case example with four components. The failure times of components followed Weibull distributions with increased failure rate. We used different reliability thresholds and maintenance intervals to define 24 different maintenance scheduling policies. The efficiency of different policies was compared with simulations. It seemed that maintenance usually becomes cheaper if it is possible to shorten the maintenance interval or lower the reliability threshold. When these policies were compared with simple and heuristic opportunistic policies we noticed that our approach creates cost savings or is equally good regardless of the chosen maintenance interval or reliability threshold. In contrast, these opportunistic policies tend to decrease the number of failures on average because they are more prone to maintain extra components simultaneously with mandatory replacements.

The developed model worked well. It gave cost-optimal maintenance scheduling policies without compromising the reliability of the system. It could solve different kinds of scheduling problems in reasonable time when the parameters affecting the size of the state space were chosen wisely. It is noteworthy that the model seems to work better with systems that have more restrictions because these tend to decrease the size of the state space.

Policy-iteration was a suitable solution method for the model. Previous studies have approached the maintenance scheduling of a multi-component system with different methods like rolling horizon. These usually include just economic dependencies while taking poorly reliability measures into account. Now, for the first time to the authors knowledge, we are including economic dependencies, structural dependencies (Geng et al., 2015) and reliability threshold (Nguyen et al., 2015) into a model which is solved analytically. Indeed, new approaches that are suitable in this field help develop maintenance scheduling further.

The next step would be to expand the model by relaxing its assumptions as discussed in Section 6.3. Including other maintenance operations than just replacements and allowing replacements between maintenance intervals could bring the model closer to satisfying the needs of a real-life operations. We could also test the sensitivity of the model with new types of parameter changes, like looking at the structural dependencies more closely.

In further studies it would also be interesting to find ways to reduce the size of the state space and to make solving bigger models possible. We learned than after knowing the failure distributions of the components we can change

the size of the state space by choosing maintenance interval or reliability threshold accordingly.

If reducing the size of the state space seems difficult, we could look for other ways to speed up the calculations. For example, in applying value-iteration algorithm we do not need to solve a system of linear equations so one iteration of value-iteration is faster than one iteration of policy-iteration. However, value-iteration algorithm typically needs more iterations to converge. One approach would be to solve the system of equations approximately by executing a limited number of value iterations. This method is a modified policy-iteration algorithm (Bertsekas and Tsitsiklis, 1996).

References

H. Ab-Samat and S. Kamaruddin. Opportunistic maintenance (om) as a new advancement in maintenance approaches: A review. *Journal of Quality in Maintenance Engineering*, 20(2):98–121, 2014.

R. Bellman, R. Bellman, and R. Corporation. *Dynamic Programming*. Rand Corporation research study. Princeton University Press, 1957. URL `https://books.google.fi/books?id=rZW4ugAACAAJ`.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-dynamic programming*, volume 5. Athena Scientific Belmont, MA, 1996.

M. Bevilacqua and M. Braglia. The analytic hierarchy process applied to maintenance strategy selection. *Reliability Engineering & System Safety*, 70(1):71–83, 2000.

K. Bouvard, S. Artus, C. Bérenguer, and V. Cocquempot. Condition-based dynamic maintenance operations planning & grouping. application to commercial heavy vehicles. *Reliability Engineering & System Safety*, 96(6): 601–610, 2011.

B. Castanier, A. Grall, and C. Bérenguer. A condition-based maintenance policy with non-periodic inspections for a two-unit series system. *Reliability Engineering & System Safety*, 87(1):109–120, 2005.

N. Chalabi, M. Dahane, B. Beldjilali, and A. Neki. Optimisation of preventive maintenance grouping strategy for multi-component series systems: Particle swarm based approach. *Computers & Industrial Engineering*, 102: 440–451, 2016.

G. Chan and S. Asgarpoor. Optimum maintenance policy with markov processes. *Electric Power Systems Research*, 76(6-7):452–456, 2006.

D. I. Cho and M. Parlar. A survey of maintenance models for multi-unit systems. *European Journal of Operational Research*, 51(1):1–23, 1991.

C. D. Dao and M. J. Zuo. Selective maintenance of multi-state systems with structural dependence. *Reliability Engineering & System Safety*, 159: 184–195, 2017.

R. Dekker, R. E. Wildeman, and R. Van Egmond. Joint replacement in an operational planning phase. *European Journal of Operational Research*, 91 (1):74–88, 1996.

P. Do, A. Voisin, E. Levrat, and B. Iung. A proactive condition-based maintenance strategy with both perfect and imperfect maintenance actions. *Reliability Engineering & System Safety*, 133:22–32, 2015a.

P. Do, H. C. Vu, A. Barros, and C. Bérenguer. Maintenance grouping for multi-component systems with availability constraints and limited maintenance teams. *Reliability Engineering & System Safety*, 142:56–67, 2015b.

P. Do Van, A. Barros, C. Bérenguer, K. Bouvard, and F. Brissaud. Dynamic grouping maintenance with time limited opportunities. *Reliability Engineering & System Safety*, 120:51–59, 2013.

J. Geng, M. Azarian, and M. Pecht. Opportunistic maintenance for multi-component systems considering structural dependence and economic dependence. *Journal of Systems Engineering and Electronics*, 26(3):493–501, 2015.

R. A. Howard. *Dynamic programming and markov processes.* John Wiley, 1960.

R. Jiang and P. Ji. Age replacement policy: a multi-attribute value model. *Reliability Engineering & System Safety*, 76(3):311–318, 2002.

M. C. O. Keizer, S. D. P. Flapper, and R. H. Teunter. Condition-based maintenance policies for systems with multiple dependent components: A review. *European Journal of Operational Research*, 261(2):405–420, 2017.

J. Kleinberg and E. Tardos. *Algorithm design*. Pearson Education India, 2006.

E. Kyriakidis and T. D. Dimitrakos. Optimal preventive maintenance of a production system with an intermediate buffer. *European Journal of Operational Research*, 168(1):86–99, 2006.

C. Love, A. Rodger, and G. Blazenko. Repair limit policies for vehicle replacement. *INFOR: Information Systems and Operational Research*, 20 (3):226–236, 1982.

D. G. Luenberger. *Investment Science*. Oxford University Press, 1998.

K. S. Moghaddam and J. S. Usher. A new multi-objective optimization model for preventive maintenance and replacement scheduling of multi-component systems. *Engineering Optimization*, 43(7):701–719, 2011.

K.-A. Nguyen, P. Do, and A. Grall. Multi-level predictive maintenance for multi-component systems. *Reliability Engineering & System Safety*, 144: 83–94, 2015.

R. P. Nicolai and R. Dekker. Optimal maintenance of multi-component systems: a review. *Complex system maintenance handbook*, pages 263–286. Springer, 2008.

F. Pargar, O. Kauppila, and J. Kujala. Integrated scheduling of preventive maintenance and renewal projects for multi-unit systems with grouping and balancing. *Computers & Industrial Engineering*, 110:43–58, 2017.

W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

M. Rausand and A. Høyland. *System reliability theory: models, statistical methods, and applications*. John Wiley & Sons, 1994.

L. Thomas. A survey of maintenance and replacement models for maintainability and reliability of multi-item systems. *Reliability Engineering*, 16 (4):297–309, 1986.

H. C. Tijms. *Stochastic modelling and analysis: a computational approach*. John Wiley & Sons, Inc., 1986.

F. A. van der Duyn Schouten and S. G. Vanneste. Analysis and computation of (n, n)-strategies for maintenance of a two-component system. *European Journal of Operational Research*, 48(2):260–274, 1990.

A. Van Horenbeek and L. Pintelon. A dynamic predictive maintenance policy for complex multi-component systems. *Reliability Engineering & System Safety*, 120:39–50, 2013.

H. C. Vu, P. Do, A. Barros, and C. Bérenguer. Maintenance grouping strategy for multi-component systems with dynamic contexts. *Reliability Engineering & System Safety*, 132:233–249, 2014.

H. Wang. A survey of maintenance policies of deteriorating systems. *European Journal of Operational Research*, 139(3):469–489, 2002.

R. E. Wildeman, R. Dekker, and A. Smit. A dynamic policy for grouping maintenance activities. *European Journal of Operational Research*, 99(3): 530–551, 1997.