Optimizing Long-Term Planning of Railway Maintenance

Maarten Giltaij

Transport, Infrastructure and Logistics

MSc Thesis

TU Delft

ARCADIS

Optimizing Long-Term Planning of Railway Maintenance

by

Maarten Giltaij

in partial fulfilment of the requirements for the degree of

Master of Science in Transport, Infrastructure and Logistics

at the

Delft University of Technology

to be defended publicly on Wednesday October 24th, 2018 at 15:00.

Thesis Committee:

Prof. Dr. Rob Goverde, Professor of Railway Traffic Operations & Management, Faculty of Civil Engineering and Geosciences, TU Delft (Chair)

Dr. Nikola Bešinović, Researcher, Faculty of Civil Engineering and Geosciences, TU Delft (Daily Supervisor TU Delft)

Ing. Mart Folkerts, Advisor, Arcadis Netherlands (Daily Supervisor Arcadis)

Dr. Valeri Markine, Specialist in Railway Engineering and Structural Optimization, Faculty of Civil Engineering and Geosciences, TU Delft

Dr. Jan Anne Annema, Lecturer, Faculty of Technology, Policy and Management, TU Delft

Preface

Dear reader,

A more fitting venue for writing this preface than the train home from Amersfoort could not be imagined. Yet here I sit, almost finished with what is undoubtedly the most daunting assignment of my years in education, which go back some twenty years already.

Many a hardworking hour has been poured into the creation of the thesis before you. Some lovingly, others hated with a passion. Many coding away with furor, a few breaking personal procrastination records. But mostly it was a matter of chugging along steadily, celebrating small success and stepping over cobblestones along the way.

At the beginning of this project, almost a year ago today, I could not have imagined the result that is here today. I have heard from many a graduate before me that you don't really know what you're doing and where you're going until you are at least half way there. They couldn't have been more right.

First off, I would like to thank Nikola. He has always been at the ready to help out and bring structure to my progress. Although we didn't always see eye to eye, I couldn't have wished for a more helpful daily supervisor. Mart, thank you for keeping my feet on the ground while my mind was full of scientific abstractness, and of course for introducing me to a world of FOTs, BDSen, KrOLs, TAOs and TVTAs. Rob, Valeri and Jan Anne all provided invaluable contributions with their wealth of expertise and experience in their field of work, as well as their experience in working on graduation projects.

Keeping my spirits up when working, or rather, while taking breaks, was a task entrusted to my colleagues from Asset Management and from Data Analytics in Amersfoort, and to those with whom I shared the graduation room when I was in Delft. Merely the foosball games and random banter made the daily commute a worthwhile ordeal. Finally, I cannot leave my friends at the graduation room at the faculty unmentioned. Seven people on a mission, a room that is on the small side for such company and a free coffee pass is all that is needed to forge companionship that I am sure is to stand the test of time and diverging careers.

As for my work over the past months, I truly hope it will result in more than me graduating. Hopefully, it will provide a useful stepping stone the operations research community, and a tool by which train travel will become just a little bit nicer. I'm hopeful someone out there will take a little bit of inspiration from this work to make theirs a little better, just as I have from many who have worked in this field before me.

My future as a proper, money-making, responsibility-taking, early-waking adult will start on November 1st, where I will see my colleagues at Arcadis again, just no longer as the intern. My starting position will be as a trainee in team Asset Management, of course hoping to develop myself further.

As the world of transportation engineering and management is fairly small, I'm sure my paths will cross with many of you who work there or will do so. I wish you good luck on all your future endeavors, and be sure to let me know if ever you have important railway business to discuss, or simply want to drink a beer.

Cheers,

Maarten Giltaij
Delft, October 2018

Executive Summary

Railways are important to modern society. It is a system that occasionally requires maintenance, which involves cost and service suspensions. However, maintenance is important to reduce failures. A trade-off is necessary. In the Netherlands, railway maintenance is performed by independent contractors. These parties are increasingly made responsible for the track they are maintaining, and also get increasing possibilities to improve their working methods. This includes the ability to innovate in scheduling maintenance.

Currently, determining when and how often to maintain individual railway components is done manually, and only for the next year. What's more, maintenance is performed within fixed bounds. There is no tradeoff between cost, reliability and availability. This is not optimal. A better solution may exist than is found by manually creating a solution. An improvement could reduce the number of track failures, the cost of doing maintenance and the number of times the track needs to be closed for performing maintenance. What is desired is a problem formulation that captures all these goals. The main research question therefore states:

"What is the optimal way of planning railway maintenance over the long term?"

Railways are made up of many components, each of which has to function in order to host the train service. The preferred approach when maintaining these components is to bundle the maintenance operations that affect a service, that is, to reduce the required possessions.

For a quantitative approach, a way to model degradation of the component is required. Failure of components is a stochastic process. Failure rate modelling can be used to estimate the probability of failure over time. Two statistical models, the additive Weibull distribution and the additive Gompertz-Makeham distribution, are presented. The parameters for these models can be determined with data on the components.

Linear programming is a widely used tool for solving numerical optimization problems. The underlying assumption is that the relation between each parameter that can be changed and the overall desirability is linear. Even though the problem is not linear in itself, an unconventional transformation of the decision variables enables successful problem formulation. In this way, the costs of maintenance, possession and failure can be included in the objective function.

In this thesis, a way to analyze the failure characteristics over time and to determine a maintenance schedule that is optimized for the cost of failure, the cost of maintenance and the cost of possessions is introduced. This is more than what has been done in other literature. Ultimately, it could serve to improve the maintenance planning of railways.

Two models are presented that achieve the set goals. In both, the objective function is an expression of the total expected cost as a function of the chosen maintenance strategy. The models are able to solve problems that cannot be solved using models in existing literature.

The formulation is very general, and could possibly be applied in many fields other than railways. They have the potential to improve the way maintenance to all kinds of degrading systems is planned.

The first model optimizes a situation in which each component in the system has its own, repeating maintenance interval. The maintenance interval differs between components, but stays constant over time. The costs of maintenance, failure and possessions are taken together and minimized.

The second model is an expansion of the first model in which the intervals are no longer constant over time, but are modelled individually. This is done by adding decision variables to the model for every single interval. The result is that the optimization produces a more efficient schedule. It does increase the required runtime substantially compared to the first model.

A fictitious case was formulated in order to test model performance. These tests relate to numerical settings and to problem characteristics. For problems with three component types, performance is acceptable over a wide range of settings. Various problems of up to several years, assuming a time step of one week, can be solved within an hour.

A test case based on real maintenance data was devised. Model input was created by estimating the failure rates of components in the data sets. Compared to the estimated performance of a conventional schedule, the optimized schedule reduced costs by 2.3%.

The long-term planning and cost estimation of maintenance can be improved by using the model presented in this thesis. As opposed to current methods, a quantification of the expected cost of failure depending on the maintenance strategy is defined.

Further development that builds on the model as presented could address certain uncertainties that are still present. The main uncertainty is, and probably remains, the predictive quality of the available data. Before major decisions can be based on the model, a more elaborate trial would be advisable.

Content

1 Introduction

1.1 Background

Railways provide affordable and sustainable transportation to many passengers and shippers. However, railways need to be maintained, and this requires train services to be reduced or suspended temporarily. If a reduction of these timetable adjustments can be achieved, rail users profit. When scheduling maintenance, a trade-off has to be made between direct cost, reliability, user preferences, safety and availability. There is a lack of knowledge, coordination and incentive in the parties involved to improve the working methods. Improving and optimizing the way scheduled maintenance is planned could potentially yield benefits to customers and maintenance contractors.

In the Netherlands, the railway network is publicly owned, but maintained by private contractors. Daily maintenance is contracted out on a regional basis. Figure 1 gives an overview of these regions. The maintenance contractors get assigned a certain amount of time to perform maintenance by the infrastructure manager (IM, ProRail in the Netherlands), which is defined in the contract between the two. This contract is based on the tender that is put out by the track manager. Included in this contract is a minimum availability of the infrastructure, expressed in a percentage of weighed time. Weights are allocated to track sections based on importance.

The nature of contracts between the IM and the contractor is shifting from direct specification of work and hourly compensation, to performance targets and fixed sum payment with bonusses and reductions for performance. The task of planning maintenance is also transferred to the contractor, which explains why there is increased interest in efficient maintenance planning.

The offer made by the contractor is a rough estimate of the time needed, and there is a strong incentive to offer a high track availability, as this increases the likelihood of being assigned the contract. The available amount of time is therefore not ideally matched to the maintenance requirement. If contractors have a better indication of the amount of resources that are needed to fulfill the requirement that is proposed in the tender prior to submitting that tender, risk is reduced for all parties involved. One way of doing this is by modelling the maintenance processes, for example in GIS or a mathematical model. The downside of this is that contractors need to invest more in a tender they are unsure to win.

Furthermore, there is a preference by passengers for large maintenance projects, which cannot be performed within a weekend, to be performed during school holidays, when travel demand by commuters is lower. This does mean that maintenance may be performed at a moment in time that is not ideal from an engineering perspective. In addition, because this increases peak demand for maintenance resources, their utilization rates are worse and capital cost higher.

Figure 1, contracting areas, adapted from http://www.spoordata.nl/sites/default/files/2017-12-11%20Contractgebiedenkaart.pdf

As can be derived from the literature review in chapter 2, much research has been performed in the fields of railway maintenance, of failure modelling, and of optimization. However, an integrated approach that includes all these elements is not yet common. The potential for such an approach is that the planning of maintenance can be improved and possibly optimized.

Using the latest advancements in data collection, the failure behavior of individual components can be modeled more accurately. When the failure probabilities and costs of components are known, the performance of the system as whole can also be evaluated.

1.2 Research Objective and Questions

The practical purpose of the research is to improve the ability of infrastructure managers and contractors to plan maintenance depending on the requirements of the infrastructure components. This can be captured in the main research question:

- What is the optimal way of planning railway maintenance over the long term?

To be able to answer this very general question, it is broken down into four topics, which contain several sub questions each.

1.2.1 Characterization

To be able to do any kind of quantitative analysis on failure processes, one must first analyze and characterize the components under consideration. This is done through answering the following sub questions:

- What are the challenges in the current way of determining a maintenance strategy for railway assets?
- How can the failure probabilities of railway components over time be estimated?

1.2.2 Optimization

Once the quantitative properties of the effects of maintenance are known, this information can be used to determine the expected performance of a certain maintenance strategy. If the maintenance strategy is found that demonstrably has the best expected performance, the maintenance is optimized.

- How can the costs resulting from failure be estimated in case the maintenance schedule is known?
- How can the optimal schedule for performing maintenance to a certain track section and its associated cost be determined?

1.2.3 Evaluation

Before the model can be put into practical use, it is necessary to demonstrate its performance compared to the current methods that are used to plan maintenance.

- How does an optimized maintenance schedule compare to a schedule that is up to the current state of the art?
- How accurately and how far into the future can the cost of performing track maintenance be predicted?

1.2.4 Valorization

Once an optimization model for the long-term maintenance strategy is available, it is important to know in which way the model is useful to maintenance planners in the railway industry, and, on a larger scale, how optimization of railway maintenance is beneficial to society.

- How can the model be used by railway maintenance engineers?
- Will better optimization of maintenance contribute to reduced unplanned maintenance?

- How can the contract between the contractor and the infrastructure manager be specified, such that all the information necessary for optimizing maintenance is exchanged?

By addressing each of these individual research questions, a well-supported answer to the main question can be given.

1.3 Methodology

The initial research, which served to find the knowledge gaps and clearly define the problem, was mainly conducted through doing interviews with engineers at Arcadis and a literature review.

As the problem is an optimization problem, a kind of programming method is needed to solve it. Linear programming is a widely used tool for optimization problems. It allows for solving a large variety of optimization problems. There are several advantages and disadvantages to using linear programming. The main advantage is the availability of specialized solvers. If the problem is well defined, the exact solution algorithm will be determined by the solver. A disadvantage is that not all problems are linear. Linear programming uses decision variables that represent the decisions that are made in operating a system, an objective function that is used to determine the cost or benefit of a certain solution, and constraints that limit the decision variables to practical values. Cases where (some of) the decision variables are integers are called mixed integer linear programming (MILP). A specific form of mixed integer linear programming is binary programming. In such problems the decision variables are restricted to two values. If non-linear functions are encountered that cannot be reformulated such that they are linear, unary encoding can be used. This is a technique to reformulate the program such that every level of the integer decision variable is represented by a single binary variable.

1.4 Thesis Outline

Before the research questions are answered in depth, initial research is conducted to provide an overview of knowledge that is already available on the topic. It can be found in chapter 2. Chapter 3 starts with the definitions and assumptions that underpin the models, after which the models are presented. The second model is an extension of the first. In chapter 4, the second model is evaluated. The performance is tested on a synthetic parameter set (the input of the model), and a test case, in which the input parameters are derived from real data, is tried. In chapter 5, conclusions and recommendations are given. A reflection on the work can be found in chapter 6. Information and documents that support the main text, but provide information that is too detailed too include in the main text, can be found in the appendices following the reflection.

2 Existing Knowledge and Practices

2.1 Current Industry Practice

The planning of maintenance is evolving under the changing priorities of both infrastructure managers and contractors. As contractors are no longer rewarded for executing maintenance, but for achieving and maintaining performance levels, the incentive for them has shifted from performing as much maintenance as billable to as little maintenance as needed.

When applying for a tender, the contractor has to submit a price for which he will ensure the promised level of performance. The lowest bid wins, but if the actual cost of performing the maintenance exceeds the bid, the contractor loses money on the contract. Thus, a good estimation of what the cost of maintenance to the contractor will be helps the contractor win the contract while still making a profit.

The planning of track decommissioning in order to perform planned track maintenance (track possession) is currently suboptimal. The industry relies mostly on the knowledge and experience of the employees. Most information is taken from previous projects. The main reasons that it is done in this way are that every contract area is different, that the cost of maintaining relies on a lot of factors as the maintenance operations are diverse and large in number, and that these maintenance operations should be coordinated with each other. This is why it is not ideal to simply plan maintenance for the individual components and combine the results into a schedule.

Ever since railways were first constructed, experience has been gained on maintaining railways. Employees that have worked in the business are a valuable resource to railway maintenance contractors, as they are able to give useful indications of what is required to perform maintenance on a railway line. However, the informal nature of this knowledge usage prevents a reliable estimation of requirements that is replicable and consistent.

A current trend in research on optimization and planning maintenance, both in the railway industry and other industries, is to use big data and predictive maintenance to improve the planning of maintenance (Núñez et al, 2014). Instead of doing maintenance according to a fixed schedule, maintenance is adjusted to the actual requirements of the elements in the system. However, in practice the planning methods used by contractors have not yet evolved along with these trends.

Currently, planners have no way to use all of the information that is available effectively. The reliance on human knowledge is a limiting factor, because the human brain is far worse at evaluating large amounts of data than a computer. A formalized, integrated method for estimation of maintenance would solve these problems.

Other railway sectors and different industries, for example train timetabling and finance, have used advanced optimization to increase performance and reduce uncertainty and unexpected results. Research into applying these techniques to railway maintenance planning may yield similar improvements. Ultimately, a better insight into the maintenance requirement over a longer time horizon will enable both contractors and IMs to better plan

maintenance and estimate the cost involved. It can contribute to a more affordable and more predictable railway system.

2.2 Literature Review

A fair amount of literature is available on optimizing maintenance scheduling in well-defined cases in other industries. Some inroads have been made into incorporating stochastic elements into maintenance models. Existing stochastic models have focused on specific types of maintenance process, e.g. ballast level degradation (Quiroga & Schnieder, 2010) and (Quiroga & Schnieder, 2012). However, a generalized approach for modeling failure and degradation on the strategical and tactical level as a function of failure processes is not available.

2.2.1 Process Improvement

Nyström (2008) did research into delay attribution by railway staff. He emphasizes the need for consistent measures of punctuality performance and requirements. Improved delay attribution can help in identifying and solving problems more quickly.

Vet (2014) has investigated the organizational problems behind predictive track geometry maintenance. He provides suggestions for reorganization that should lead to improved maintenance procedures.

2.2.2 Deterministic Models for Maintenance Planning

Budai et al. (2006) present the Preventive Maintenance Scheduling Problem (PMSP). A solution and a heuristic are provided and compared. The objective function consists of a generalized possession cost and the maintenance cost.

Budai et al. (2009) did further research on this problem, and provided a solution using a genetic algorithm, a memetic algorithm and a two-phase heuristic. These solutions are less conventional than most, and may provide building blocks for further research.

Forsgren et al. (2013) have developed a model that reschedules trains to fit within windows of track possessions. However, scheduling maintenance itself is not considered.

Higgins (1998) has written one of the earlier papers on optimizing maintenance schedules. His objectives were to minimize the finishing time, and to minimize interference with the regular timetable.

Higgins et al. (1999) build on this research by introducing minor refinements and applying it to a different case.

Huisman et al. (2005) have given a broad overview of the operations research challenges experienced by a train operating company, in this case NS (Nederlandse Spoorwegen, Dutch Railways). This does not include infrastructure maintenance planning, however maintenance availability is of importance to the train operating company and its customers. Train timetables have to be adapted to possessions for maintenance.

Li & Roberti (2017) have focused their research on large project maintenance, where track is closed for a longer period of time. They call this the Railway Track Possession Scheduling Problem (RTPSP). A mixed-integer linear program (milp) is used to find a solution. The objective is to minimize construction costs while satisfying operational constraints.

Lidén & Joborn (2016a & 2016b) have written two papers on optimizing maintenance scheduling. An approach is used that incorporates traffic planning into maintenance planning. This approach is applied to a single-track railway in northern Sweden.

Luan et al. (2017) developed a method for planning Preventive Maintenance Time Slots (PMTSs), whereby the timetable and the maintenance schedule are optimized simultaneously. The objective for the planning is to minimize deviation from the original timetable. Maintenance is planned as a virtual train. This approach simplifies the integration of maintenance into timetabling.

Oh et al. (2006) created a solution for a Track Tamping Scheduling Problem (TTSP) in the context of the Korean high-speed railway system.

Oyama and Miwa (2006) have made a model that optimizes scheduling of a multi tie tamper, based on a prediction of track condition, under a constraint of equipment availability. The objective function is therefore not related to the timetable, but rather to the physical state of the track. Optimizing a combination of track condition with for example cost and disruption would be an improvement to their model.

Peng et al. (2011) have made a model for a Track Maintenance Scheduling Problem (TMSP), which minimizes an objective function consisting of maintenance crew travel time and impact on operation. To this end, an iterative heuristic model is used. What makes this research interesting is the incorporation of track availability into the model.

Ultrasonic Inspection Vehicles gather large amounts of useful data on the condition of the rail. Ideally, the vehicle detects each crack before the rail head fails. However, their availability is limited. Podofillini et al. (2006) propose a genetic algorithm that optimizes the use. Parallels may be found with other maintenance tasks to which the modelling structure may also be applied.

Van Aken at al. (2017a, 2017b) formulated the Train Timetable Adjustment Problem (TTAP). The goal is to optimally reschedule trains to allow for maintenance works. Track possession itself is not incorporated into the model but taken as a given input.

Vansteenwegen et al. (2015) have made an algorithm (the maintenance conflict avoidance algorithm) that adjusts the timetable to maintenance works. Allowing the algorithm to make small modifications to the routing and the timetable improves robustness.

2.2.3 Stochastic Models of Railway Maintenance Requirements

Consilvio et al. (2016) proposed a more general model, which includes stochastic elements into scheduling problems. This is very relevant to maintenance operations because the

amount of preventive maintenance and the probability of corrective maintenance are dependent on failures, which are stochastic in nature.

Quiroga and Schnieder (2010) provide a heuristic to optimize the tamping schedule of high-speed lines. The underlying model assumes the used quantification of track condition is a stochastic variable that is normally distributed. The maximum amount of track possession is taken as a boundary condition. The heuristic is very fast, which makes it ideal for comparing different scenarios.

In 2011, the same authors published a paper on making a Monte Carlo model on longitudinal levelling of the ballast bed. Historical data is used to estimate distributions and the accompanying parameters. These are then used to optimize the tamping schedule.

Zhang et al. (2012) have developed a maintenance optimization algorithm in the context of condition-based maintenance. The deterioration processes are considered, as are safety, maintenance cost and travel cost of the maintenance crews.

A model for optimizing the number of inspections versus the number of failures of a single component subject to a non-homogenous Poisson process is formulated by Zhao et al. (2007).

2.2.4 Stochastic Models of Processes in Other Applications

Reliability engineering is applied in many industries, and several handbooks are available, e.g. Birolini (2007). It details many aspects of modeling and improving reliability of physical systems.

Goei and Meisel (2013) have published on a scheduling problem for electricity network maintenance. Many parallels can be drawn between electricity network maintenance and railway maintenance. For example, edges in the network (power lines or track sections) have to be taken out of service before maintenance can be performed. Hence, techniques developed in this field of operations research can be applied to railway maintenance planning.

Vromans (2005) has made an overview of the state-of-the-art of railway timetabling, as well as a model that incorporates stochastic elements. It can be used for creating and evaluating resilient timetables. According to the author, a unique feature is that it takes delay propagation into account. Without naming it as such, it is a form of Monte Carlo simulation.

2.2.5 Reliability and Survival Models

There has been a lot of research into modelling failure and reliability.

An overview of techniques concerning the modeling of failure rates is found in Finkelstein (2008). Of particular interest is the description of different classes of lifetime distributions. The most common is the exponential distribution. However, such models assume that failure rates are constant over time and therefore not suitable for optimizing maintenance. A more useful, but still common and simple model is the Weibull distribution. It allows for a change in failure rate over time. A somewhat more advanced model is the Gompertz-

Makeham distribution. A characteristic and possibly desirable property of this distribution is that the failure rate increases exponentially over time. In practice, the reliability of parts can drop off quickly after a threshold has been exceeded.

Xie and Lai (1996) have used the Weibull distribution to propose the additive Weibull distribution. One Weibull distribution can either model the wear in (phase in which failure is mainly due to manufacturing, handling or installation errors) or wear out (phase in which failure is mainly due to aging, use and maintenance errors) of a part. A failure rate for a case where both are present is possible by simply adding up the two. The model has four parameters that enable fitting to data. This technique could possibly be applied to other distributions as well.

2.2.6 Railway Infrastructure Maintenance

Maintaining railway infrastructure can be decomposed into several different tasks. Esveld (2001) gives an overview of usual maintenance operations, a quick overview of key components is found in Amtrak (n.d.). The most important objects that need to be maintained and their respective maintenance operations are:

- The catenary system (only on electrified lines)
    - Replacing the contact wire. As the carbon strips of trains pass along the contact wire, the wire, which is either made out of pure copper or a copper alloy. Specialized trucks with working platforms on the roof are used to provide access for the maintenance crew. New contact wire arrives on a coil and is installed.
- The signaling and train protection system
    - Very strict safety measures are in place regarding the signaling and train protection system.
- The rail
    - Rail grinding restores the original profile of the rail head. This is important, because the wheel-rail interface can deteriorate substantially if the contact patch deforms due to rail wear. This can impact vehicle running characteristics and the rate of wear of the wheels.
    - As rail grinding removes away a layer of the rail head, the rail head shrinks. After some time, grinding is no longer possible, and the rail needs to be replaced completely. Another reason for rail replacement can be the formation of cracks in the rail.
- The sleepers
    - Occasionally sleepers (or ties in American English) have to be replaced because of deterioration. Sleepers can be made out of either wood or concrete, with the latter material being the most common one in current installations. It requires less maintenance.
- The ballast bed
    - The ballast may become less supportive of the track. Tamping is supposed to solidify the ballast bed again. If there is not enough ballast available in the bed. After tamping, reprofiling the ballast is necessary. Advanced tamping machines can do all these actions in one run. Therefore, it can be regarded as one maintenance operation.

    - Another measure that can be taken in addition to tamping operations is stone blowing. It fills the spaces that emerge when ballast assumes the shape of the sleepers.
    - The ballast will over time get contaminated with dirt. This reduces the stability. A solution to this is ballast cleaning.
    - As with rail grinding, ballast maintenance can only be done a limited number of times. At a certain point, the ballast bed will have to be replaced completely.
- The switches
    - The moving parts of the switch need to be lubricated occasionally.

What is lacking in the current scientific literature is an approach that builds on the knowledge on the physical characteristics of railway systems by integrating it into a quantitative optimization. Such an approach would fill a void that exists between the fields of optimization and railway engineering.

3 Definitions, Assumptions and Model

3.1 Introduction

Several steps are required to get to a well-supported optimization model. First, some definitions and assumptions need to be laid out. This is the foundation for all further modelling. Next, two statistical models that describe the failure characteristics of railway components are presented. These are then incorporated into a linear optimization model which is able to produce an optimal maintenance schedule.

3.2 Definitions and Assumptions

Railway maintenance can be characterized as either preventive maintenance or corrective maintenance. The two are interrelated, because effective preventive maintenance reduces the need for corrective maintenance. Both maintenance types will be included in this research, although corrective maintenance is not modelled.

Preventive maintenance is intended to keep the asset operational. It is planned in advance, and ideally in such a way that normal operations are not affected. The degree to which it can be planned in advance varies. Preferably, it is performed at fixed intervals, but if deterioration of the component is faster than expected, it may have to planned on short notice.

The state of the component is only dependent on the time since last maintenance. Other factors that may influence the failure probability are not considered. After each maintenance operation, the component functions as if it were new.

A type of maintenance is only performed to an element once within one time period. In practice, there are no components that require multiple maintenance operations per week. Therefore, the time step is a week. This means that specific operational details, such as resource assignment and order of maintenance operations, are left out. The model does not have the accuracy needed to incorporate it. As such, the consequences of possessions with individual train services are not relevant.

Corrective maintenance is either a repair of a part that is already broken, or maintenance that has to be performed urgently because the probability of breaking before the next scheduled repair is too high. If a component is broken or exceeds a safety limit that prohibits operations, the asset cannot be used until corrective maintenance is performed, and it is considered to have failed. Even though technically it may still function, the asset is no longer available for use and the cost of repair is incurred. Ideally, corrective maintenance is not needed, but in practice it is not economically efficient to attempt to prevent all failures.

Repairs are considered to be minimal, as defined in Finkelstein (2008). The component that undergoes repair is in the exact state it was in before the failure occurred. This means the failure behavior does not change as a consequence of repairs, and that maintenance schedule does not have to be adapted. In fact, modelling of specific failure occurrences is not needed at all.

An element either functions or fails. There is no state of limited functionality. One instance where limited functionality would be possible in practice is a switch that is locked in a certain position. Trains can still run over the switch, but in one direction only. However, repair is still necessary.

The costs for a possession are dependent on the moment of possession. This is done through using different costs for possessions depending on the week. Furthermore, possessions may have different sets of components and time lengths. This may be introduced by including multiple possessions within one week.

Different individual components that have the same set of characteristics are aggregated into one category. It is not useful to model these separately, as the resulting schedule applicable to the components should be the same.

A maintenance interval is the time between two maintenance operations. The expected number of failures in an interval is a function of only the length of the interval.

3.3 Modeling Failure Distributions and Rates

3.3.1 The Additive Weibull Distribution

Failures occur randomly over time. When modeling the maintenance requirements associated with a single part of the railway system, one needs to know what the failure probabilities of this part are. The failure rate is a metric that is often used in reliability engineering. A definition can be found in (Verma et al., 2010, pp. 31-34). It represents the number of failures that can be expected within one unit of time if the failure rate is constant; a failure rate of three per year implies that on average three failures occur each year. Even if the failure rate is not constant, the expected number of failures over time can be calculated. Using this variable has one very convenient property: its integral is the cumulative hazard function (Rodriguez, n.d.). The cumulative hazard curve can be interpreted as the expected number of failures between 0 and t. If the cost of an occurrence of failure and the probability of failure are both known, the expected cost of failure can be calculated by multiplying the two:

$$CoF \times \Lambda(t) \tag{3.1}$$

with:
$CoF: Cost\ of\ Failure$
$t: time$
$\Lambda(t): cumulative\ hazard\ at\ t$

This means that, although the process of failure is stochastic by nature, it is not necessary to apply stochastic programming to obtain the cost of failure as a function of time.

Failure in railway components, or indeed many physical structures, is driven by two factors: Break in (also called infant mortality) and wear out (Xie and Lai, 1995). These occur in consecutive phases, usually with a stable phase in between. Therefore, the failure

probability over time exhibits a certain behavior, that can be described as following a bathtub curve. Research into this (very specific) field has also been done by Chen et al. (2011) and Wang et al. (2002).

For the purposes of optimization, a hazard function that approximates a bathtub curve, that is at the same time easily differentiable, is desired. Based on Xie and Lai, the additive Weibull model is used:

$$\lambda(t) = ab(at)^{b-1} + cd(ct)^{d-1} \tag{3.2}$$

with:
$\lambda(t)$: $hazard\ rate\ at\ t$
a: $scale\ parameter\ for\ break\ in$
b: $shape\ parameter\ for\ break\ in$
c: $scale\ parameter\ for\ wear\ out$
d: $shape\ parameter\ for\ wear\ out$

Which can be rewritten into:

$$\lambda(t) = a^b bt^{b-1} + c^d dt^{d-1} \tag{3.3}$$

As a^b and c^d are constant with respect to t, these can also be expressed in a single parameter a and c. This simplifies calculations, at the expense of the scale parameters (a and c) being less intuitive in use. A constant is introduced to allow for adjustment to the absolute level. The following hazard rate follows:

$$\lambda(t) = abt^{b-1} + cdt^{d-1} + f \tag{3.4}$$

with:
f: $vertical\ location\ parameter$
$a \begin{cases} < 0, if\ b < 0 \\ > 0, if\ b > 0 \end{cases}$
$b < 1$
$c > 0$
$d > 2$

If the parameters $a = -1$, $b = -2$, $c = 1$, $d = 3$ are chosen, the curve of the hazard rate over time is visible in figure 2.

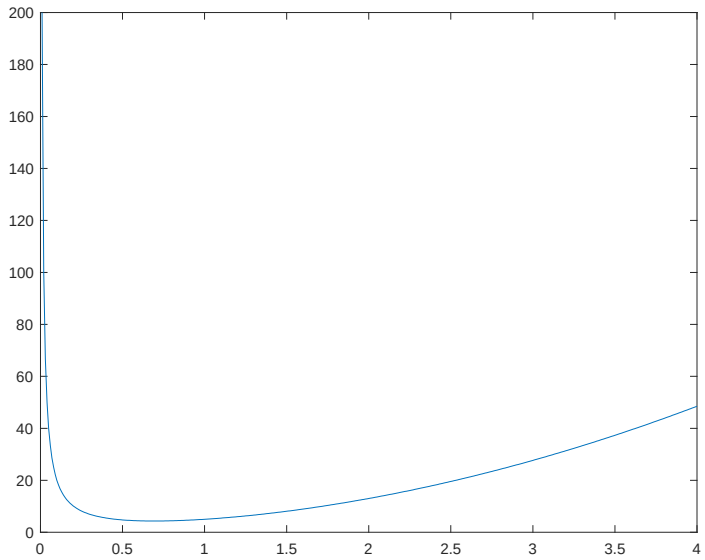


Figure 2, Example of the additive Weibull failure rate

On the vertical axis is the failure rate, on the horizontal axis the time.

The first term represents the break in phase, while the second one models the wear out. By adapting the five parameters, the model can be made to fit the behavior of the component. The constants will have to be estimated from observed data. The objective here is to minimize the sum of squared differences between predicted and observed values. Note that in cases where there is no break in period, the first function may simply be left out.

If the estimated value of f results in a minimum lower than zero, the model will predict parts "unbreaking". When applying real data to estimate the parameters, this may pose a risk. It is of course unrealistic and should be prevented by guaranteeing that the failure rate is nonnegative for all positive values. The time at which the failure rate is minimal can be obtained as follows:

$$\frac{d\lambda(t_{min})}{dt} = 0 \tag{3.5}$$

Taking the derivative of (3.4) yields:

$$ab(b-1){t_{min}}^{b-2} + cd(d-1){t_{min}}^{d-2} = 0 \tag{3.6}$$

Which can be rewritten into:

$$t_{min} = \left(-\frac{cd(d-1)}{ab(b-1)}\right)^{\frac{1}{b-d}} \tag{3.7}$$

with:
t_{min}: $t\ at\ minimal\ failure\ rate$

The minimum value of the hazard rate should be at least zero. It is however possible for the estimated failure rate to drop below zero in case parameter f is chosen incorrectly. The following inequality has to hold:

$$\lambda(t_{min}) \geq 0 \tag{3.8}$$

Inserting t_{min} from (3.7) into (3.4) gives:

$$ab\left(\left(-\frac{cd(d-1)}{ab(b-1)}\right)^{\frac{1}{b-d}}\right)^{b-1} + cd\left(\left(-\frac{cd(d-1)}{ab(b-1)}\right)^{\frac{1}{b-d}}\right)^{d-1} + f \geq 0 \tag{3.9}$$

Which can be rewritten into the (3.10).

$$ab\left(-\frac{cd(d-1)}{ab(b-1)}\right)^{\frac{b-1}{b-d}} + cd\left(-\frac{cd(d-1)}{ab(b-1)}\right)^{\frac{d-1}{b-d}} + f \geq 0 \tag{3.10}$$

Bringing everything that is not f to the other side yields (3.11).

$$f \geq -ab\left(-\frac{cd(d-1)}{ab(b-1)}\right)^{\frac{b-1}{b-d}} - cd\left(-\frac{cd(d-1)}{ab(b-1)}\right)^{\frac{d-1}{b-d}} \tag{3.11}$$

Using the hazard function, the cumulative hazard rate as in (.1) may be determined. Its definition is the integral of (7.4) over 0 to t. The integral for the additive Weibull distribution models is in (3.12).

$$\Lambda(t) = \int_0^t \lambda(t)dt = at^b + ct^d + ft \tag{3.12}$$

If one is interested in minimizing the number of failures over a set time horizon, the average number of failures per unit of time as a function of the maintenance interval is of interest. Depending on the distribution, the formula for the average failure rate over time is described in (3.13).

$$AFR(t) = \frac{\Lambda(t)}{t} = \frac{at^b + ct^d + ft}{t} = at^{b-1} + ct^{d-1} + f \tag{3.13}$$

with:
$AFR(t)$: $Average\ Failure\ Rate\ at\ t$

The optimal maintenance strategy from a reliability point of view would be one that minimizes the average number of failures. This can be found analytically by minimizing (3.13). Its first derivative is:

$$\frac{dAFR(t)}{dt} = a(b-1)t^{b-2} + c(d-1)t^{d-2} \tag{3.14}$$

The maintenance interval that leads to the lowest average failure rate can then be found by equating (3.14) to zero and rewriting. For the additive Weibull distribution, it results in:

$$a(b-1)t^{b-2} + c(d-1)t^{d-2} = 0 \tag{3.15}$$

Extracting t gives (3.16).

$$t = \left(-\frac{c(d-1)}{a(b-1)}\right)^{d-b} \tag{3.16}$$

However, this does not take the direct cost of maintenance into account. To determine the average total cost of a maintenance strategy, it should. The total cost over one maintenance period including both cost of maintenance and cost of failure is a function of the cost of failure, the expected number of failures (3.12) and the cost of maintenance:

$$TC(t) = CoF \times \Lambda(t) + CoM = CoF \times (at^b + ct^d + ft) + CoM \tag{3.17}$$

with:
$TC(t)$: $Total\ Cost\ of\ a\ maintenance\ interval\ of\ length\ t$
CoF: $The\ cost\ of\ a\ failure\ occurence\ of\ the\ part$
CoM: $The\ cost\ of\ performing\ maintenance\ once\ on\ the\ part$

The total cost rate is (3.17) over time. It represents the average cost per unit of time of a maintenance interval, as opposed to the marginal cost or the total cost (3.17). This is relevant, because an expensive maintenance interval may be efficient if the cost is spread over a sufficient amount of time. It is defined as:

$$TCR(t) = CoF \times \mathrm{AFR}(t) + \frac{CoM}{t} = CoF \times (at^{b-1} + ct^{d-1} + f) + \frac{CoM}{t} \tag{3.18}$$

$TCR(t)$: $Total\ Cost\ Rate\ of\ a\ maintenance\ interval\ of\ length\ t$

The economically optimal maintenance interval assuming fixed maintenance intervals is found at the minimum of the total cost rate:

$$\frac{dTCR(t)}{dt} = 0 \tag{3.19}$$

Which can be rewritten using (3.18).

$$CoF \times \frac{dAFR(t)}{dt} - \frac{CoM}{t^2} = 0 \tag{3.20}$$

After inserting (3.14), (3.21) is obtained

$$CoF \times \left(a(b-1){t_{opt}}^{b-2} + c(d-1){t_{opt}}^{d-2}\right) - \frac{CoM}{{t_{opt}}^2} = 0 \tag{3.21}$$

with:
t_{opt}: $the\ optimal\ maintenance\ interval$

3.3.2 The Additive Gompertz-Makeham Distribution

In the previous subsection, a model for determining the cost resulting from a maintenance strategy is presented that is based on a Weibull distribution. However, any distribution can be used, as long as a failure rate is defined, and that failure rate has an elementary integral. Another distribution that satisfies these criteria is the Gompertz-Makeham distribution (Finkelstein, 2008). Similar to the previous deterioration model, calculations can be made and the failure rate can be expressed in a function of t. This subsection serves the same purpose as the previous. In order to keep the subsections independently legible, most of this section's content are the same as the previous one's. The letters used for the parameters are the same, and have a similar meaning.

$$\lambda(t) = abe^{bt} + cde^{dt} + f \tag{3.22}$$

with:
a: $location\ parameter\ for\ break\ in$
b: $scale\ parameter\ for\ break\ in$
c: $location\ parameter\ for\ wear\ out$
d: $scale\ parameter\ for\ wear\ out$
f: $vertical\ location\ parameter$
$a, b < 0$
$c, d > 0$

Using the parameters $a = -1$, $b = -1$, $c = 0.0001$, $d = 1$, the failure rate looks is graphed in figure 3.

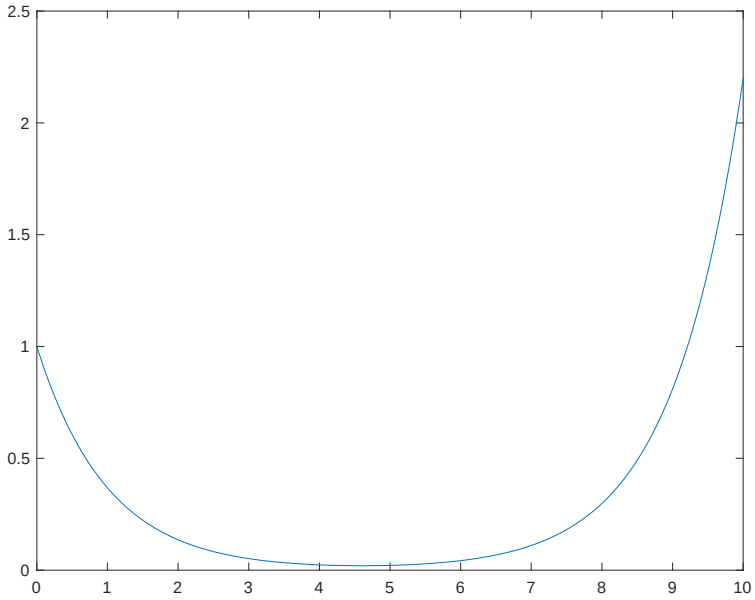


Figure 3, Example Additive Gompertz-Makeham failure rate

On the vertical axis is the failure rate, on the horizontal axis the time.

The first term represents the break in phase, while the second one models the wear out. By adapting the five parameters, the model can be made to fit the behavior of the component. The constants will have to be estimated from observed data. The objective here is to minimize the sum of squared differences between predicted and observed values. Note that in cases where there is no break in period, the first function may simply be left out.

If the estimated value of f results in a minimum lower than zero, the model will predict parts "unbreaking". When applying real data to estimate the parameters, this may pose a risk. It is of course unrealistic and should be prevented by guaranteeing that the failure rate is nonnegative for all positive values. The time at which the failure rate is minimal can be obtained as follows:

$$\frac{d\lambda(t_{min})}{dt} = 0 \tag{3.23}$$

Taking the derivative of (3.23) yields:

$$ab^2 e^{bt_{min}} + cd^2 e^{dt_{min}} = 0 \tag{3.24}$$

Which can be rewritten into (3.25).

$$t_{min} = \frac{\ln\left(-\frac{cd^2}{ab^2}\right)}{b-d} \tag{3.25}$$

with:
t_{min}: $t\ at\ minimal\ failure\ rate$

The minimum value of the hazard rate should be at least zero. It is however possible for the estimated failure rate to drop below zero in case parameter f is chosen incorrectly. The following inequality has to hold:

$$\lambda(t_{min}) \geq 0 \tag{3.26}$$

Inserting t_{min} from (3.25) into (3.22) gives:

$$abe^{b\frac{\ln\left(-\frac{cd^2}{ab^2}\right)}{b-d}} + cde^{d\frac{\ln\left(-\frac{cd^2}{ab^2}\right)}{b-d}} + f \geq 0 \tag{3.27}$$

Which can be rewritten into (3.28):

$$ab\left(-\frac{cd^2}{ab^2}\right)^{\frac{b}{b-d}} + cd\left(-\frac{cd^2}{ab^2}\right)^{\frac{d}{b-d}} + f \geq 0 \tag{3.28}$$

Bringing everything that is not f to the other side yields (3.29).

$$f \geq -a\left(-\frac{cd^2}{ab^2}\right)^{\frac{b}{b-d}} - c\left(-\frac{cd^2}{ab^2}\right)^{\frac{d}{b-d}} \tag{3.29}$$

Using the hazard function, the cumulative hazard rate may be determined. Its definition is the integral of (3.22) over 0 to t.

$$\begin{aligned}\Lambda(t) = \int_0^t \lambda(t)dt &= (ae^{bt} + ce^{dt} + ft + C) - (ae^{b\times 0} + ce^{d\times 0} + f \times 0 + C) \\ &= ae^{bt} + ce^{dt} + ft - a - c\end{aligned} \tag{3.30}$$

Note that the constant component (ft) is identical between the two models. If one is interested in minimizing the number of failures over a set time horizon, the average number of failures per unit of time as a function of the maintenance interval is of interest. The formula for the average failure rate over time is described in (3.31):

$$AFR(t) = \frac{\Lambda(t)}{t} = \frac{ae^{bt}}{t} + \frac{ce^{dt}}{t} + f - \frac{a}{t} - \frac{c}{t} \tag{3.31}$$

with:
$AFR(t)$: $Average\ Failure\ Rate\ at\ t$

The optimal maintenance strategy from a reliability point of view would be one that minimizes the average number of failures. This can be found analytically by minimizing (3.31). Its first derivative is:

$$\frac{dAFR}{dt} = \frac{abe^{bt}}{t} - \frac{ae^{bt}}{t^2} + \frac{cde^{dt}}{t} - \frac{ce^{dt}}{t^2} + \frac{a}{t^2} + \frac{c}{t^2} \tag{3.32}$$

The maintenance interval that leads to the lowest average failure rate can then be found by equating (3.32) to zero and rewriting. For the additive Gompertz-Makeham distribution, it results in (3.33).

$$\frac{abe^{bt}}{t} - \frac{ae^{bt}}{t^2} + \frac{cde^{dt}}{t} - \frac{ce^{dt}}{t^2} + \frac{a}{t^2} + \frac{c}{t^2} = 0 \tag{3.33}$$

This equation can be simplified into (3.34). The solution has to be computed numerically.

$$abte^{bt} - ae^{bt} + cdte^{dt} - ce^{dt} + a + c = 0 \tag{3.34}$$

However, this does not take the direct cost of maintenance into account. To determine the average total cost of a maintenance strategy, it should. The total cost over one maintenance period including both cost of maintenance and cost of failure is a function of the cost of failure, the expected number of failures (3.30) and the cost of maintenance:

$$TC(t) = CoF \times \Lambda(t) + CoM = CoF \times (ae^{bt} + ce^{dt} + ft - a - c) + CoM \tag{3.35}$$

with:
$TC(t)$: $Total\ Cost\ of\ a\ maintenance\ interval\ of\ length\ t$
CoF: $The\ cost\ of\ a\ failure\ occurence\ of\ the\ part$
CoM: $The\ cost\ of\ performing\ maintenance\ once\ on\ the\ part$

The total cost rate is (3.35) over time. It represents the average cost per unit of time of a maintenance interval, as opposed to the marginal cost or the total cost (3.35). This is relevant, because an expensive maintenance interval may be efficient if the cost is spread over a sufficient amount of time. It is defined as:

$$TCR(t) = \frac{TC}{t} = CoF \times \mathrm{AFR} + \frac{CoM}{t} = CoF \times \left(\frac{ae^{bt}}{t} + \frac{ce^{dt}}{t} + f - \frac{a}{t} - \frac{c}{t}\right) + \frac{CoM}{t} \tag{3.36}$$

with:
$TCR(t)$: $Total\ Cost\ Rate\ of\ a\ maintenance\ interval\ of\ length\ t$

The economically optimal maintenance interval assuming fixed maintenance intervals is found at the minimum of the total cost rate:

$$\frac{dTCR(t)}{dt} = 0 \tag{3.37}$$

Which can be rewritten using (3.36)

$$CoF \times \frac{dAFR(t)}{dt} - \frac{CoM}{t^2} = 0 \tag{3.38}$$

After inserting (3.32), (3.39) is obtained

$$CoF \times \left(\frac{abe^{bt}}{t} - \frac{ae^{bt}}{t^2} + \frac{cde^{dt}}{t} - \frac{ce^{dt}}{t^2} + \frac{a}{t^2} + \frac{c}{t^2} \right) - \frac{CoM}{{t_{opt}}^2} = 0 \tag{3.39}$$

with:
t_{opt}: $the\ optimal\ maintenance\ interval$

3.4 Examples of Each Distribution

The model for the Weibull model as discussed above is applied to a small fictitious case. The parameters are:

```
a = [0.2 0.3 0.4];
b = [0.2 0.3 0.4];
c = [0.002 0.005 0.0035];
d = [2.8 2.2 2.5];
f = [0 0 0];
```

This case consists of three components. Each component has one value within each vector. The first parameter in each vector refers to the first component, et cetera.

The failure rates for this case are presented in figure 4.

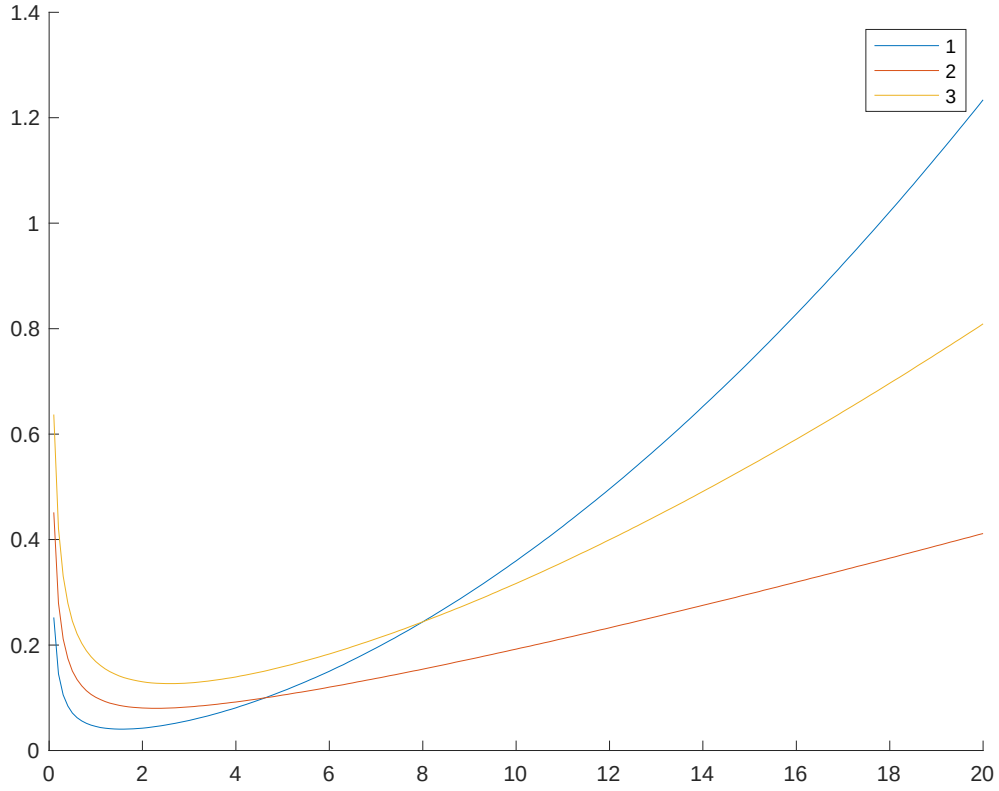


Figure 4, failure rates of the Weibull example

The optimal maintenance intervals (t axis) for these functions are 17.1, 16.8 and 13.0 for the blue, orange and yellow lines respectively. This is caused by the higher CoF of the last component.

An example of the Gompertz-Makeham specification can also be presented in a graph. The following parameter set was used:

```
a = [-2 -3 -4];
b = [-0.2 -0.3 -0.4];
c = [2 5 8];
d = [0.016 0.016 0.02];
f = [0 0 0 0 0];
```

This is one of the parameter sets that is used to test the performance of the optimization model. The failure rates of these components are shown in figure 5.

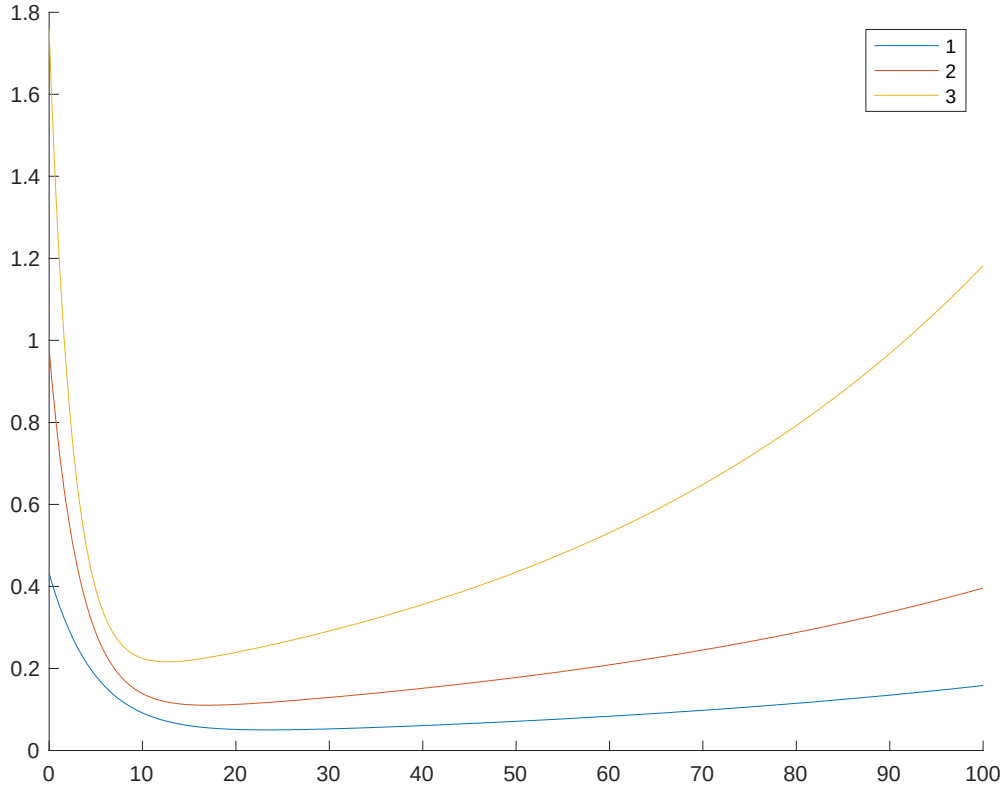


Figure 5, failure rates of three components

The three components have slightly different characteristics, with the first instance of each parameter being related to the blue line, the second one to the red line and finally the third instance to the yellow line.

Which of these distributions is the most suited and what the corresponding parameters are, should be determined using observed data for each case individually. Generally, the model that produces the smallest sum-of-squares of the error is most suitable. This also means that the predictive and prescriptive qualities of the model are possibly highly dependent on the quality of the data. A post-hoc sensitivity analysis can confirm the model's robustness. An example of such a process is in section 4.2.

3.5 Optimization Models

Using the calculations for the individual components, a model is set up to achieve an optimal maintenance strategy as a function of maintenance intervals. The objective of this model is to minimize the cost of the maintenance strategy by adjusting the schedule. This can be done by binary programming.

3.5.1 Linear Programming and Optimizing Maintenance

Arguably the most common tool in operations research is linear programming (LP). It allows for solving a large variety of optimization problems. Linear programming uses decision variables that represent the decisions that are made in operating a system, an objective function that is used to determine the cost or benefit of a certain solution, and constraints

that limit the decision variables to feasible values. A graphical representation of linear constraints in two dimensions with three constraints can be seen in figure 6. The colored areas are infeasible, so the solution has to be found within the white triangle that is not cut off by the constraints. Cases where (some of) the decision variables are integers are called mixed integer linear programming (MILP). The solution space is not a continuum, and instead consists of discrete points. Binary programming is a specific form of mixed integer programming, where each decision variable can only be zero or one.

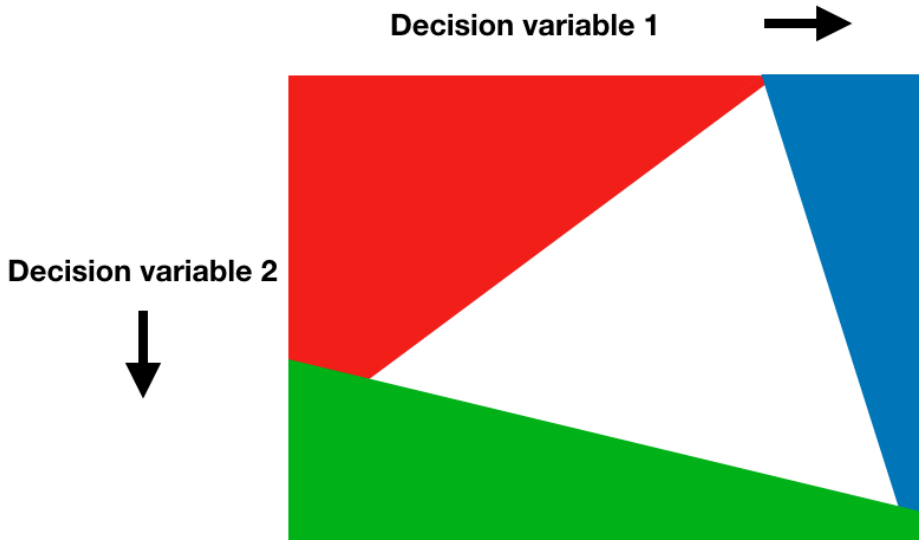


Figure 6, linear constraints

3.5.2 Objective Function and Decision Variables

The core of the linear optimization model is the objective function. It is used to calculate the cost of an alternative. It has to be a linear function of the decision variables. The decision variables of the model are the maintenance intervals for the different components and the necessity of possessions. As the maintenance interval cannot be specified more accurately than with a precision of one week, it is appropriate to use integer decision with a step of one week. The decision variable x_i represents the fixed interval for each component i while y_j represents a possession at j. If maintenance is performed and a possession is needed at time j, $y_j = 1$, if not then $y_j = 0$. Depending on the statistical model that is assumed (see section 3.3), an objective function is specified.

An underlying assumption of linear programming is that the objective value changes linearly with each decision variable. However, the problem at hand has different characteristics. First, the expected cost of failure is not linear. Second, there have to be constraints on the decision variables for the possessions to be taken up that are not linearly related to the decision variables for the maintenance intervals. For example, a decision variable z for a maintenance operation and for y_3 for a possession may have a relationship as shown in figure 7. Only the situation where there is maintenance (z=3), but no possession (y3=0) needs to be eliminated. With an ordinary implementation of a maintenance interval as a decision variable, the constraint cannot be expressed in linear programming. The red lines represent the possible constraints. Every constraint would also cut off feasible solutions, in which case an optimal solution can no longer be guaranteed. A solution is needed that makes it possible to solve this problem.

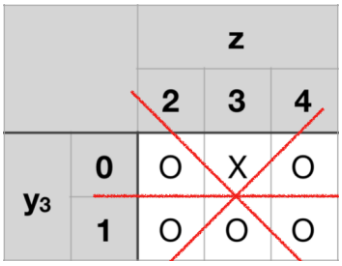


Figure 7, impossible constraint in integer linear programming

3.5.3 Unary Encoding of Decision Variables

A solution is provided by unary encoding (Refalo, 2000, Belov et al., 2016). Integrity enables a different way of specifying the decision variables: instead of using one integer decision variable, one binary variable for each possible value of the integer decision variable can be used. Conventionally, a single variable x with the following properties would be used:

$$x \leq u,$$
$$x, u \in \mathbb{N} \tag{3.40}$$

In other words, a nonnegative integer with upper bound u. For example, consider a decision variable with an upper bound of 5. By applying unary encoding, it is replaced by a set of binary variables x_0 to x_5, where each one represents a value of the single variable x. The following constraints apply:

Every x_i is binary for all values of i.

$$x_i \in \{0,1\} \; \forall \; i \in \{0 \dots u\} \tag{3.41}$$

Exactly one of the decision variables x_i is one.

$$\sum_{i \in \{0 \dots u\}} x_i = 1 \tag{3.42}$$

In all equations, x can be replaced by the expression in (3.43)

$$\sum_{i \in \{0 \dots u\}} i x_i = x \tag{3.43}$$

The conventional term in the objective function would be:

$$cx \tag{3.44}$$

This is now replaced by:

$$\sum_i c_i x_i \tag{3.45}$$

It is important to note that c_i can take any real value. The effect, of the maintenance interval on the objective function no longer has to be linear.

The same problem as in figure 7 is expressed in figure 8. However, the decision variable z is now programmed using unary encoding. Because there is now a corner of the solution space that needs to be cut off, the problem is able to be solved by linear programming.

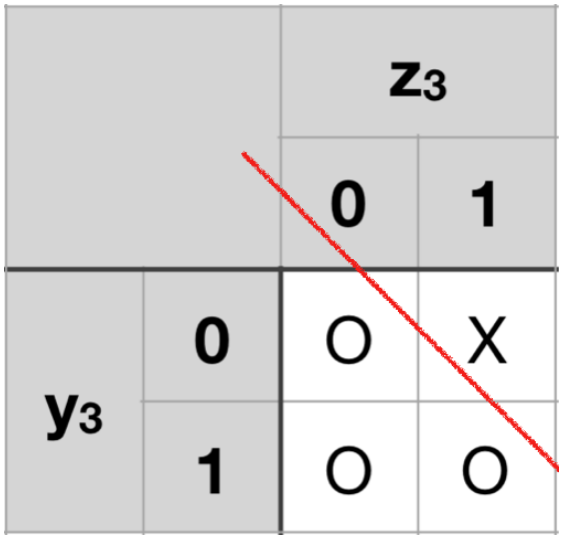


Figure 8, constraint with unary encoding

Possible negative effects include a large increase in the number of decision variables and being less efficient due to the inability for the optimization solver to take advantage of a continuous slope of the objective function.

3.6 Fixed Interval Railway Maintenance Optimization Model

3.6.1 Objective Function

This model describes a situation in which the maintenance interval to each component is fixed, that is, the time between each different maintenance operations is always the same. This means it is not necessary to determine each interval length individually.

Minimizing the total cost of the interval would simply result in using the shortest possible interval. Hence, the appropriate objective is to minimize the average cost over time. the sum of the Total Cost Rate over the planning horizon and the cost of possessions will be minimized. The formula (3.18) has to be adapted into a linear form through unary encoding. Note that the decision variables x_{ik} have two indices; i and k. Index i represents the component, while k is an index that is added to allow for unary encoding.

$$\min \sum_{i=1}^{C} \sum_{k=1}^{V_i} \left(CoF_i \times \left(a_i k^{(b_i-1)} + c_i k^{(d_i-1)} + f_i \right) + \frac{CoM_i}{k} \right) \times W \times x_{ik} + \sum_{i}^{W} CoP_i y_i \tag{3.35}$$

where:
a_i, b_i, c_i, d_i, f_i: The parameters of the failure distribution as described in section 3.3.
C: $The\ number\ of\ components\ under\ consideration$
V_i: $The\ number\ of\ possible\ values\ of\ the\ interval\ length$
CoF_i: $cost\ of\ failure\ of\ component\ i$
x_{ik}: $The\ binary\ decision\ variable\ representing\ component\ i\ having\ a\ maintenance\ interval\ of\ k$
CoM_i: $cost\ of\ maintenance\ of\ component\ i$
W: $The\ number\ of\ weeks\ within\ the\ planning\ horizon$
CoP_i: $cost\ of\ possession\ at\ time\ i$
y_i: $binary\ variable\ for\ possession\ at\ time\ i$

If the objective function is based on the additive Gompertz-Makeham distribution (3.36), it becomes:

$$\min \sum_{i=1}^{C} \sum_{k=1}^{V_i} \left(CoF_i \left(\frac{a_i e^{b_i k}}{b_i k} + \frac{c_i e^{d_i k}}{d_i k} + f_i - \frac{a_i}{b_i k} - \frac{c_i}{d_i k} \right) + \frac{CoM_i}{k} \right) \times W \times x_{ik} + \sum_{i=1}^{W} CoP_i y_i \tag{3.46}$$

As the value of f_i is constant, it can be taken out of the objective function at will. The meaning of the variables is the same as for the other distribution.

3.6.2 Constraints

The model is subject to two constraints. Inequality (3.47) represents the requirement to have a possession at every multiple of the possession interval. This is used to include the cost of possession into the objective. It promotes a schedule in which maintenance is synchronized to reduce costs resulting from possessions.

$$x_{ik} - \sum_{j=1}^{\left\lfloor \frac{W}{k} \right\rfloor} \frac{1}{\left\lfloor \frac{W}{k} \right\rfloor} \times y_{j \times k} \leq 0 \ \forall\ i \in \{1 \ldots C\}, k \in \{1 \ldots W\} \tag{3.47}$$

For example, consider a planning horizon of 10 weeks and a maintenance interval of 3 weeks for component i. The number of maintenance operations is 3. Maintenance happens in week 3, 6 and 9. Possession need to be assigned in these weeks. Filling in (3.47) yields:

$$x_{ik} - \sum_{j=1}^{\left\lfloor \frac{W}{k} \right\rfloor} \frac{1}{\left\lfloor \frac{W}{k} \right\rfloor} \times y_{j\times k} = 1 - \sum_{j=1}^{\left\lfloor \frac{10}{3} \right\rfloor} \frac{1}{\left\lfloor \frac{10}{3} \right\rfloor} \times y_{j\times 3} = 1 - \left(\frac{1}{\left\lfloor \frac{10}{3} \right\rfloor} y_{1\times 3} + \frac{1}{\left\lfloor \frac{10}{3} \right\rfloor} y_{2\times 3} + \frac{1}{\left\lfloor \frac{10}{3} \right\rfloor} y_{3\times 3} \right)$$
$$= 1 - \left(\frac{1}{3} y_3 + \frac{1}{3} y_6 + \frac{1}{3} y_9 \right) \leq 0 \; \forall \; i \in \{1 \dots C\}, k \in \{1 \dots W\} \tag{3.48}$$

Evidently, (3.47) is satisfied if and only if y_3, y_6 and y_9 all have a value of one.

This constraint cannot be modelled as an MILP problem without unary encoding, as the number of elements of the summation would depend on a decision variable.

Additionaly, to ensure exactly one interval length is selected for each component, constraint (3.49) is added.

$$\sum_{k=1}^{V_i} x_{ik} = 1 \; \forall \; i \in \{1 \dots C\} \tag{3.49}$$

Appendix II contains the MATLAB code for this model.

3.6.3 Properties, Possibilities and Limitations

The fixed interval railway maintenance optimization problem is able to optimally design maintenance schedules, given that the maintenance intervals for each component are constant. This is actually fairly common in practice and allows for a relatively simple model formulation which in turn enables solving large problems, for example with many components and a long planning horizon. However, the state of maintenance at the beginning of the schedule is not included. in the model. Hence, a component will not be maintained until after one full interval has passed. Furthermore, a fixed interval may not be the best possible solution as the ability to make maintenance operations coincide by making small adjustments to the schedule is not available.

3.7 Individual Interval Railway Maintenance Optimization Model

The formulation for this optimization problem is an extension of the previous problem. The major difference is that instead of using one variable for a component, which describes its interval, a set of variables is used to define each individual interval. The advantage is that a more accurate planning can be made. For example, it may be beneficial to deviate from the ideal interval to have the maintenance coincide with other maintenance operations.

3.7.1 Decision Variables

The model has three sets of decision variables. First, x_{ijk} defines if the jth maintenance interval to component i has a value of k. The third index is needed in this model to define the different successive intervals between maintenance operations. Second, y_i defines the

occurrence of a possession at time i. Finally, z_{ijk} is an auxiliary variable that is needed to properly apply constraints to the other decision variables. It can be thought of as the point in time on which a maintenance operation takes place. For example, if last maintenance was at -5, and the first two intervals are 10 long, the first two maintenance operations are at 5 and 15.

3.7.2 Objective Function

The objective function for the expected cost of failure that is presented here is derived from the additive Gompertz-Makeham lifetime distribution. The objective function becomes:

$$\min \sum_{i=1}^{C} \sum_{j=1}^{Imax_i} \sum_{k=0}^{V_i} \left(CoF_i \times (ae^{bk} + ce^{dk} + fk) + CoM_{ijk}\right) \times x_{ijk} + \sum_{i}^{W} CoP_i y_i \tag{3.50}$$

where:
C: the components in the system under consideration.
$Imax_i$: The maximum number of maintenance intervals of component i. Note that a maximum of n maintenance intervals implies a maximum of n-1 maintenance operations. This is because it is assumed that there is no maintenance at the beginning or the end of the contract period.
V_i: The possible values of the interval length for component i.
CoF_i: The cost of failure of component i.
a_i, b_i, c_i, d_i, f_i: The parameters of the failure distribution as described in section 3.2.
x_{ijk}: The decision variable representing interval j of component i being k long.
CoM_{ijk}: The cost of maintenance at interval j to component i for interval length k.
W: The weeks within the planning horizon.
CoP_j: The cost of possession at week j.
y_i: The decision variable representing a possession at time j.

A difficulty is that the number of maintenance operations to each component should ideally not be fixed, but also optimized. This is made possible by introducing an interval of 0. For each interval of 0, one less maintenance operation is performed. This is a fictitious situation in which two maintenance situations are planned at the same time. Of course, in reality, maintenance would only be performed once. To account for this, there is no cost of maintenance in case of an interval of 0. Thus:

CoM_{ijk} equals the cost of maintenance if $j > 1$ and $k > 0$.
CoM_{ijk} equals 0 else.

Function (3.50) differs from the objective function from the model for constant intervals, as the objective function no longer contains the average expected cost per unit of time, but the total expected cost per interval.

3.7.3 Constraints

There are three groups of constraints. The constraints are organized in such a way that whenever maintenance is planned, a possession is required. The first constraints (3.51) restrict the auxiliary variables. First, the auxiliary decision variables are linked to the decision variables for intervals. Through this constraint, the auxiliary variables are defined such that they can be used in constraint (3.53).

$$\sum_{j=1}^{l}\sum_{k=0}^{V_i} kx_{ijk} - \sum_{k=0}^{W} kz_{ikl} = 0 \;\forall\; i \in \{1\dots C\}, l \in \{1\dots O_i\} \tag{3.51}$$

with:
l: The maintenance operations of component i. As there is a maintenance operation in between each interval, there are always one fewer maintenance operations than intervals.
V_i: The possible values of the interval length for component i.
x_{ijk}: The decision variable representing interval j of component i being k long.
z_{ikl}: The decision variable representing maintenance operation l of component i taking place at k.
O_i: The set of maintenance operations to component i.

Next up, constraint (3.52) ensures the maintenance intervals cover all of the planning horizon.

$$\sum_{j=1}^{Imax_i}\sum_{k=0}^{V_i} kx_{ijk} = W \;\forall\; i \in \{1\dots C\} \tag{3.52}$$

The last category (3.53) connects the maintenance operations to the possessions, ensuring there is a possession when there is maintenance.

$$\sum_{i=1}^{C}\sum_{l=1}^{O_i} z_{ikl} - My_k < 0.5 \;\forall\; k \in \{1\dots W\} \tag{3.53}$$

The right-hand side is 0.5, because the function has to be at most zero and always has an integer outcome. Any value that is greater than or equal to 0, but smaller than 1 can be used, and will result in the same model outcome. Using 0.5 instead of 0, which might be a more obvious choice, yields slight performance benefits. For the same reason, this technique is also applied elsewhere.

A Big M notation is used. In this case, M should have the largest value $\sum_i^C \sum_l^{O_i} z_{ikl}$ could theoretically take on. In other words, all maintenance operations to all components. Hence:

$$M = \sum_{i=1}^{C} O_i \tag{3.54}$$

Additionally, constraints are included that prevent outcomes that violate the requirements of modelling using binary variables. This means a situation where a single interval or auxiliary does not have exactly one value.

$$\sum_{k=0}^{V_i} x_{ijk} = 1 \; \forall \; i \in \{1 \dots C\}, j \in \{1 \dots Imax_i\} \tag{3.55}$$

$$\sum_{k=0}^{W} z_{ikl} = 1 \; \forall \; i \in C\{1 \dots C\}, l \in \{1 \dots O_i\} \tag{3.56}$$

This model as coded in MATLAB is found in appendix III.

3.7.4 Comparison with the Fixed Interval Railway Maintenance Optimization Problem

Compared to the problem presented in section 3.6, the problem in this section has some key advantages. The limitation regarding state of maintenance at the beginning of the planning horizon is solved, as is the limitation on varying the schedule. It is also able to make small adjustments to the schedule, for example extending an interval by a week to have it coincide with another maintenance operation.

The solution of this problem requires the addition of many more decision variables. This will mean that for the same problem size, the complexity is much larger. In other words, for given maximum computing resources, the possible problem size is smaller. The model using fixed intervals should be more suited to very large problems, as there are fewer decision variables and constraints. All in all, these two different optimization problems have their own different strengths and weaknesses, and they should be regarded as complementary to each other.

3.7.5 Improving Performance of the Individual Interval Railway Maintenance Optimization Model

The script in which the model is implemented was written iteratively. After all desired functionality was added, measures were taken to reduce the computational complexity as much as possible. These measures include:

1. Eliminating decision variables whose information can also be taken from other decision variables. In order to do this, some of the constraints had to be rewritten such that all the coefficients for the decision variable that was to be eliminated are zero.

2. Reducing the number of nonzero elements for constraints. Experimentation has shown that a large number of nonzeros can increase runtime dramatically. By using the variables for the points in time to constrain the next point in time rather than taking the sum of all intervals has reduced the number of nonzeros in the equations.
3. Transforming equality constraints into inequality constraints. Each equality constraint in a numerical model is in practice a set of two inequality constraints of the form: $c - e \leq x \cdot a \leq c + e$, where c is the constant in right hand side, e is a small numerical error, x is the decision variable and a is the coefficient associated with x. If the requirement for the equation to be equal to c can be transformed into a requirement to be equal to or smaller than c, such a constraint can be transformed into an inequality and performance can be increased.
4. Adding constraints that ensure a unique solution. For each interval, a value of zero means that there is effectively no interval. This can lead to different solutions being exactly the same in practice, and also having the same objective function. For example, it means that a solution where the second interval is forty and the third one zero, and a solution where the second interval is zero and the third forty are both optimal. Eliminating such possibilities improves runtimes.

3.7.6 Improved Reformulation of the Individual Interval Railway Maintenance Optimization Model Using Unary Encoding

The formulation of the problem in 3.7.1 through 3.7.3 was improved by taking the measures in 3.7.5. The objective function is now expressed as follows:

$$\min \sum_{i=1}^{C} \left(\sum_{k=0}^{V_i} CoF_i \times (ae^{bk} + ce^{dk} + fk) \times x_{i1k} + \sum_{j=2}^{I_i} \sum_{k=1}^{V_i} (CoF_i \times (ae^{bk} + ce^{dk} + fk) + CoM_i) \times x_{ijk} \right) + \sum_{i}^{W} CoP_i y_i \tag{3.57}$$

The difference is that the intervals from number 2 up can no longer take on a value of zero. A zero interval for component i at interval j now emerges as an absence of any x_{ijk}. This is why a different summation (starting at $k = 0$) is applied to the first interval than to all subsequent intervals (starting at $k = 1$). As CoM_i only appears before decision variables where it is applicable ($j > 1$ and $k > 0$), the indices j and k are no longer needed.

Some of the auxiliary decision variables can be eliminated, as the relevant expression can also be expressed other decision variables. To be exact, the variables related to the moment of maintenance operation 1 and $n - 1$ are not needed. The graphical representation becomes:

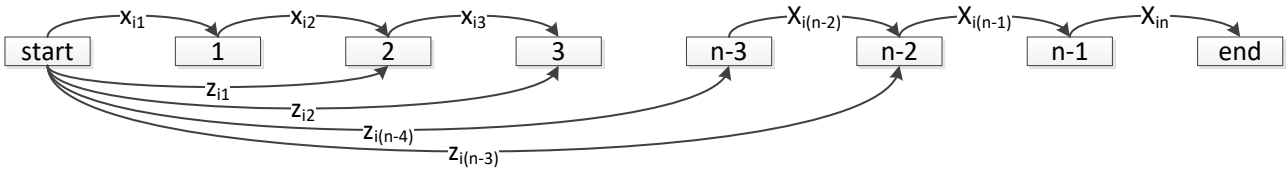


Figure 9, graphical representation of the decision variables in the improved model

The first moment of maintenance (z_{i1}) is no longer expressed in an auxiliary variable, as the value of the first interval (x_{i1}) is identical per definition (see also figure 9). The moment of the second maintenance operation is expressed as follows:

$$\sum_{k=0}^{V_i} kx_{i1k} + \sum_{k=1}^{V_i} kx_{i2k} + \sum_{k=0}^{W-1} (W-k)\, z_{i1k} = W \;\forall\, i \in \{1 \dots C\} \tag{3.58}$$

In the case of $k = Weeks$ the value of the coefficient for z_{i1k} becomes zero. The summation related to z only goes to $Weeks - 1$. As this is the case for the coefficients in all other constraints that apply to this decision variable, it can be eliminated altogether. It is an example of application of technique 1 and 2.

Every moment from 3 up is defined as the sum of the previous moment plus the interval in between, instead of as the sum of all preceding intervals. This also reduces the number of nonzero elements in the constraints, and is another application of technique 2.

$$\sum_{k=0}^{V_i} kx_{ijk} - \sum_{k=0}^{W-1} (W-k)\, z_{i(j-2)k} + \sum_{k=0}^{W-1} (W-k)\, z_{i(j-1)k} = 0 \;\forall\, i \in \{1 \dots C\}, j \in \{3 \dots Imax_i - 2\} \tag{3.59}$$

The moment of the last maintenance operation is also no longer expressed separately. The constraint that applies to the possession for the last maintenance operation is now based on the final maintenance interval. As such, the constraint defining the total planning horizon is defined as the sum of the second to last maintenance operation and the two last intervals:

$$\sum_{k=1}^{V_i} kx_{i(Imax_i-1)k} + \sum_{k=1}^{V_i} kx_{i(Imax_i)k} - \sum_{k=0}^{W-1} (W-k)\, z_{i(Imax_i-3)k} = 0 \;\forall\, i \in \{1 \dots C\} \tag{3.60}$$

The next constraint (3.61) is used to ensure that a possession is applied in case any element gets a maintenance at the beginning of the planning horizon.

$$\sum_{i=1}^{C} x_{i1\,0} - M_1 y_0 < 0.5 \tag{3.61}$$

Again, a big M expression is used. In this case, the big M should account for the case that all components are maintained at the beginning of the planning horizon. The following applies:

$$M_1 = C \tag{3.62}$$

The constraints applicable to weeks after zero are:

$$\sum_{i=1}^{C} \left(x_{i1k} + x_{i(Imax_i)(W-k)} + \sum_{j=1}^{Imax_i-3} z_{ijk} \right) - M_2 y_k < 0.5 \; \forall \; k \in \{1 \dots W-1\} \tag{3.63}$$

The value of big M is the same as in the initial model:

$$M_2 = \sum_{i=1}^{C} (Imax_i - 1) \tag{3.64}$$

The unary encoding constraints for each first interval are unchanged from the initial formulation:

$$\sum_{k=0}^{V_i} x_{i1k} = 1 \; \forall \; i \in \{1 \dots C\} \tag{3.65}$$

As there is now a possibility that the intervals other than the first have no value at all (meaning the interval is zero), the constraints for each subsequent interval are:

$$\sum_{k=1}^{V_i} x_{ijk} < 1.5 \; \forall \; i \in \{1 \dots C\}, j \in \{2 \dots Imax_i\} \tag{3.66}$$

These constraints are now inequalities as proposed in point 3. As with (3.53), 1.5 is used instead of 1 for performance benefits.

Finally, for the moments of maintenance the unary encoding constraints are now also inequalities. It is no longer needed for at least one of the decision variables for each moment to be a one.

$$\sum_{k=0}^{W-1} z_{ikl} < 1.5 \; \forall \; i \in \{1 \dots C\}, l \in \{1 \dots Imax_i - 3\}$$

(3.67)

If $\sum_{k=0}^{Weeks-1} z_{ijk} = 0$, the end of the planning horizon was reached using only earlier maintenance operations. Explicit definition of this moment is not necessary.

Finally, there is a set of constraints (3.58) to ensure a unique solution as mentioned in point 4. While not strictly necessary to arrive at an optimal solution, experimentation has shown that runtimes are reduced by including this constraint. This is probably due to the solver not having to explore all optimal branches of the problem. They can be cut off before the next optimal solution is found due to this constraint.

$$\sum_{k=1}^{V_i} x_{i(j-1)k} - \sum_{k=1}^{V_i} x_{ijk} > -0.5 \; \forall \; i \in \{1 \dots C\}, j \in \{3 \dots Imax_i\} \tag{3.68}$$

This model as implemented in MATLAB can be found in appendix IV.

4 Results

4.1 Experimentation

For any model to be relevant to industry users, it should be able to perform in practical situations. The input data as well as the output should be clearly defined and interpretable. The model should produce results reliably and within a practical amount of time.

Experimentation was performed to determine the behavior of the model under different circumstances. The inputs were changed, and the resulting runtimes evaluated and compared. First off, three standard parameter sets are defined, which are used for all performance testing. The input set was varied to account for specific characteristics of the problem. The order of the values of each parameter was randomly changed between each component. This step should account for any typical results that stem from the specific test problem.

This data is synthetic. While the values are in the order of magnitude that may be expected when using real data, there is no direct basis in practice. All simulations were run three times consecutively on the same parameter set to account for randomness in computer activity. The process was automated using the script in appendix XI. The numbers given are the medians of the three obtained figures for the first parameter set. The latter two parameter sets showed the same general trend without exception, hence a thorough analysis of this data is superfluous. Nonetheless, the resulting schedules are included in the appendices.

The results of each experiment are the maintenance intervals for the components and the necessary possessions. To make comparisons easier, a graph is used to present each result.

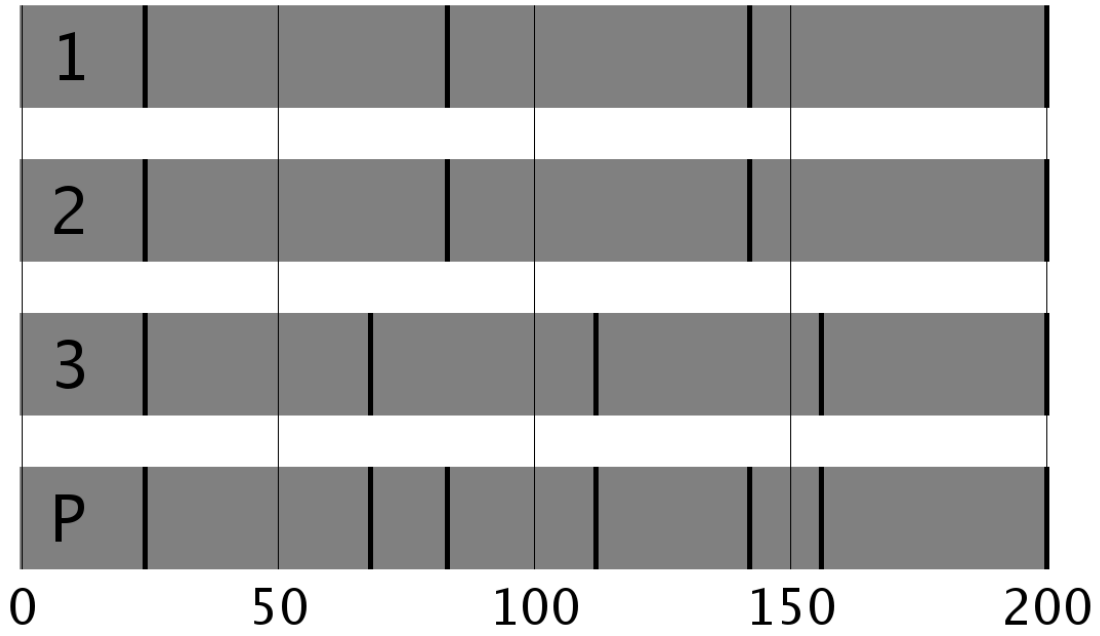


Figure 10, Example solution graph

An example of such a graph is in figure 10. In this case, a system with three components and a planning horizon of 200 weeks was optimized. On the horizontal axis is the time in weeks. The grey horizontal bands marked one to three represent the three components in the system. Each black vertical bar is a maintenance operation to the component. The horizontal grey bar marked P works in the same way but visualizes the possessions. When two black bars are in the same position horizontally, the components are maintained at the same time. In the example, it is clear to see that components 1 and 2 are always maintained at the same time, and that the maintenance operations are spaced equally.

A test loop was used to vary individual parameters. These parameters are chosen because they are suspected to have an influence on the computational performance of the model. The parameters that are varied are:

- Computational input
    - Optimality gap
    - Presolve setting
    - Number of processor threads
- Problem input
    - Wear out shape parameter
    - Cost of possession
        - Level of cost of possession
        - Varying cost of possession over time
    - Planning horizon
    - Number of components

All computing was performed on a 2012 Dell XPS 8500, with an Intel Core i7-8770 quad core, eight thread 3.4 GHz processor, and 8 GB 1,600 MHz DDR3 memory. The software used is Gurobi Optimizer 8.0.0 and MATLAB R2018a. All runtimes are in seconds, unless otherwise noted.

4.1.1 Optimality Gap

The optimality gap is the difference between the best feasible solution and the lower bound. An optimality gap of 0 means that the absolute optimal solution is found. The default setting for Gurobi is 0.0001, which means the solution returned is within 0.01% of the global optimum. Care should be taken in the interpretation of this value in case the objective value does not have an absolute zero point, for example if negative profit is a possibility. However, as an objective value of zero in this case represents zero cost, a relative definition of the optimality gap is meaningful.

Reducing the required optimality gap can help to lower runtimes of the model, while still giving a satisfactory result. To determine the difference between optimal and suboptimal solutions, a solution for several levels of optimality gap were requested. The exact experiment is found in appendix XIII. The runtimes are presented in table 1.

Table 1, performance as a function of the specified optimality gap

Specified optimality gap (%)	Median runtime	Actual optimality gap (%)

0	112.4	0
0.1*	91.3	0.10
0.3*	66.2	0.30
1	17.9	0.74
3	7.7	2.58
10	1.8	6.65
30	1.7	6.65

In the first two cases of nonzero optimality gap (noted by a *), the optimal solution was found (this was checked against the first run), but the optimality gap had not yet fully closed. The results are presented in figure 11.

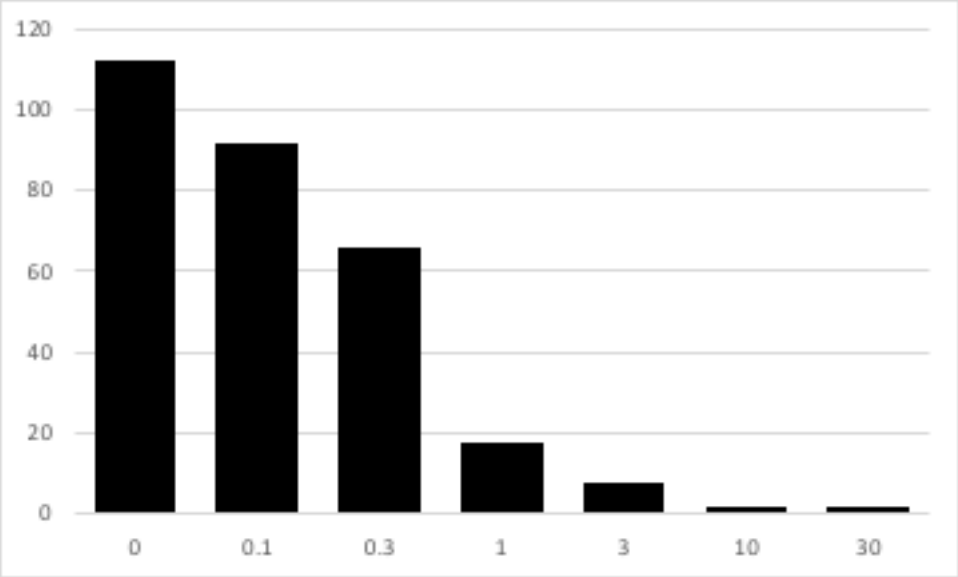


Figure 11, runtimes as a function of optimality gap setting

The resulting schedules can be found in appendix XXV. As the runtime improvements are very significant, specifying a higher optimality gap may be useful should the resources not be available to solve to optimality. Depending on the demands of the end user, the suboptimal solutions may be usable in practice. The results where the optimality gap is still very large are clearly suboptimal. For example, there are multiple maintenance intervals with a value of 1 in the cases of a 30% optimality gap. A longer solution time would definitely be worthwhile in those cases.

4.1.2 Presolve Setting

Presolve is an automatic operation by Gurobi before numerical computation of the objective takes place. Presolve is a procedure that simplifies the problem by eliminating redundant constraints. The number of nonzero elements in the constraints is reduced, and the solution time is less. However, presolve takes up resources by itself, which may negate the gains from faster processing. A description of the methods and purpose of presolve is given at (Gurobi, n.d.). A thorough evaluation of presolve in Gurobi is given by Achterberg et al. (2016).

By running the experiment in appendix XIV, the performance of the model using all different presolve settings was tested. The results of this test are in table 2 and figure 12.

Table 2, performance as a function of presolve setting

Presolve setting	Presolve time (s)	Removed columns	Removed rows	Median runtime	Median remaining optimality gap (%)
-1 (automatic)	0.03	70	0	363.9	0
0 (off)	-	-	-	3,600	0.46
1 (conservative)	0.03	70	0	551.3	0
2 (aggressive)	0.23	285	198	112.7	0

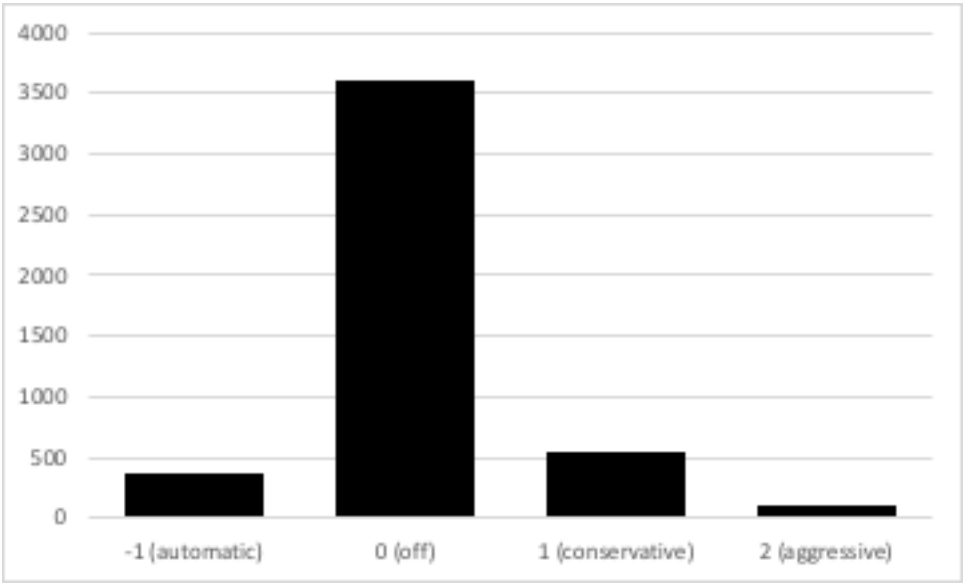


Figure 12, runtimes as a function of presolve setting

Clearly, presolve has a beneficial effect of solution times. The aggressive setting results in the most benefits. The additional time needed for presolve is negligible compared to the improvement in runtime. Therefore, it is advisable to manually set aggressive presolve when using this model. Interestingly, automatic and conservative presolve removed the same number of columns and no rows, in the same amount of time, yet there is a large performance gap between the two. The spread in runtimes was very small for each setting (under 15 seconds), so either different columns were removed or the presolve setting alters the actual solving strategy.

4.1.3 Number of Processor Threads

The processor in the computer that was used for simulation has four physical cores and can execute eight threads. For some problems and solvers, parallelization can significantly improve runtime, which offers potential for benefitting from computers with many processor cores or distributed computing. To find out whether this is the case for this problem, the number of used processor threads was varied using the params.threads parameter. In addition of one to eight, there is an automatic setting 0. The experiment in appendix XV yielded the results in table 3 and figure 13.

Table 3, performance as a function of thread setting

Thread setting	Median runtime	Improvement over next lower setting (%)

0 (automatic)	205.8	
1	450.5	
2	263.0	41.6
3	243.8	7.3
4	192.9	20.9
5	229.8	-19.1
6	144.5	37.1
7	187.3	-29.7
8	113.3	39.5

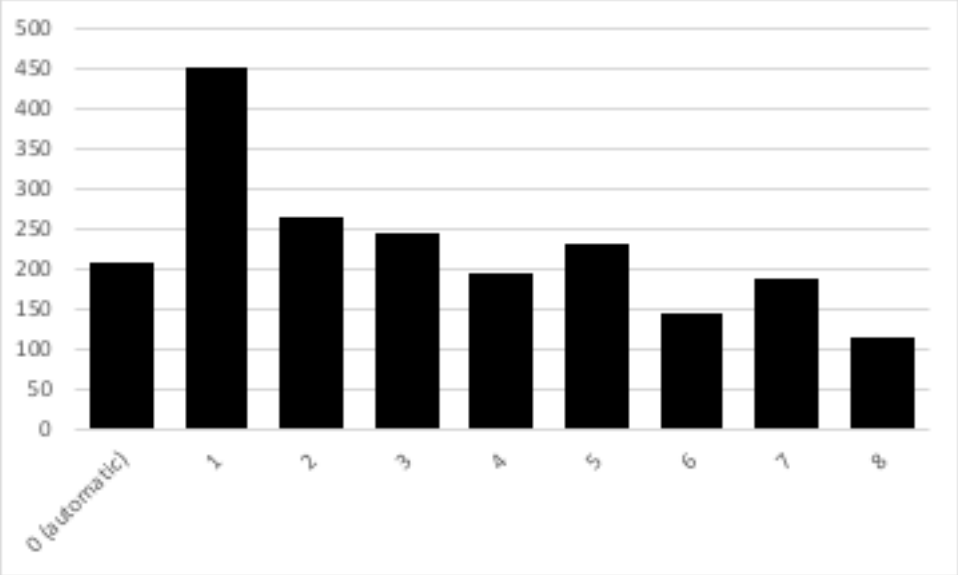


Figure 13, runtimes as a function of processor threads

As can be seen, the improvement from utilizing more threads is meaningful. It is likely that increasing the processor core count will improve runtimes. During computations, the computer was not memory limited. This leaves core count, clock speed and possibly memory speed as limitations in hardware to computation times. While stepping up from an odd number of threads to an even number yields clear benefits, the opposite seems to be true for stepping up from an even number to an odd number. In general, it is advisable to use every thread available in the computer. However, should the user of the model be inclined to restrict this number (e.g. to retain processing power for other tasks), specifying n-2 threads is smarter than using n-1. Note that this does not hold for systems with 4 threads. Furthermore, using the automatic setting is ill advised. Performance when using a manually selected setting is better, except when using very few threads.

4.1.4 Wear Out Shape Parameter d

The first parameter related to the actual component that was varied was d. It represents the shape parameter of the wear out rate. Also presented in the table is the theoretical optimal maintenance interval for each component. The exact specification and meaning of these variables can be found in chapter 7. The experiment (appendix XVI) results in the performance in case d is either half or twice the normal value. The performance is in table 4 and figure 14.

Table 4, performance as a function of the wear out shape parameter

d	t_{opt}	Median runtime	Number of possessions
0.008 0.008 0.01	132 108 79	1.2	3
0.016 0.016 0.02	66 54 40	111.7	6
0.032 0.032 0.04	33 27 20	3,600	14

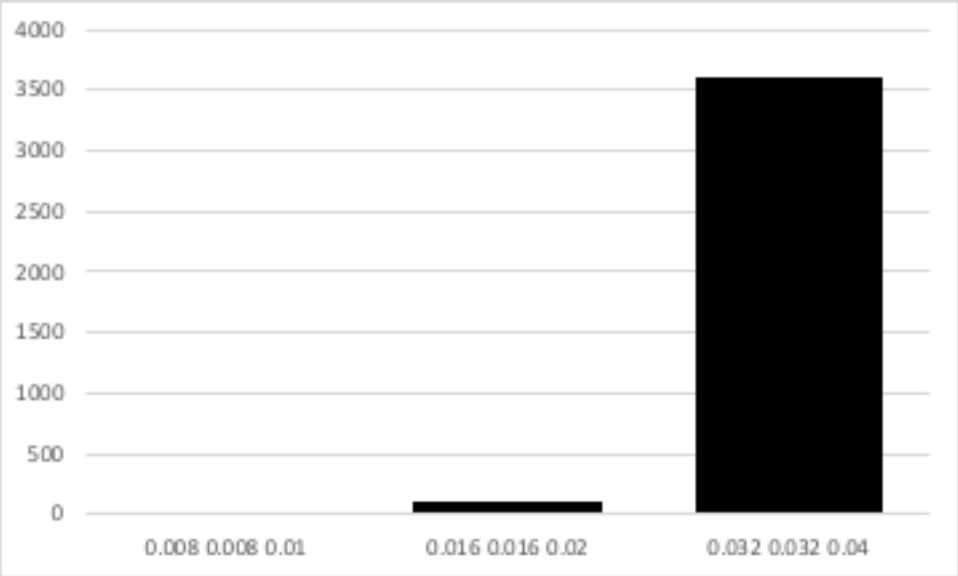


Figure 14, runtimes as a function of d

The results show that components that wear out faster increase the problem size and slow down the optimization significantly. This is simply due to having to plan more maintenance operations. It can clearly be seen in the solution graphs in appendix XXIV. Techniques to speed up the program are mentioned in paragraph 11.7, should it be required that such a schedule is solved quickly.

Cost of Possession

Next, the influence of the cost of possession on the runtime was investigated. The difficulty of determining the optimal solution may be increased as the cost of possession is larger. The algorithm may not be able to cut of sets of decision variables as easily if both the decision variables related to failure and those related to possession have a strong effect on the outcome. By varying the cost of possession from 0.25 to 25000 in exponential steps of roughly $10^{1/2}$, varying results are found. The exact experiment specification is in appendix XVII, and the resulting schedules in appendix XXII. The results of this experiment are in table 5 and figure 15.

Table 5, performance as a function of the cost of possession

CoP	runtime	number of possessions
0.25	2.8	11
0.8	3.1	11
2.5	6.9	9
8	19.6	8
25	28.3	7
80	112.4	6

250	1770.3	4
800	424.4	3
2500	643.8	3
8000	462.8	2
25000	148.6	2

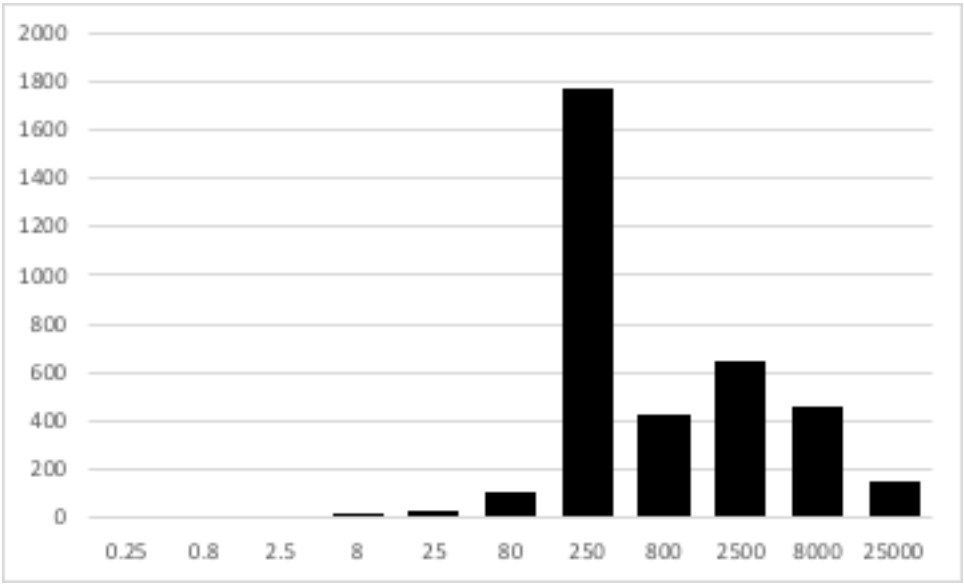


Figure 15, runtime as a function of cost of possession

As the cost of possession tends to zero, the problem simply involves minimizing the cost of maintenance and the cost of failure, and the decision variables related to cost of possession do not influence the cost of possession. However, for very large values of CoP, the problem simplifies into minimizing the number of possessions as the role of failure and maintenance becomes secondary. Any solution that requires more possessions than the minimum can be cut off immediately.

4.1.5 Varying Cost of Possession

Realistically, some points in time may be more suitable for performing maintenance than others. The reasons for this can be manifold, but in any case, the consequence is that a small adjustment in the planning of maintenance can have tangible benefits. To simulate such a problem, two parameter sets in which the cost of possession alternates between 50 and 150 is generated. In the first case, this happens for every other possession, in the second one the lower value is applied every fifth possession. The experiment loop is described in appendix XVIII. It is expected that the maintenance will be assigned to the cheaper possessions. In table 6 and figure 16, the effect on performance is displayed.

Table 6, performance as a function of alternating cost of possession

Cost of possession	Median runtime	Possessions in expensive week
always 80	113.3	-
once 150, once 50, etc.	37.6	0
four times 150, once 50, etc.	13.8	0

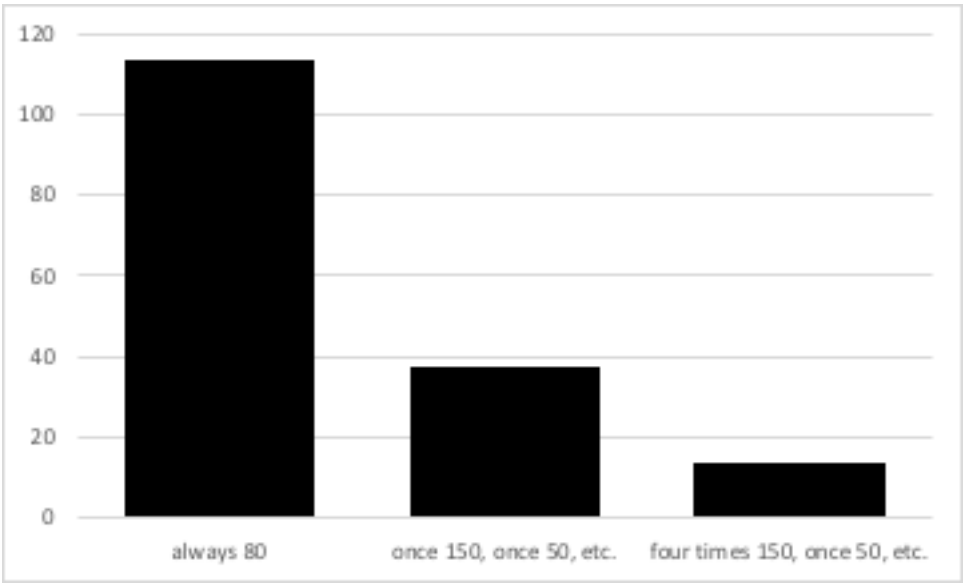


Figure 16, runtime as a function of cost of possession

As expected, the cheaper weeks were used to perform maintenance. While not dramatically different, the differences in schedule are visible in appendix XXIII. The problem solved quicker in the cases where the cheap possession was rare. As with 11.5, the algorithm will attempt to cut off certain decision variables. As soon as a feasible solution is found (through heuristics) that uses the decision variables associated with cheap weeks, the expensive weeks can be cut off. In practice, varying costs of possession over time are no obstacle to planning maintenance using the model. In fact, this experiment has shown that it will probably be solved quicker than a situation where all possession costs are equal.

4.1.6 Planning Horizon

The script was tested for runtime performance to give an indication of the problem size that can be solved in a reasonable amount of time. In this case, a situation with three components was optimized. The parameters were chosen such that the optimal maintenance intervals are between 18 and 36 weeks. The test specified that times up to 3,600 seconds were evaluated, and that three consecutive overruns would terminate the test. An exact specification is in appendix XIX. The results in table 7 and figure 17 were obtained.

Table 7, performance as a function of the planning horizon

planning horizon (weeks)	runtime (s)	time at which best solution was found (s)	remaining optimality gap (%)
200	111.1		0
205	443.0		0
210	386.4		0
215	2,522.7		0
220	3,600		0.16
225	3,600		0.23
230	3,177.8		0
235	3,600		0.24
240	3,600	1769	0.07
245	1,914.1	55	0

250	1,576.3	74	0
255	2,057.3	690	0
260	1,252.2	63	0
265	1,381.1	40	0
270	3,600	51	0.04
275	3,600	483	0.30
280	3,600	1,764	0.38

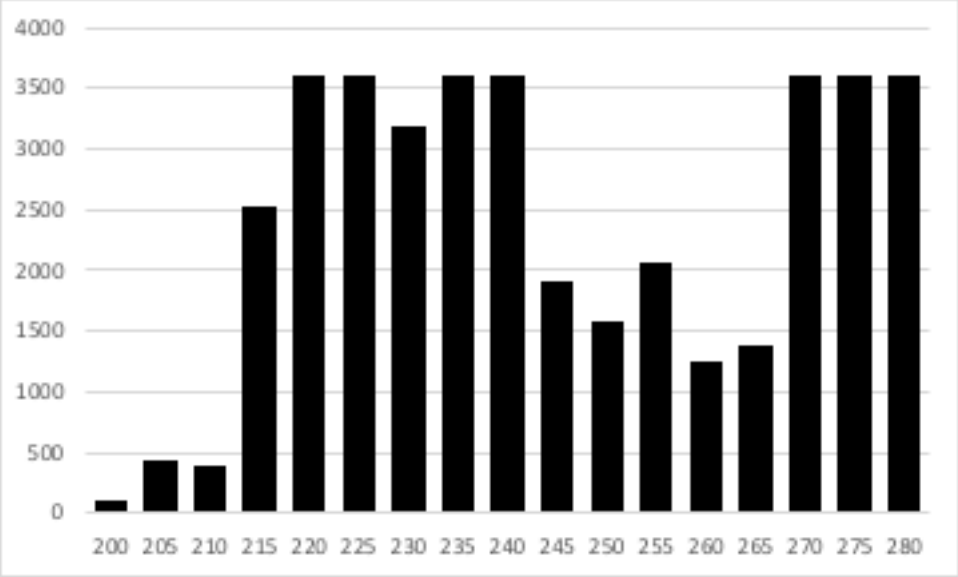


Figure 17, runtime as a function of planning horizon with an optimality gap of 0

In practice, optimization of up to 265 timesteps can be solved quickly. As a timestep in this case represents one week, a planning horizon of five years is realistic. It is noteworthy that in the last three cases convergence continued at a steady rate, and that given enough time, a solution can be expected. Given that for long term maintenance planning runtimes of days or even weeks are not problematic, and that much better computing hardware is available, the limitation on problem size with three components should not be an objection to practical implementation.

To explore the possibilities of the model, should abundant computing resources be available, an experiment was performed on very long planning horizons. It is specified in appendix XXI. The value of the planning horizon was increased in steps of 100 until there was no solution found after one hour of solution time. The results are in table 8 and figure 18.

Table 8, performance over long planning horizons

planning horizon	runtime	time at which first solution was found	remaining optimality gap (%)
450	8.7	8	2.65
500	251.6	101	7.87
550	215.9	111	9.06
600	1,615.2	1268	8.69
650	257.0	183	9.92
700	427.3	205	9.57

750	876.4	357	9.70
800	1,599.5	410	9.44
850	70.6	70	2.74
900	730.2	488	2.40
950	2,538.5	577	7.89
1000	3,273.5	1120	9.86
1050	2,113.3	558	9.73
1100	3,232.1	916	8.91
1150	3,600	1277	13.06

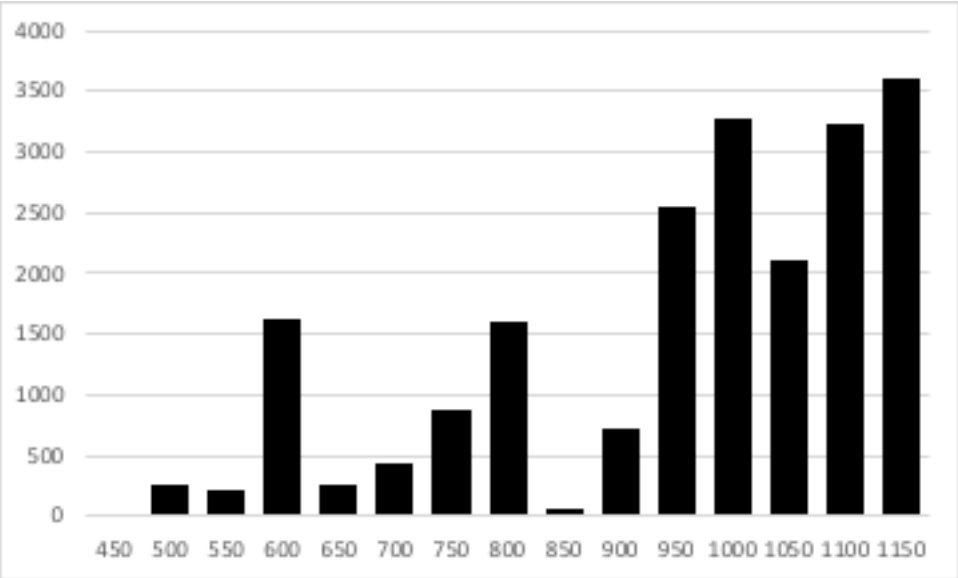


Figure 18, runtime as a function of planning horizon with an optimality gap of 10%

Should the problem still be too large to solve, rescaling the problem will decrease problem complexity and runtimes significantly. For example, a step size of one week may be replaced by a step size of two weeks, effectively halving the total number of time steps. Of course, this goes at the expense of accuracy. Another strategy for improving the time needed to solve the problem is to apply the model for individual interval planning to the first n weeks, and resorting to fixed intervals thereafter, or to apply the model for individual interval planning consecutively. At the end of the planning horizon of the model for individual interval planning, all components should be in a similar, degraded state. This effectively breaks the total problem down into smaller subproblems that will be faster to solve than the problem as a whole. However, the schedule that is produced will probably not be globally optimal.

4.1.7 Number of Components

All previous trials were based on a case of three component categories. However, if more maintenance categories should be included in the optimization, the solution time increases. Based on the initial scenario, a scenario with five components was created, where the additional components have characteristics that lay outside the bounds of the characteristics of the existing components. It can be found in appendix IX through XI. These parameter sets were experimented with using the test loop in appendix XX.

Table 9, performance as function of number of components

Number of components	Runtime	Remaining optimality gap (%)
1	0.5	0
2	1.1	0
3	110.6	0
4	207.9	0
5	3,600	0.12

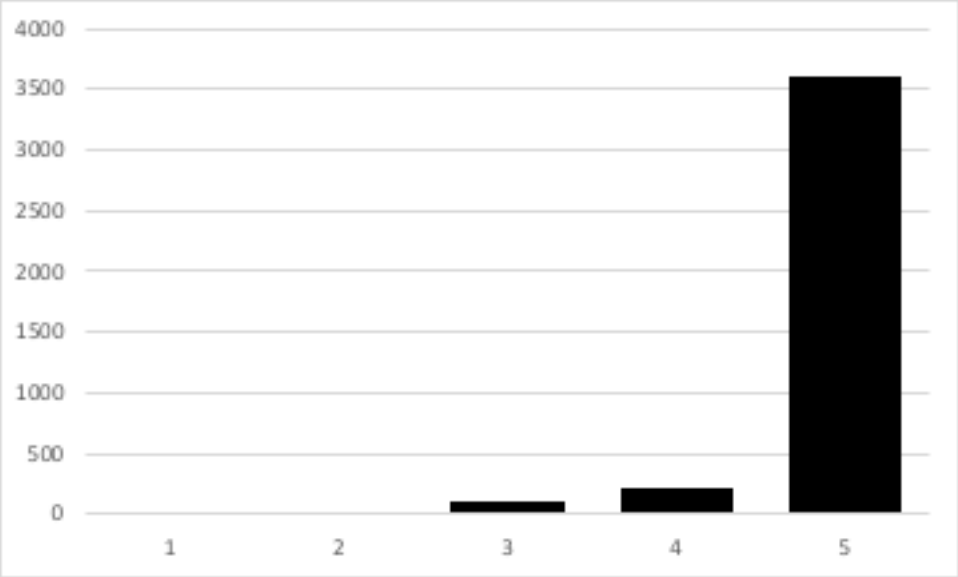


Figure 19, runtime as a function of number of components

As can be derived from table 9 and figure 19, the problem is substantially increased in complexity as the number of components is increased. The usability of the model may decrease as the problem is complicated by a larger number of components.

4.1.8 Comparison with Initial Implementation of the Model

An initial version of the model, where the decision variables and constraints are defined as straight forward as possible, was used for establishing the effectiveness of the improvement. Compared to the improved version of the model, the runtimes are orders of magnitude higher. To be precise, a model where none of these measures are applied, but which solves the same problem and results in the same objective value, at 3,600 seconds, the model had not yet converged to optimality. The remaining optimality gap was 0.24%. As the improved model had a runtime of 111 seconds, the initial model is more than 32 times slower.

In addition, the improved model without the unicity constraints (fourth bullet in list) was evaluated. Even though the number of nonzero elements in the linear constraint matrix is smaller, having multiple solutions that have to be evaluated increases the runtime markedly. In fact, without these constraints, the model performs worse than the initial model. All results are in table 10 and figure 20.

Table 10, performance of optimized versus unoptimized models

	Runtime	Remaining optimality gap (%)	Rows in linear constraint matrix	Columns in linear constraint matrix	Nonzeros in linear constraint matrix
Improved	112.3	0	247	3560	12992
Initial	3,600	0.24	239	3869	12359
Improved, but without unicity constraints	3,600	0.53	247	3560	10746

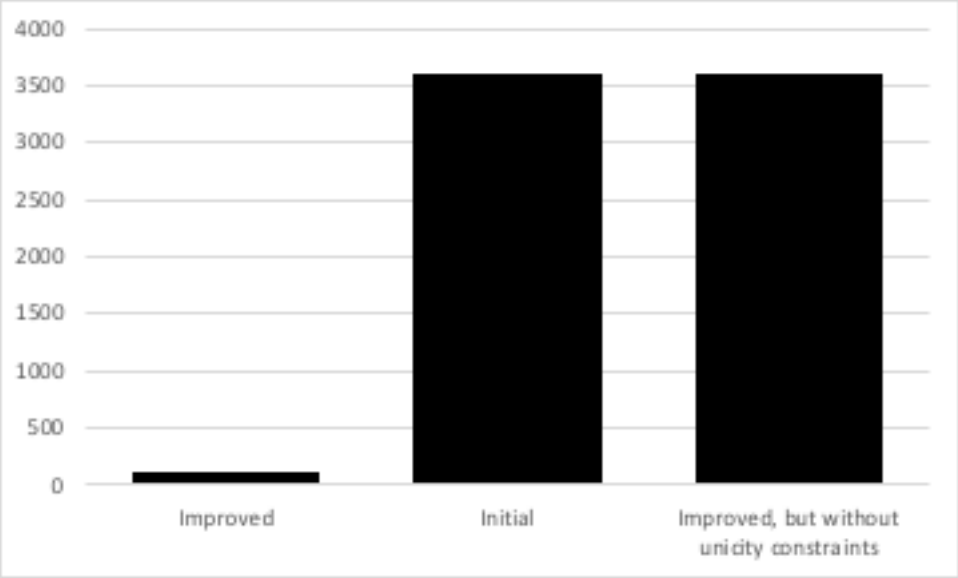


Figure 20, runtime of different model versions

4.1.9 Conclusion

The model can be applied to realistic problem sizes and specifications without taking problematically long computation times. By changing specific settings, the performance of the model was optimized. Certain values of the parameters affect the performance of the model, without changing the model structure. This means it is hard to draw general conclusions on the model performance, that can be applied to input data with different characteristics.

4.2 Test Case

A realistic case for two different components was formulated. The components chosen are switches and insulated joints. These components are both load-carrying parts of the rail, and fairly maintenance intensive. It should be noted that it may be regarded as a proof of concept rather than a practically applicable solution of the maintenance schedules. For this, more accurate data as well as differentiation to component subtypes and failure types would be advisable.

A realistic case for two different components was formulated. The components chosen are switches and insulated joints. These components are both load-carrying parts of the rail, and fairly maintenance intensive. It should be noted that it may be regarded as a proof of

concept rather than a practically applicable solution of the maintenance schedules. For this, more accurate data as well as differentiation to component subtypes and failure types would be advisable.

Real data were used to get to parameters that reflect realistic failure behavior. Tables that contain all preventive and corrective maintenance operations were available for the two component types. The information in the table includes the object, whether the maintenance was corrective or preventive, the specification code of the maintenance, a short description of the work, and the planned and actual execution dates. The procedure to obtain the parameters is in appendix XXXIII.

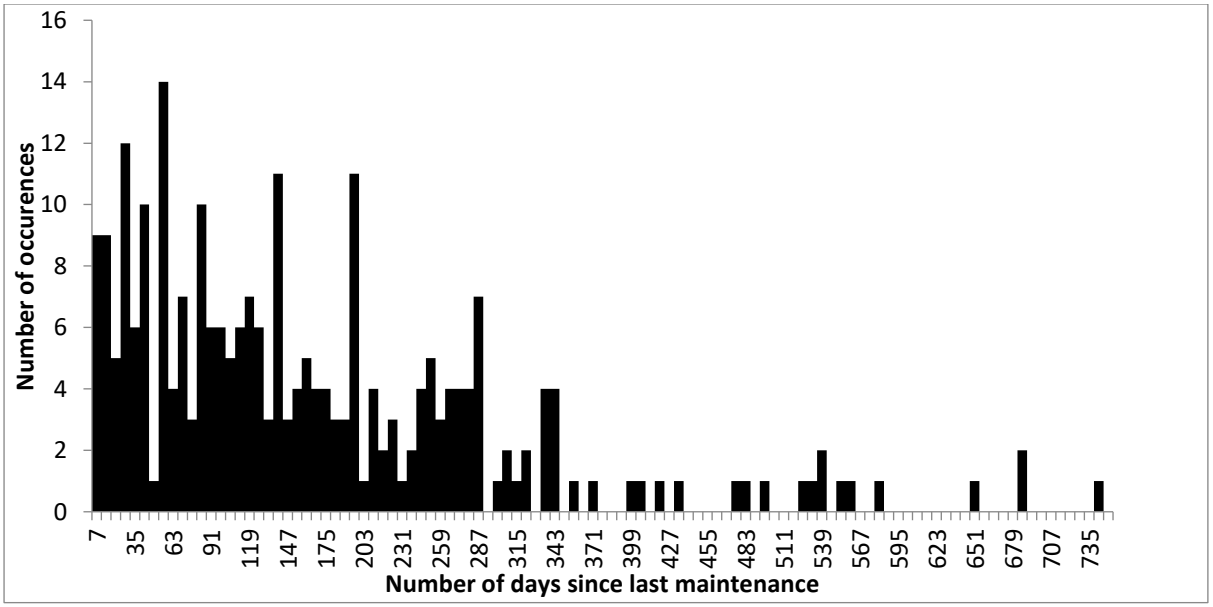


Figure 21, histogram of failure of switches as a function of time since last maintenance with bin size 7 days

For this histogram, the times of corrective maintenance since preventive maintenance to switches are binned in weeks. As can be seen in figure 21, the number of occurrences per value is very low for higher values. The failure times are grouped into bins of a more practical size (50 days) to smooth out the data. The resulting histogram is in figure 22.

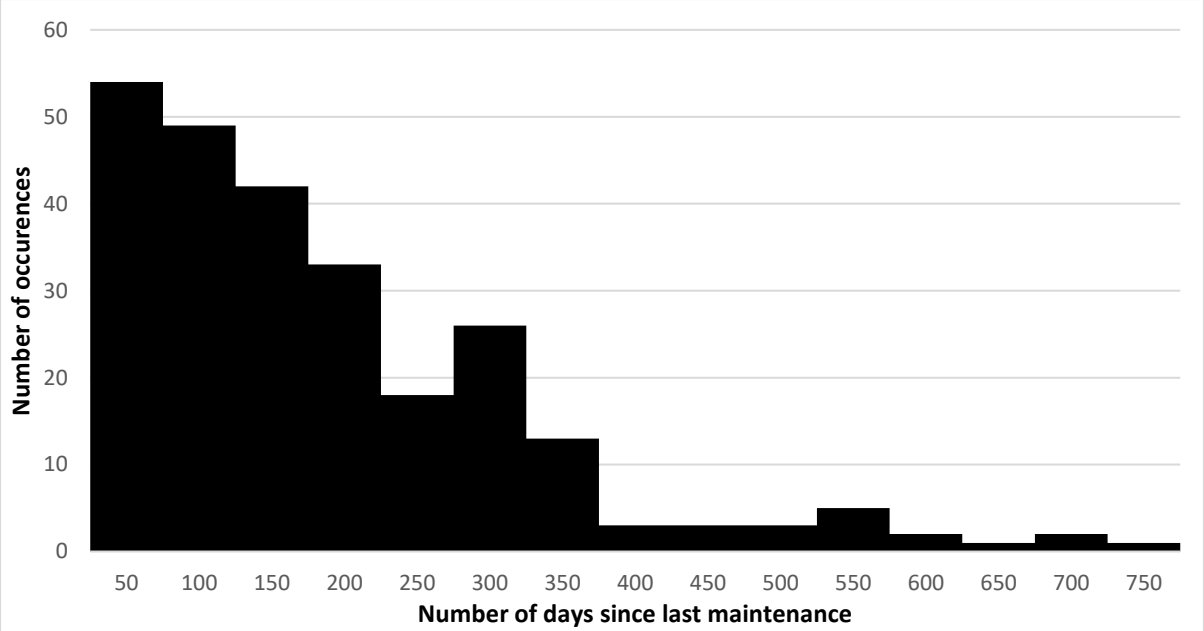


Figure 22, histogram of failure of switches as a function of time since last maintenance with bin size 50 days

After adjustment for already maintained components (step 7, Appendix XXXIII), the histogram becomes heavier on the right side. A bathtub shape that is typical of degrading components emerges. This can be seen in figure 22.

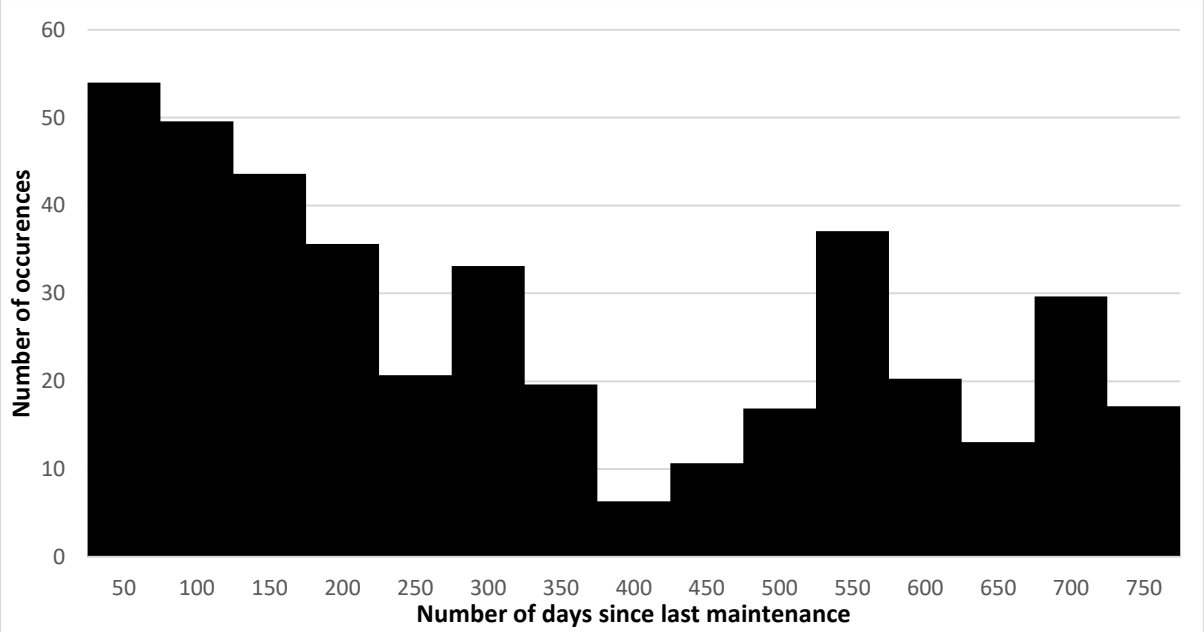


Figure 23, adjusted histogram of failure of switches as a function of time since last maintenance with bin size 50 days

The adjusted failure count for switches is estimated using both the Gompertz-Makeham and the Weibull distribution. A least squares of errors optimization yields the parameters of the distribution and the goodness of fit. The Gompertz-Makeham distribution provided more explained variance than Weibull distribution; 68.8 percent and 63.4 percent respectively. The adjusted observed failure count and predicted failure count can be seen in black and striped respectively in figure 23.

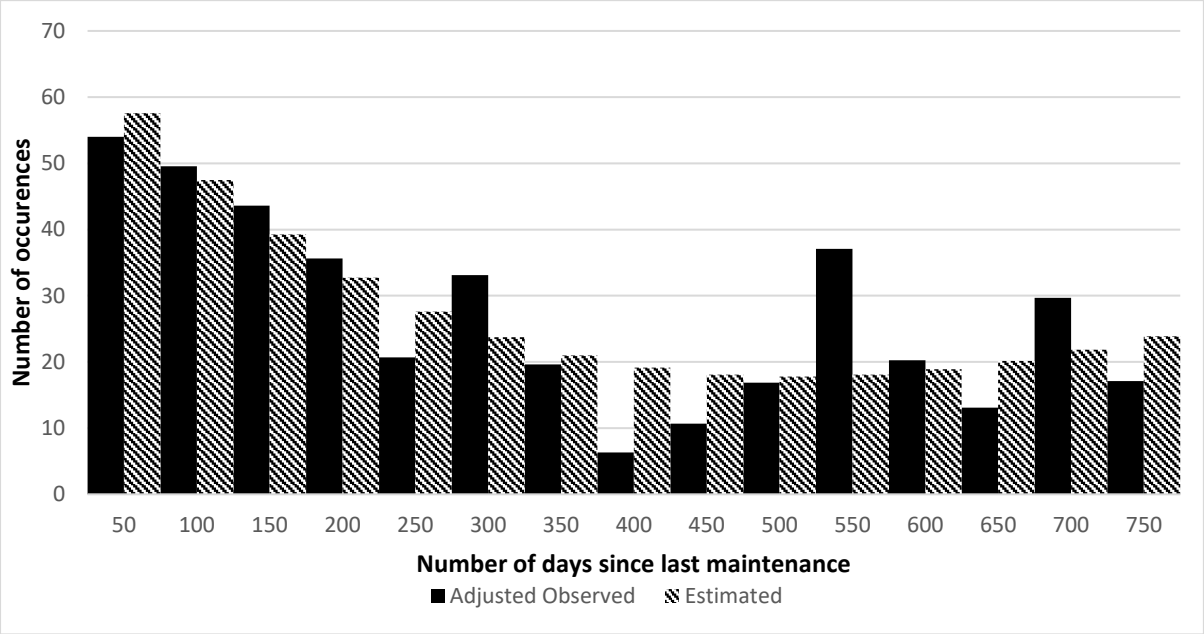


Figure 24, observed and predicted failure counts of switches

The same procedure was followed with the data on insulated joints. The fitting of the Gompertz-Makeham function to the insulated joint data resulted in an explained variance of 36.0 percent, while with the Weibull function 35.2 percent of variance can be explained. The resulting histogram is in figure 24.

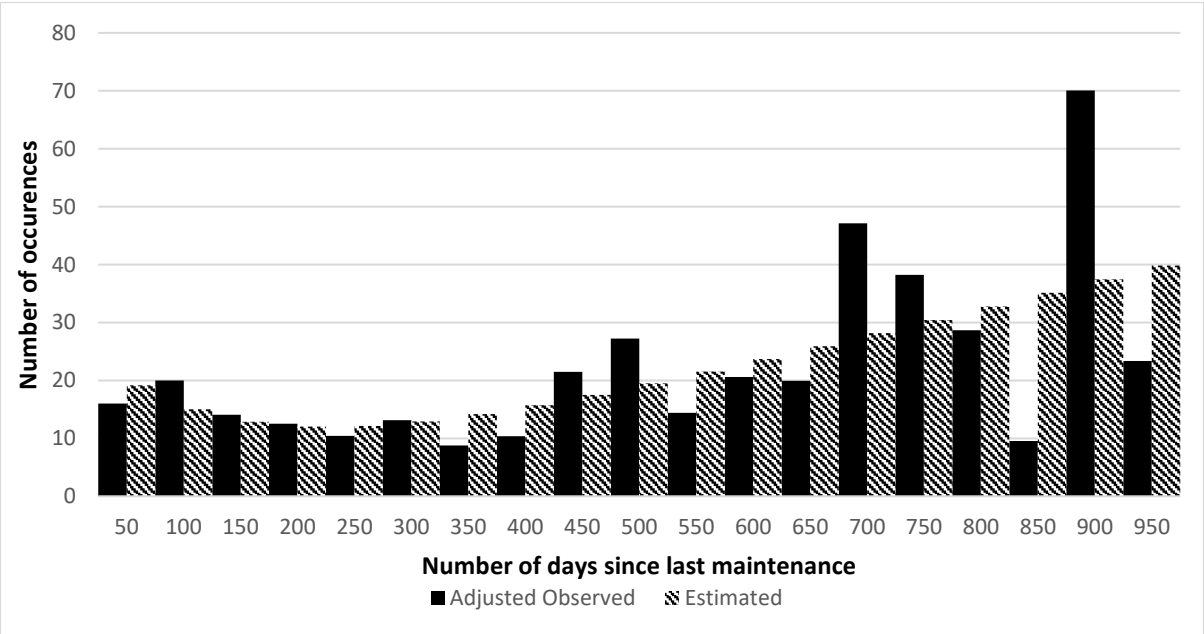


Figure 25, observed and predicted failure counts of insulated joints

The obtained parameters as defined in section 3.3 are presented in table 13. These parameters are ready to be used in the optimization model.

Table 11, estimated parameters on switch and insulated joint data

object type	a	b	c	d	f

switch	-6.603	-0.02028	1524	0.0006557	-0.1511
insulated joint	-0.05677	-0.05067	120.1	0.0005460	-0.009412

These parameters were introduced into the optimization model. Additionally, realistic values (confidential) were used for cost of maintenance, failure and possession. The planning horizon was set to 261 weeks, or 5 years, and the ratio of switches to insulated joints was 1 to 10. This is a rough estimation, based on the fact that for every switch at least 4 insulated joints are needed, and that each section of straight track is also separated by two insulated joints to the next. This ratio is in the same order of magnitude as in a station area. As the exact number of switches and insulated joints does not change the optimization a long as their ratio is the same, this assumption suffices to produce a representative test case. The time since the last maintenance operation was assumed to be 30 weeks for switches and 20 for insulated joints. If a maintenance interval of 52 weeks is applied consistently, the schedule in figure 25 is produced, where 1 denotes the switches, 2 denotes the insulated joints and P denotes the possessions.

Using the statistical analysis only, optimized intervals can already be determined. This is done by solving equation (3.39) for each component individually. The results show that the maintenance intervals for the components without taking the cost of possession into account, is 126 and 61 weeks for switches and insulated joints respectively. Based on this figure, a maintenance schedule that continuously applies this interval can be used. This schedule is pictured in figure 26.

The total optimization model also minimizes the cost of possession. This can be seen when After 6.91 seconds of runtime, an optimality gap of zero was achieved. The resulting schedule has one maintenance operation for the switches and three for the insulated joints. The completely optimized schedule is in figure 27.

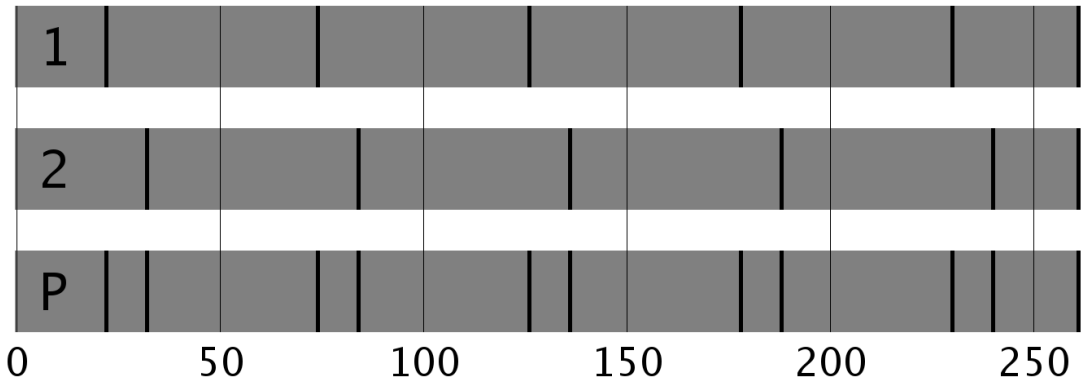


Figure 26, conventional schedule

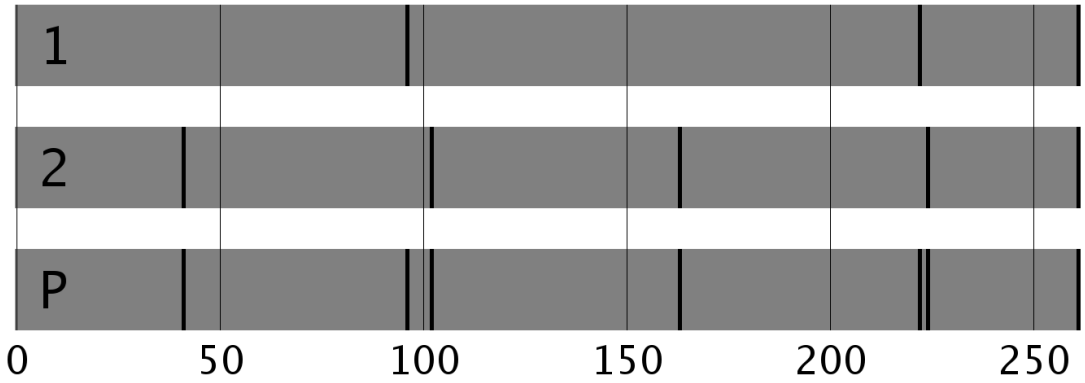


Figure 27, schedule when intervals are optimized individually

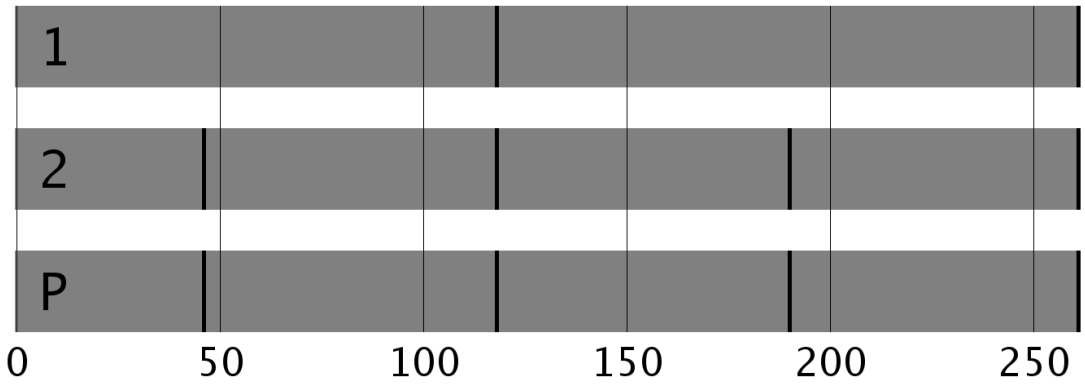


Figure 28, optimized schedule

Table 12, comparison of different schedules applied to the test case

	Conventional intervals	Optimized intervals	Optimized schedule
Value of the objective function	716.4	703.4	699.9
Number of possessions	8	6	3
Number of maintenance operations to switches	5	2	1
Number of maintenance operations to insulated joints	5	4	3
Objective function compared to conventional intervals	100 %	98.2 %	97.7 %
Objective function compared to optimized intervals	101.8 %	100 %	99.5 %
Objective function compared to optimized schedule	102.4 %	100.5 %	100 %

Comparing the three schedules in table 12, the difference in objective function is small. Even so, small improvements such as these would amount to sizeable profits over the longer term. As the difference in number of maintenance operations and number of possessions is substantial, it seems that the cost of failure is a large factor in this case. Either that is true, or the parameters are based on an overestimation of failures.

Considering that better analysis methods and the use of better data could improve the performance of the model, there is potential for delivering further improvements. Furthermore, differentiated maintenance, as is already applied to switches, can also improve the usefulness of the model. This could be included by splitting components to which it is applied into multiple separate groups, for example based on frequency of use.

5 Conclusion and Recommendations

5.1 Conclusion

Based on the research that was performed, the research questions can be fully answered.

5.1.1 Characterization

- What are the challenges in the current way of determining a maintenance strategy for railway assets?

The current practice is to maintain all components that need maintenance according to a fixed schedule, whereby the intervals are usually multiples or fractions of a year. Based on decades of experience, the resulting performance is acceptable. However, this is by no means optimal. Statistical analysis of failures in railway components will give a better indication of what maintenance intervals are cost efficient.

Furthermore, there is no incentive for the contractor to have the assets perform well beyond the end of the contract. This means maintenance that is desirable to the owner of the asset may not be performed by the contractor, and left to the next contractor. A model for failure processes can be used by for example the infrastructure manager to assess the impact of a maintenance strategy in the long run.

- How can the failure probabilities of railway components over time be estimated?

The failure probability of a railway component is not constant. In fact, if it were, maintenance would not have any benefit. Whether or not a component will actually fail depends on many different factors. Failure rates of mechanical components are typically increasing over time. This is called wear-out. Two ways of modeling increasing failure rates is by a Weibull or a Gompertz-Makeham distribution. In addition, such components may have a decreasing failure rate in the initial stages after maintenance. This can be modelled by having a separate function for this break-in phase, and adding it to the existing function for wear-out. In a test case, the Gompertz-Makeham distribution was better able to explain the variance in the moment of failure of switches.

5.1.2 Optimization

- How can the costs resulting from failure be estimated in case the maintenance schedule is known?

As performing maintenance and applying possessions is not stochastic, it does not need to be estimated. However, the maintenance schedule affects the reliability of the components, which is stochastic. Using the estimated failure rate function, the expected number of failures can be computed. Assuming the cost of failures is constant, the expected cost due to failures can be computed by multiplying the expected number of failures with the cost of each failure.

- How can the optimal schedule for performing maintenance to a certain track section and its associated cost be determined?

What constitutes an optimal schedule depends on three main factors; maintenance, possessions and expected number of failures. All of these can be expressed as a linear function of the maintenance schedule. Therefore, the problem of optimizing the

maintenance schedule for railway components can be solved by a binary optimization model. A model is presented that produces a maintenance schedule that is optimal regarding cost of maintenance, cost of possession and cost of failure.

By applying the model, a feasible maintenance schedule is created. The outcome of the optimization is a solution for which the objective value is optimal. This objective value represents an estimation of the aforementioned costs. Hence, if this objective value is minimized, the expected cost is minimized. It can be used by contractors to determine an appropriate bid for a maintenance contract for the area that is optimized. It contains the number of times as well as the times each component is maintained. This can be used to allocate sufficient resources including workers, equipment and materials, and to acquire these if necessary.

5.1.3 Evaluation

- How does an optimized maintenance schedule compare to a schedule that is up to the current state of the art?

A case study of two component types revealed that the optimal maintenance interval is in both cases higher than the current maintenance interval. Percentage wise, the differences are small, but given the large expenditure in an absolute sense, the gains may still be significant. Furthermore, better data analysis may contribute to further improvement over the current results.

- How accurately and how far into the future can the cost of performing track maintenance be predicted?

The model is not theoretically limited to a certain planning horizon. Depending on the model parameters, it is possible to plan many years ahead. However, in practice it may not make sense to do this. As the planning horizon increases, so does the uncertainty about the quality of the prediction on failure. Besides, the usefulness of planning further ahead than the contract extends has no added use in practice.

5.1.4 Valorization

- How can the model be used by railway maintenance engineers?

Given that there is enough data available, maintenance engineers can make a long-term planning for managing assets. The model is able to produce a solution that is optimal under the assumptions as defined. While these assumptions are not perfect, they resemble reality closely enough for the model to produce a good starting point for more accurate planning. In this way, it is a useful addition to the skills and tools that are already available.

- Will optimization of maintenance contribute to reduced unplanned maintenance?

For each case individually, the expected number of failures of the optimal solution and the current solution can be compared. There is a tradeoff between investing in maintenance and suffering failures. Consequently, an optimal maintenance schedule could work both ways. In case there is too much maintenance in the current situation, there may be less overall cost despite increasing the number of failures if the frequency of maintenance is reduced. Furthermore, failure is a stochastic process. Care should be taken to avoid drawing conclusions from results of small samples.

- How can the contract between the contractor and the infrastructure manager be specified, such that all the information necessary for optimizing maintenance is exchanged?

To be able to apply the model effectively, data on the assets is needed, preferably of high quality. In case the contractor is new to the area, this data will have to be supplied by either the infrastructure manager or the previous contractor. What is needed is a standard procedure for logging maintenance and failures, and making the resulting data available to both the IM and the contractor.

5.2 Recommendations for Further Research

5.2.1 Quadratic Reformulation

An implementation using quadratic constraints is possible. The non-linear part of the objective function would have to be locally approximated by a quadratic function by matching the second derivative in a specified point. The constraint that defines the relationship between the intervals and the possessions would be:

$$\left(x_{i,j} - k\right)^2 + y_k > 0.5 \; \forall \; i \in components, j \in intervals \; of \; i, k \in number \; of \; weeks \tag{5.1}$$

where:
$x_{i,j}$ is the moment of maintenance operation j to component group i
y_k is whether there is a possession at time k

While the number of decision variables is sharply reduced, the number of constraints increases significantly. Performance may be better or worse. A implementation of this formulation, and a comparison with the model as proposed should answer this question.

5.2.2 Practical Limitations

Practical limitations, such as limits on the numbers and types of maintenance operations carried out concurrently, are not included in the model as it is now. Formulating these as linear or quadratic constraints and adding them to the existing model could enhance the practical value of the model.

5.2.3 Application in Other Domains

The formulation of the model was based on the requirements from the railway industry, but very similar problems will exist in other industries. The essence of the optimization, balancing cost due to failure, cost due to maintenance and cost due to inoperability, should be familiar to any industry that has systems with multiple stochastically failing components. The general nature of the model formulation allows for quick adaption to optimizing other systems with these characteristics.

5.2.4 Use in Cost-Benefit Analysis

A cost-benefit analysis contains estimations of cost over the lifetime of a system. The model has the potential to provide a standardized approach for estimating the failure and maintenance cost of a railway and enable a quantitative comparison.

5.2.5 Data Collection

As with any model that relies on observed data, the quality of that data largely determines the quality of the model result. Unfortunately, the data that is required for the model is not yet available as accurately as possible. Having an accurate estimation of the failure rate function of all the components that are part of the optimization is a requirement for having a schedule that is not only optimal according to the model, but also performs well in practice.

6 Reflection

Working from both TU Delft and Arcadis offered me the opportunity to approach the problem at hand from both a scientific and a more practical point of view. It has definitely been useful to attempt to bring these two drastically different ways of working together, as the resulting work has both a solid scientific foundation as well as a possibility for application in practice.

What anybody who starts working in railway engineering will notice is that the business is rather conservative. Even though more attention is being given to cutting off inefficient ways of working, breaking down old attitudes and procedures takes time. Solid proof and testing are required to convince and reassure actors within the industry of the added value of innovative approaches to asset management.

Operations research offers many new techniques that will improve the way railway assets are managed in the future. Big data and optimization can enable contractors to more accurately predict failures, both in the short term by monitoring developing problems, and in the long term by applying statistical models. It is in the latter category that this research comes in.

A less common technique within linear optimization was required to get to the end product. While it required a lot of trial and error, the result is a model that can be used to help the railway industry and possibly other industry where deteriorating systems are maintained. The model as it stands today has growth potential, as I am sure the possibilities from improving the model are not yet fully exhausted.

What has been difficult is getting access to data that is accurate enough to make a good case study on. Unfortunately, the data that was readily available was only on switches. As the strength of the model is to design maintenance schedules for multiple components in parallel, application to a real-life case study has proven to be difficult.

Which brings us to what is still a weakness of this approach. Good data is needed to make reliable and accurate predictions and plans over the long term, and this is not always available. What's more, the required computing power for five components is already rather high. A more efficient implementation of the model would probably be needed in cases where the maintenance to many more components is planned simultaneously.

All in all, the project and the resulting product have yet to prove itself, but are hopefully provide a positive influence to both maintenance engineering and my personal and professional development.

I References

Achterberg, T., Bixby, R. E., Gu, Z., Rothberg, E., & Weninger, D. (2016). Presolve reductions in mixed integer programming. *ZIB Report*, 16-44. https://opus4.kobv.de/opus4-zib/files/6037/Presolve.pdf

Amtrak. (n.d.). *A Guide to Key Railroad Terminology for the Northeast Corridor.* https://www.amtrak.com/content/dam/projects/dotcom/english/public/documents/corporate/NYPRenewal/Guide-to-Key-Railroad-Terminology-Northeast-Corridor-v2.pdf

Belov, G., Stuckey, P. J., Tack, G., & Wallace, M. (2016, September). Improved linearization of constraint programming models. In *International Conference on Principles and Practice of Constraint Programming* (pp. 49-65). Springer, Cham. https://doi.org/10.1007/978-3-319-44953-1_4

Birolini, A. (2007). *Reliability engineering* (Vol. 5). Heidelberg: Springer. https://doi.org/10.1007/978-3-662-54209-5

Budai, G., Huisman, D., & Dekker, R. (2006). Scheduling preventive railway maintenance activities. *Journal of the Operational Research Society, 2006*(57), 1035–1044. https://doi.org/10.1057/palgrave.jors.2602085

Budai, G., Dekker, R., & Kaymak, U. (2009). Genetic and memetic algorithms for scheduling railway maintenance activities. *Econometric Institute Report, 2009*(30), 1-23. http://hdl.handle.net/1765/17513

Chen, Y., Wang, Z., Qiu, J., Zheng, B., & Huang, H. Z. (2011, June). Adaptive bathtub hazard rate curve modelling via transformed radial basis functions. In *Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE), 2011 International Conference on* (pp. 110-114). IEEE. https://doi.org/10.1109/ICQR2MSE.2011.5976579

Consilvio, A., Di Febbraro, A., & Sacco, N. (2016). Stochastic Scheduling Approach for Predictive Risk-Based Railway Maintenance. *2016 IEEE International Conference on Intelligent Rail Transportation (ICIRT)*, Birmingham, 197-203. https://doi.org/10.1109/ICIRT.2016.7588732

Dekker, R. (1996). Applications of maintenance optimization models: a review and analysis. *Reliability Engineering and System Safety, 51*(3), 229-240. https://doi.org/10.1016/0951-8320(95)00076-3

Dekker, R., Smit, A., & Losekoot, J. (1992). Combining maintenance activities in an operational planning phase: a set-partitioning approach. *IMA Journal of Mathematics Applied in Business & Industry, 3*(4), 315-331. https://doi.org/10.1093/imaman/3.4.315

Famurewa, S. M. (2015). Maintenance analysis and modelling for enhanced railway infrastructure capacity (doctoral thesis). http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A990112

Ferreira, L., & Murray, M. H. (1997). Modelling rail track deterioration and maintenance: current practices and future needs. *Transport Reviews*, *17*(3), 207-221. https://doi.org/10.1080/01441649708716982

Finkelstein, M. (2008). *Failure rate modelling for reliability and risk*. Springer Science & Business Media. https://doi.org/10.1007978-1-84800-986-8

Forsgren, M., Aronsson, M., & Gestrelius, S. (2013). Maintaining tracks and traffic flow at the same time. *Journal of Rail Transport Planning & Management*, *3*(3), 111-123. https://doi.org/10.1016/j.jrtpm.2013.11.001

Goel, A., & Meisel, F. (2013). Workforce routing and scheduling for electricity network maintenance with downtime minimization. *European Journal of Operational Research*, *231*(1), 210-228. https://doi.org/10.1016/j.ejor.2013.05.021

Gurobi. (n.d.). Mixed-Integer Programming (MIP) - A Primer on the Basics. Retrieved from http://www.gurobi.com/resources/getting-started/mip-basics

Hansen, I. A. (2006). State-of-the-art Of Railway Operations Research. *WIT Transactions on The Built Environment*, *88*. https://doi.org/10.2495/CR060561

Higgins, A. (1998). Scheduling of railway track maintenance activities and crews. *Journal of the Operational Research Society*, *49*(10), 1026-1033. https://doi.org/10.1057/palgrave.jors.2600612

Higgins, A., Ferreira, L., & Lake, M. (1999). Scheduling rail track maintenance to minimise overall delays. *The 14th International Symposium on Transportation and Traffic Theory*, Jerusalem, 1999. https://eprints.qut.edu.au/2531/

Hommes, T. (2016). Optimisation model for scheduling preventive tamping maintenance: A study into the effect of using the alignment in combination with the level to schedule preventive tamping maintenance (master thesis, Delft University of Technology). http://resolver.tudelft.nl/uuid:53b1444c-5e19-44ee-92a1-956976ba0faa

Høyland, A., & Rausand, M. (1994). *System reliability theory: models and statistical methods*. Wiley.

Huisman, D., Kroon, L. G., Lentink, R. M., & Vromans, M. J. (2005). Operations research in passenger railway transportation. *Statistica Neerlandica*, *59*(4), 467-497. https://doi.org/10.1111/j.1467-9574.2005.00303.x

Kobbacy, K. A. H., & Murthy, D. P. (Eds.). (2008). Complex system maintenance handbook. Springer Science & Business Media. https://doi.org/10.1007/978-1-84800-011-7

Li, R., & Roberti, R. (2017). Optimal Scheduling of Railway Track Possessions in Large-Scale Projects with Multiple Construction Works. *Journal of Construction Engineering and Management*, *143*(6), 04017007. https://doi.org/10.1061/(ASCE)CO.1943-7862.0001289

Lidén, T. (2014). *Survey of railway maintenance activities from a planning perspective and literature review concerning the use of mathematical algorithms for solving such planning and scheduling problems*. http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A754942

Lidén, T., & Joborn, M. (2016a). Dimensioning windows for railway infrastructure maintenance: Cost efficiency versus traffic impact. *Journal of Rail Transport Planning & Management*, *6*(1), 32-47. https://doi.org/10.1016/j.jrtpm.2016.03.002

Lidén, T., & Joborn, M. (2017). An optimization model for integrated planning of railway traffic and network maintenance. *Transportation Research Part C: Emerging Technologies*, *74*, 327-347. https://doi.org/10.1016/j.trc.2016.11.016

Luan, X., Miao, J., Meng, L., Corman, F., & Lodewijks, G. (2017). Integrated optimization on train scheduling and preventive maintenance time slots planning. *Transportation Research Part C: Emerging Technologies*, *80*, 329-359. https://doi.org/10.1016/j.trc.2017.04.010

Lyngby, N., Hokstad, P., & Vatn, J. (2008). RAMS management of railway tracks. In *Handbook of performability engineering* (pp. 1123-1145). Springer, London. https://doi.org/10.1007/978-1-84800-131-2_68

Núñez, A., Hendriks, J., Li, Z., De Schutter, B., & Dollevoet, R. (2014, October). Facilitating maintenance decisions on the Dutch railways using big data: The ABA case study. In *Big Data (Big Data), 2014 IEEE International Conference on* (pp. 48-53). IEEE.

Nyström, B. (2008). *Aspects of improving punctuality: from data to decision in railway maintenance* (Doctoral dissertation, Luleå tekniska universitet). http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A999787

Oh, S. M., Lee, J., Park, B., Lee, H., & Hong, S. (2006). A study on a mathematical model of the track maintenance scheduling problem. *Computers in Railways X*, *88*, 85-96. https://doi.org/10.2495/CR060091

Ouyang, Y. (2013). Improving Railroad Track Maintenance Scheduling with Operations Research Techniques. *TR News*, (286), 55-56. http://www.trb.org/Publications/Blurbs/169265.aspx

Oyama, T., & Miwa, M. (2006). Mathematical modeling analyses for obtaining an optimal railway track maintenance schedule. *Japan Journal of Industrial and applied mathematics*, *23*(2), 207. https://doi.org/10.1007/BF03167551

Peng, F., Kang, S., Li, X., Ouyang, Y., Somani, K., & Acharya, D. (2011). A heuristic approach to the railroad track maintenance scheduling problem. *Computer-Aided Civil and Infrastructure Engineering*, *26*(2), 129-145. https://doi.org/10.1111/j.1467-8667.2010.00670.x

Peng, F., & Ouyang, Y. (2014). Optimal clustering of railroad track maintenance jobs. *Computer-Aided Civil and Infrastructure Engineering*, *29*(4), 235-247. https://doi.org/10.1111/mice.12036

Podofillini, L., Zio, E., & Vatn, J. (2006). Risk-informed optimisation of railway tracks inspection and maintenance procedures. *Reliability Engineering & System Safety*, *91*(1), 20-35. https://doi.org/10.1016/j.ress.2004.11.009

Pour, S. M., & Benlic, U. (2016, December). Clustering of Maintenance Tasks for the Danish Railway System. In *International Conference on Intelligent Systems Design and Applications,* 791-799. https://doi.org/10.1007/978-3-319-53480-0_78

Quiroga, L. M., & Schnieder, E. (2010). A heuristic approach to railway track maintenance scheduling. *WIT Transactions on The Built Environment*, *114*, 687-699. https://doi.org/10.2495/CR100631

Quiroga, L. M., & Schnieder, E. (2012). Monte Carlo simulation of railway track geometry deterioration and restoration. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, *226*(3), 274-282. https://doi.org/10.1177/1748006X11418422

Rausand, M., & Arnljot, H. Ã. (2004). *System reliability theory: models, statistical methods, and applications* (Vol. 396). John Wiley & Sons. https://doi.org/10.1002/9780470316900

Refalo, P. (2000, September). Linear formulation of constraint programming models and hybrid solvers. In *International Conference on Principles and Practice of Constraint Programming* (pp. 369-383). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45349-0_27

Rodríguez, G. (2007). *Lecture Notes on Generalized Linear Models*. http://data.princeton.edu/wws509/notes/

Tamparas, D. (2016). *Railway Track Geometry: A framework to optimally evaluate track condition and assess maintenance needs based on vehicle response.* (master thesis, Delft University of Technology). http://resolver.tudelft.nl/uuid:5a7f6611-a83d-43cf-a2dc-93debf46feeb

Umiliacchi, P., Lane, D., Romano, F., & SpA, A. (2011, May). Predictive maintenance of railway subsystems using an Ontology based modelling approach. In *Proceedings of 9th world conference on railway research, May,* 22-26.

Van Aken, S., Bešinović, N., & Goverde, R. M. (2017). Designing alternative railway timetables under infrastructure maintenance possessions. *Transportation Research Part B: Methodological*, *98*, 224-238. https://doi.org/10.1016/j.trb.2016.12.019

Van Aken, S., Bešinović, N., & Goverde, R. M. (2017). Solving large-scale train timetable adjustment problems under infrastructure maintenance possessions. *Journal of Rail Transport Planning & Management*, *7*(3), 141-156. https://doi.org/10.1016/j.jrtpm.2017.06.003

Van der Kamp, M.V. (2014). *Strategic maintenance on the right track* (master thesis, Delft University of Technology). http://resolver.tudelft.nl/uuid:83c8bec4-4933-4809-9fed-ca76a4294ecc

Vansteenwegen, P., Dewilde, T., Burggraeve, S., & Cattrysse, D. (2016). An iterative approach for reducing the impact of infrastructure maintenance on the performance of railway systems. *European Journal of Operational Research*, *252*(1), 39-53. https://doi.org/10.1016/j.ejor.2015.12.037

Verma, Ajit Kumar, Srividya Ajit, and Durga Rao Karanki. *Reliability and safety engineering*. Vol. 43. London: Springer, 2010. https://doi.org/10.1007/978-1-4471-6269-8

Vet, L. (2014). *Het verbeteren van predictief liggingsonderhoud bij ASSET Rail: de interactie tussen medewerkers, techniek, organisatie en omgeving op het trans-organisatorisch onderhoudsproces* (master thesis, University of Twente). http://essay.utwente.nl/65797/1/VetL_1242032_openbaar.pdf

Vromans, M. (2005). *Reliability of railway systems* (doctoral thesis, Erasmus University Rotterdam). http://hdl.handle.net/1765/6773

Wang, K. S., Hsu, F. S., & Liu, P. P. (2002). Modeling the bathtub shape hazard rate function in terms of reliability. *Reliability Engineering & System Safety*, *75*(3), 397-406. https://doi.org/10.1016/S0951-8320(01)00124-7

Wang, W. (2008). Condition-based maintenance modelling. In *Complex system maintenance handbook* (pp. 111-131). Springer, London. https://doi.org/10.1007/978-1-84800-011-7_5

Wang, W. (2012). An overview of the recent advances in delay-time-based maintenance modelling. *Reliability Engineering & System Safety*, *106*, 165-178. https://doi.org/10.1016/j.ress.2012.04.004

Xie, M., & Lai, C. D. (1996). Reliability analysis using an additive Weibull model with bathtub-shaped failure rate function. *Reliability Engineering & System Safety*, *52*(1), 87-93. https://doi.org/10.1016/0951-8320(95)00149-2

Zhang, T., Andrews, J., & Wang, R. (2013). Optimal scheduling of track maintenance on a railway network. *Quality and Reliability Engineering International*, *29*(2), 285-297. https://doi.org/10.1002/qre.1381

Zhao, J., Chan, A. H. C., Roberts, C., & Madelin, K. B. (2007). Reliability evaluation and optimisation of imperfect inspections for a component with multi-defects. *Reliability*

Engineering & System Safety, *92*(1), 65-73. https://doi.org/10.1016/j.ress.2005.11.003

Zoeteman, A. (2001). Life cycle cost analysis for managing rail infrastructure. *European Journal of Transport and Infrastructure Research*, *1*(4), 391-413. http://resolver.tudelft.nl/uuid:7d9bb393-f21e-44b3-b2c8-9e0d90781971

II Implementation of the Fixed Interval Railway Maintenance Optimization Problem

```
ncomponents = size(a,2);

ObjectiveCoefficient = zeros((ncomponents+1)*nweeks,1);

% Cost of maintenance interval
for i = 1:ncomponents
    ObjectiveCoefficient((i-1)*nweeks+1:i*nweeks) =
(CoF(i)*(a(i)*exp(b(i)*[1:nweeks])./[1:nweeks]+c(i)*exp(d(i)*[1:nweeks])./[
1:nweeks]-a(i)./[1:nweeks]-
c(i)./[1:nweeks])+CoM(i)./[1:nweeks])*NumberOfComponentsOfType(i)*nweeks;
end

% Cost of possession
ObjectiveCoefficient(ncomponents*nweeks+1:(ncomponents+1)*nweeks) = CoP;

% Preallocate LinearConstraintMatrix
LinearConstraintMatrix =
zeros(ncomponents*(nweeks+1),(ncomponents+1)*nweeks);

% Possession constraints
LinearConstraintMatrix(1:ncomponents*nweeks,1:ncomponents*nweeks) =
eye(ncomponents*nweeks);

for i = 1:nweeks
    NumberOfMoments = floor(nweeks/i);
    for j = 1:ncomponents
        LinearConstraintMatrix((j-
1)*nweeks+i,ncomponents*nweeks+(1:NumberOfMoments)*i) = -1/NumberOfMoments;
    end
end
RightHandSide = zeros(ncomponents*nweeks,1);
Sense = repmat('<',ncomponents*nweeks,1);

% Exactly one interval per component
for i = 1:ncomponents
    LinearConstraintMatrix(ncomponents*nweeks+i,(i-1)*nweeks+1:i*nweeks)=1;
end
RightHandSide(ncomponents*nweeks+1:ncomponents*(nweeks+1)) = 1;
Sense(ncomponents*nweeks+1:ncomponents*(nweeks+1)) = '=';

model.A = sparse(LinearConstraintMatrix);
model.obj = ObjectiveCoefficient;
model.sense = Sense;
model.vtype = 'B';
model.rhs = RightHandSide;

solution = gurobi(model)

sparse(solution.x)
```

III Initial Implementation of the Individual Interval Railway Maintenance Optimization Problem

```
ncomponents = size(a,2);

% Preallocate variables
t_opt = zeros(1,ncomponents);

for i=1:ncomponents
    % Determine the optimal maintenance interval
    syms t
    eqn = CoF(i)*(a(i)*b(i)*exp(b(i)*t)/t-
a(i)*exp(b(i)*t)/t^2+c(i)*d(i)*exp(d(i)*t)/t-
c(i)*exp(d(i)*t)/t^2+a(i)/t^2+c(i)/t^2)-CoM(i)/t^2 == 0;
    solution = vpasolve(eqn,t);
    t_opt(i) = double(solution);
end

nintervalvalues = min(nweeks,ceil(2*t_opt));
nintervals = ceil(nweeks./t_opt);

% Preallocate variables
ObjectiveCoefficient =
zeros(sum(nintervals.*(nintervalvalues+1))+sum(nintervals-
1)*(nweeks+1)+nweeks,1);
LinearConstraintMatrix = zeros(2*sum(nintervals)+sum(nintervals-
1)+nweeks,sum(nintervals.*(nintervalvalues+1))+sum(nintervals-
1)*(nweeks+1)+nweeks);
ModelSense = zeros(2*sum(nintervals)+sum(nintervals-1)+nweeks,1);
ModelSense = char(ModelSense);
RightHandSide = zeros(2*sum(nintervals)+sum(nintervals-1)+nweeks,1);

% Cost of failure as a function of maintenance interval
for i = 1:ncomponents
    ObjectiveCoefficient(sum(nintervals(1:i-1).*(nintervalvalues(1:i-
1)+1))+(1:nintervalvalues(i)+1)) =
NumberOfComponentsOfType(i)*(CoF(i)*(a(i)*exp(b(i)*((0:nintervalvalues(i))+
TimeSinceLastMaintenance(i)))+c(i)*exp(d(i)*((0:nintervalvalues(i))+TimeSin
ceLastMaintenance(i)))+f(i)*((0:nintervalvalues(i))+TimeSinceLastMaintenanc
e(i))-a(i)-c(i))-
CoF(i)*(a(i)*exp(b(i)*TimeSinceLastMaintenance(i))+c(i)*exp(d(i)*TimeSinceL
astMaintenance(i))+f(i)*TimeSinceLastMaintenance(i)-a(i)-c(i)));
    for j = 1:nintervals(i)-1
        ObjectiveCoefficient(sum(nintervals(1:i-1).*(nintervalvalues(1:i-
1)+1))+j*(nintervalvalues(i)+1)+(1:nintervalvalues(i)+1)) =
NumberOfComponentsOfType(i)*(CoF(i)*(a(i)*exp(b(i)*(0:nintervalvalues(i)))+
c(i)*exp(d(i)*(0:nintervalvalues(i)))+f(i)*(0:nintervalvalues(i))-a(i)-
c(i))+[0 repmat(CoM(i),[1,nintervalvalues(i)])]);
    end
end

% Add cost of possessions
ObjectiveCoefficient(sum(nintervals.*(nintervalvalues+1))+sum(nintervals-
1)*(nweeks+1)+(1:nweeks)) = CoP;
```

```
% Define the points in time
for i = 1:ncomponents
    for j = 1:nintervals(i)-1
        for k = 1:j
            LinearConstraintMatrix(sum(nintervals(1:i-
1))+j,sum(nintervals(1:i-1).*(nintervalvalues(1:i-1)+1))+(k-
1)*(nintervalvalues(i)+1)+(1:nintervalvalues(i)+1)) = 0:nintervalvalues(i);
        end
        LinearConstraintMatrix(sum(nintervals(1:i-
1))+j,sum(nintervals.*(nintervalvalues+1))+sum(nintervals(1:i-1)-
1)*(nweeks+1)+(j-1)*(nweeks+1)+(1:nweeks+1)) = -(0:nweeks);
    end
    LinearConstraintMatrix(sum(nintervals(1:i)),sum(nintervals(1:i-
1).*(nintervalvalues(1:i-1)+1))+(1:nintervals(i)*(nintervalvalues(i)+1))) =
repmat(0:nintervalvalues(i),[1,nintervals(i)]);
    RightHandSide(sum(nintervals(1:i))) = nweeks;
end

ModelSense(1:sum(nintervals)) = '=';

% Define the possessions
LinearConstraintMatrix(sum(nintervals)+(1:nweeks),sum(nintervals.*(ninterva
lvalues+1))+(1:sum(nintervals-1)*(nweeks+1))) = repmat([eye(nweeks)
zeros(nweeks,1)],[1,sum(nintervals-1)]);
LinearConstraintMatrix(sum(nintervals)+(1:nweeks),sum(nintervals.*(ninterva
lvalues+1))+sum(nintervals-1)*(nweeks+1)+(1:nweeks)) = -sum(nintervals-
1)*eye(nweeks);
ModelSense(sum(nintervals)+(1:nweeks)) = '<';
RightHandSide(sum(nintervals)+(1:nweeks)) = 0.5;

% Ensure every interval has one value
for i = 1:ncomponents
    for j = 1:nintervals(i)
        LinearConstraintMatrix(sum(nintervals)+nweeks+sum(nintervals(1:i-
1))+j,sum(nintervals(1:i-1).*(nintervalvalues(1:i-1)+1))+(j-
1)*(nintervalvalues(i)+1)+(1:nintervalvalues(i)+1)) = 1;
    end
end

ModelSense(sum(nintervals)+nweeks+(1:sum(nintervals))) = '=';
RightHandSide(sum(nintervals)+nweeks+(1:sum(nintervals))) = 1;

% Ensure every moment has at most one value
for i = 1:ncomponents
    for j = 1:nintervals(i)-1
       LinearConstraintMatrix(2*sum(nintervals)+nweeks+sum(nintervals(1:i-
1)-1)+j,sum(nintervals.*(nintervalvalues+1))+sum(nintervals(1:i-1)-
1)*(nweeks+1)+(j-1)*(nweeks+1)+(1:nweeks+1)) = 1;
    end
end

ModelSense(2*sum(nintervals)+nweeks+(1:sum(nintervals-1))) = '=';
RightHandSide(2*sum(nintervals)+nweeks+(1:sum(nintervals-1))) = 1;

model.A = sparse(LinearConstraintMatrix);
model.obj = ObjectiveCoefficient;
model.sense = ModelSense;
model.vtype = 'B';
model.rhs = RightHandSide;
```

```
solution = gurobi(model,params);

Indices = find(round(solution.x)==1);

% Preallocate variables
Intervals = zeros(ncomponents,max(nintervals));
Moments = zeros(ncomponents,max(nintervals-1));

% Get the intervals
for i = 1:ncomponents
    for j = 1:nintervals(i)
        Intervals(i,j) = Indices(sum(nintervals(1:i-1))+j)-
sum(nintervals(1:i-1).*(nintervalvalues(1:i-1)+1))-(j-
1)*(nintervalvalues(i)+1)-1;
    end
end

Intervals

% Get the moments
for i = 1:ncomponents
    for j = 1:nintervals(i)-1
        Moments(i,j) = Indices(sum(nintervals)+sum(nintervals(1:i-1)-1)+j)-
sum(nintervals.*(nintervalvalues+1))-sum(nintervals(1:i-1)-1)*(nweeks+1)-
(j-1)*(nweeks+1)-1;
    end
end

Moments

% Get the possessions
Possessions = Indices(sum(nintervals)+sum(nintervals-1)+1:end)-
sum(nintervals.*(nintervalvalues+1))-sum(nintervals-1)*(nweeks+1)-1
```

IV Improved Implementation of the Individual Interval Railway Maintenance Optimization Problem

```
ncomponents = size(a,2);

% Preallocate variables
t_opt = zeros(1,ncomponents);

for i=1:ncomponents
    % Determine the optimal maintenance interval
    syms t
    eqn = CoF(i)*(a(i)*b(i)*exp(b(i)*t)/t-
a(i)*exp(b(i)*t)/t^2+c(i)*d(i)*exp(d(i)*t)/t-
c(i)*exp(d(i)*t)/t^2+a(i)/t^2+c(i)/t^2)-CoM(i)/t^2 == 0;
    solution = vpasolve(eqn,t);
    t_opt(i) = double(solution);
end

nintervalvalues = min(ceil(2*t_opt),nweeks);
nintervals = max(ceil(nweeks./t_opt)+1,4);

% Preallocate variables
ObjectiveCoefficient =
zeros(sum(nintervals.*nintervalvalues+1)+sum(nintervals-3)*(nweeks-
1)+nweeks,1);
LinearConstraintMatrix = zeros(2*sum(nintervals-
2)+nweeks+sum(nintervals)+sum(nintervals-
3),sum(nintervals.*nintervalvalues+1)+sum(nintervals-3)*(nweeks-1)+nweeks);
ModelSense = zeros(2*sum(nintervals-
2)+nweeks+sum(nintervals)+sum(nintervals-3),1);
ModelSense = char(ModelSense);
RightHandSide = zeros(2*sum(nintervals-
2)+nweeks+sum(nintervals)+sum(nintervals-3),1);

% Cost of failure as a function of maintenance interval
for i = 1:ncomponents
    ObjectiveCoefficient(sum(nintervals(1:i-1).*nintervalvalues(1:i-
1)+1)+(1:nintervalvalues(i)+1)) =
NumberOfComponentsOfType(i)*(CoF(i)*(a(i)*exp(b(i)*((0:nintervalvalues(i))+
TimeSinceLastMaintenance(i)))+c(i)*exp(d(i)*((0:nintervalvalues(i))+TimeSin
ceLastMaintenance(i)))+f(i)*((0:nintervalvalues(i))+TimeSinceLastMaintenanc
e(i))-a(i)-c(i))-
CoF(i)*(a(i)*exp(b(i)*TimeSinceLastMaintenance(i))+c(i)*exp(d(i)*TimeSinceL
astMaintenance(i))+f(i)*TimeSinceLastMaintenance(i)-a(i)-c(i)));
    for j = 1:nintervals(i)-1
        ObjectiveCoefficient(sum(nintervals(1:i-1).*nintervalvalues(1:i-
1)+1)+j*nintervalvalues(i)+1+(1:nintervalvalues(i))) =
NumberOfComponentsOfType(i)*(CoF(i)*(a(i)*exp(b(i)*(1:nintervalvalues(i)))+
c(i)*exp(d(i)*(1:nintervalvalues(i)))+f(i)*(1:nintervalvalues(i))-a(i)-
c(i))+CoM(i));
    end
end

% Add cost of possessions
```

```
ObjectiveCoefficient(sum(nintervals.*nintervalvalues+1)+sum(nintervals-
3)*(nweeks-1)+(1:nweeks)) = CoP;


% Define the points in time
for i = 1:ncomponents
    LinearConstraintMatrix(sum(nintervals(1:i-1)-2)+1,sum(nintervals(1:i-
1).*nintervalvalues(1:i-1)+1)+(1:nintervalvalues(i)+1)) =
0:nintervalvalues(i);
    LinearConstraintMatrix(sum(nintervals(1:i-1)-2)+1,sum(nintervals(1:i-
1).*nintervalvalues(1:i-1)+1)+nintervalvalues(i)+1+(1:nintervalvalues(i)))
= 1:nintervalvalues(i);
    LinearConstraintMatrix(sum(nintervals(1:i-1)-
2)+1,sum(nintervals.*nintervalvalues+1)+sum(nintervals(1:i-1)-3)*(nweeks-
1)+(1:nweeks-1)) = nweeks-(1:nweeks-1);

    for j = 1:nintervals(i)-4
        LinearConstraintMatrix(sum(nintervals(1:i-1)-
2)+1+j,sum(nintervals(1:i-1).*nintervalvalues(1:i-
1)+1)+(j+1)*nintervalvalues(i)+1+(1:nintervalvalues(i))) =
1:nintervalvalues(i);
        LinearConstraintMatrix(sum(nintervals(1:i-1)-
2)+1+j,sum(nintervals.*nintervalvalues+1)+sum(nintervals(1:i-1)-3)*(nweeks-
1)+(j-1)*(nweeks-1)+(1:nweeks-1)) = (1:nweeks-1)-nweeks;
        LinearConstraintMatrix(sum(nintervals(1:i-1)-
2)+1+j,sum(nintervals.*nintervalvalues+1)+sum(nintervals(1:i-1)-3)*(nweeks-
1)+j*(nweeks-1)+(1:nweeks-1)) = nweeks-(1:nweeks-1);
    end

    LinearConstraintMatrix(sum(nintervals(1:i)-2),sum(nintervals(1:i-
1).*nintervalvalues(1:i-1)+1)+(nintervals(i)-
2)*nintervalvalues(i)+1+(1:nintervalvalues(i))) = 1:nintervalvalues(i);
    LinearConstraintMatrix(sum(nintervals(1:i)-2),sum(nintervals(1:i-
1).*nintervalvalues(1:i-1)+1)+(nintervals(i)-
1)*nintervalvalues(i)+1+(1:nintervalvalues(i))) = 1:nintervalvalues(i);
    LinearConstraintMatrix(sum(nintervals(1:i)-
2),sum(nintervals.*nintervalvalues+1)+sum(nintervals(1:i-1)-3)*(nweeks-
1)+(nintervals(i)-4)*(nweeks-1)+(1:nweeks-1)) = (1:nweeks-1)-nweeks;

    RightHandSide(sum(nintervals(1:i-1)-2)+1) = nweeks;
    RightHandSide(sum(nintervals(1:i-1)-2)+(2:nintervals(i)-2)) = 0;
end

ModelSense(1:sum(nintervals-2)) = '=';

% Define the possessions
for i = 1:ncomponents
    LinearConstraintMatrix(sum(nintervals-
2)+(1:nintervalvalues(i)+1),sum(nintervals(1:i-1).*nintervalvalues(1:i-
1)+1)+(1:nintervalvalues(i)+1)) = eye(nintervalvalues(i)+1);
    LinearConstraintMatrix(sum(nintervals-2)+(nweeks-
nintervalvalues(i)+1:nweeks),sum(nintervals(1:i).*nintervalvalues(1:i)+1)-
(0:nintervalvalues(i)-1)) = eye(nintervalvalues(i));
    if nintervalvalues(i) == nweeks
        LinearConstraintMatrix(sum(nintervals-
2)+nintervalvalues(i)+1,sum(nintervals(1:i-1).*nintervalvalues(1:i-
1)+1)+nintervalvalues(i)+1) = 0;
    end
end
```

```
LinearConstraintMatrix(sum(nintervals-
2)+(2:nweeks),sum(nintervals.*nintervalvalues+1)+(1:sum(nintervals-
3)*(nweeks-1))) = repmat(eye(nweeks-1),[1,sum(nintervals-3)]);
LinearConstraintMatrix(sum(nintervals-
2)+(1:nweeks),sum(nintervals.*nintervalvalues+1)+sum(nintervals-3)*(nweeks-
1)+(1:nweeks)) = -sum(nintervals-1)*eye(nweeks);
ModelSense(sum(nintervals-2)+(1:nweeks)) = '<';
RightHandSide(sum(nintervals-2)+(1:nweeks)) = 0.5;

% Ensure every interval has at most one value
for i = 1:ncomponents
    LinearConstraintMatrix(sum(nintervals-2)+nweeks+sum(nintervals(1:i-
1))+1,sum(nintervals(1:i-1).*nintervalvalues(1:i-
1)+1)+(1:nintervalvalues(i)+1)) = 1;
    ModelSense(sum(nintervals-2)+nweeks+sum(nintervals(1:i-1))+1) = '=';
    RightHandSide(sum(nintervals-2)+nweeks+sum(nintervals(1:i-1))+1) = 1;
    for j = 1:nintervals(i)-1
        LinearConstraintMatrix(sum(nintervals-2)+nweeks+sum(nintervals(1:i-
1))+1+j,sum(nintervals(1:i-1).*nintervalvalues(1:i-
1)+1)+j*nintervalvalues(i)+1+(1:nintervalvalues(i))) = 1;
        ModelSense(sum(nintervals-2)+nweeks+sum(nintervals(1:i-1))+1+j) =
'<';
        RightHandSide(sum(nintervals-2)+nweeks+sum(nintervals(1:i-1))+1+j)
= 1.5;
    end
end

% Ensure every moment has at most one value
for i = 1:ncomponents
    for j = 1:nintervals(i)-3
       LinearConstraintMatrix(sum(nintervals-
2)+nweeks+sum(nintervals)+sum(nintervals(1:i-1)-
3)+j,sum(nintervals.*nintervalvalues+1)+sum(nintervals(1:i-1)-3)*(nweeks-
1)+(j-1)*(nweeks-1)+(1:nweeks-1)) = 1;
    end
end

ModelSense(sum(nintervals-2)+nweeks+sum(nintervals)+(1:sum(nintervals-3)))
= '<';
RightHandSide(sum(nintervals-2)+nweeks+sum(nintervals)+(1:sum(nintervals-
3))) = 1.5;

% No interval n without interval n-1
for i = 1:ncomponents
    for j = 1:nintervals(i)-2
        LinearConstraintMatrix(sum(nintervals-
2)+nweeks+sum(nintervals)+sum(nintervals-3)+sum(nintervals(1:i-1)-
2)+j,sum(nintervals(1:i-1).*nintervalvalues(1:i-
1)+1)+j*nintervalvalues(i)+(1:nintervalvalues(i))) = 1;
        LinearConstraintMatrix(sum(nintervals-
2)+nweeks+sum(nintervals)+sum(nintervals-3)+sum(nintervals(1:i-1)-
2)+j,sum(nintervals(1:i-1).*nintervalvalues(1:i-
1)+1)+(j+1)*nintervalvalues(i)+(1:nintervalvalues(i))) = -1;
    end
end

ModelSense(sum(nintervals-2)+nweeks+sum(nintervals)+sum(nintervals-
3)+(1:sum(nintervals-2))) = '>';
RightHandSide(sum(nintervals-2)+nweeks+sum(nintervals)+sum(nintervals-
3)+(1:sum(nintervals-2))) = -0.5;
```

```
model.A = sparse(LinearConstraintMatrix);
model.obj = ObjectiveCoefficient;
model.sense = ModelSense;
model.vtype = 'B';
model.rhs = RightHandSide;

solution = gurobi(model,params);

Indices = find(round(solution.x)==1);

% Preallocate variables
Intervals = zeros(ncomponents,max(nintervals));
Moments = zeros(ncomponents,max(nintervals-3));

% Get the intervals
for i = 1:ncomponents
    Intervals(i,1) = Indices(Indices>sum(nintervals(1:i-
1).*nintervalvalues(1:i-1)+1) & Indices<=sum(nintervals(1:i-
1).*nintervalvalues(1:i-1)+1)+nintervalvalues(i)+1)-sum(nintervals(1:i-
1).*nintervalvalues(1:i-1)+1)-1;
    for j = 1:nintervals(i)-1
        if any(Indices>sum(nintervals(1:i-1).*nintervalvalues(1:i-
1)+1)+j*nintervalvalues(i)+1 & Indices<=sum(nintervals(1:i-
1).*nintervalvalues(1:i-1)+1)+(j+1)*nintervalvalues(i)+1)
            Intervals(i,j+1) = Indices(Indices>sum(nintervals(1:i-
1).*nintervalvalues(1:i-1)+1)+j*nintervalvalues(i)+1 &
Indices<=sum(nintervals(1:i-1).*nintervalvalues(1:i-
1)+1)+(j+1)*nintervalvalues(i)+1)-sum(nintervals(1:i-
1).*nintervalvalues(1:i-1)+1)-j*nintervalvalues(i)-1;
        end
    end
end

Intervals

% Get the moments
for i = 1:ncomponents
    for j = 1:nintervals(i)-3
        if
any(Indices>sum(nintervals.*nintervalvalues+1)+sum(nintervals(1:i-1)-
3)*(nweeks-1)+(j-1)*(nweeks-1) &
Indices<=sum(nintervals.*nintervalvalues+1)+sum(nintervals(1:i-1)-
3)*(nweeks-1)+j*(nweeks-1))
            Moments(i,j) =
Indices(Indices>sum(nintervals.*nintervalvalues+1)+sum(nintervals(1:i-1)-
3)*(nweeks-1)+(j-1)*(nweeks-1) &
Indices<=sum(nintervals.*nintervalvalues+1)+sum(nintervals(1:i-1)-
3)*(nweeks-1)+j*(nweeks-1))-sum(nintervals.*nintervalvalues+1)-
sum(nintervals(1:i-1)-3)*(nweeks-1)-(j-1)*(nweeks-1);
        end
    end
end

Moments

% Get the possessions
Possessions =
Indices(Indices>sum(nintervals.*nintervalvalues+1)+sum(nintervals-
```

```
3)*(nweeks-1))-sum(nintervals.*nintervalvalues+1)-sum(nintervals-
3)*(nweeks-1)-1
```

V First input parameter set

```
a = [-2 -3 -4];
b = [-0.2 -0.3 -0.4];
c = [2 5 8];
d = [0.016 0.016 0.02];
f = [0 0 0 0 0];
CoF = [6 8 12];
CoM = [2 3 4];
CoP = 80;
NumberOfComponentsOfType = [40 30 20];
TimeSinceLastMaintenance = [40 30 20];
nweeks = 200;
params.MIPGap = 0;
params.threads = 0;
params.presolve = 2;
```

VI Second Input Parameter Set

```
a = [-4 -1 -5];
b = [-0.1 -0.4 -0.2 -0.5 -0.3];
c = [11 5 1 8 2];
d = [0.016 0.012 0.02 0.024 0.016];
f = [0 0 0 0 0];
CoF = [8 4 12 16 6];
CoM = [3 1.5 6 2 4];
CoP = 80;
NumberOfComponentsOfType = [10 50 40 30 20];
TimeSinceLastMaintenance = [20 30 10 40 50];
nweeks = 200;
params.MIPGap = 0;
params.Threads = 8;
params.Presolve = 2;
params.TimeLimit = 3600;
```

VII Third Input Parameter Set

```
a = [-4 -5 -2];
b = [-0.4 -0.1 -0.2 -0.3 -0.5];
c = [1 5 11 2 8];
d = [0.016 0.016 0.024 0.012 0.02];
f = [0 0 0 0 0];
CoF = [12 16 4 6 8];
CoM = [3 2 6 4 1.5];
CoP = 80;
NumberOfComponentsOfType = [40 50 30 10 20];
TimeSinceLastMaintenance = [50 40 10 20 30];
nweeks = 200;
params.MIPGap = 0;
params.Threads = 8;
params.Presolve = 2;
params.TimeLimit = 3600;
```

VIII First Input Parameter Set for up to Five Components

```
a = [-2 -3 -4 -1 -5];
b = [-0.2 -0.3 -0.4 -0.1 -0.5];
c = [2 5 8 1 11];
d = [0.016 0.016 0.02 0.012 0.024];
f = [0 0 0 0 0];
CoF = [6 8 12 4 16];
CoM = [2 3 4 1.5 6];
CoP = 80;
NumberOfComponentsOfType = [40 30 20 50 10];
TimeSinceLastMaintenance = [40 30 20 50 10];
nweeks = 200;
params.MIPGap = 0;
params.Threads = 8;
params.Presolve = 2;
params.TimeLimit = 3600;
```

IX Second Input Parameter Set for up to Five Components

```
a = [-4 -1 -5 -3 -2];
b = [-0.1 -0.4 -0.2 -0.5 -0.3];
c = [11 5 1 8 2];
d = [0.016 0.012 0.02 0.024 0.016];
f = [0 0 0 0 0];
CoF = [8 4 12 16 6];
CoM = [3 1.5 6 2 4];
CoP = 80;
NumberOfComponentsOfType = [10 50 40 30 20];
TimeSinceLastMaintenance = [20 30 10 40 50];
nweeks = 200;
params.MIPGap = 0;
params.Threads = 8;
params.Presolve = 2;
params.TimeLimit = 3600;
```

X Third Input Parameter Set for up to Five Components

```
a = [-4 -5 -2 -3 -1];
b = [-0.4 -0.1 -0.2 -0.3 -0.5];
c = [1 5 11 2 8];
d = [0.016 0.016 0.024 0.012 0.02];
f = [0 0 0 0 0];
CoF = [12 16 4 6 8];
CoM = [3 2 6 4 1.5];
CoP = 80;
NumberOfComponentsOfType = [40 50 30 10 20];
TimeSinceLastMaintenance = [50 40 10 20 30];
nweeks = 200;
params.MIPGap = 0;
params.Threads = 8;
params.Presolve = 2;
params.TimeLimit = 3600;
```

XI Overall Test Loop

```
clear
for k = 1:3
    TestLoopd
    TestLoopCoP
    TestLoopCoPvarying
    TestLoopMIPGap
    TestLoopThreads
    TestLoopPresolve
end

AllIntervals =
{IntervalsCoP,IntervalsCoPvarying,Intervalsd,IntervalsMIPGap};

Mediand = median(RunTimed);
MedianCoP = median(RunTimeCoP);
MedianCoPvarying = median(RunTimeCoPvarying);
MedianThreads = median(RunTimeThreads);
MedianPresolve = median(RunTimePresolve);
MedianGapPresolve = median(GapPresolve);

save('TotalTestLoop2506.mat')
```

XII Test Loop for Optimality Gap

```
ThreeComponents

MIPGapset = [0 0.001 0.003 0.01 0.03 0.1 0.3];
RunTimeMIPGap = zeros(1,size(MIPGapset,2));
clear IntervalsMIPGap

for MIPGap = MIPGapset
    params.MIPGap = MIPGap;
    BinaryIndividual5
    RunTimeMIPGap(MIPGapset==params.MIPGap) = solution.runtime;
    IntervalsMIPGap(:,:,MIPGapset==params.MIPGap) = Intervals;
end
```

XIII Test Loop for Presolve Setting

```
ThreeComponents

RunTimePresolve = zeros(1,4);

for Presolve = -1:2
    params.presolve = Presolve;
    BinaryIndividual5
    RunTimePresolve(Presolve+2) = solution.runtime;
end
```

XIV Test Loop for Number of Processor Threads

```
ThreeComponents

RunTimeThreads = zeros(1,5);

for Threads = 0:4
    params.threads = Threads;
    BinaryIndividual5
    RunTimeThreads(Threads+1) = solution.runtime;
end
```

XV Test Loop for Wear Out Shape Parameter d

```
ThreeComponents

dset = [d/2;d;d*2];

for iteration = 1:size(dset,1)
    d=dset(iteration,:);
    BinaryIndividual5
    RunTimed(k,iteration) = solution.runtime;
    Possessionsd(iteration) = size(Possessions,1);
    Intervalsd{iteration} = Intervals;
    TOpts(iteration,:) = t_opt;
end
```

XVI Test Loop for Cost of Possession

```
ThreeComponents

CoPset = [0.25 0.8 2.5 8 25 80 250 800 2500 8000 25000];

for CoP = CoPset
    BinaryIndividual5
    RunTimeCoP(k,CoP==CoPset) = solution.runtime;
    IntervalsCoP{CoP==CoPset} = Intervals;
    PossessionCoP(CoPset==CoP) = size(Possessions,1);
end
```

XVII Test Loop for Varying Cost of Possession

```
ThreeComponents

BinaryIndividual5
RunTimeCoPvarying(k,1) = solution.runtime;
IntervalsCoPvarying{1} = Intervals;

CoP = repmat([150 50],[1,nweeks/2]);
BinaryIndividual5
RunTimeCoPvarying(k,2) = solution.runtime;
IntervalsCoPvarying{2} = Intervals;

CoP = repmat([150 150 150 150 50],[1,nweeks/5]);
BinaryIndividual5
RunTimeCoPvarying(k,3) = solution.runtime;
IntervalsCoPvarying{3} = Intervals;
```

XVIII Test Loop for Planning Horizon

```
ThreeComponents

iteration = 0;
step = 5;
exceedances = 0;

while exceedances < 3
    BinaryIndividual5
    iteration = iteration+1;
    RunTimenweeks(k,iteration) = solution.runtime;
    Gapnweeks(k,iteration) = solution.mipgap;
    nweeks = nweeks+step;
    if solution.runtime < params.TimeLimit-1
        exceedances = 0;
    end
    if solution.runtime > params.TimeLimit-1
        exceedances = exceedances+1;
    end
end
```

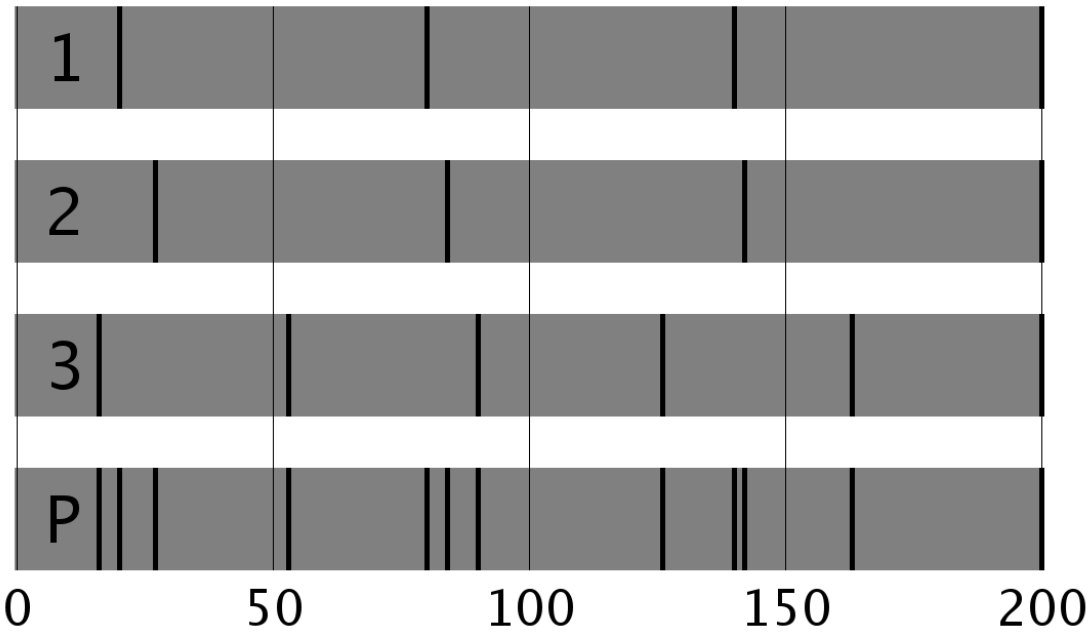
XIX Test Loop for Number of Components

```
FiveComponents
astable = a;

for l = 1:size(astable,2)
    a = astable(1:l)
    BinaryIndividual5
    RunTimeSize(k,l) = solution.runtime;
    IntervalsSize{k,l} = Intervals;
end
```

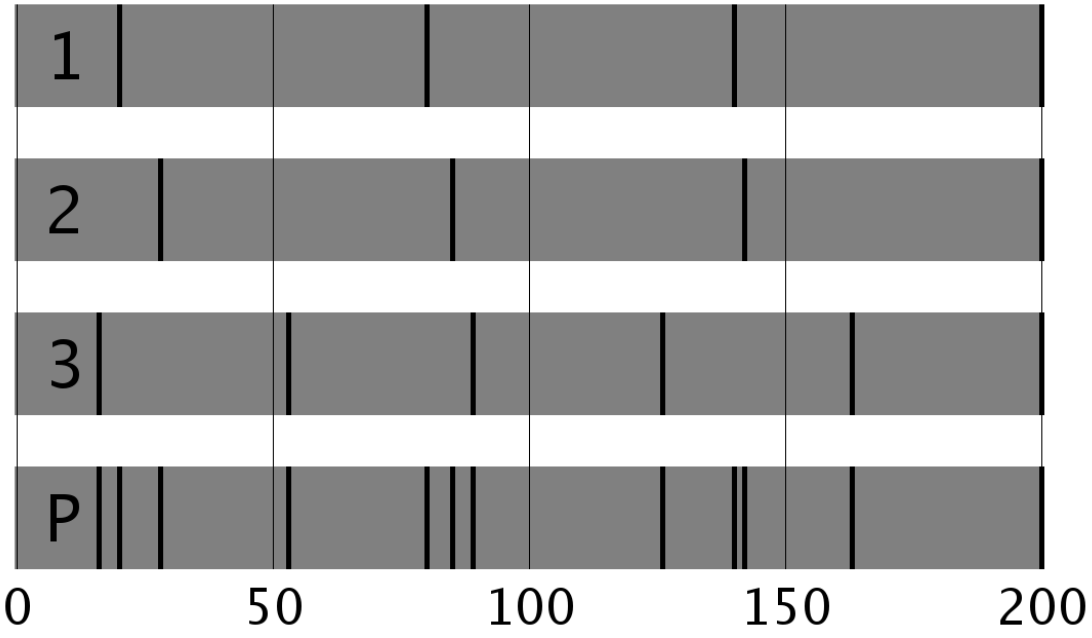
XX Test Cycle that Explores Long Planning Horizons

```
ThreeComponents

iteration = 0;
params.MIPGap = 0.1;
nweeks = 400;
clear solution

while not(exist('solution') && isequal(solution.status,'TIME_LIMIT'))
    iteration = iteration+1;
    nweeks = nweeks+50;
    BinaryIndividual5
    RuntimeConvergence(iteration,1) = nweeks;
    RuntimeConvergence(iteration,2) = solution.runtime;
    IntervalsConvergence{iteration} = Intervals;
end
```

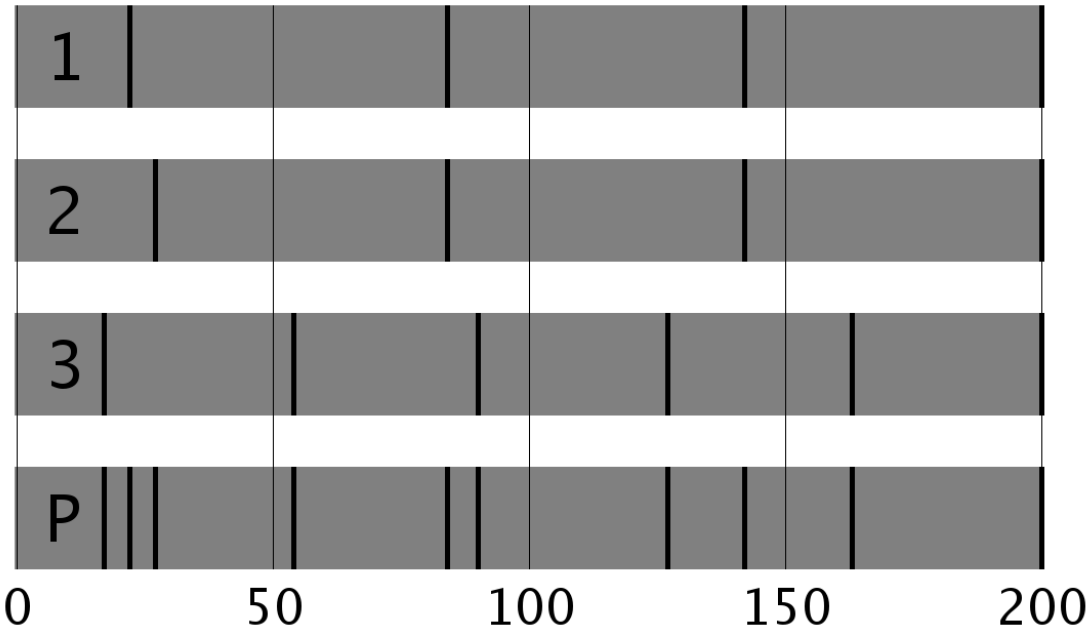
XXI Resulting Schedules from Experimentation on Cost of Possession, Parameter Set 1



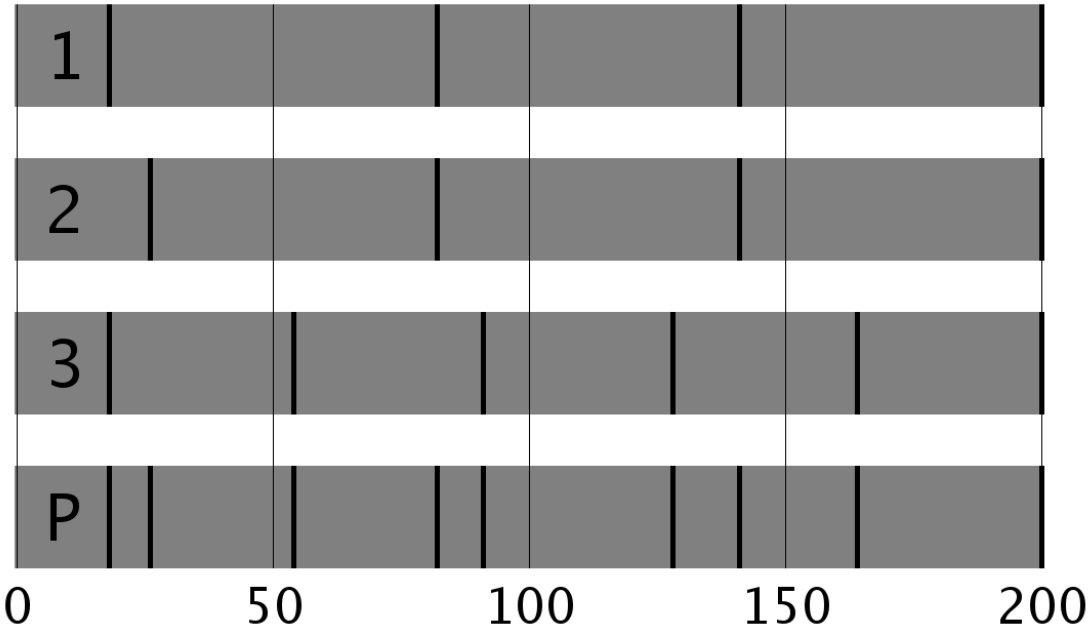
CoP = 0.25



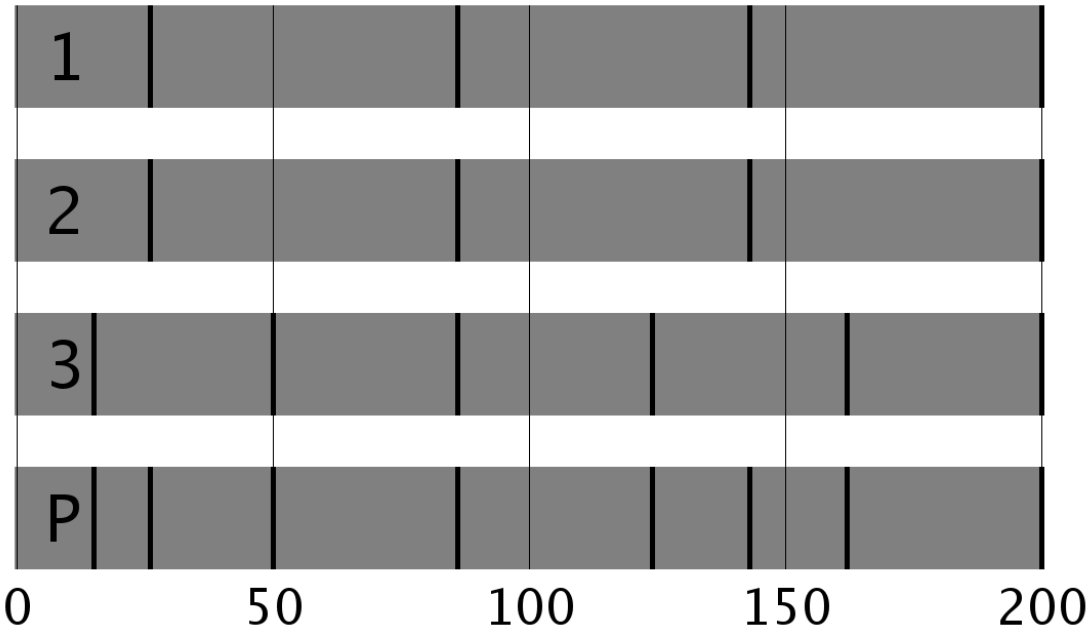
CoP = 0.8



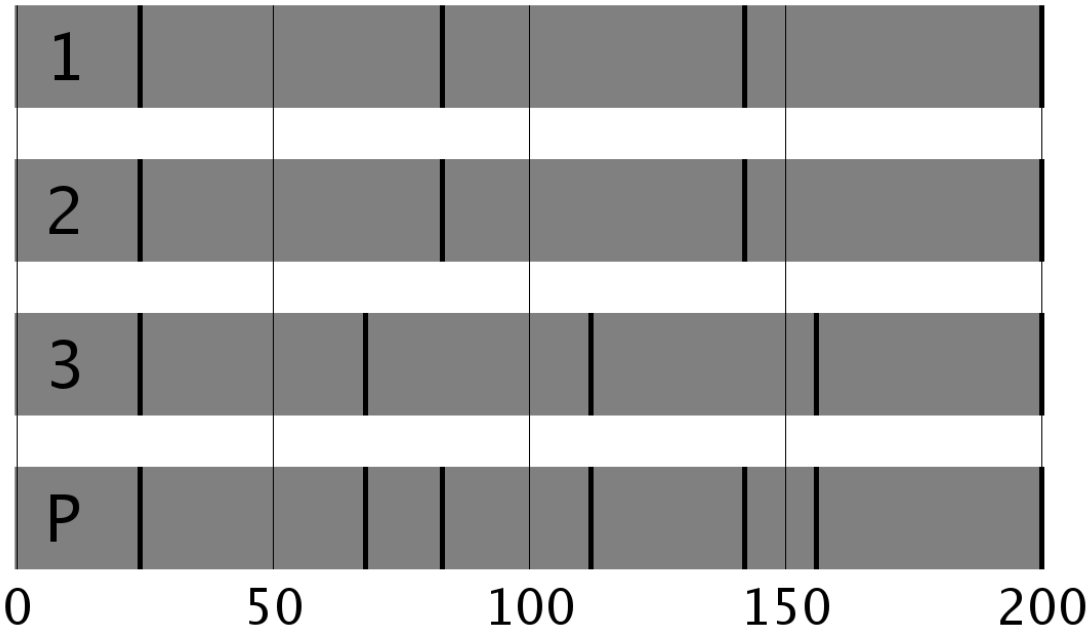
CoP = 2.5



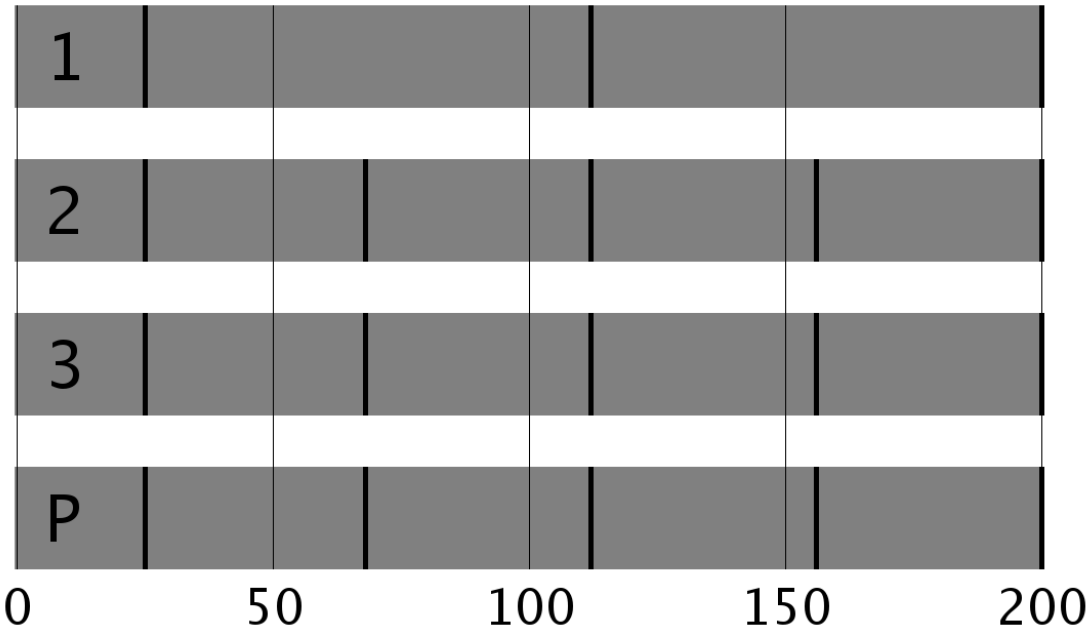
CoP = 8



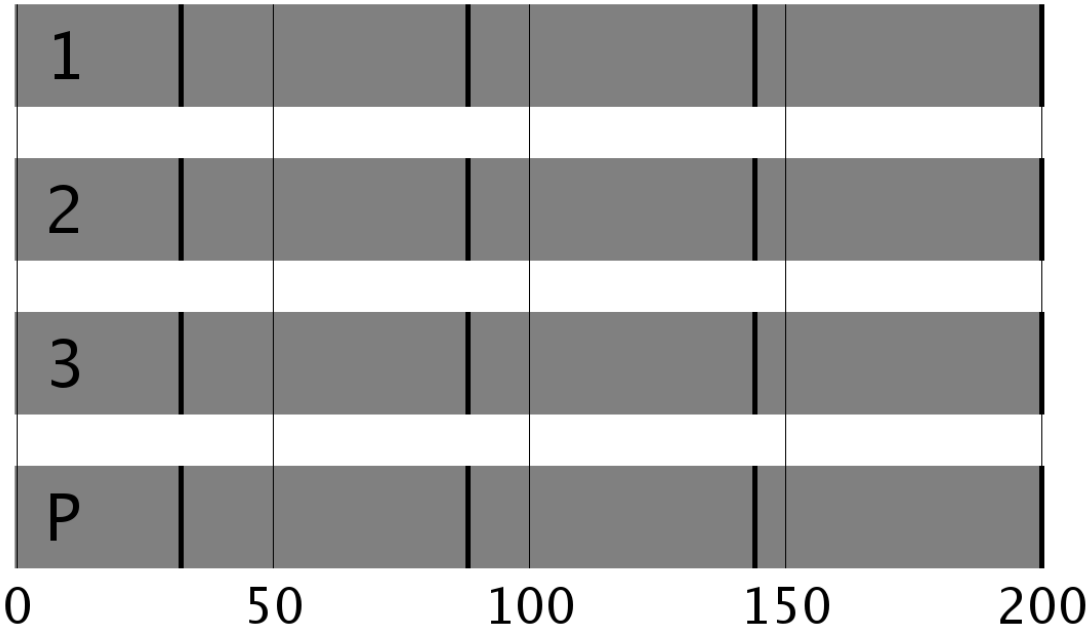
CoP = 25



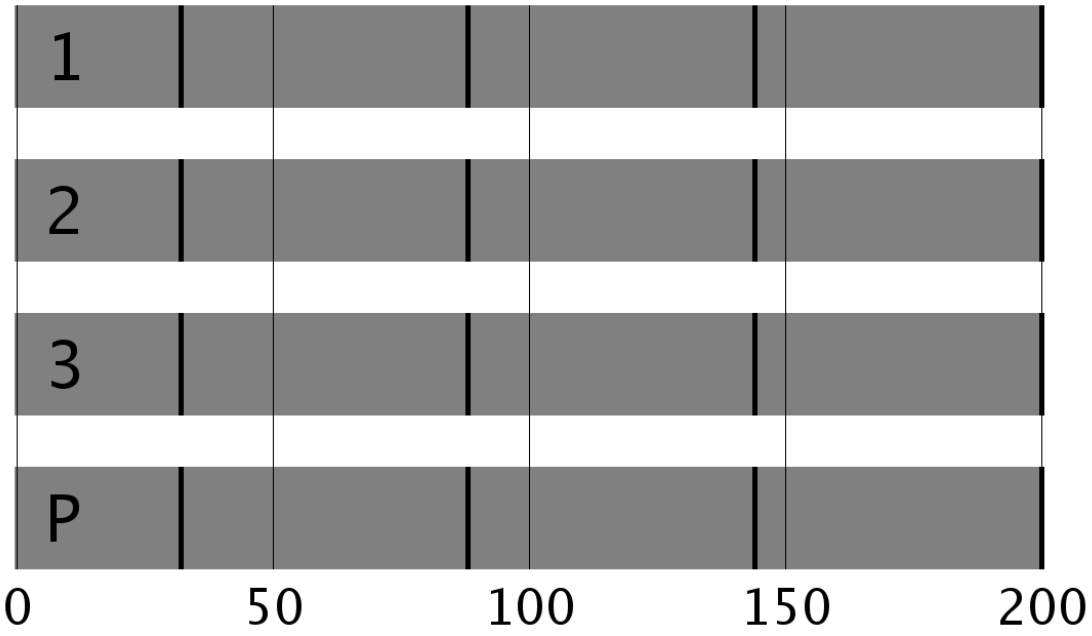
CoP = 80



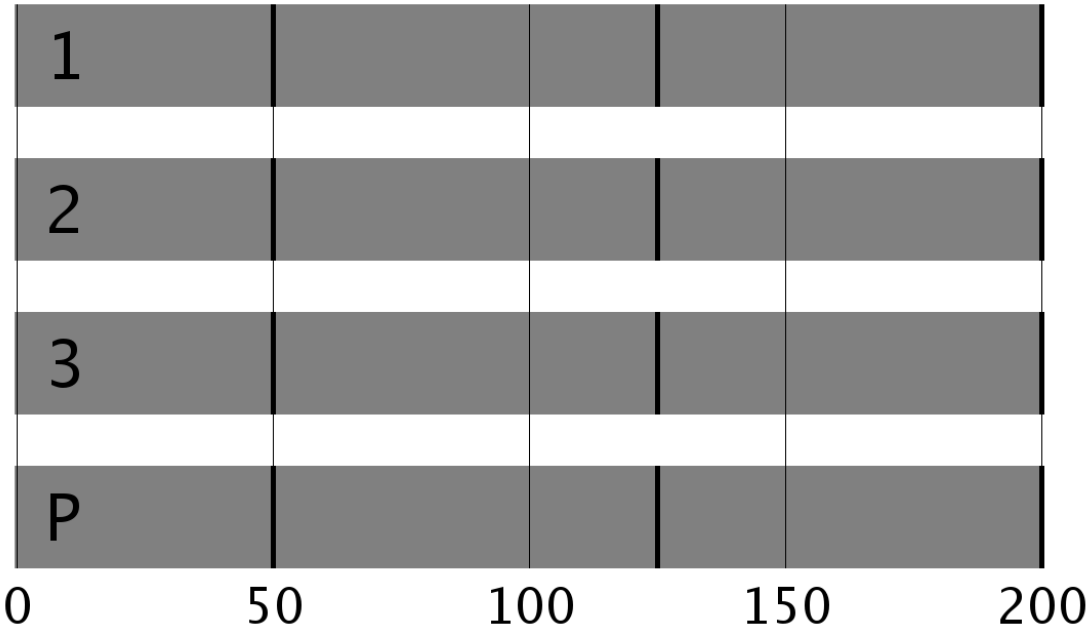
CoP = 250



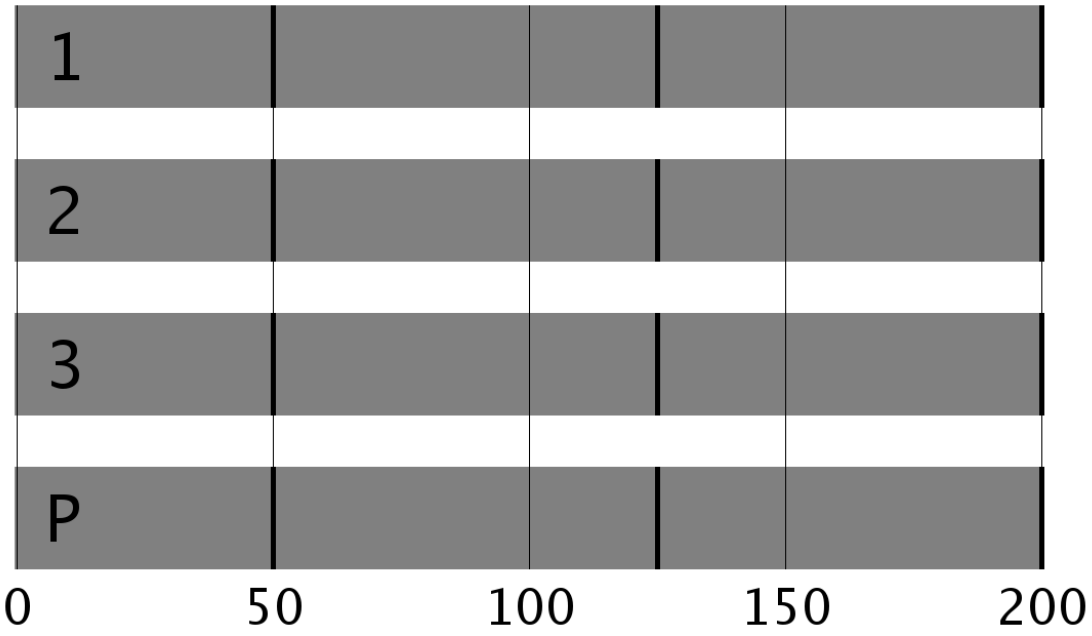
CoP = 800



CoP = 2,500

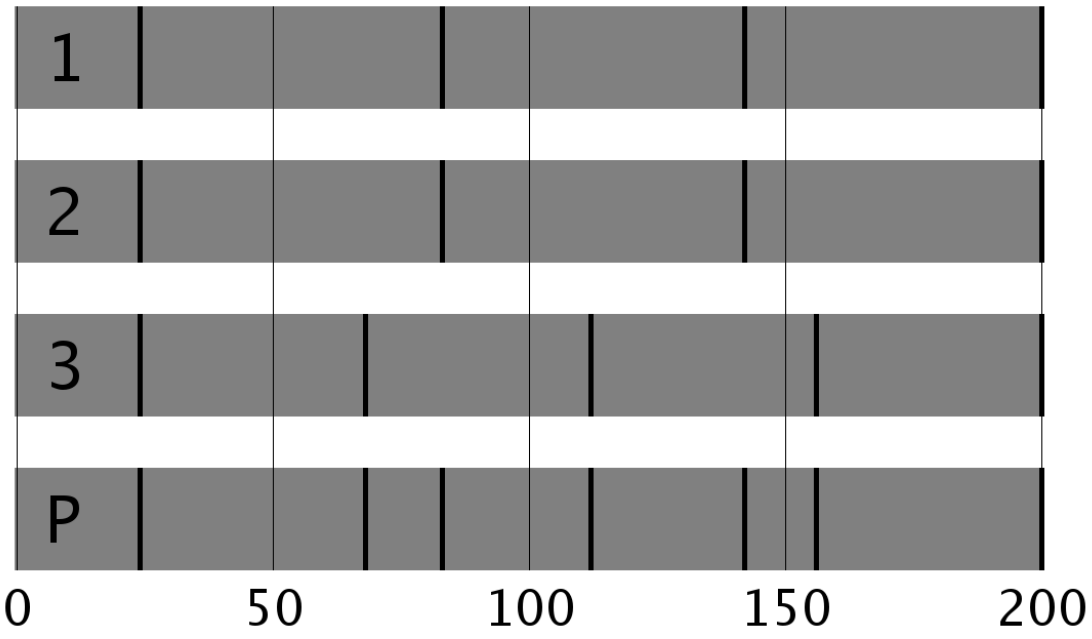


CoP = 8,000

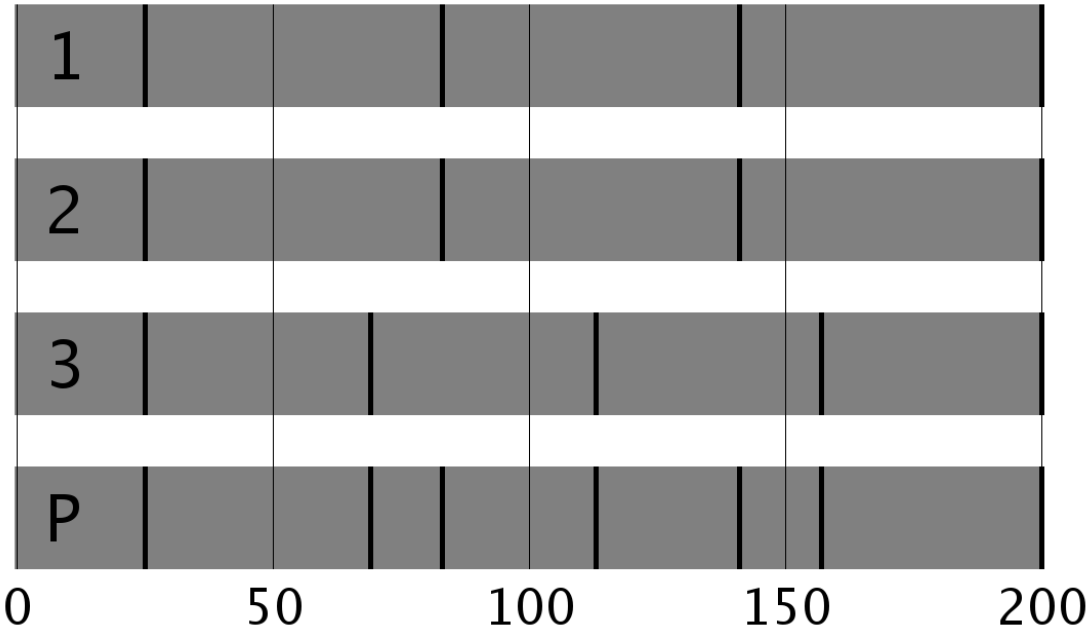


CoP = 25,000

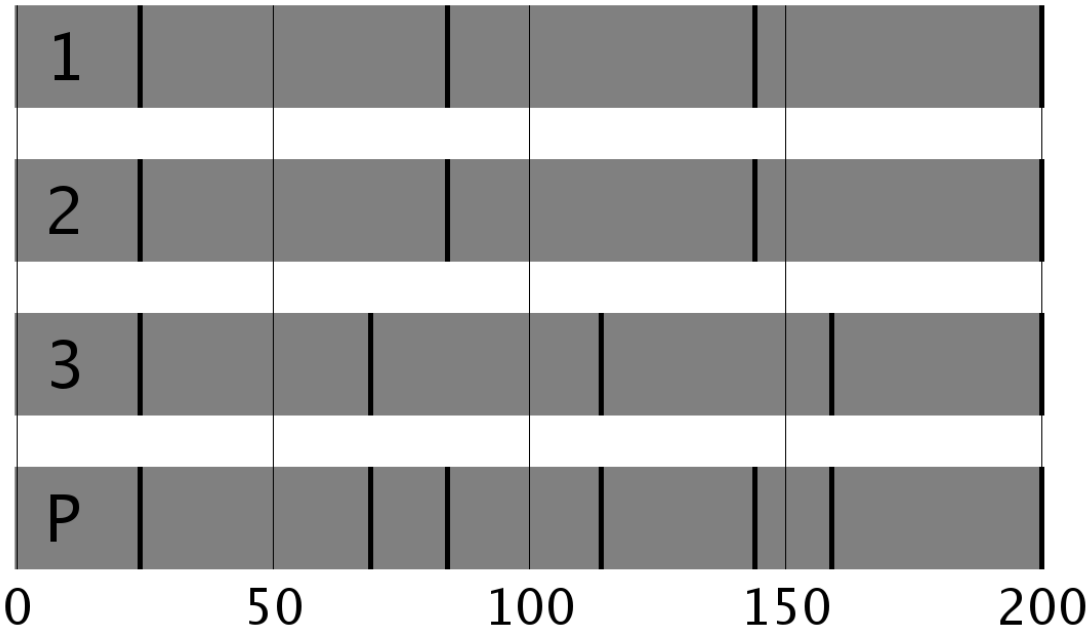
XXII Resulting Schedules from Experimentation on Alternating Cost of Possession, Parameter Set 1



CoP = 80

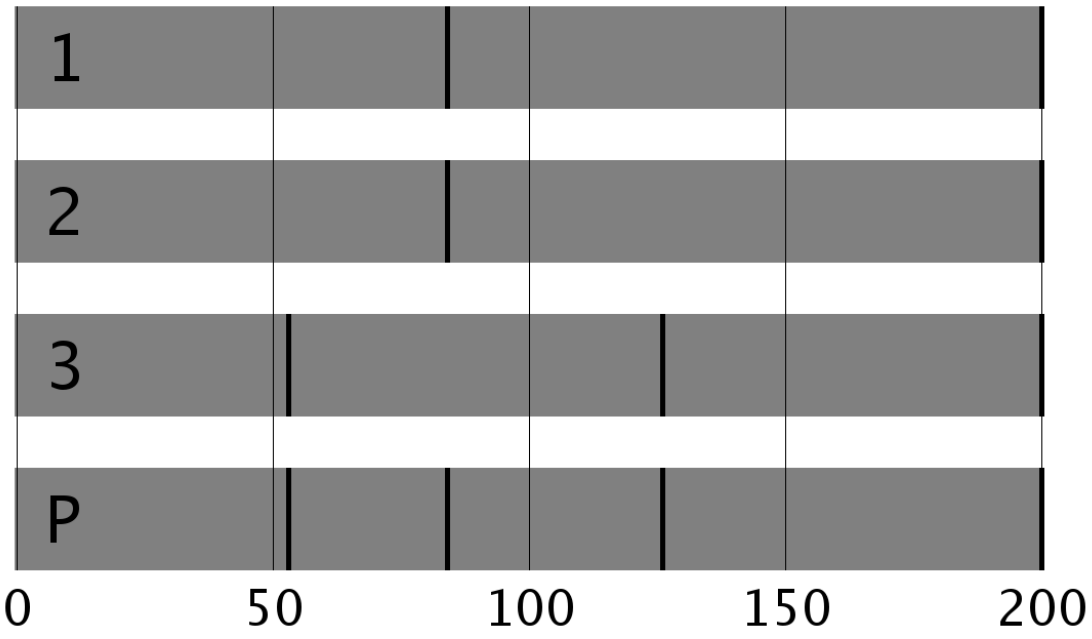


CoP = 50, 150, etc.

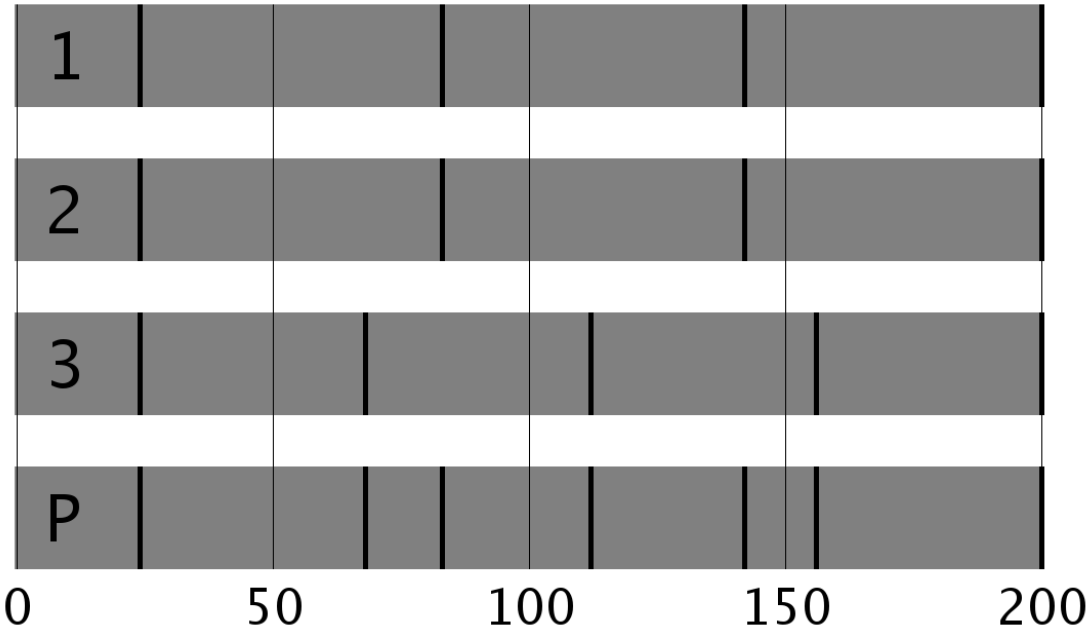


CoP = 50, 50, 50, 50, 150, etc.

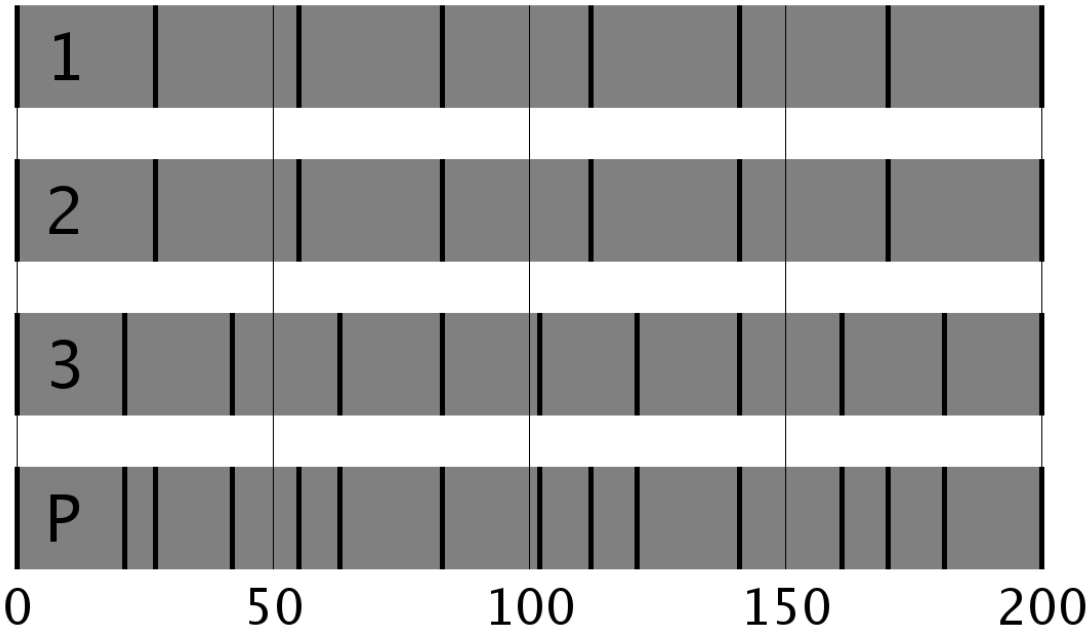
XXIII Resulting Schedules from Experimentation on Wear Out Shape Parameter, Parameter Set 1



d is half of normal

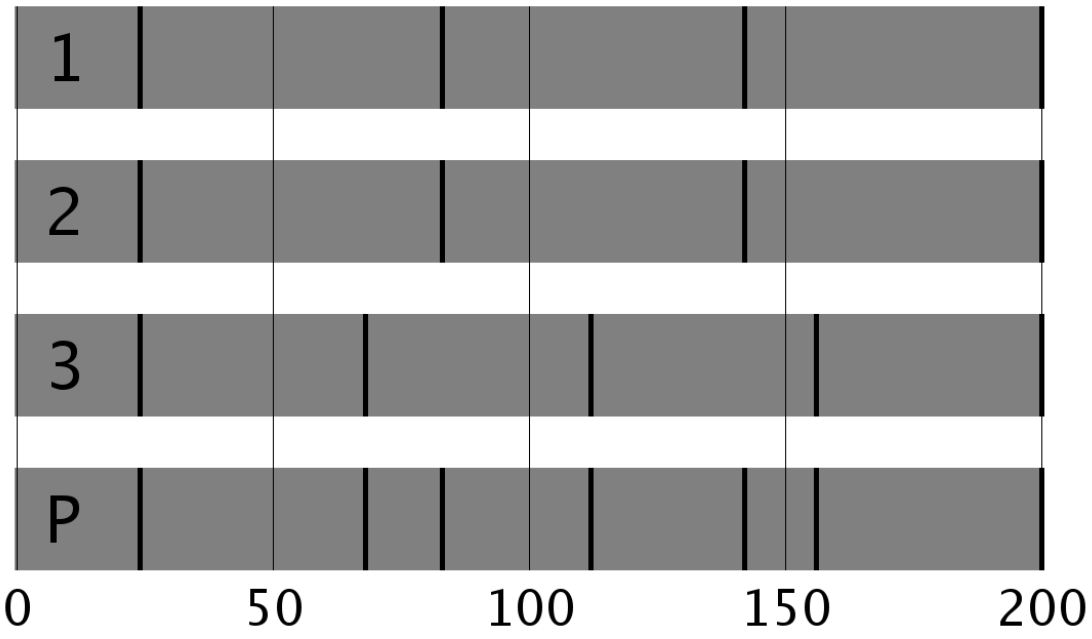


d is normal

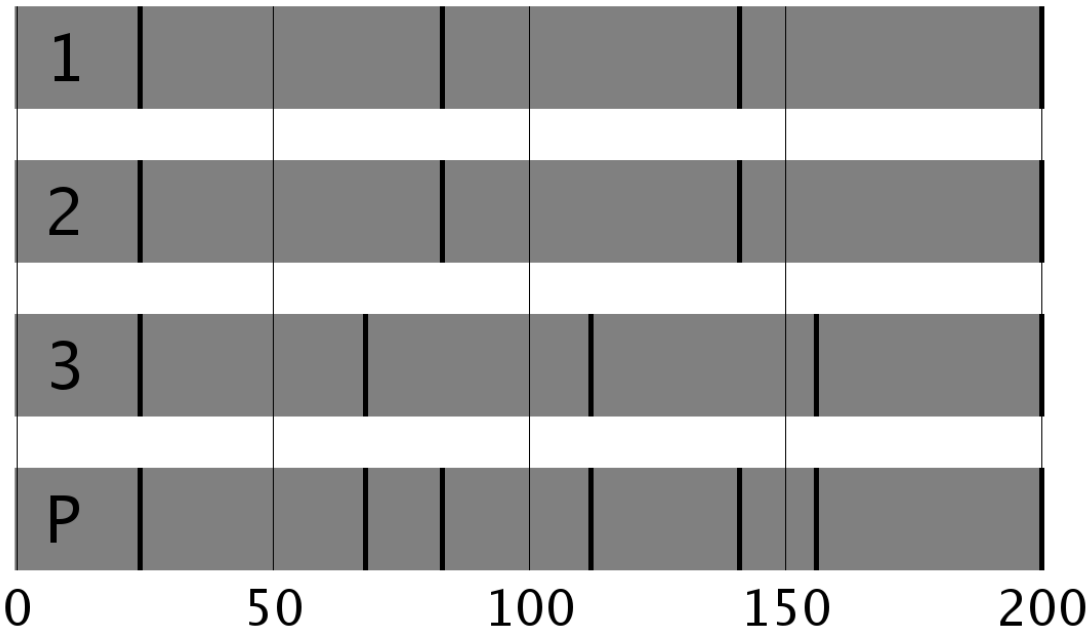


d is double of normal

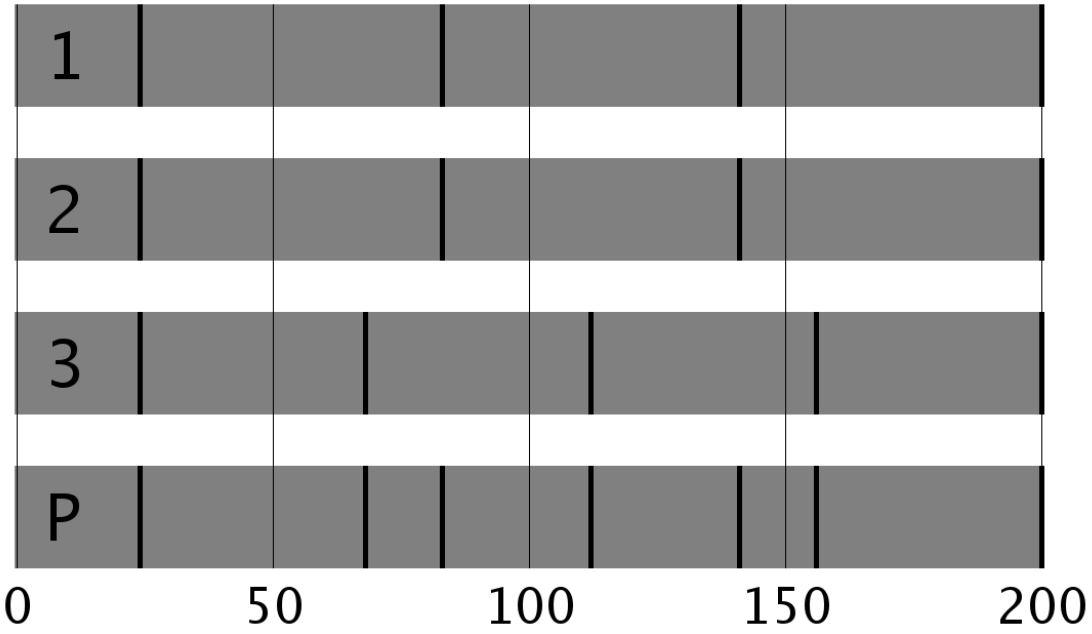
XXIV Resulting Schedules from Experimentation on Optimality Gap, Parameter Set 1



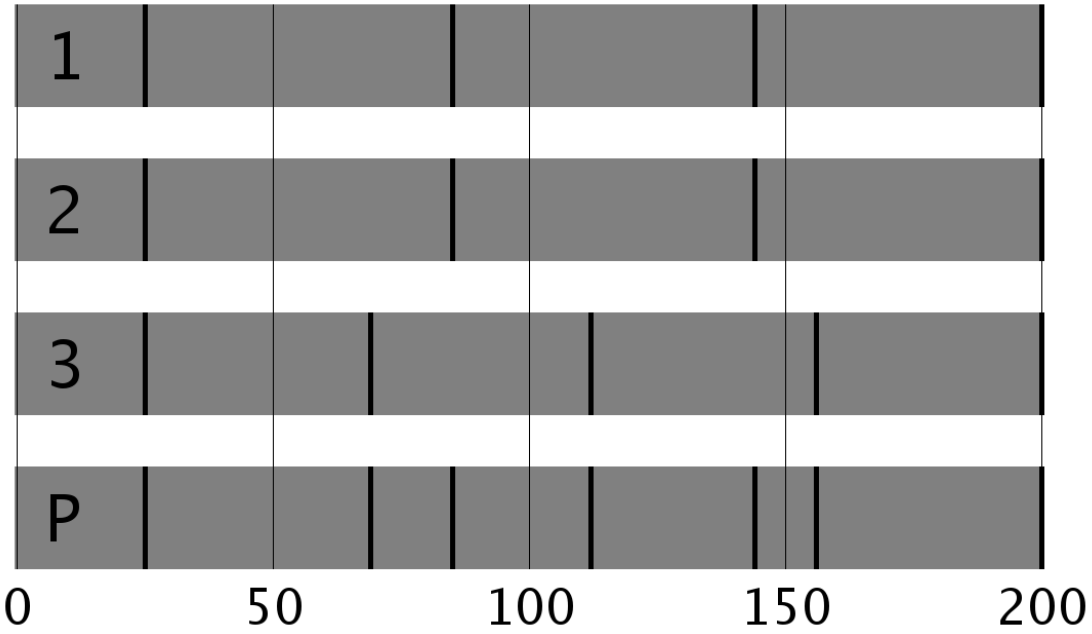
Optimality gap = 0%



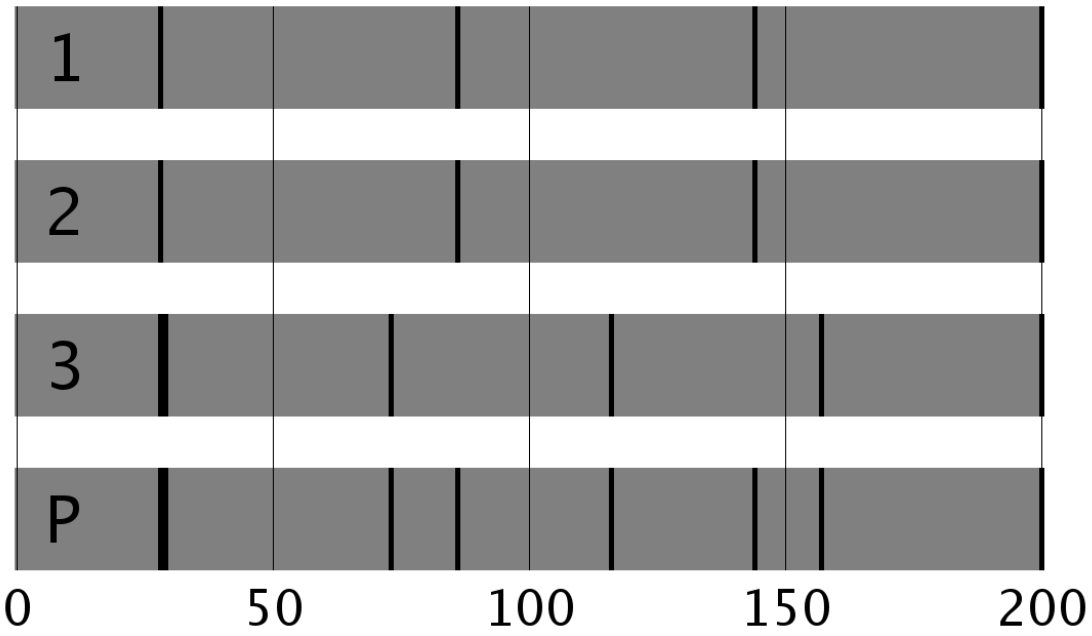
Optimality gap = 0.1%



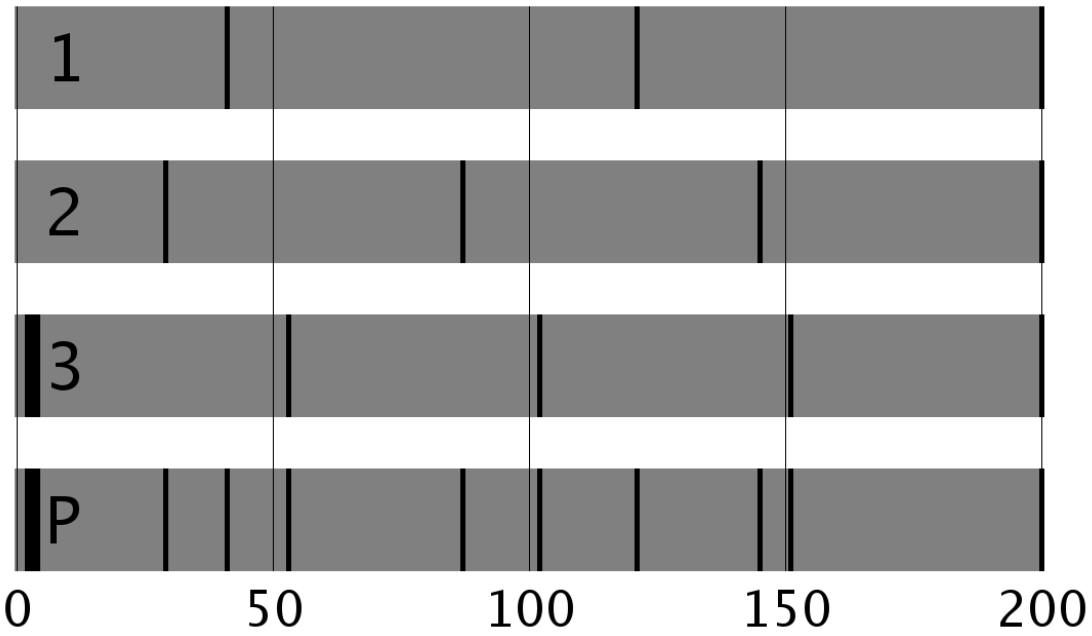
Optimality gap = 0.3%



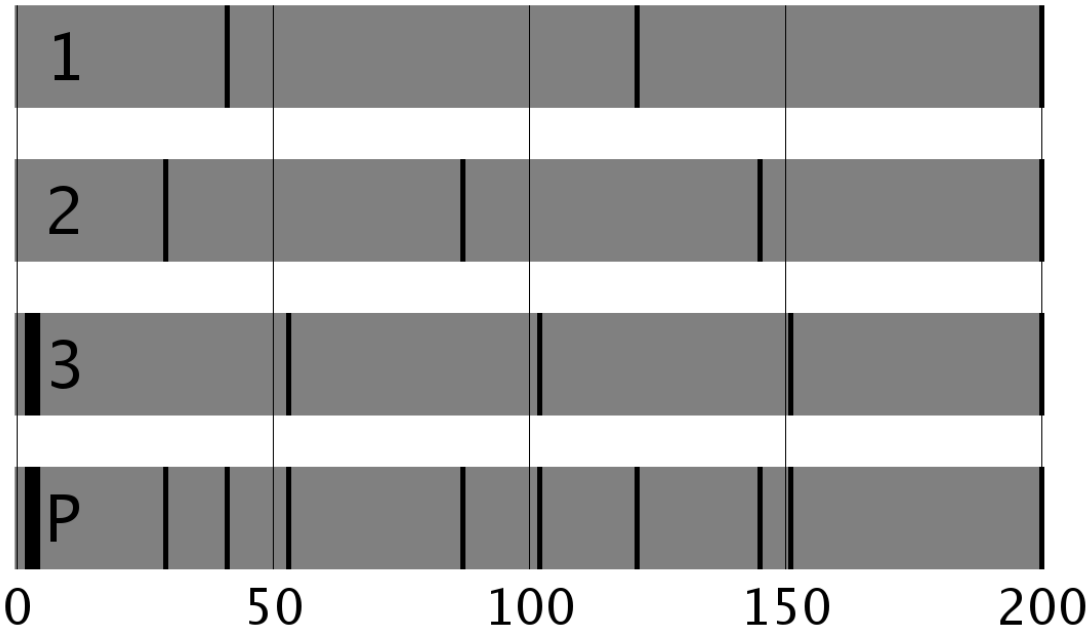
Optimality gap = 1%



Optimality gap = 3%

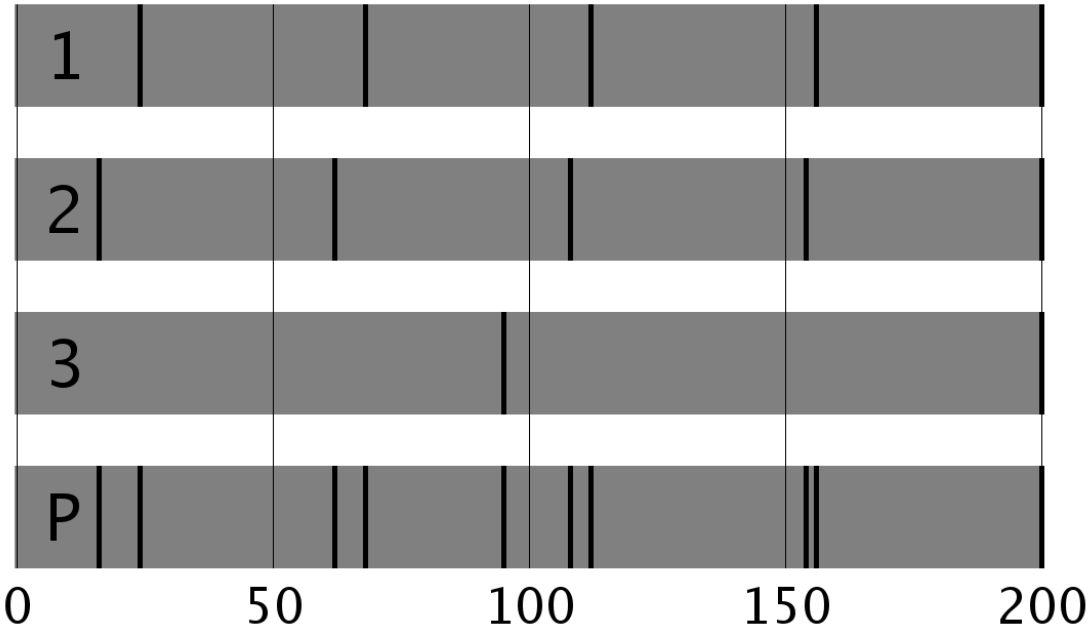


Optimality gap = 10%

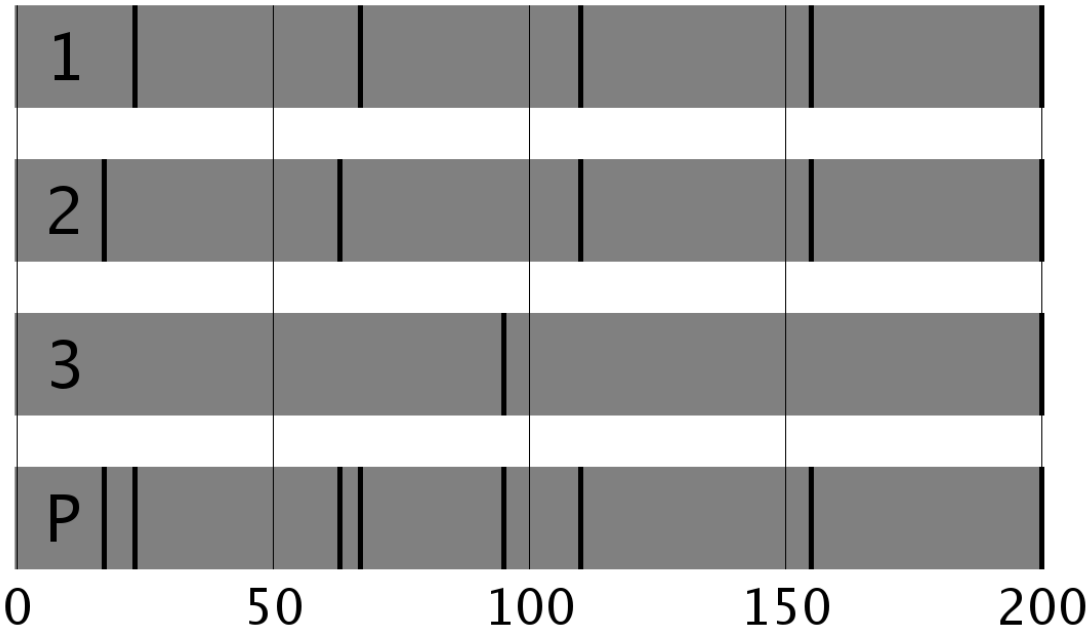


Optimality gap = 30%

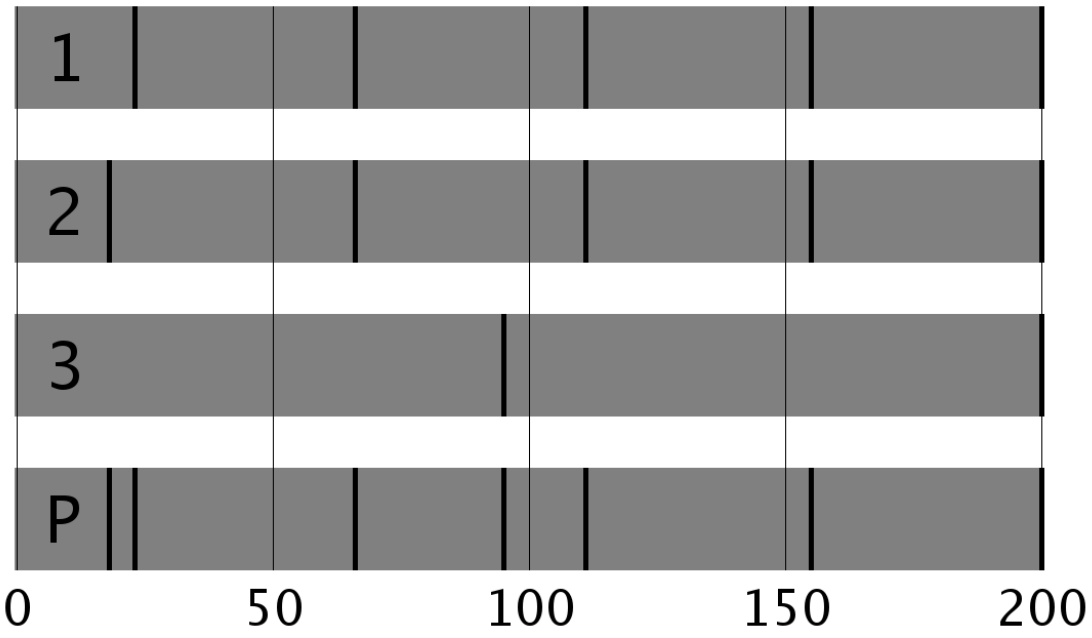
XXV Resulting Schedules from Experimentation on Cost of Possession, Parameter Set 2



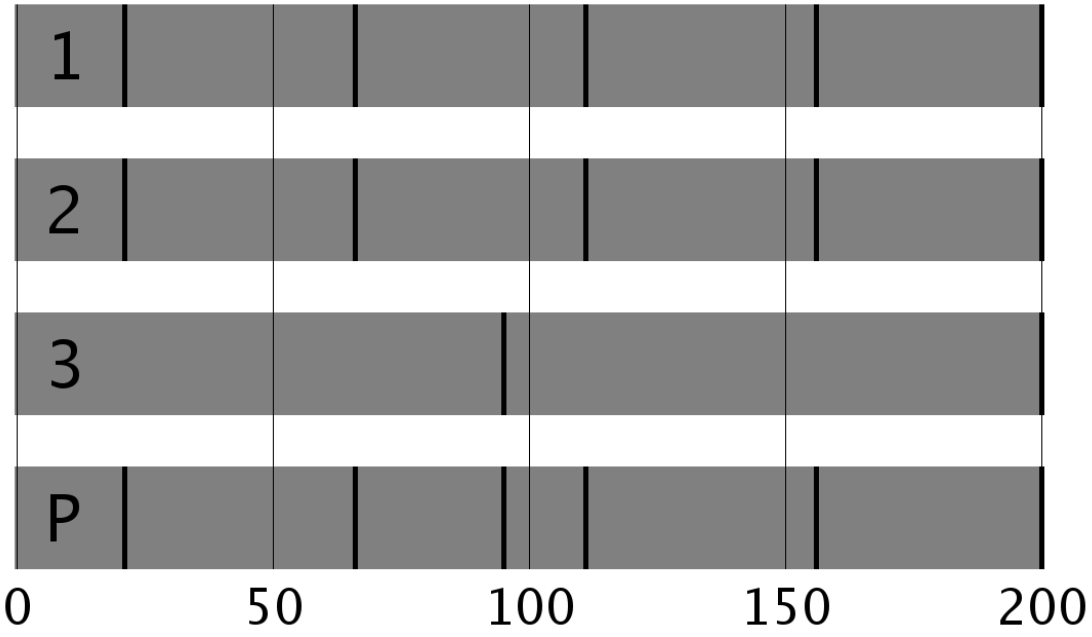
CoP = 0.25



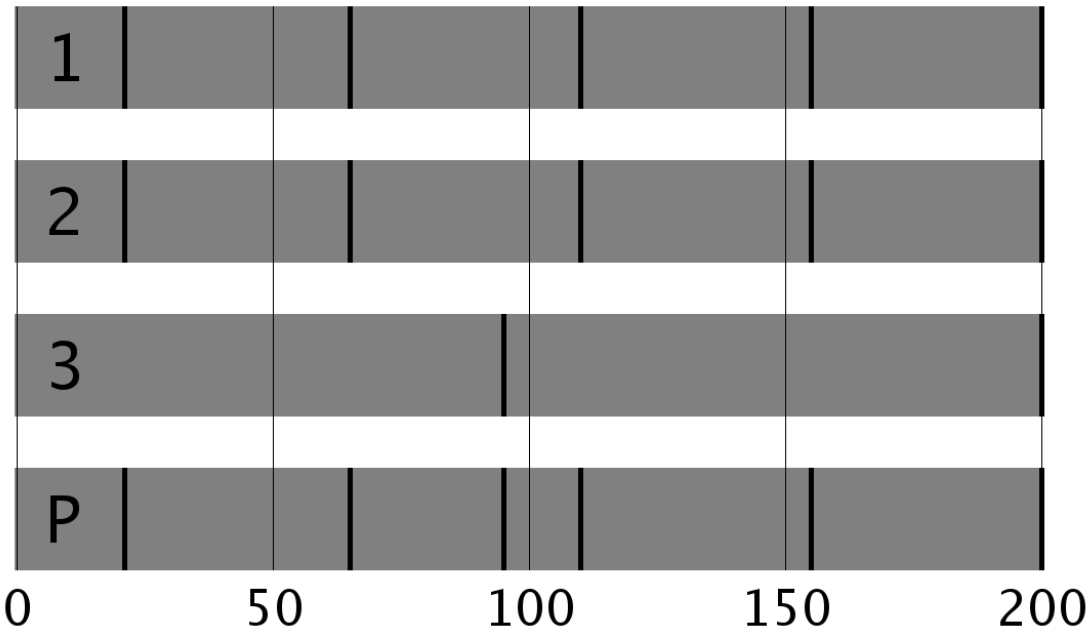
CoP = 0.8



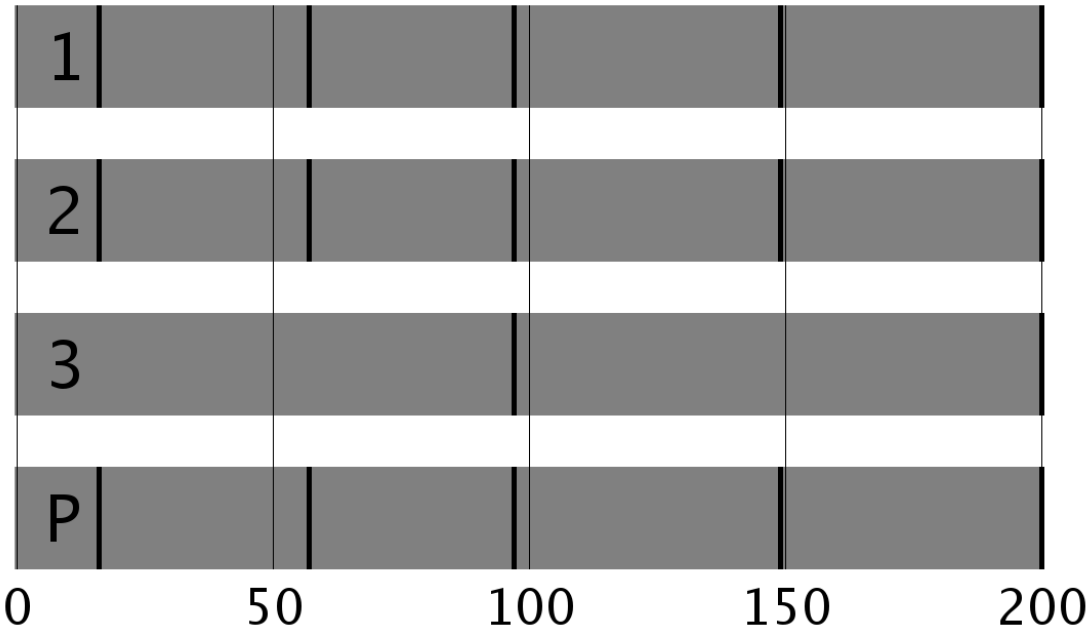
CoP = 2.5



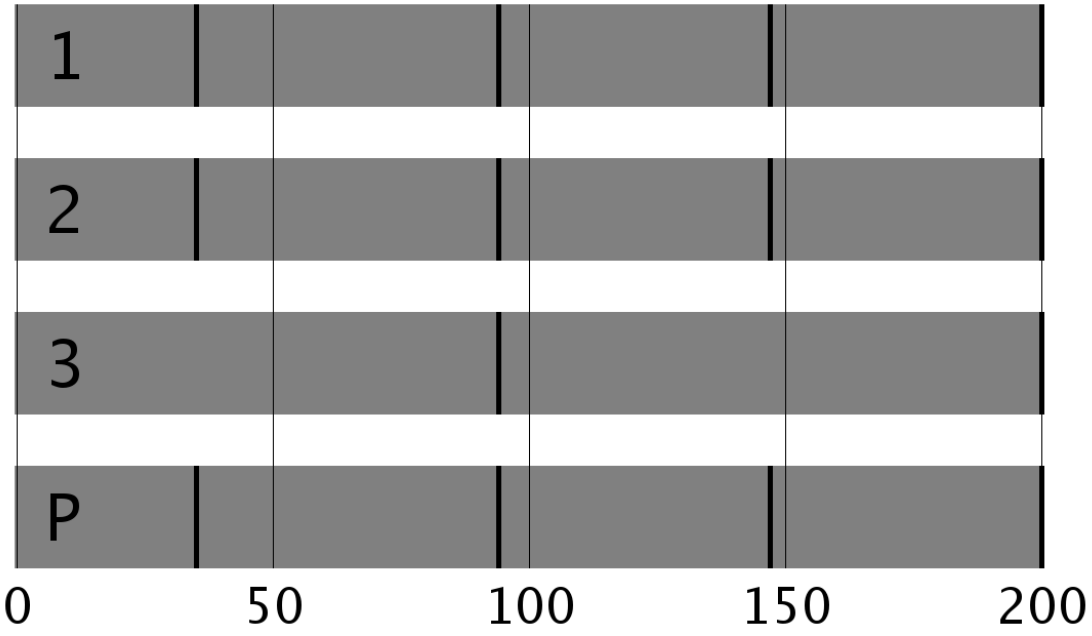
CoP = 8



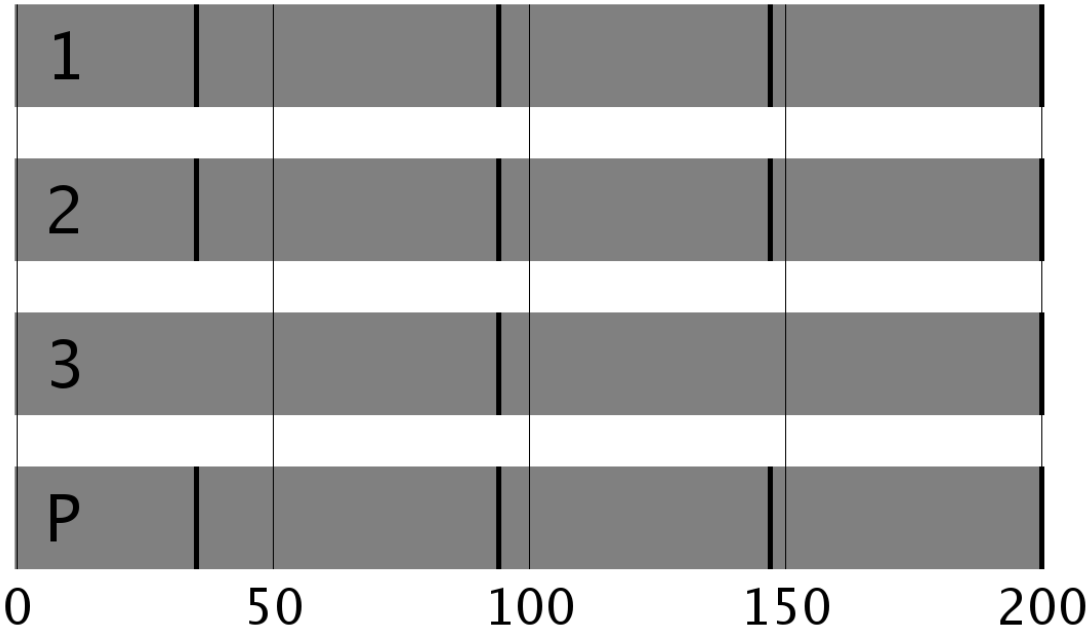
CoP = 25



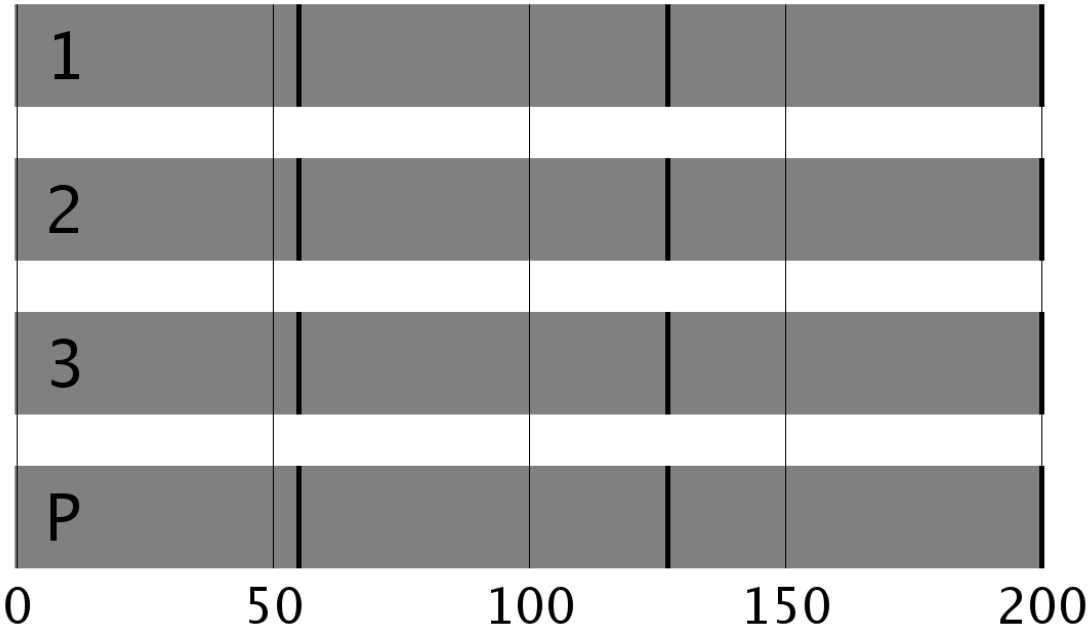
CoP = 80



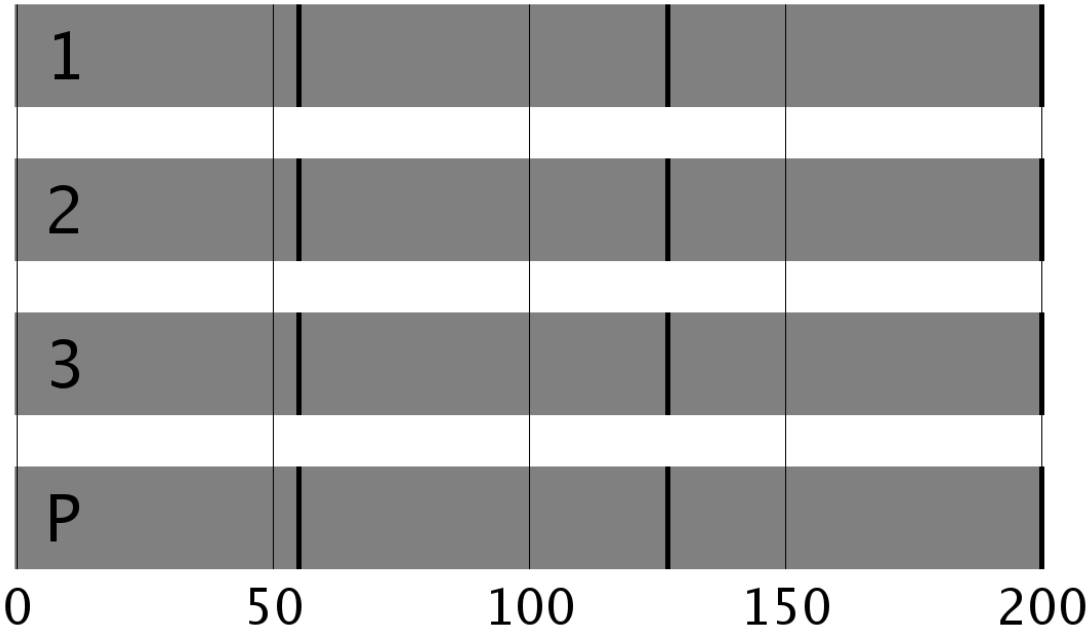
CoP = 250



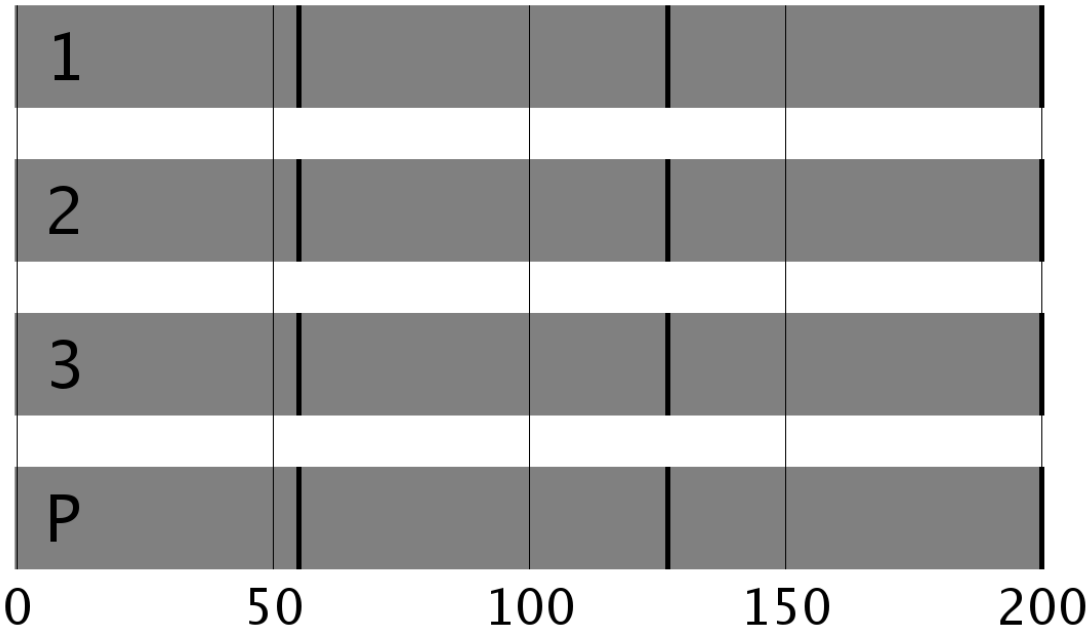
CoP = 800



CoP = 2,500

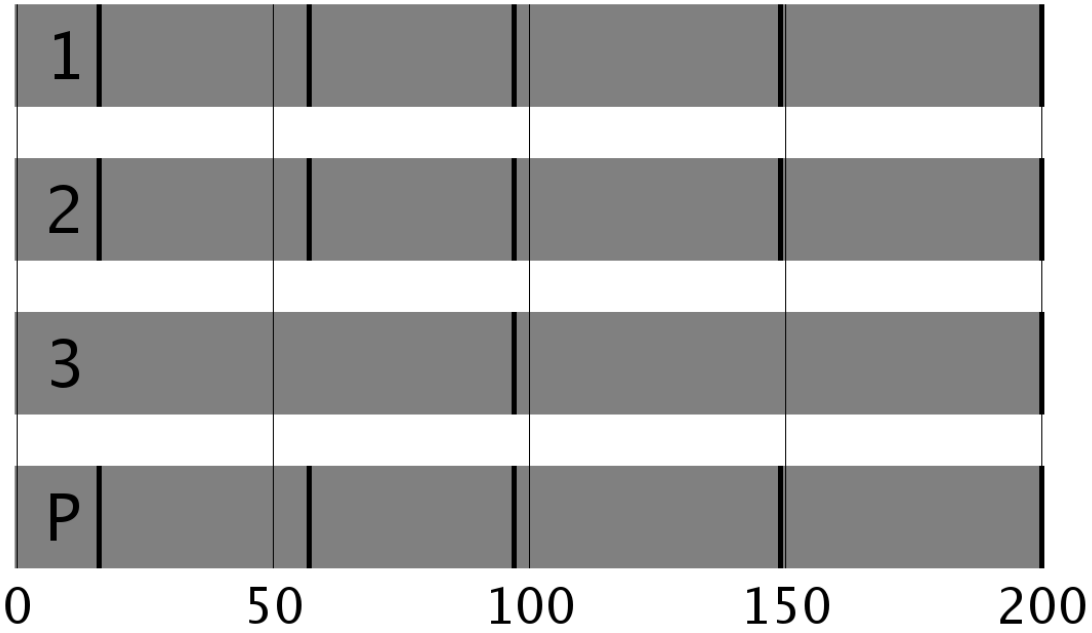


CoP = 8,000

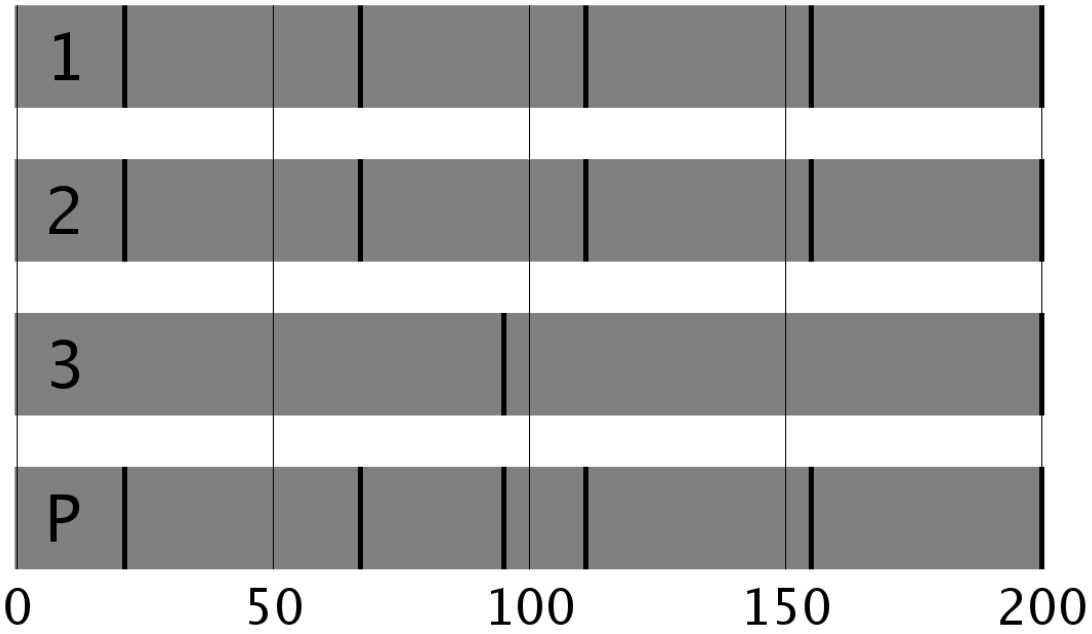


CoP = 25,000

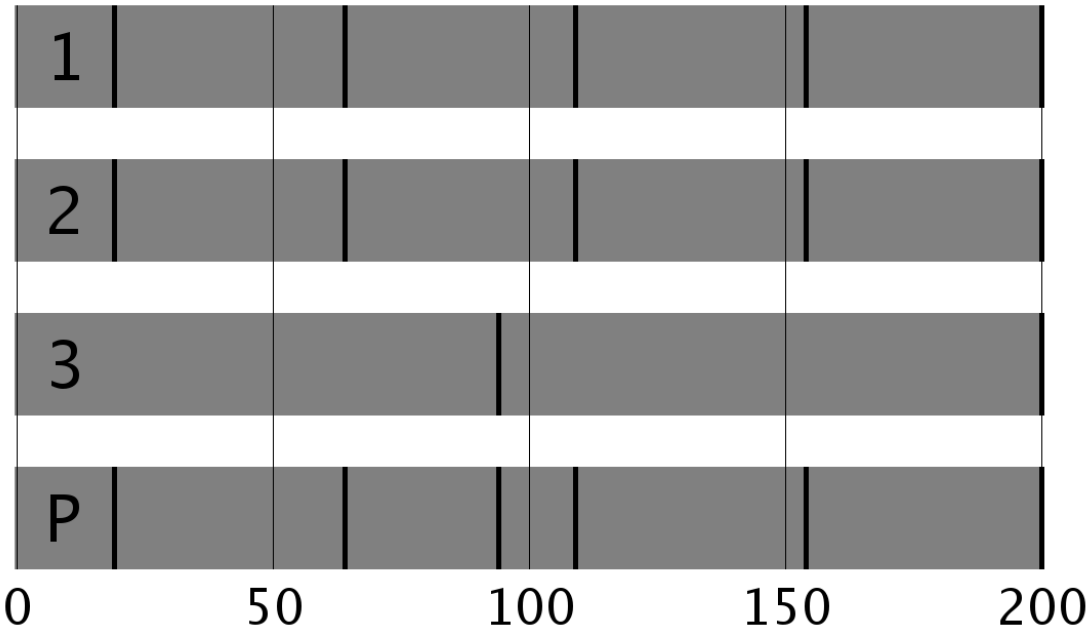
XXVI Resulting Schedules from Experimentation on Alternating Cost of Possession, Parameter Set 2



CoP = 80

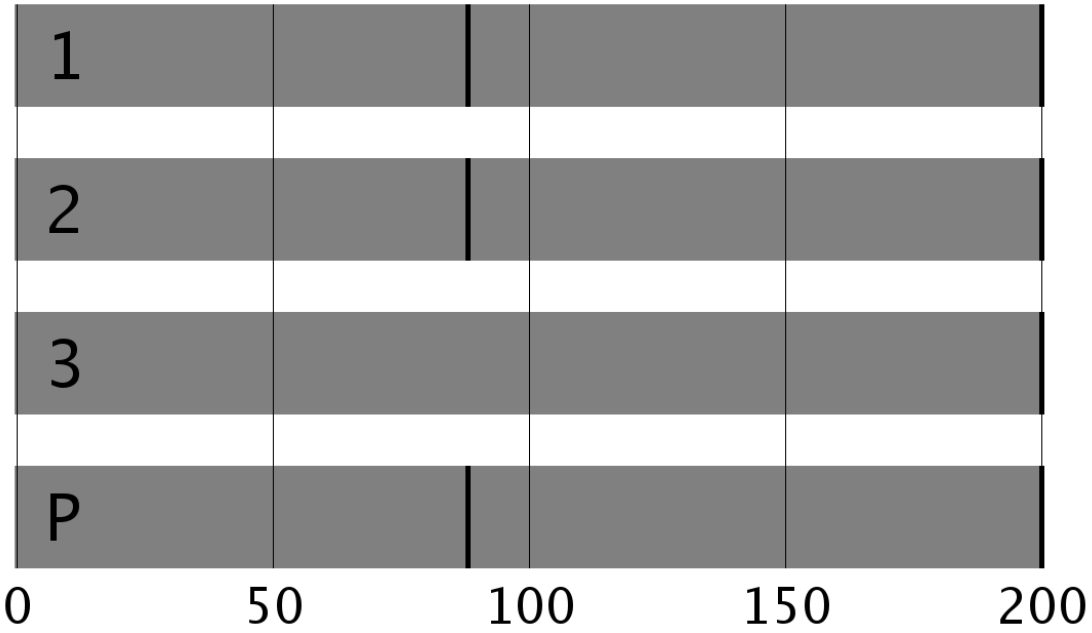


CoP = 50, 150, etc.

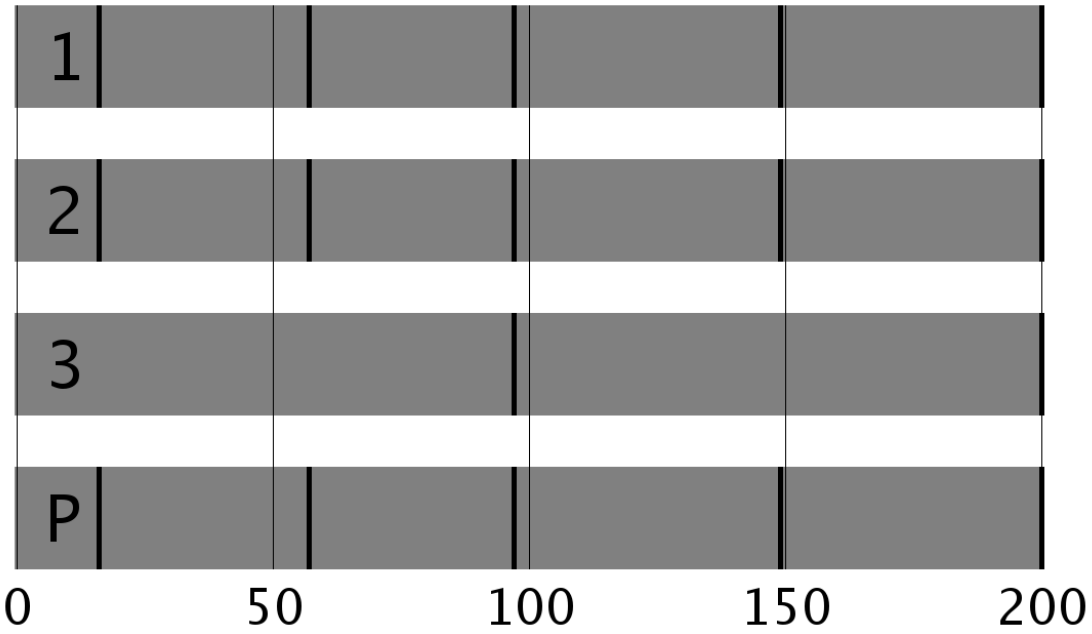


CoP = 50, 50, 50, 50, 150, etc.

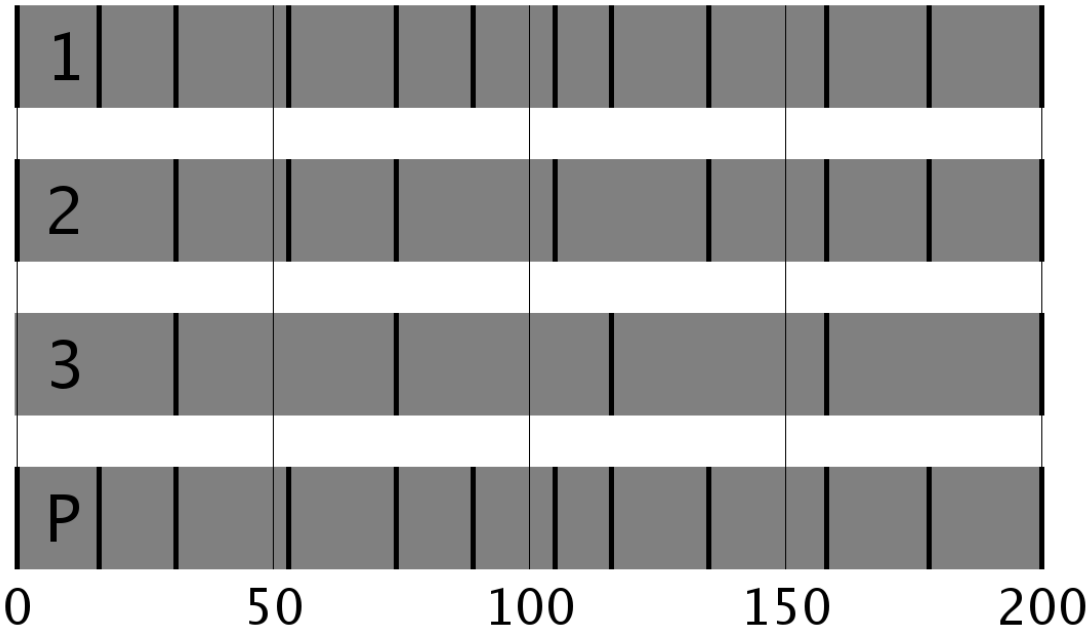
XXVII Resulting Schedules from Experimentation on Wear Out Shape Parameter, Parameter Set 2



d is half of normal

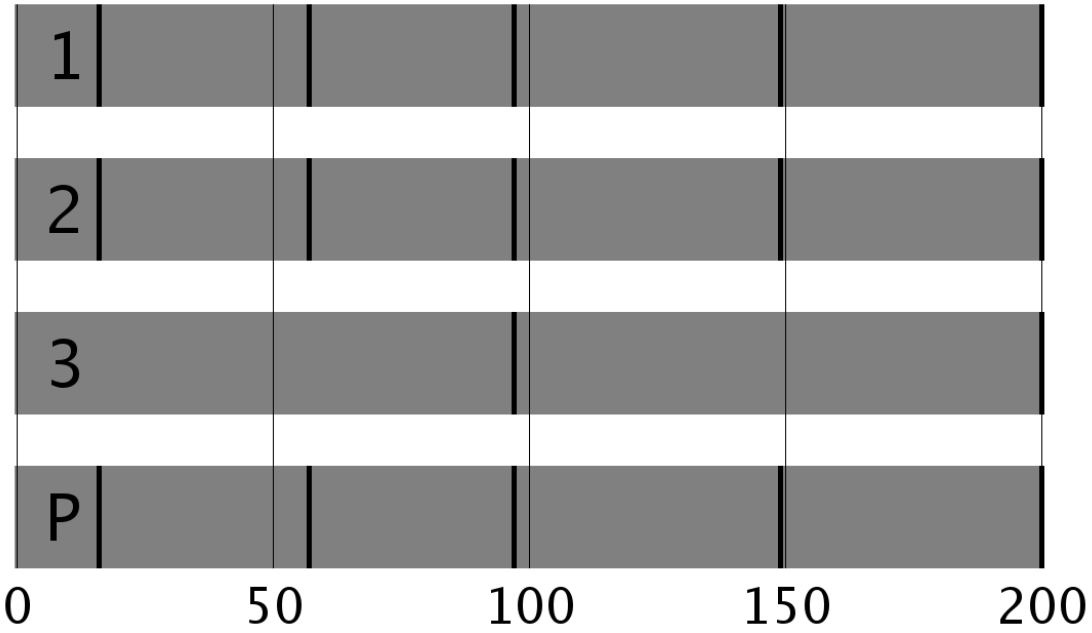


d is normal

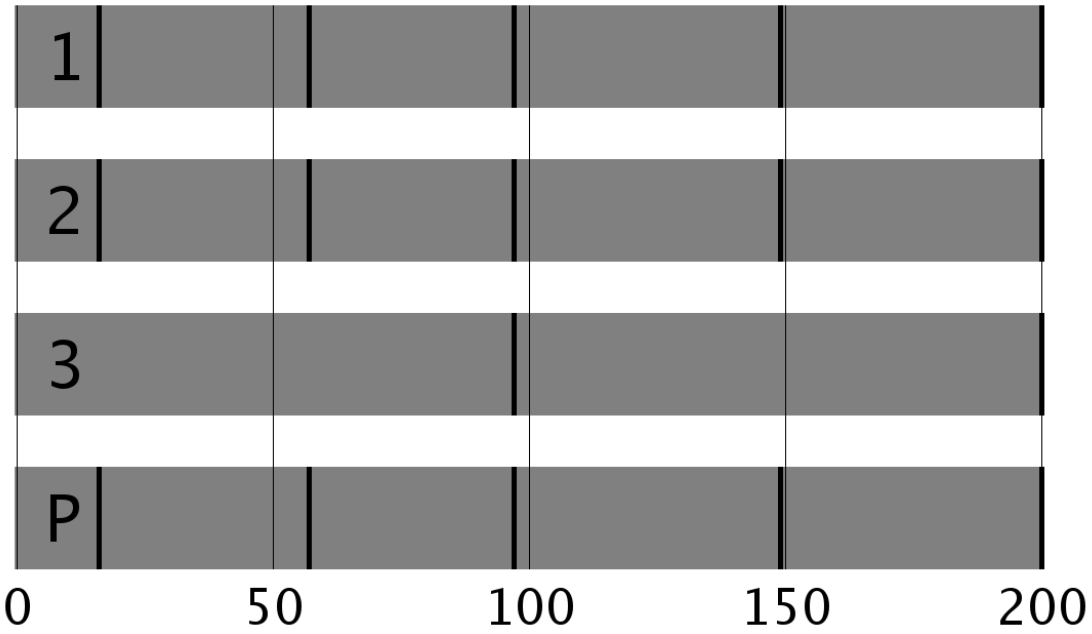


d is double of normal

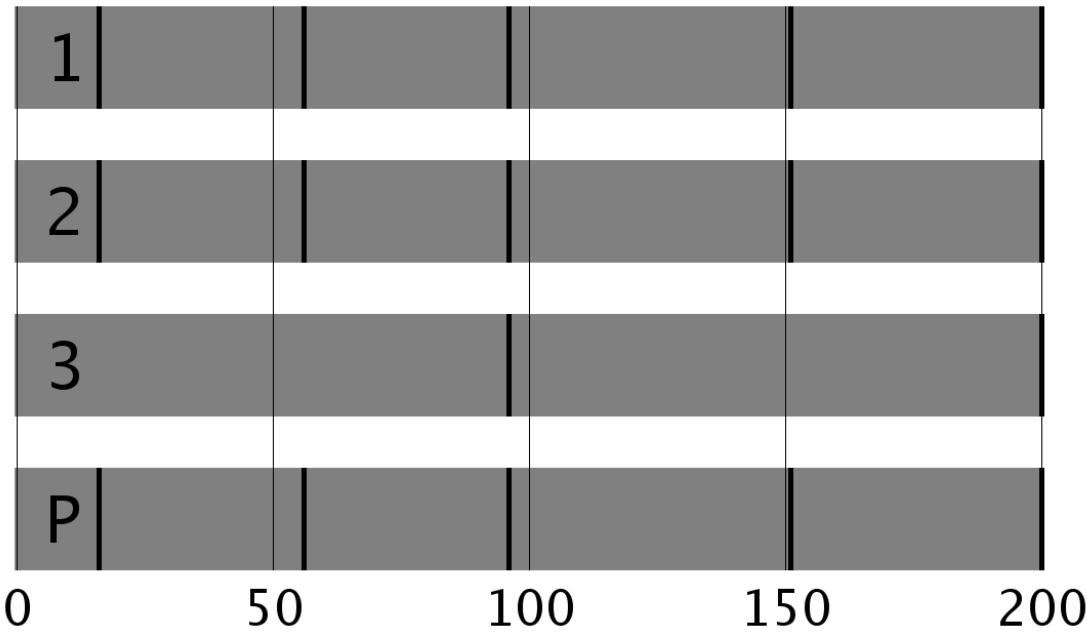
XXVIII Resulting Schedules from Experimentation on Optimality Gap, Parameter Set 2



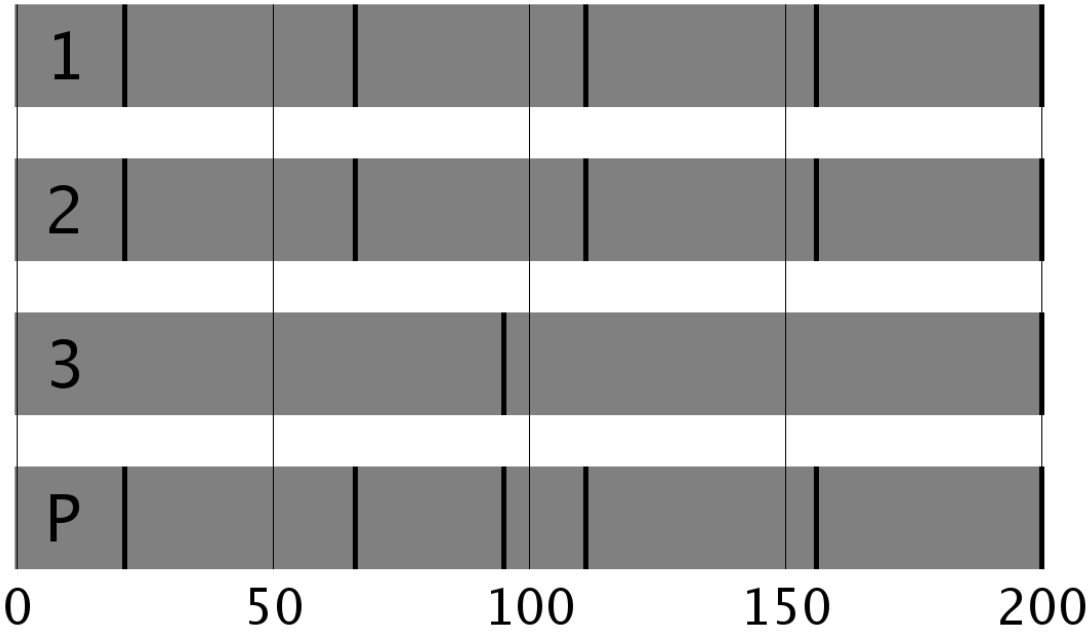
Optimality gap = 0%



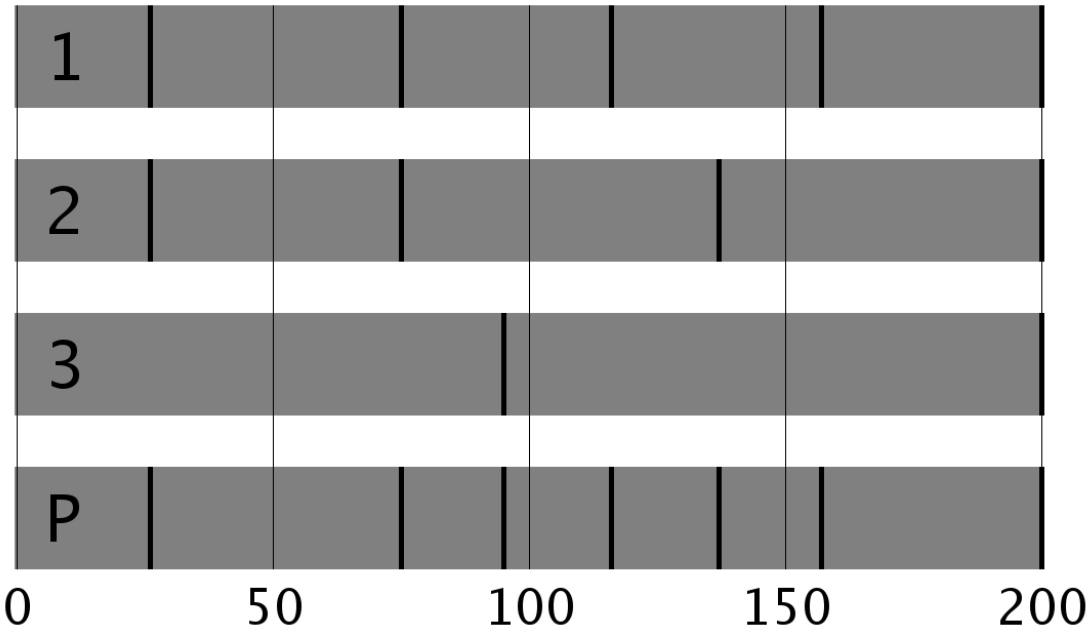
Optimality gap = 0.1%



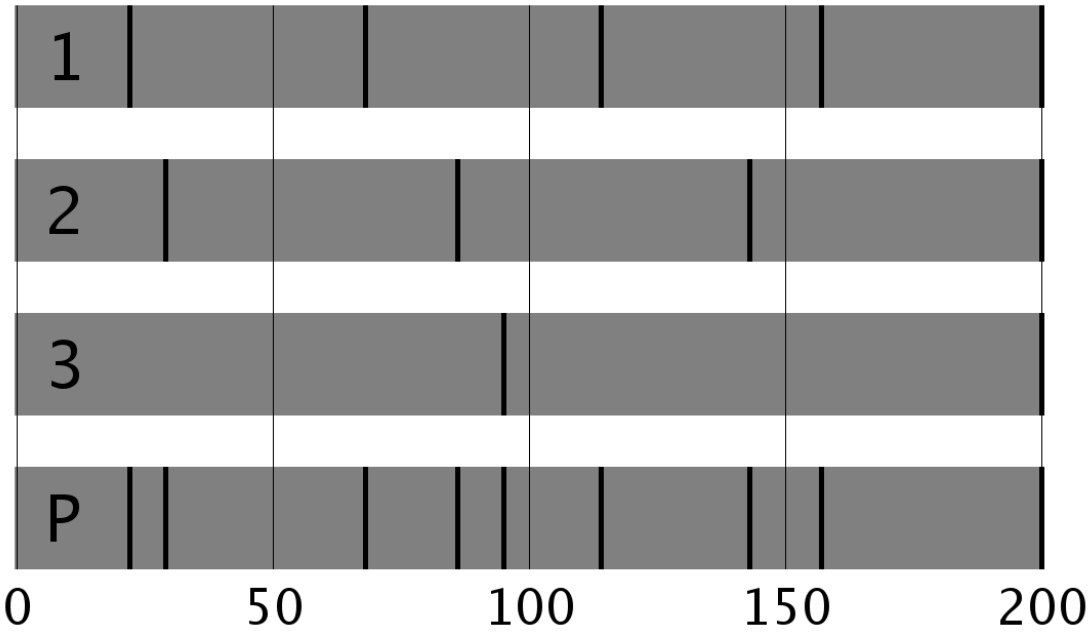
Optimality gap = 0.3%



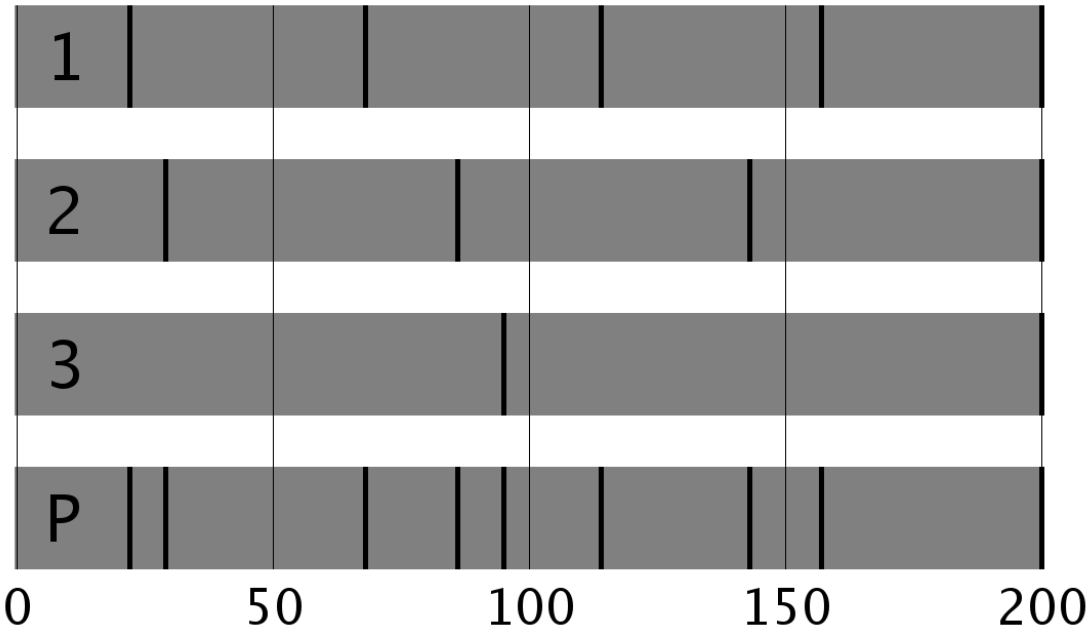
Optimality gap = 1%



Optimality gap = 3%

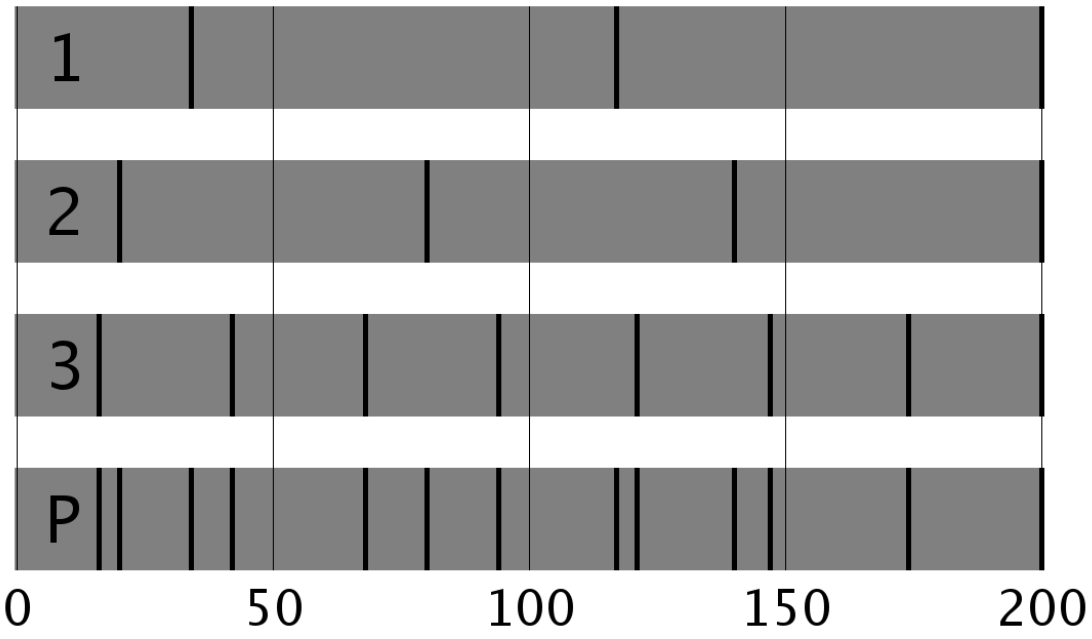


Optimality gap = 10%

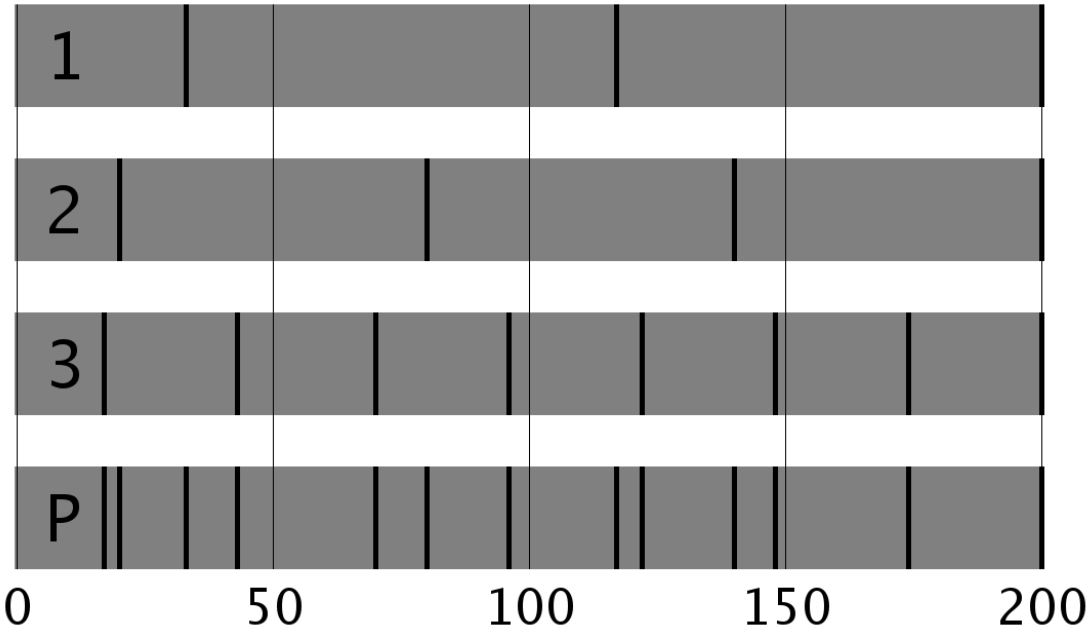


Optimality gap = 30%

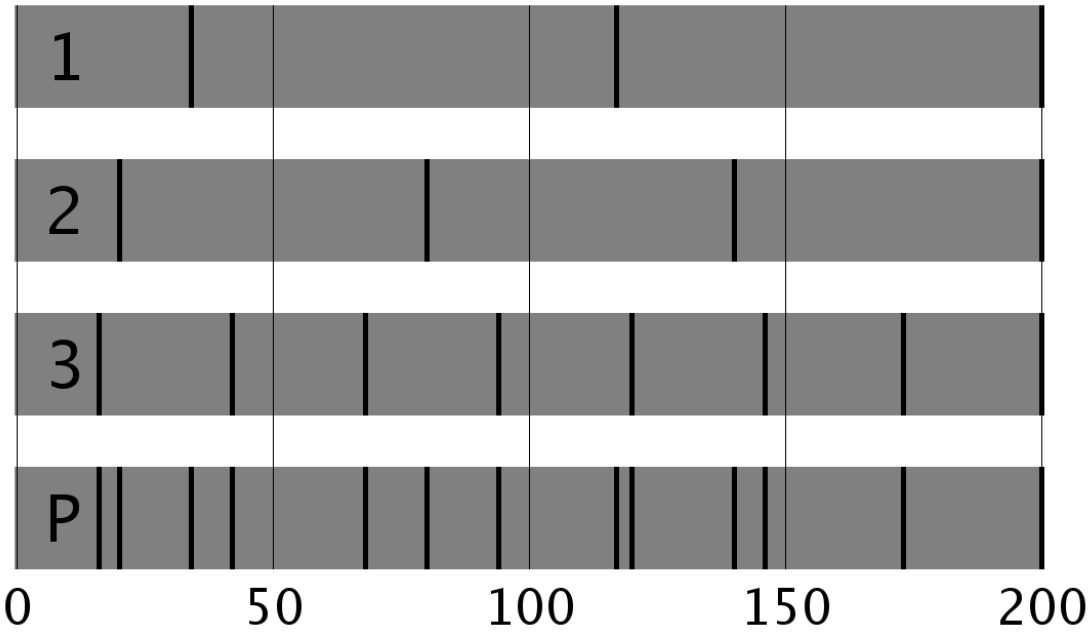
XXIX Resulting Schedules from Experimentation on Cost of Possession, Parameter Set 3



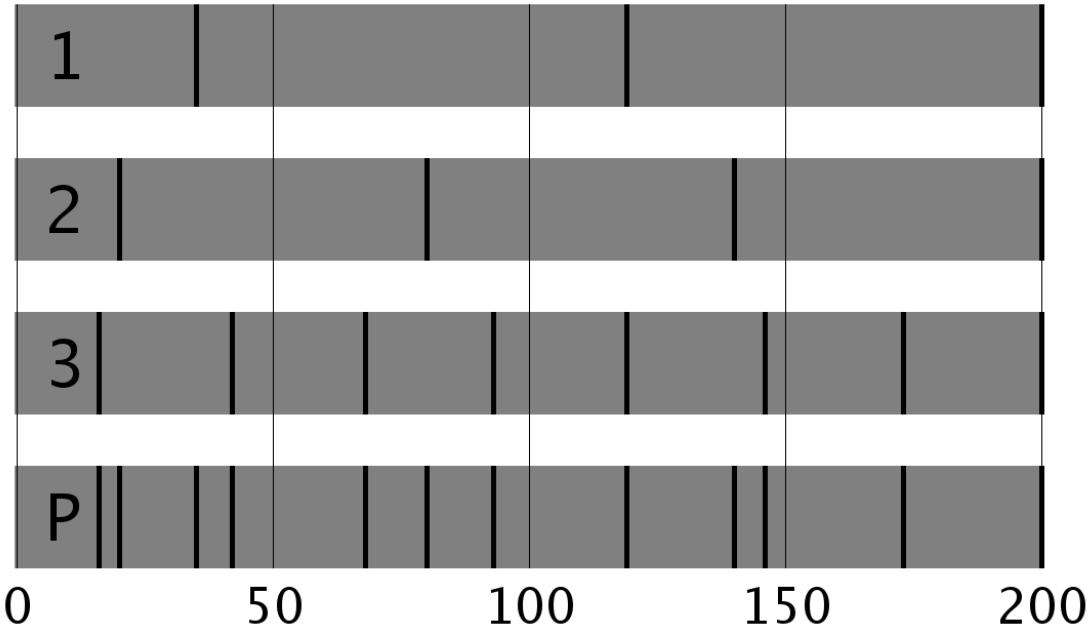
CoP = 0.25



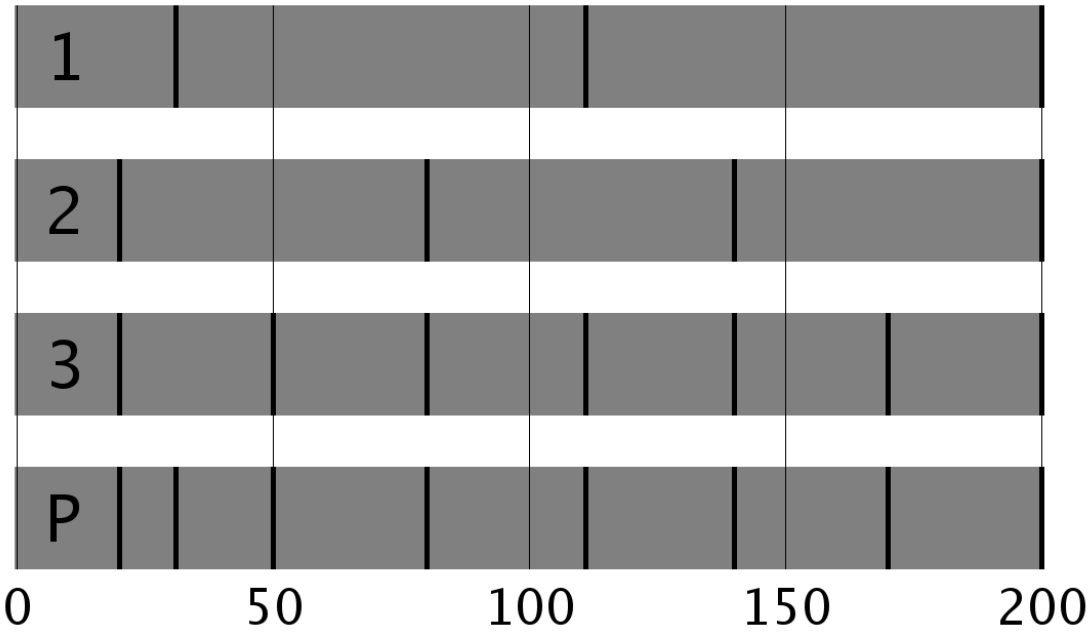
CoP = 0.8



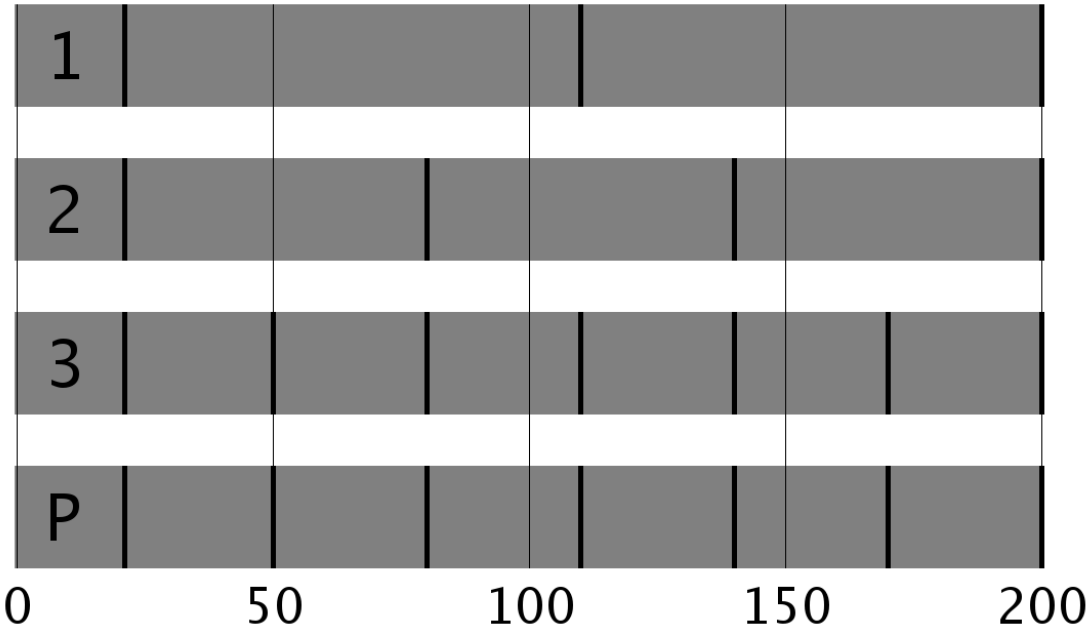
CoP = 2.5



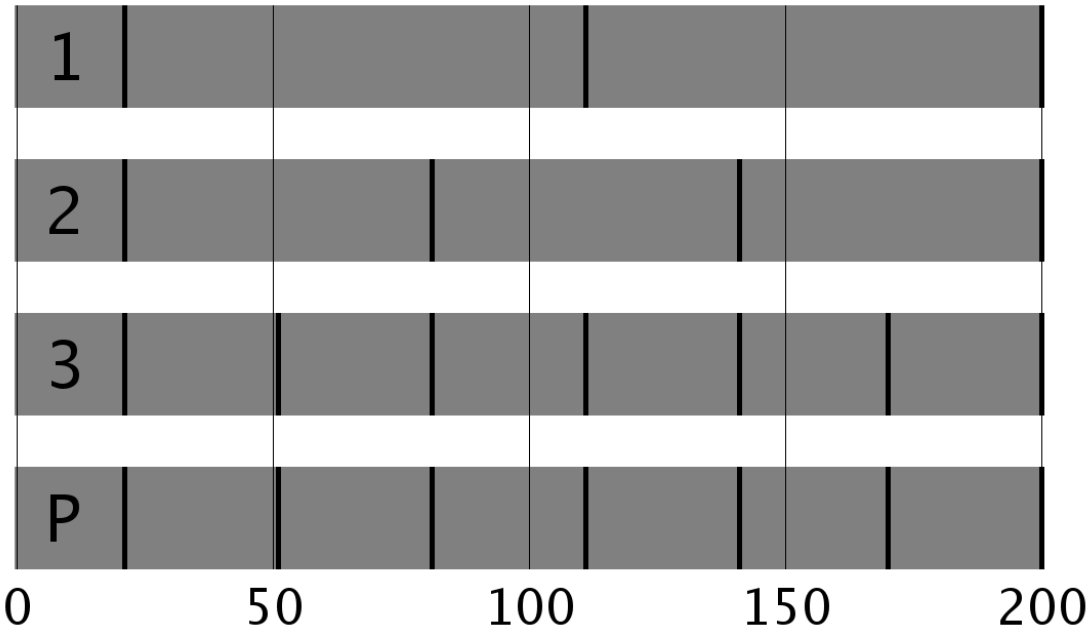
CoP = 8



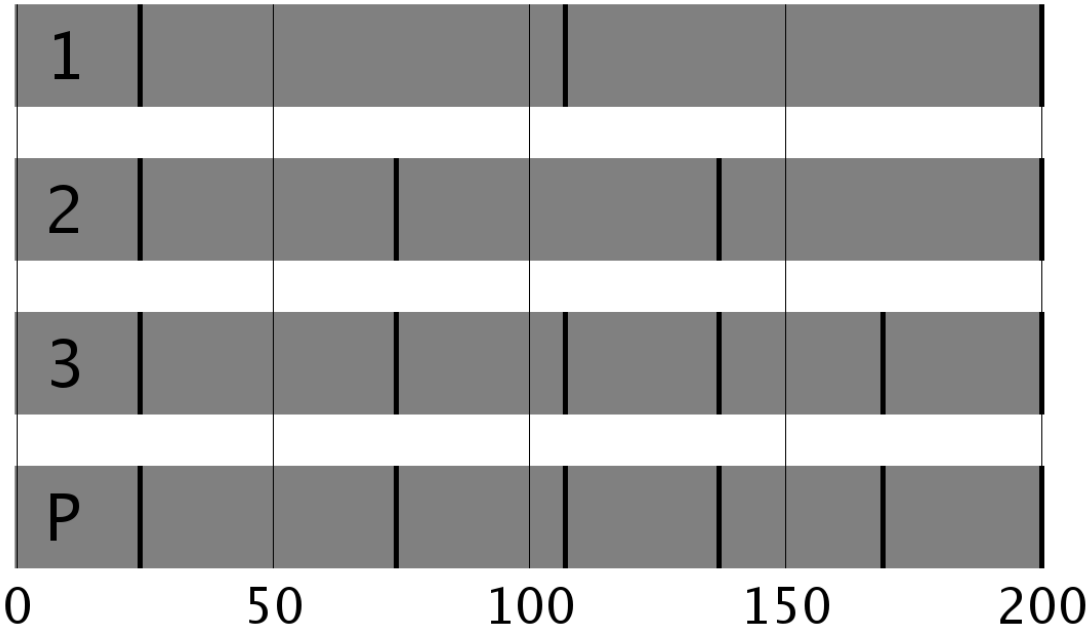
CoP = 25



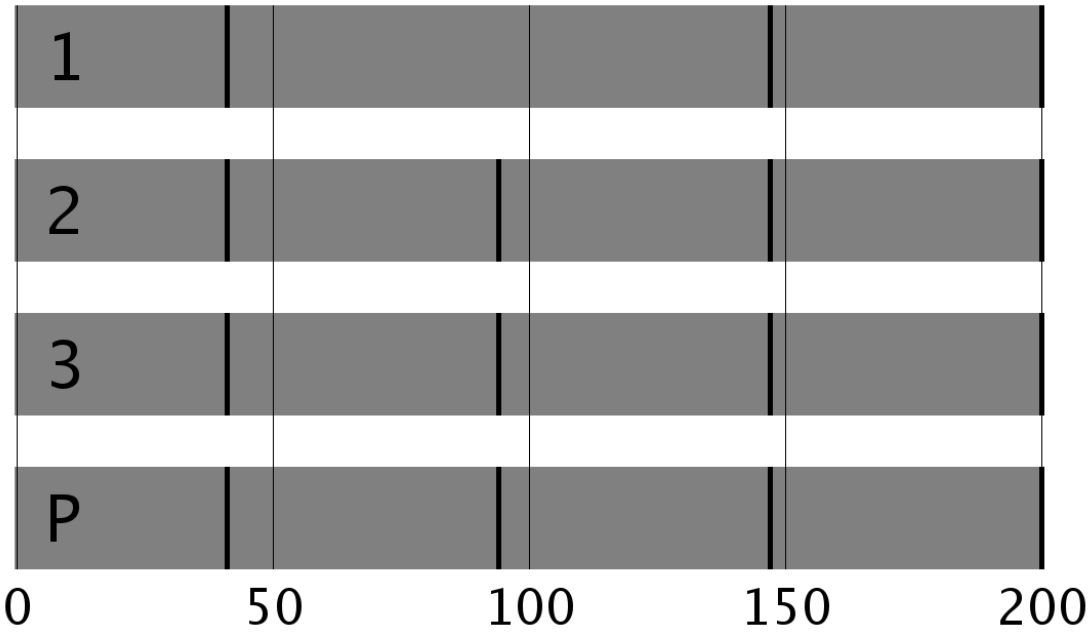
CoP = 80



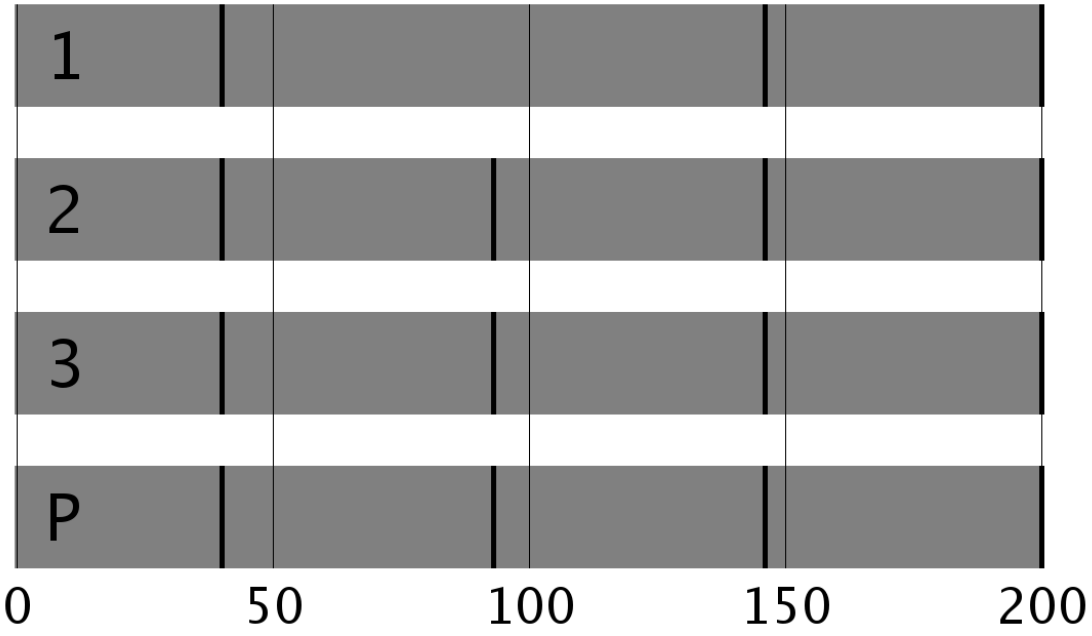
CoP = 250



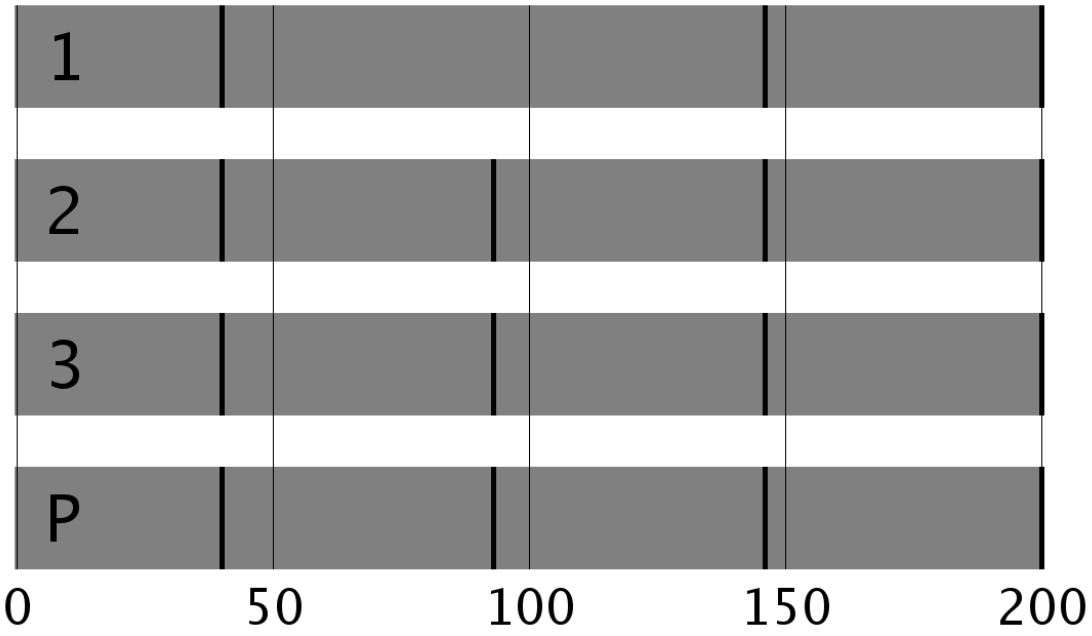
CoP = 800



CoP = 2,500

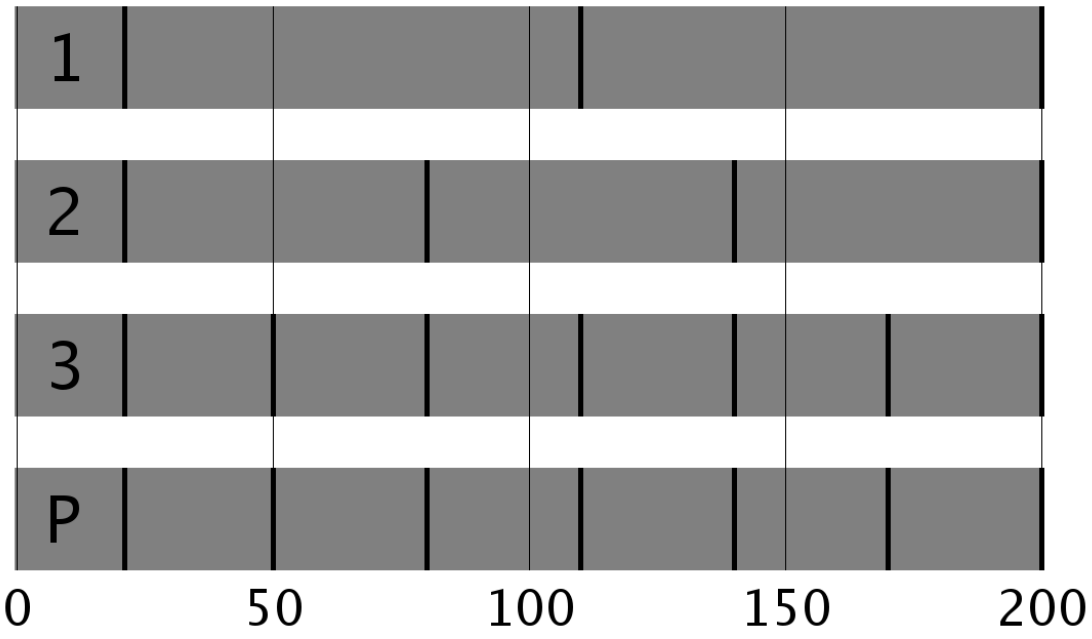


CoP = 8,000

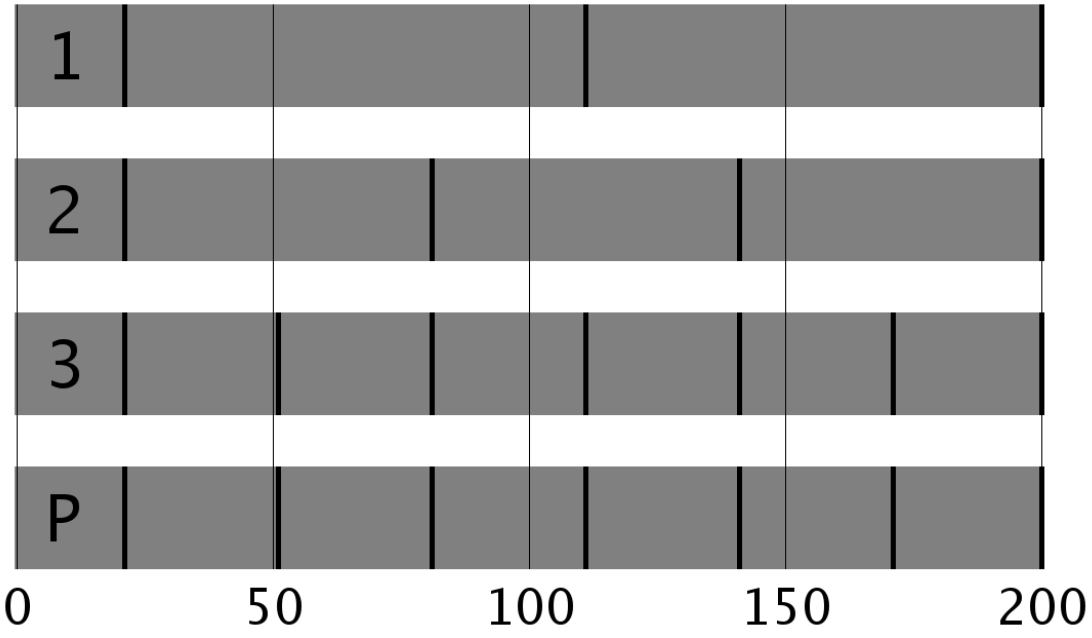


CoP = 25,000

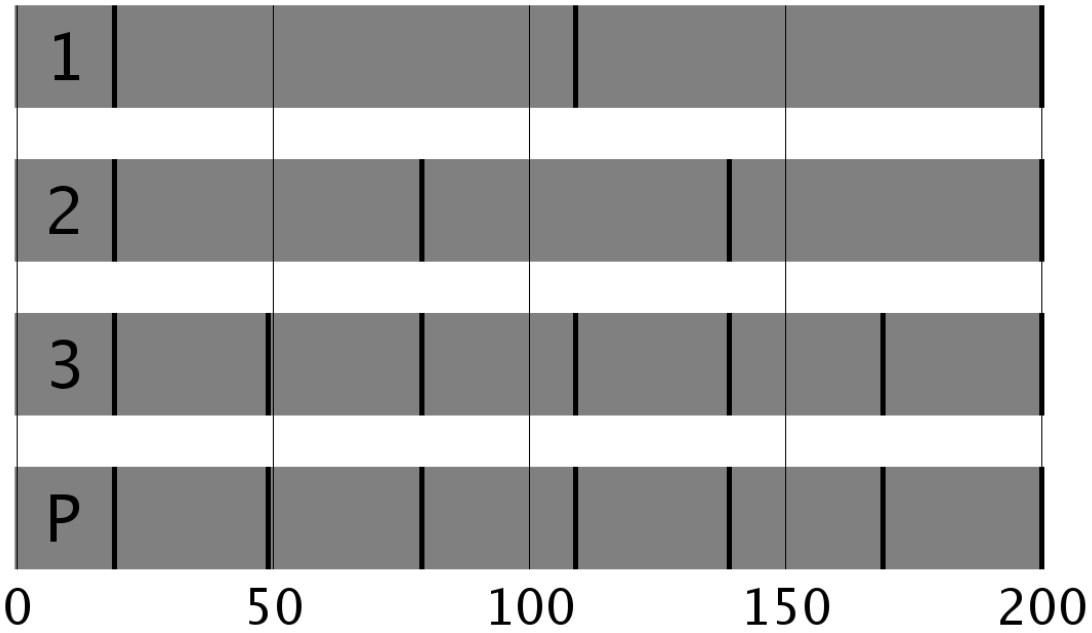
XXX Resulting Schedules from Experimentation on Alternating Cost of Possession, Parameter Set 3



CoP = 80

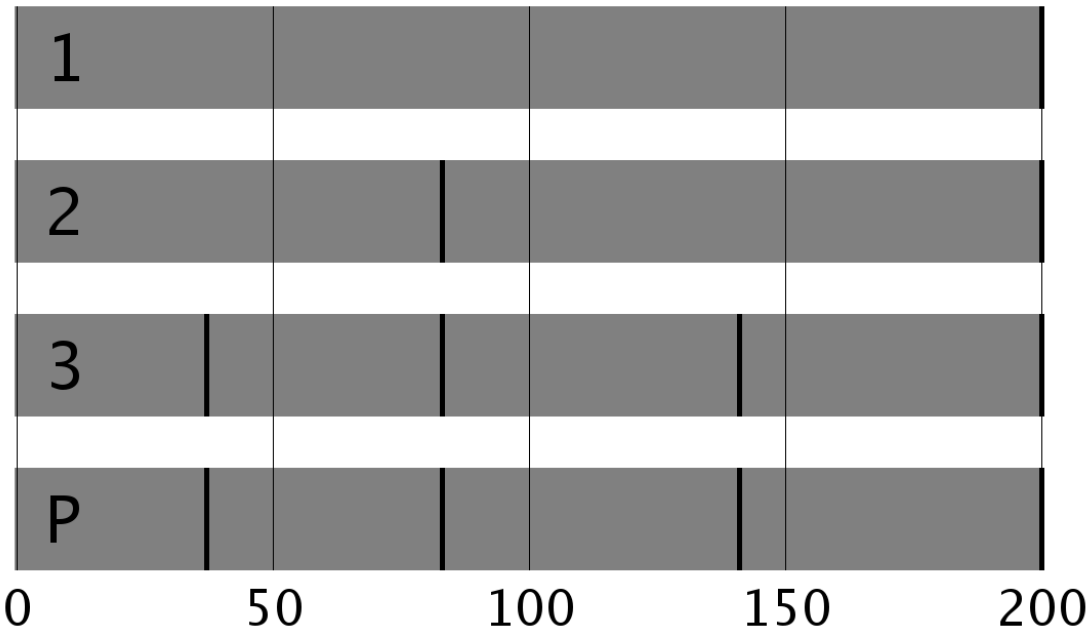


CoP = 50, 150, etc.

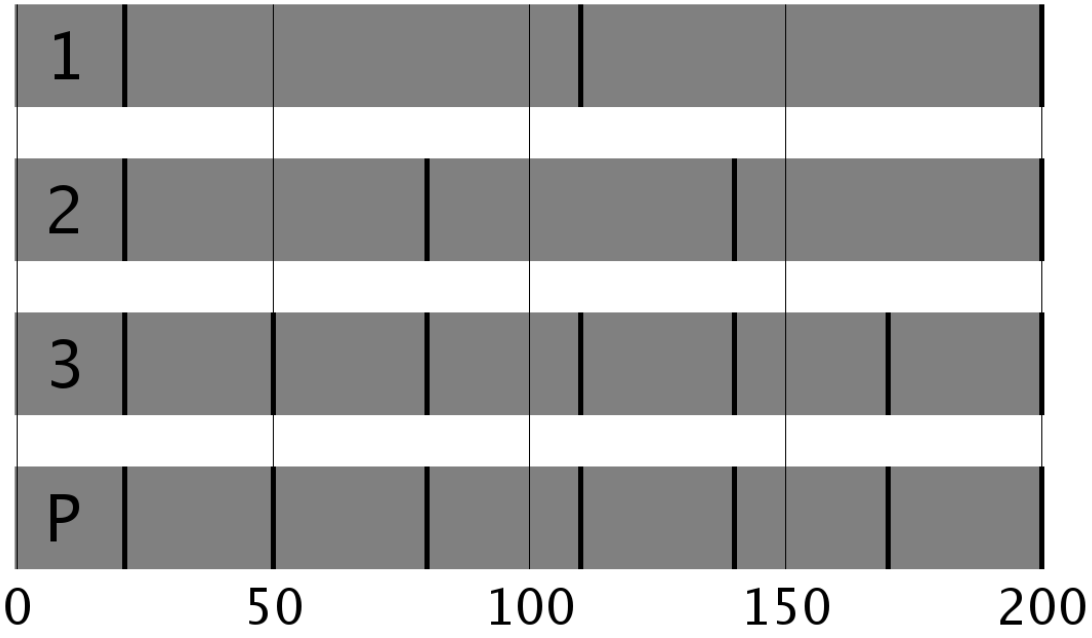


CoP = 50, 50, 50, 50, 150, etc.

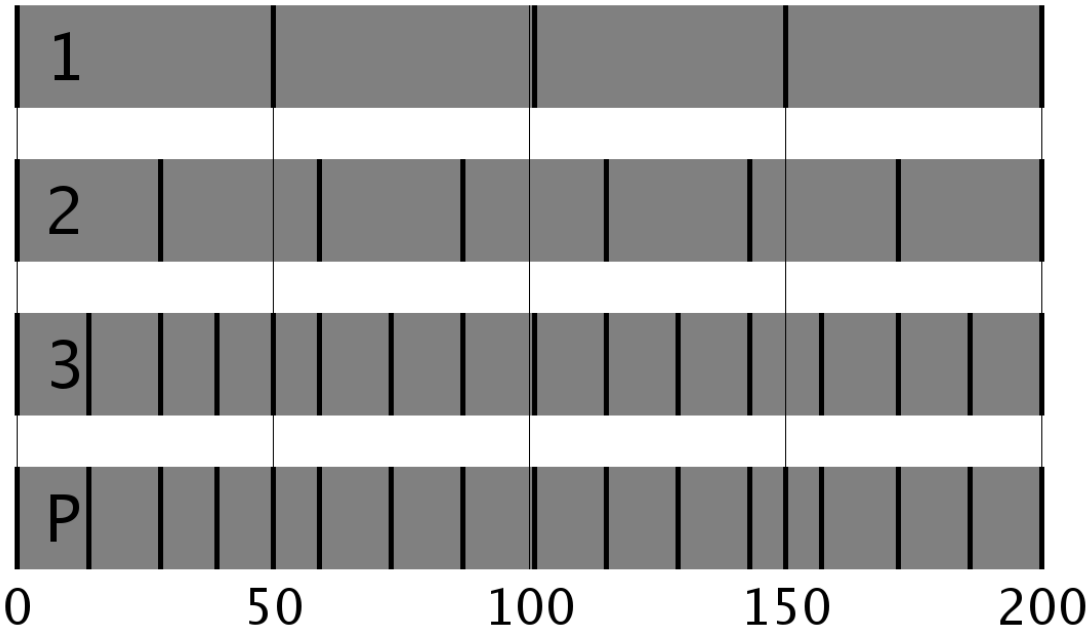
XXXI Resulting Schedules from Experimentation on Wear Out Shape Parameter, Parameter Set 3



d is half of normal

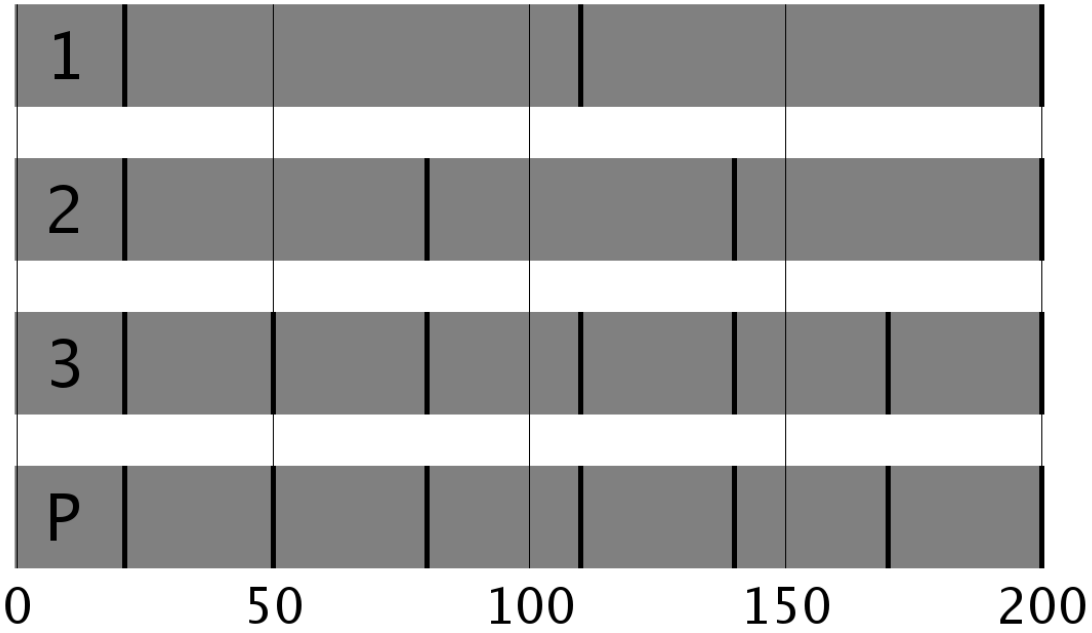


d is normal

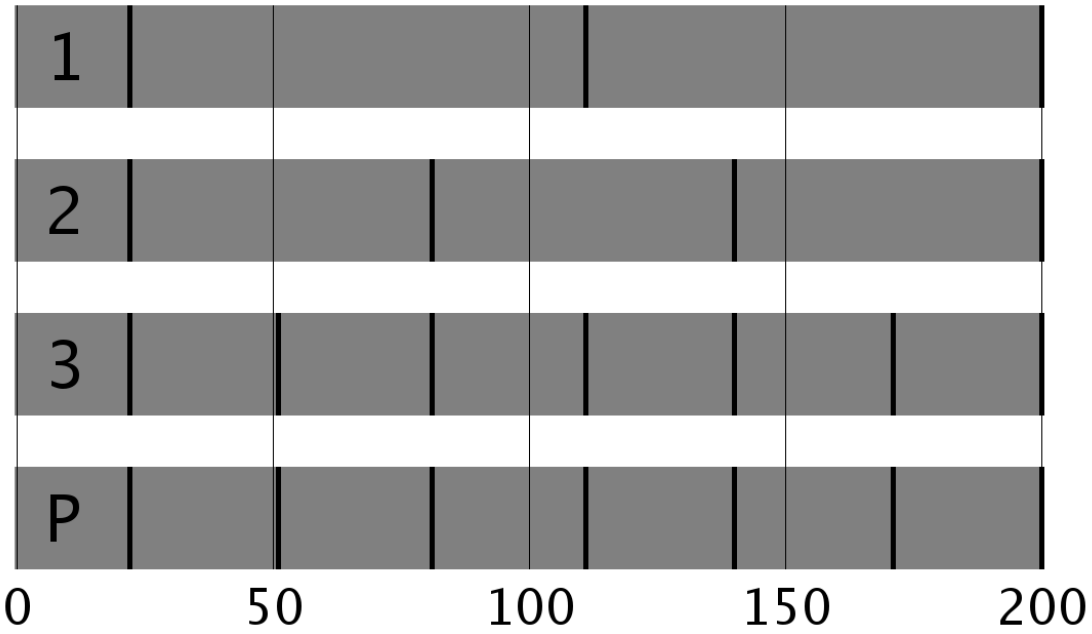


d is double of normal

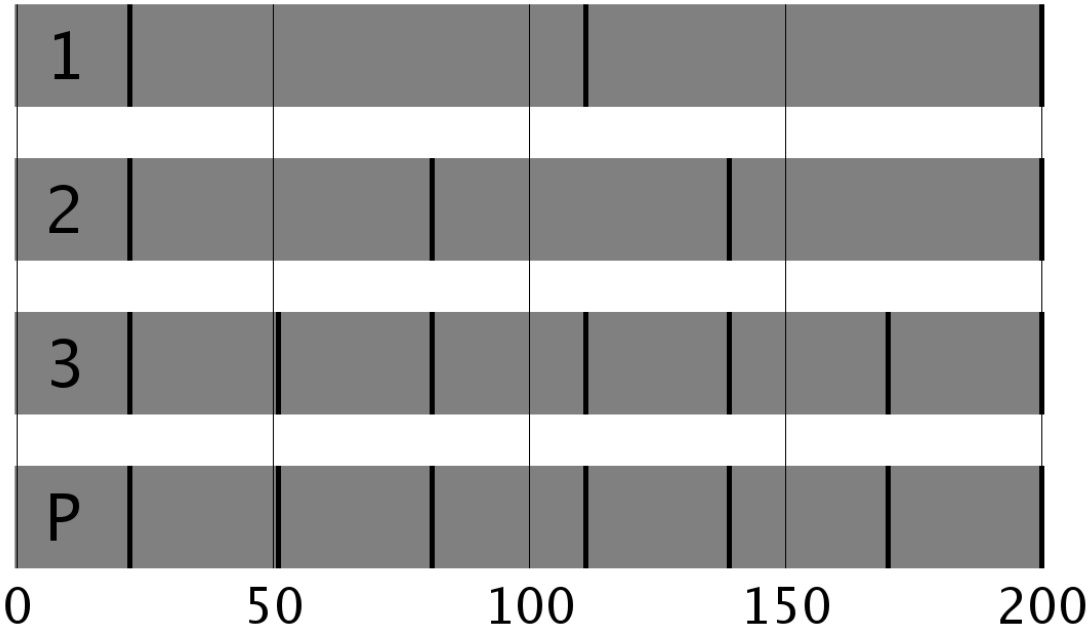
XXXII Resulting Schedules from Experimentation on Optimality Gap, Parameter Set 3



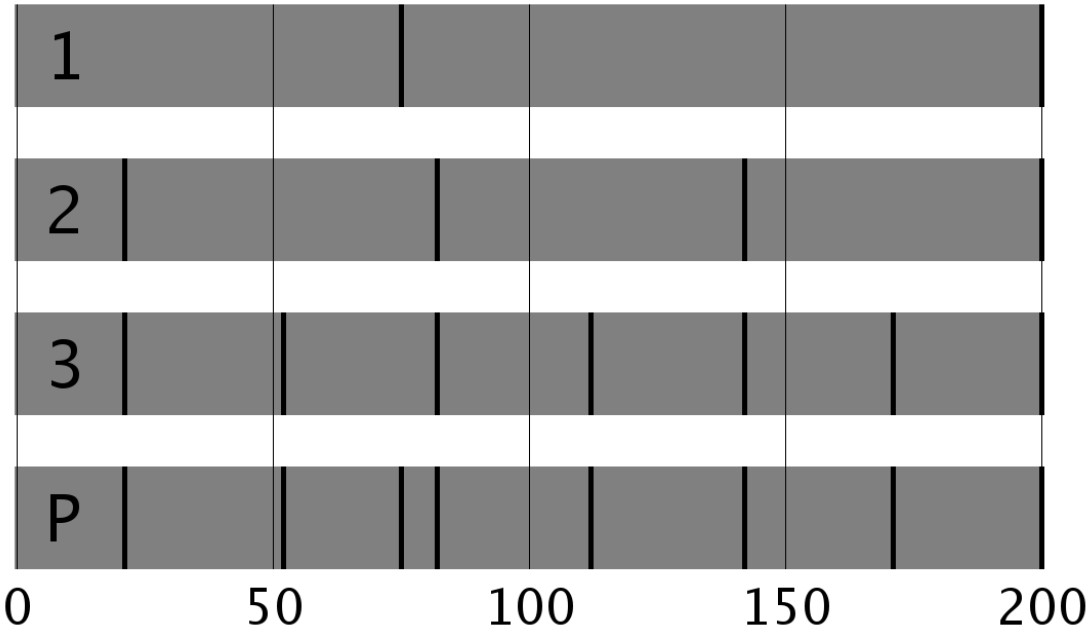
Optimality gap = 0



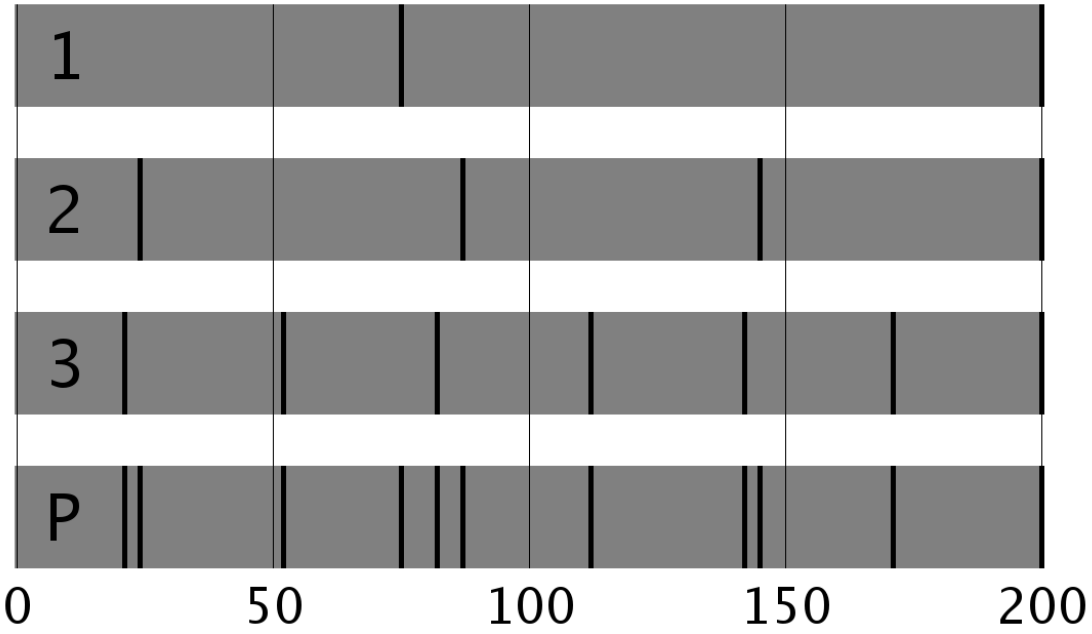
Optimality gap = 0.001



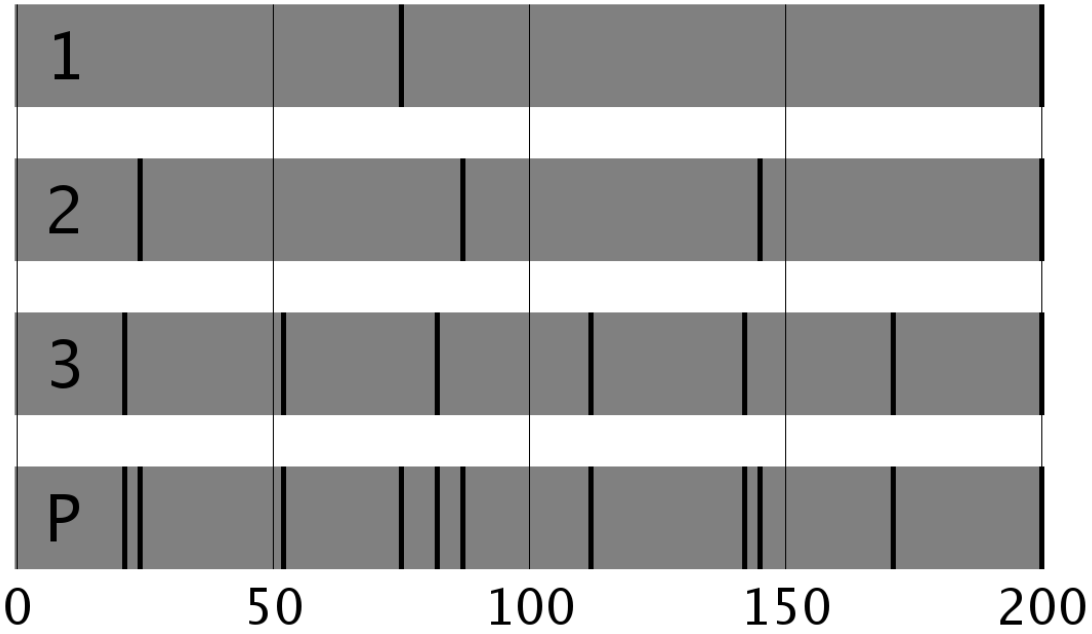
Optimality gap = 0.003



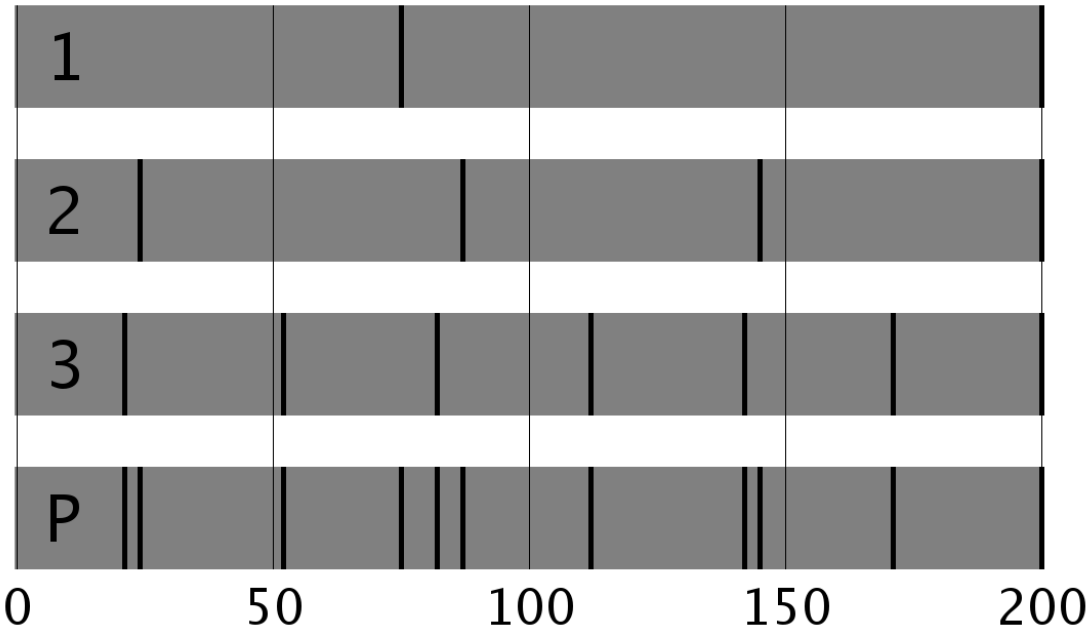
Optimality gap = 0.01



Optimality gap = 0.03



Optimality gap = 0.1



Optimality gap = 0.3

XXXIII Method for Applying the Optimization Model

The data consists of individual maintenance records. Each line in the spreadsheet is a maintenance operation. For each maintenance operation, several properties are available, including the object to which it is applied, the date on which it was executed.

1. The lines are ordered by object, then by date of execution.
2. Lines that do not contain the required data are excluded.
3. The lines are classified as either preventive or corrective maintenance. In this case, only the lines with the most common maintenance code for each object (ba.ar.05 for switches and ba.0006 for insulated joints) are classified as preventive. All other maintenance operations are considered to be corrective.
4. For each occurrence of preventive maintenance, the time between the repair of the component and the last time regular maintenance was performed, is determined.
5. These data points are divided into bins. The bins are chosen such that irregularities are mostly smoothed out, while enough detail remains to enable the fitting of a distribution. For the two examples in the case study, a bin size of 50 days was chosen.
6. If available, the time since the previous maintenance operation is calculated.
7. Not all maintenance intervals are equal. To compensate for the reduced occurrence of longer maintenance intervals, the proportion of components not having received maintenance for each time is determined. The failure count of each bin is then divided over this number to get an adjusted failure count. That number decreases, as maintenance reduces the number of components associated with each consecutive bin. If for example half the components in the dataset are maintained within the year, then failures that occur one year after the last maintenance has taken place, should be weighed with a factor of two.
8. For each bin, the corrected failure rate is determined by dividing the number of failures by the correction factor related to that bin.
9. A failure rate function is formulated. It is a function of the statistical parameters.
10. Initial values for the statistical parameters are chosen. This can be done manually as the initial values only serve as a starting point for the optimization in the next step.
11. Through optimization (for example using the Analysis ToolPak in Microsoft Excel), the parameters are adjusted such that the sum of squared differences between the estimated and the actual number of failures is minimized.
12. The parameters are checked for feasibility. If the constraint in equation (3.10) is violated, the optimization in step 11 should be redone with this constraint explicitly defined.
13. For the estimation of usable parameters, data of sufficient quality is needed. This means that the failure rate function is fitted in a way that explains a substantial amount of the variance. A post-hoc check is performed to prove that this is the case.

The explained variance, expressed in a percentage, is calculated with formula (XXXIII.1).

$$explained\ variance = \left(1 - \frac{\sum_i \left(X_i - \widehat{X}_i\right)^2}{\sum_i (X_i - \bar{X})^2}\right) \times 100\% \qquad \text{(XXXIII.1)}$$

with:

X_i: The observed number of failures in bin i

$\widehat{X}_i$: The predicted number of failures in bin i

$\bar{X}$: The average number of failures over all bins