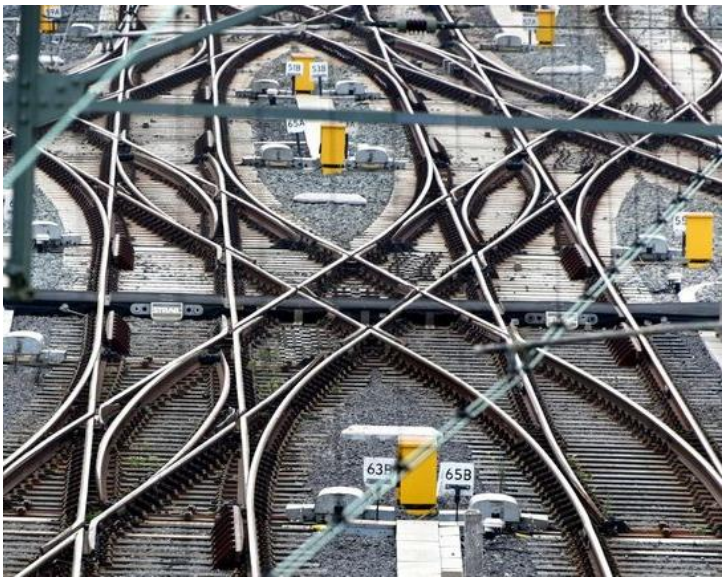

REDUCING THE PASSENGER HINDRANCE ON SNOWY WINTER DAYS BY PRIORITIZING SWITCH FAILURE REPAIRS

ProRail B.V.

NICK SLIGTING
n.g.sligting@student.utwente.nl
s1596977

INDUSTRIAL ENGINEERING & MANAGEMENT
Bachelor thesis
4 April 2020

PRORAIL SUPERVISOR
Saskia Wevers

UNIVERSITY OF TWENTE SUPERVISORS
Dr. E. Topan (1st chair)
Dr. I. Seyran Topan (2nd chair)

UNIVERSITY OF TWENTE
7522NB Enschede
Drienerlolaan 5

PRORAIL B.V.
3527KV Utrecht
Admiraal Helfrichlaan 1

Preface

Before you lies my bachelor thesis: “Reducing passenger hindrance on winter days through prioritizing infra-failure repairs”, which concludes my Bachelor’s programme in Industrial Engineering & Management at the University of Twente. The study has been performed at asset management department of ProRail B.V.

I would like to thank everyone at ProRail that has helped me during my time within the company and especially my Supervisor Saskia Wevers. I’ve learned a lot during my time at ProRail, not only about the subject matter of the assignment but also about the railway operation as a whole. It has given me a newfound admiration and understanding of all the effort that goes into keeping the railway system operating safe and smoothly.

Furthermore I would like to thank my university supervisor Dr. Engin Topan, Whose feedback and insights have been a great help along the way.

Finally I would like to thank my family and my friends for their continuous support throughout the years.

Nick Sligting,
Enschede, April 2020

Management summary

In extreme cases of snowy winter days the number of switch failures increases so drastically that the contractors do not know in what order to perform repairs. This increase of failures also leads to greatly increased levels of passenger hindrance as trains are delayed or have to be cancelled completely. The aim of this study is to develop a way for ProRail to determine an optimal way of ordering these repairs that they could potentially use to give the contractors directions that minimize the passenger hindrance.

In this study we measure the total passenger hindrance over a day by multiplying the sum of the number of trains that have to be cancelled on that day as a result of all switch failures by 30 minutes. This stems from the fact that on average, the next train that a passenger can take to his or her destination departs 30 minutes later than the cancelled train.

The machine scheduling problem

Each contract area has a single contractor that is responsible for all the maintenance and repair within this area and each contractor has at any given time, a single crew available to perform repairs.

Therefore this can be described as a single machine job scheduling problem, in which the machine is the contractors crew and the jobs are the switch failure repairs. The objective of minimizing the total weighted completion time lines up perfectly with our goal of minimizing the passenger hindrance if we take the number of trains cancelled per hour as result of the failure as weight. As for any failure the crew has to first travel to the location of the failure, the length of which of course depends on the crew's current location, the problem the characteristic of sequence dependent setup times. Formally notated this makes it a $1 \parallel S_{jk} \parallel \sum w_j C_j$ problem.

The performed Literature research has shown us, that 9 out of 11 observed articles aim to solve similar machine scheduling problems by using a WSPT based heuristic. The WSPT heuristic, which Pinedo (2008) proves is optimal for $1 \parallel \sum w_j C_j$ problems therefore serves as a starting point for our approach. As we're dealing with stochastic repairs times however, we have to use the WSEPT heuristic instead. The difference is that you then work with an expected repair time E(x) rather than a deterministic one. The weighted shortest expected processing time heuristic orders the jobs (j) by decreasing order of weight over expected processing time. $\frac{\mathrm{w}_j}{\mathrm{E}(\mathrm{x}_j)}$

A different approach would be to look at the problem from a routing perspective. We then say that we always want to go the failure that is closest to our current location next. To do so we can use the Nearest neighbor algorithm which constructs a order by always selecting the job with the lowest required traveling time from its current location, to be scheduled next. TTj

Proposed solution

We propose solving the $1 \parallel S_{jk} \parallel \sum w_j C_j$ by substituting the sequence dependent setup times with the nearest neighbor algorithm and then solving it as a WSEPT heuristic. We do so by including the travelling time in the processing time. The processing time then consists of the traveling time and the expected repair time. This results in a greedy constructive heuristic that essentially balances between the nearest neighbor algorithm and WSEPT heuristic. It constructs a order by selecting the job with the highest ratio of weight over the sum of the expected repair time and the travelling time between the current location and the locations of the failures, to be scheduled next. $\frac{\mathrm{w}_j}{\mathrm{E}(\mathrm{x}_j)+\mathrm{TTj}}$

Testing the solution

We test the performance of the alternatives and their influence on the passenger hindrance by performing Monte Carlo simulations. We test for two actual historic days, and four randomly generated days. Two of the random days have roughly the same number of failures as the observed historic days and two have strongly increased numbers of failures.

Results

The simulations results show us that for both the historic days as well as the random days with roughly the same number of failures the weight based (WSEPT) approach performs almost identical as our proposed heuristic. They both perform significantly better than the average random order and even more so than the order in which the repairs were actually performed. On the two days with the increased number of failures however, the proposed heuristic performs a lot better than the simple weight based heuristic. Notably, in none of the simulation runs has any of the alternatives nor the random order resulted in a lower total passenger hindrance than the proposed heuristic.

Conclusion

We conclude that it is indeed possible to reduce the passenger hindrance significantly on these extreme cases of snowy winter days by optimizing the order in which repairs are performed. The greedy constructive heuristic that we propose should very closely approximate the optimal order in the sense that is finds the order that leads to the least amount of total passenger hindrance. The tool that we've created can be used by ProRail employees to very easily find this optimal order of repairs. However the big question remains wether the contractors will actually in practice ask for ProRail's advice when it comes to the order of performing repairs which limits the practical use of the tool and the results of this study.

Recommendations

Due to the very small chance of the situation in which ProRail can really influence the order of repairs ever happening, we recommend ProRail not to put to much effort in the prioritization of switch failures repairs. However, as it should require relatively little effort to expand the tool to cover the entire nation rather than just the handshake area, we would recommend ProRail to do so. Because the results show that the proposed heuristic that the tool applies can significantly reduce the passenger hindrance. Therefore it can't hurt to have the tool laying around incase the situation in which it can be applied does occur.

Table of Contents

Glossary

Infra-failure	Failure of a infrastructural piece of the railway like switches, signal or overhead cables
Infra-availability	The extent to which the infrastructural parts of the railway are available for use by the operators
(Railway) Operators	Here NS and other companies that drive passenger trains over the dutch railway system.
Prio 2	Failures that require urgent repair are marked prio 2 within ProRail.
Prio 5	Failures that do not require urgent repair and can for example just be repair during the night are marked prio 5 within ProRail.
OCCR	Operationeel controle centrum rail, which is the the operational control centre for the railway system located in Utrecht.
PAB	The prestatie analyse bureau or performance analysis bureau is a department within ProRail that measures all kinds of performances.
BO	The backoffice is an assembly of departments that together are responsible for, among other things, the intake and monitoring of failures.
NS	Primary railway passenger transporter in the netherlands
PGO (contracts)	Performance based contract form in which contractors are responsible for maintaining a certain level of performance.
Prioritizing / ordering / scheduling repairs	Here used interchangeably as: setting the order in which repairs are to be performed

1. Context

ProRail works together with NS and contractors at the *Operationeel Controle Centrum Rail* (OCCR) in order to keep train traffic running smooth and safe. Part of their operation is the intake and management of infra-failures. E.g. Defect switches or overhead lines. After ProRail is notified of a failure they record the failure in their systems and alert the responsible contractor. The contractor is then responsible for repairing the failure within a certain time-frame depending on the urgency of the failure, which is determined by ProRail. The chance of more than one failure, that requires urgent repair, occurring simultaneously within a single contract area is normally so low that the contractors are more than capable of determining in what order to perform repair and maintenance. Snowfall however drastically increases the number of switch failures to the extent that multiple failures occur in the same contract area simultaneously. This can lead to a situation in which the contractors do not know what order of repairs would be preferred / optimal. In this research we look at these situations and we assume that the contractors then ask ProRail what order of repairs they would prefer. Currently ProRail has no real answer to this, other than one based on the gut feeling of whoever is working the phone at that moment, which is presumably very loosely based on the list of critical switches. Which is a list that essentially ranks switches based on how many trains have to be cancelled when a failure occurs.

We examine individual winter days on which this situation could occur. We're talking about extreme cases that really don't occur often and are highly dependent of the weather. Some years there are none of these days and other years there are up to three. Examples of days on which this could apply are 11 December 2017 and 3 February 2012. The increase in passenger hindrance on these days is so noticeable that news sites run liveblogs during the day to report about it as well as it being a hot topic for news articles the following days. See for example: Chaos op spoor door sneeuw (2012, February 3), rtl nieuws.[1] In this study we focus on switch failures that require urgent repair, which are marked *prio 2* within ProRail.

2. Problem analysis

In this chapter we described the problem at hand, the research aim and the research strategy.

2.1. Problem identification

The problem that ProRail faces is that there is too much passenger hindrance on snowy winter days. This hindrance arises out of the drastic increase in the number of delays and cancelations as well as the increased durations of delays. These are on one side caused by variety of problems at railway operators (NS), unrelated to ProRail. On the other side they are caused by the drastic decrease of infra availability. I.e. the availability of railway that can be driven on safely. The decrease in infra availability has two causes: the drastic increase in the number of infra-failures and the limited to no prioritization in infra-failure repairs. Lastly the drastic increase in the number of infra-failures is the result of the winter weather, e.g. the freezing of switches rendering them inoperable and thus stuck in a position.

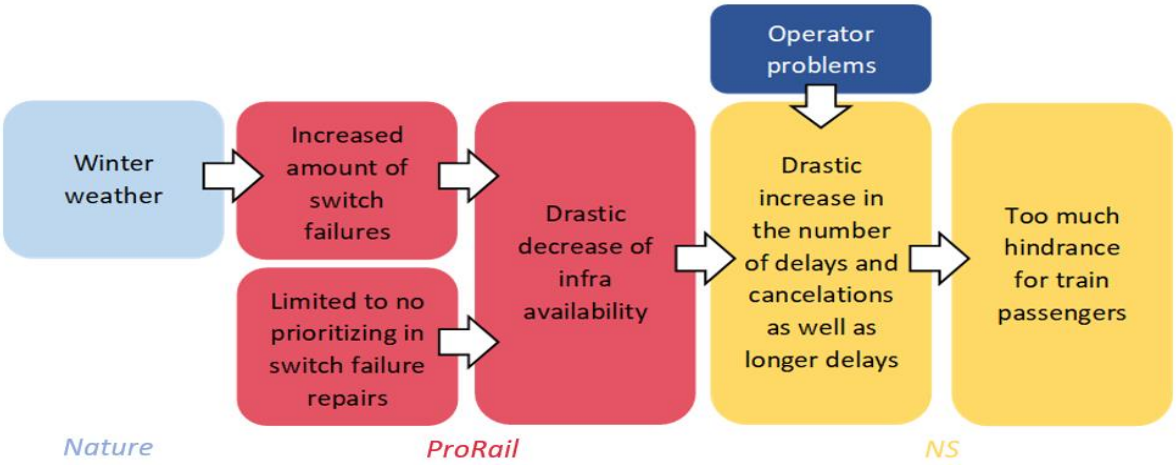

Figure 1: Problem cluster

2.1.1. Common goal

As can be seen in the problem cluster, *Too much passenger hindrance* falls in the yellow NS / operator section. NS is by far the biggest railway operator in the Netherlands, therefore references to NS in this report also extend to the other smaller operators. It falls in this section because strictly speaking ProRail's influence and responsibility ends at the infra availability. Their task is to make sure that the rail network is available and safe. So technically beyond the infra availability it is NS' responsibility to make sure that the trains depart and arrive at the right time. In practice however ProRail and NS share the common goal of having the least possible amount of passenger hindrance. The decreased infra availability plays a huge part in causing passenger hindrance, but it is definitely not the only cause. As even with a 100% availability of the rail network there can still be quite a lot of passenger hindrance. This due to various external factors, almost all of which are NS / operator related problems. (see Operator problems Section 2.1.2). However the infra availability is essential as a railway that is not available can simply not be driven on, on-time or otherwise. The infra availability works as a limiting factor on the entirety of the problem cluster. Simply put, if only 80% of the rail network is available, NS can only drive their trains hindrance free on less than 80% of the network even if they do everything perfectly. So in order to achieve this common goal of minimizing passenger hindrance, both ProRail and NS need to keep their commitments. their commitments.

2.1.2. Operator problems

In the problem cluster *Operator problems* is listed as a cause of the drastic increase in the number of delays and cancelations as well as the increased durations of delays. This signals all external problems, problems that are not related to or caused by ProRail. These are grouped into one entity because the list of possible problems is too extensive and are not fully clear. They are not further specified because they are not problems related to ProRail. Think for example of a defective train or personnel problems. E.g. NS not being able to get their personnel to the right location on time resulting in a train being delayed or cancelled. For a train can simply not drive without a train driver regardless of the state of the railway.

2.1.3. Core problem

Now that the involved problems and their relations are clear, the core problem can be selected. Heerkens and van Winden (2012) state that to find the core problem you must look at the problems in the problem cluster that are the furthest away and thus have no causes themselves. The winter weather itself is obviously not something that ProRail can change or even influence in any shape or way. Therefore it cannot be the core problem.[2]

Moving one step further we have the increased number of failures as a result of the winter weather and the limited to no prioritizing of failure repairs. One of these should be selected as the core problem. The increase in failures is already being both actively combatted as well as being widely researched within ProRail. Whereas the prioritization of failure repairs has yet to be looked at. This together with the notion that the increased number of failures for the most part relate to other fields than that of industrial engineering, like physics or mechanical engineering, make the limited prioritization in failure repairs the most important problem to tackle. This means that the core problem of this research is the limited to no prioritizing of infra-failure repairs.

2.1.4. Measuring the problem

As mentioned earlier we're dealing with extreme cases that occur in single days. Therefore, we want to measure the total passenger hindrance over a day. We measure this by summing up the number of trains that had to be cancelled because of the switch failures and then counting 30 minutes for each cancelled train. The 30 minutes are based on the average inter-departure times as a train between arbitrary points A and B departs every 30 minutes on average. How this is calculated in the simulation model is explained in section 6.

2.1.5. Norm and reality

As this research is largely of exploratory nature, exact realities had not been measured yet. Note that realities is plural as we are observing individual single day cases and each case therefore has its own reality. A reality being the result of the problem measurement in the current situation. I.e. a quantification of the observed problem. In our case this is the total passenger hindrance as it occurred on the observed historic day. As explained earlier the common goal of ProRail and NS is to minimize passenger hindrance. So in a broad sense the norm is for there to be as little passenger hindrance as possible. This means that any decrease in the total passenger hindrance would be an improvement. However, after deliberation with my ProRail supervisors we decided that anything higher than a 10% decrease in passenger hindrance can be considered a big enough improvement. Therefore, we say that a proposed solution meets the norm if it can result in a decrease in passenger hindrance of 10% or more. The exact realities were measured during the research and are used for comparison in the results section. (section 7) This is done by running the simulation setup as explained in section 6 for the order in which repairs were performed on the actual historic days.

2.2. Scope

In this study we look at single snowy winter days on which the weather causes drastically increased numbers of switch failures. We look specifically at the former *Handshake area* which is a triangular area between Schiphol, Amsterdam and Utrecht. The name Handshake area refers the handshake agreement between ProRail and contractors that used to go into effect in this area on the snowy winter days that are the subject of this study. Simply put this agreement basically meant the contractors would go to work immediately and the paperwork would be done at a later point in time, which allowed the functionality of the rail network to be recovered much faster. This area consists of four contract areas, which are areas in which a single contractor is responsible for all maintenance and repair work. In this study we assume that each contractor only has a single crew of repairmen available to perform repairs. This is something that may not apply to the entirety of the nation yet however it holds true for the contract areas in the handshake area. Also as all contract areas transition to the new contract form in the near future it will soon be the case for all areas.

2.3. Deliverables

The primary goal of this study is to deliver insight into the effect of scheduling failures on the passenger hindrance and the optimal way of scheduling them. The secondary goal is to create a tool that ProRail can use to easily determine the optimal sequence of repairs for each contract area. Subsequently the

deliverables are this report containing the insights this study has provided and the tool that we have created as described in section 8.

2.4. Research questions

To solve the problem several research questions needed to be answered:

1. How can we schedule switch failure repairs (optimally)?

We first determined possible ways of scheduling repairs. This included looking at literature as well as investigating what data was available to us.

2. How can we test schedules?

We then had to determine how we could test schedules and thereby possible solutions. This included setting up a test and analyzing input and output data, but also determining in what environment we would build the test, e.g. are we going to use something like plant simulation 13 or are we going to write everything from scratch in a programming language of our choice.

3. To what extent can we reduce passenger hindrance by changing the order in which repairs are performed?

After we had devised a way of testing our schedules we had to actually test them and see if and how much we can affect the passenger hindrance.

4. How can we implement our solution?

Finally we investigated what the best way of implementing our solution was. We decided that the best way would be to build a simple to use tool.

Question 1 is answered in chapter 4 and partially in chapter 5. Question 2 is answered in chapter 5 and 6. Question 3 is answered in chapter 7 and again in the conclusion in chapter 9. Finally question 4 is answered in chapter 8.

2.5. Limitations

In this study we're dealing with some limitations. First of all the 10 week time limit. As this study is performed as a bachelor assignment for Industrial engineer and management, we have to deal with the time limit that has been set for these. As a result not everything can be researched and not everything can be done in as much detail. Secondly we do not have access to passenger data. This data is held by NS rather than ProRail and there was no feasible way of obtaining this data, at least not given the time limit. This means that we cannot directly use any data regarding the number of passengers travelling between points like we originally wanted.

3. Research Design

Now that the Problem and research questions are clear, we will explain the research design.

3.1. Research subjects

The research subjects are the switches and failures thereof in the handshake area, because they are what the research is essentially about.

3.2. Key variables

The key variables that we used in this research and in the tool that we developed are:

- (*Total) passenger hindrance,* this is how we measure the problem and the performance of alternative solutions. It is measured by counting 30 minutes for every train that has to be cancelled due to switch failures on that day.

- *Failure type,* this is the state in which the switch failed. A switch can be stuck in one of three positions: to the left (LL), to the right (RL) or somewhere in the middle (T). In the first two cases trains can still drive over the switch in the respective direction the switch is stuck in, in the last case no train can drive over the switch irregardless of the direction it is headed to.

3.3. Theoretical perspective

The theoretical perspective of this study is mainly based on machine scheduling and for a minor part on routing. More precisely heuristics and algorithms used to solve single machine scheduling problems and routing problems. We take on this perspective because it fits best with the context and can be solved in the limited available time. It also allows us to easily build a tool that ProRail can use to implement the solution in practice.

4. Scheduling repairs

In this section we discuss possible solutions to the scheduling problem as well as the process and ideas behind them.

4.1. The machine scheduling problem

As explained in the theoretical perspective, we view the problem as a machine scheduling problem. The problem can be described as a collection of single machine scheduling problems, as each contract area has a separate single repair crew responsible for all repairs in that area. As these areas are independent of each other in regards to failure repairs, we solve each problem separately. The individual problems are formally noted as $1 \parallel S_{jk} \parallel \sum w_j C_j$ machine scheduling problems. In which the machine is the contractor's repair crew and a job (j) is the process of repairing a failure, I.e. restoring full functionality to the switch.

4.1.1. Objective

The objective is to minimize the total weighted completion time ($\sum w_j C_j$) as this directly represents the passenger hindrance. I.e. a hindrance factor (weight) multiplied by the time it takes to restore functionality (completion time). By minimizing the weighted completion time for each area individually we minimize the overall weighted completion time.

4.1.2. Characteristics

The notable characteristic of this problem are the sequence dependent setup times (S_{jk}). After all we're dealing with a repair crew that has to first travel to the location of the failure before they can start repairing the failure. Therefore the current location of the crew determines the time needed to travel to the failure. The time needed to travel to the location of failure represents the setup time require before processing the job. So therefore the setup times are dependent on the sequence in which the jobs are processed.

4.1.3. Weights

As weight we want to use some factor that represents the passenger hindrance that is occurred per time unit. Therefore we use the number of trains cancelled per hour as a result of a specific failure. Conveniently ProRail has already indexed this for all switches in the so called critical switch list. This number depends on the situation in which the switch has failed. I.e. is it stuck to the right hand side, to the left hand side, or somewhere in the middle. The worst case is when it's stuck somewhere in the middle as this means no train whatsoever can drive over the switch.

4.1.4. Approach

The $1 \parallel S_{jk} \parallel \sum w_j C_j$ scheduling problem is strongly NP-hard. (Pinedo, 2008). Therefore, we approach the problem a little differently than we would approach a regular $1 \parallel \sum w_j C_j$ scheduling problem. Rather than determining the full order at once, we formulate a greedy constructive heuristic which constructs an order step by step.

This allows us to omit the sequence dependent setup times as a characteristic as we can then just include the setup times in the processing time. The processing time of job j then becomes the sum of the time needed to repair switch j and the time needed to travel from the current location to the location of switch j. This adds an aspect of routing problems, as we now also regard the traveling time between points and hence the distance between them. We essentially balance between a nearest neighbor heuristic, (shortest travel time / distance) and a weighted shortest processing time heuristic. The specifics of this heuristic and how we came to it are described in the rest of this section.

4.2. Literature

We performed a systematic literature review to see how similar $1 \parallel S_{jk} \parallel \sum w_j C_j$ problems are solved in literature. The full literature review protocol can be found in appendix B. As these problems are strongly NP-hard we're looking for a heuristic and not an exact way of solving the problem as it is suspected that there are no polynomial-time algorithms capable of solving said problems.[3]

Table 1 displays the conceptual matrix constructed as a result of this literature review. It notes per article the heuristic(s) used, the specific characteristics that applied to the problem and the applicability of the solution to the problem at ProRail which is a combination of the complexity, the relevance and the proven efficiency. The applicability is rated 1 to 5, 1 being the lowest and 5 being the highest.

Article	Heuristic	Characteristics	Applicability
Moghaddam, A., Teghem, J., Tuyttens, D., Yalaoui, F., Amodeo, L.	*4 types of Implicit enumeration following a branching scheme*	*Bi-objective, rejection*	*medium-low (2)*
Lee, W.-C., Wang, W.-J., Shiau, Y.-R., Wu, C.-C.	*Branch-and-bound based on a combination of WSPT and EDD*	*Deteriorating jobs, multi-agent common resource*	*Low (1)*
Mosheiov, G., Sarig, A.	*Dynamic programming with a WSPT based heuristic*	*Due dates*	*High (5)*
Arroyo, J.E.C., Dos Santos Ottoni, R., De Paiva Oliveira, A.	*Multi-objective algorithm based on Variable neighbourhood search (VNS)*	*Multi-objective, Due windows, Sequence dependent setup times*	*Medium (3)*
Ángel-Bello, F., Álvarez, A., Pacheco, J., Martínez, I.	*GRASP with Tabu search*	*Preventive maintenance, Sequence dependent setup times*	*Low (1)*
Lin, M.-Y., Kuo, Y	*Mixed integer programming with a WSPT based heuristic*	*Sequence dependent setup times with job families*	*Medium-high (4)*
Wang, X.-Y., Wang, J.-J.	*WSPT*	*Deterioration and learning effects, past sequence dependent setup times*	*medium-low (2)*
Lee, W.-C.	*WSPT*	*Learning effects, past sequence dependent setup times*	*medium-low (2)*
Wang, J.-B., Li, J.-X.	*WSPT*	*Learning effects, past sequence dependent setup times*	*medium-low (2)*
Wang, J.-B., Wang, D., Wang, L.-Y., Lin, L., Yin, N., Wang, W.-W.	*WSPT*	*Exponential learning effects, past sequence dependent setup times*	*low (1)*

Table 1: conceptual matrix

Notable is that we haven't found a single document about a scheduling problem with the exact same characteristics as our problem at hand at ProRail. They either don't have all characteristics or have extra properties that change the nature of the scheduling problem. E.g. deteriorating or learning effects.

Almost all, I.e. nine out of eleven, articles based their method on the WSPT heuristic, often referred to as the WSPT-rule. This is in line with what we already knew about simplistic scheduling problems with the objective of minimizing the total weighted completion time. As the WSPT-rule is proven to result in an optimal solution for those more simplistic variants of the problem.[3] It therefore seems clear that we should also look to apply a WSPT-rule based heuristic to our scheduling problem.

4.3. The WSPT heuristic

We start by observing the $1 \parallel \sum w_j C_j$ problem. Pinedo (2008)[3] proves that the WSPT rule is optimal for $1 \parallel \sum w_j C_j$ problems under deterministic processing times. The WSPT or *Weighted Shortest Processing Time* rule dictates that jobs should be ordered in decreasing order of $\frac{w_j}{p_j}$. I.e. the weight of job j over the processing time of job j.

4.4. Stochastic vs deterministic processing times

In practice processing times are stochastic rather than deterministic, as they are subject to inherent randomness. This means that we will have to work witch expected processing times that follow a distribution rather than set processing times. Under stochastic processing times (X_j) WSPT converts to **WSEPT** *Weighted shortest expected processing time*. The objective becomes $E(\sum w_j C_j)$ and jobs are then ordered in decreasing ratio of $\frac{w_j}{E(x_j)}$. I.e. weight of job j over the expected processing time of job j. This WSEPT rule then minimizes the expected weighted total completion time. *(Pinedo, Scheduling, 2008, Springer Science+Business Media, p. 264)*

4.5. Routing aspect

As explained earlier we are dealing with sequence dependent setup times, as the contractor's crew has to travel to the location of the switch failure before he can start repair. In this section we look at this as a routing problem rather than a characteristic of a machine scheduling problem.

To do so, we take into account the traveling times between the repair crew's current location and the locations of the outstanding switch failures.

4.5.1. Minkowski distance

Rather than simply using the Euclidean distance between point A and B we use the Minkowski distance. This give us a better representation of the actual distance one needs to travel between two point in (semi) urban areas than the Euclidean distance.

Shahid, R., Bertazzon, S., Knudtson, M.L., Ghali, W.A. (2009)[4] show that using the Minkowski method with a factor of 1.31 gives the best approximation of the actual road distance between two points in (semi) urban areas. This is calculated as follows:

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

In which $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_n$ are the two points between which you want to calculate the distance and *p* is the factor that we set to 1.31 based on Shadid, R., et al. (2009).

When examining what failure to schedule first we look at the distance between the area's starting location (section 5.4) and all outstanding switch failures. For all subsequent scheduled failures we look at the distances between the last scheduled switch and all outstanding switch failures.

4.5.2. Travel times

Based on the *Minkowski distances* we can calculate travel times by dividing this distance by a speed that we suppose a contractor's crew can travel between these points. Obviously in practice this speed is impacted by things like weather and traffic conditions. (see section 5.5.)

4.5.3. Nearest neighbor

If we were to approach the problem purely as a routing problem, the nearest neighbor algorithm is one of the possible ways to solve it. This algorithm constructs an order step by step, selecting the nearest neighbor to be scheduled next. I.e. the failed switch that is closest to the current location and therefore has the shortest traveling time. (TT_j) It does this until all locations have been included.

This closely resembles what we want to incorporate in our $1 \parallel \sum w_j C_j$ problem as it would intuitively not make sense to go all the way to the other side of the area next while there is also a failed switch right next to our current location.

By simply including the traveling times in the processing times we incorporate the routing aspect as a substitute for the sequence dependent setup times characteristic of the machine scheduling problem. To give an example, if we have two failures of switch A and switch B, with respectively weights 4 and 5. Now if we do not take the travelling time into account, we would obviously schedule failure B next as $\frac{4}{1.5} < \frac{5}{1.5}$. (WSEPT) But if we do include the traveling time and we find that switch A takes 15 minutes to reach switch B takes 45 minutes to reach then scheduling switch A next would be the better choice as $\frac{4}{1.75} > \frac{5}{2.25}$.

4.6. Greedy constructive heuristic

We propose a greedy constructive heuristic based on the WSEPT rule and the nearest neighbor algorithm that can be used to approximate an optimal schedule for each contract area. Constructive refers to the fact that it starts with an empty solution which it repeatedly extends until a full solution (order) is found. Greedy means that it does so by making the locally optimal choice at each stage with the intent of finding a global optimum. Figure 2 gives and overview of the process flow of the proposed heuristic as well as the alternatives, in which the priority factor depends on the alternative. Figure 3 shows the different priority factors for each of the alternatives. Note that in case of the routing based heuristic it chooses the failures with the lowest priority factor instead of the highest.

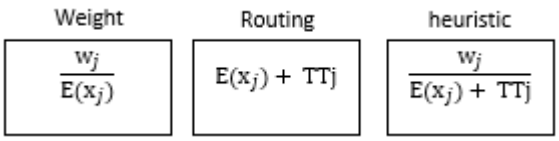

Figure 3: the priority factors for each of the alternatives

Obviously the random order just chooses a random failure to schedule next instead of basing it on a priority factor.

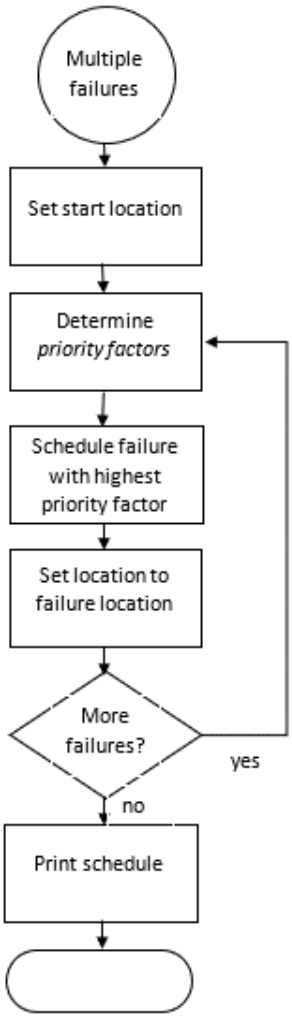

Figure 2 - process flow of the heuristics

To describe the process for the proposed heuristic: It first set the location to the contract area's start location. It then calculates the travel time (TT_j) to each outstanding switch failure location. (see section 4.5) It then looks up the weight (W_j) for each outstanding failure depending on the direction the switch is stuck in. (see section 5.3.1) It then calculates $\frac{w_j}{E(x_j)+ TTj}$ for each outstanding failure in which $E(x_j)$ is the expected repair time of 1.5 hours. (see section 5.2). The failure with the highest ratio of $\frac{w_j}{E(x_j)+ TTj}$ is then selected to be scheduled next. It then sets the location of this failure as the current location and removes the failure from the list of outstanding failures. This is process is repeated until all failures in the area are scheduled. The resulting schedule should be (near) optimal for the $1 \parallel S_{jk} \parallel \sum w_j C_j$ problem. I.e. it results in the lowest possible passenger hindrance.

How the heuristic actually performs is tested using a Monte Carlo simulation as explained in section 6. The results of these tests can be found in the results section of this report. (section 7)

The heuristic solves the problem step by step rather than in one go like a regular WSEPT heuristic does. It also requires a lot more calculations than a regular WSEPT heuristic. However the number of failures that need to be scheduled, looking at historic data, is nowhere near being big enough to affect the required processing time. The heuristic still solves almost instantaneously, i.e. the Heuristic determines the optimal solution in less than a second.

4.7. Alternatives

Next to the greedy constructive heuristics that we proposed in section 4.6 there are two other possible solutions that we test. These are the two parts that the heuristic is based on: the nearest neighbor algorithm (section 4.5) and the WSEPT rule. (section 4.4). The first one is a purely distance / travel time based solution and the latter becomes a purely weight based solution as the expected repair times are all equal. (1.5 hours). These two are also tested and compared to our constructive heuristic as described in section 6. Table 2 shows what aspects are included in each of the tested alternatives.

Where Random is a randomly generated order, Weight is the WSEPT heuristic, Travel is the nearest neighbor algorithm and Heuristic is the greedy constructive heuristic as proposed in section 4.6.

	aspect		
alternative	*weights*	*repair time*	*Travel time*
Random	-	-	-
Weight	*X*	*X*	-
Travel	-	*X*	*X*
Heuristic	*X*	*X*	*X*

Table 2: Aspects included in the alternative solutions.

5. Data analysis

In this section we discuss all data relevant to this research, both data that serves as input as well as static data that is used along the way.

5.1. Failure data

In the test we make use of both actual historical failure data as well as randomly generated failure data. We look at all switch failures that required urgent repair in the handshake area over a single day. E.g. all failures on the 4th January 2012 or the 17th of December 2018. These data records contain for each failure a unique equipment ID and the type (direction) as described in section 5.4.

5.1.1. Selecting relevant historic days

We inspected all switch failure data ranging from 2012 up to 2020. This data starts in 2012 because that is when the current way of recording failure data started. Doing so we found multiple days on which snow caused a considerable increase in switch failures that required urgent repair. We reduced this by excluding days that did not really affect the Handshake area all that much. This left us with three days to consider see table 3. Finally we looked at how these failures were spread over the contract areas in the handshake area while filtering out the failures with no impact (weight 0). All the failures that fall in the *No impact* category are failures that were marked prio 2 by ProRail employees, meaning that they believed they required urgent repair, but don't actually impact train traffic. I.e. they have a weight of 0 as they do not directly lead to any cancelations.

Contract area	10 Dec. 2017	11 Dec. 2017	16 Dec. 2018
Amsterdam	*0*	*3*	*2*
Kennemerland	*2*	*4*	*1*
Amstelpoort	*0*	*2*	*4*
Midden-Nederland	*0*	*1*	*0*
No impact	*11*	*8*	*11*
Total	***13***	***18***	***18***

Table 3: Categorized number of failures per historic day.

This analysis of the historic failure data leaves us with only two days that are interesting for testing possible solutions; 11 December 2017 and 16 December 2018. Therefore we decide to run our tests for these two days as well as randomly generated data.

5.1.2. Randomly generated failure data

Alongside the two historic day we also test four randomly generated days. Two of which have a total number of failures roughly equal to the observed historic days and two who have a strongly increased number of failures. We also test two random days with a strongly increased number of failures because we want to see how the heuristics perform on a day on which circumstances are even worse than they so far have actually been in the past. Table 4 shows for each of these four days how the failures are divided over the contract areas.

Contract area	Random day 1	Random day 2	Random day 3	Random day 4
Amsterdam	*5*	*3*	*10*	*13*
Kennemerland	*3*	*3*	*9*	*16*
Amstelpoort	*1*	*2*	*4*	*4*
Midden-Nederland	*1*	*4*	*5*	*8*
No impact	*8*	*3*	*25*	*27*
Total	***18***	***15***	***53***	***68***

Table 4: Categorized number of failures per randomly generated day.

These failures are generated using a combination of random, index and rank functions in excel to ensure we draw a sample from the list of all switches in the handshake area that contains no duplicates.

5.2. Distribution of repair times

In this section we discuss the distribution that the repair times follow. We look at the data of all switch failures over the past two years that required urgent repair (prio 2). There have been 381 of these switch failures in this period. We look at the time between the moment that the repair crew arrives at the location of the failure and the moment functionality is fully restored. This is what we call the repair time.

Based on this list of 381 repair times we fitted a distribution. We started by studying the summary statistics and making a histogram, see figure 4. The number of bins was determined using the Square-root rule: $\sqrt{381} \approx 20$

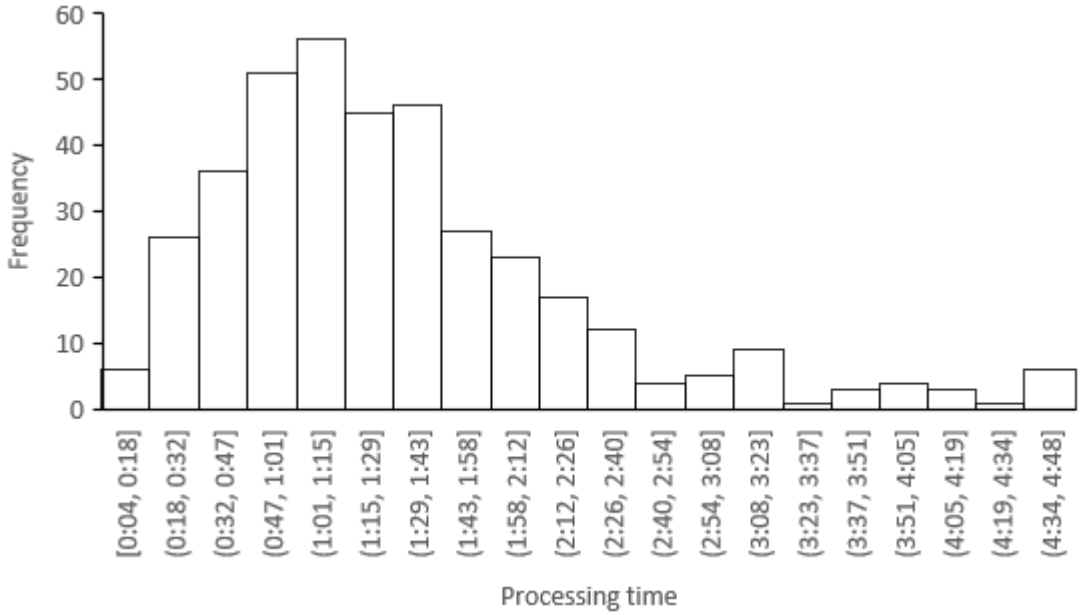

Figure 4 - Histogram of observed repair times

We came to the conclusion that the repair times follow a gamma distribution. Using some online distribution fitter we obtained the parameters k = 3.18 and θ = 0.47. We then proceeded to test whether this distribution actually fits the observed data. First we took the graphical approach and plotted the histogram against the gamma distribution as well as constructing a P-P plot. See figure 5 and figure 6. Based on this we can say that that the distribution fits quite well.

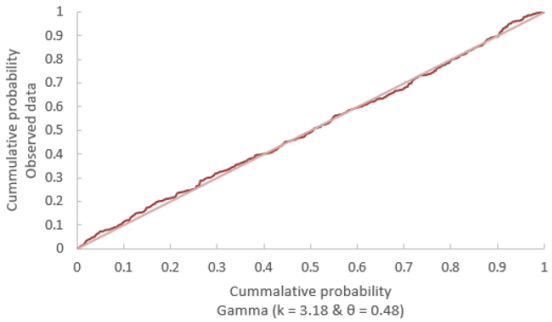

Figure 5 - p-p plot of observed cummulative distribution function vs that of the gamma distribution

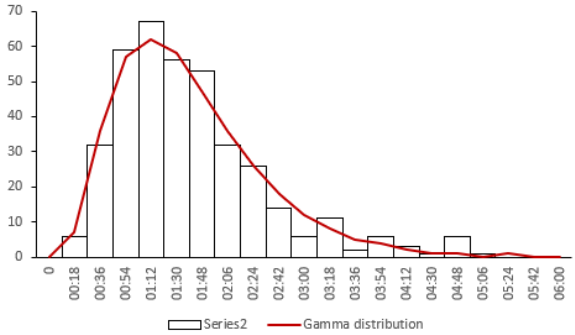

Figure 6 - Histogram of observed data plotted against gamma distribution

To further verify the assumption that this is indeed the right distribution we performed a statistical goodness-of-fit test, more specifically a chi-square test. Which means we test the Hypothesis H_0: "The repair times are independent, identically distributed random variables following a gamma distribution with k = 3.18 and θ = 0.47." We test this at the common level of significance of 95%.

We first divided the data over 20 adjacent bins, again based on the square-root rule. We then calculated the chi-square value according to $X^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$. Where O_i is the observed frequency in bin i and E_i is the expected frequency in bin i according to the gamma distribution. If this value is equal to or greater than the critical value then we reject H_0. This value came out to be 26.042.

The critical value depends on the level of significance and the degrees of freedom, in which the degrees of freedom is the number of bins – 1. Given a level of significance of 95% and 19 degrees of freedom this value is 30.143. Since 26.042 < 30.143 we do not reject H_0.

Based on the results of both the graphical approach as well as the chi-square test we conclude that it is safe to assume that the repair times follow a Gamma distribution with k = 3.18 and θ = 0.47.

Consequently this means that the expected repair time that we use in our heuristic ($\mathrm{E}(\mathrm{x}_j)$) is equal to k * θ = 3.18 * 0.47 = 1.495 or 1:29:41 ≈ 1.5 hours (1:30:00).

5.3. Switch data

This is all relevant static data pertaining to the actual switches. E.g. the contract area it belongs to. We index these records by the equipment ID which is a unique 8 digit number, So that we can trace the needed data based on this equipment ID. We have compiled all this data obtained from different sources into a single excel file consisting of eight columns. I.e. equipment ID, contract area, weight if LL, weight if RL, weight if T, X, Y.

5.3.1. Weights

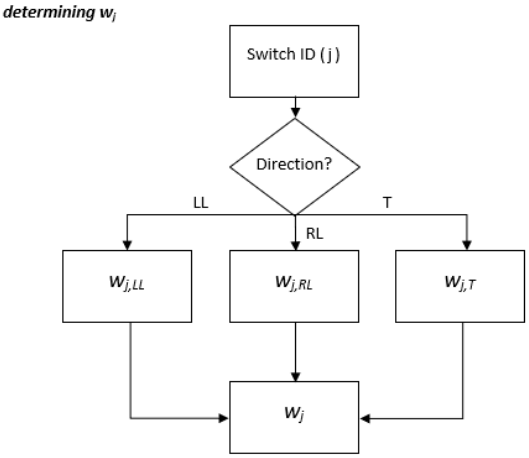

Figure 7 - process flow of selecting the weight of failure j, based on the type of failure.

As explained earlier, ProRail has previously made lists containing the number of trains cancelled per hour if a switch fails, depending on the direction it is stuck in. This is the before mentioned critical switch list. We say a failure has 3 possible types, LL, RL and T, respectively, left sided, right sided and total. The system has to read the right number from the switch data file based on this type. E.g. if switch x breaks down while stuck to the right hand side (RL) 8 trains have to be cancelled per hour as there are 8 trains per hour that need the switch to be in the LR position in order to be able to drive to their destination. If it was stuck to the left hand side only 4 trains would have to be cancelled, but if it was stuck somewhere inbetween, all 12 trains would have to be cancelled.

Figure 8 shows the different directions a switch can be stuck in, in which the red path is the path that can still be driven on.

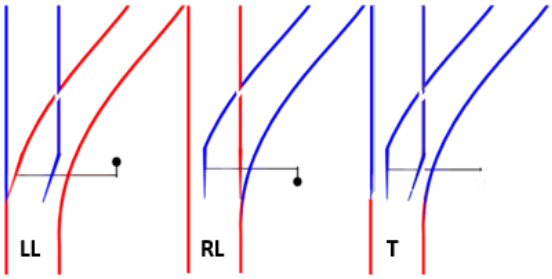

Figure 8 - the 3 different positions a switch can be stuck in

These weights come directly from the critical switch list, which is a list that among other things notes for all switches how many trains have to be cancelled if it fails and is stuck in a certain position. This is the biggest validity concern, as the proposed heuristic is only as accurate as the list. I.e. if the critical switch list is wrong, the weights are wrong and therefore the solution generated by the heuristic is wrong. It is therefore essential that the critical switch list is regularly checked and updated, if ProRail wants to use the results of this study and the tool that we've created in the future.

5.3.2. Location

The location of a switch has two aspects, the contract area is lies is and the actual coordinates. The contract area determines which group or which *machine* a possible failure would belong to.

The coordinates are used to determine the actual physical location of a switch which is used in calculating distances between switches and/or the repair crews' locations. This distance is calculated using the Minkowski method. (section 4.5.1)

The geo location of a switch is based on *rijksdriehoekscoördinaten* also often referred to as RD-coordinates. This is a coordinate system used in the Netherlands that centers around a point in Amersfoort. This point is given the coordinates; x = 155 000, y = 463 000 so that no point in the Netherlands has negative x or y coordinates. These coordinates are known for all switches.

5.4. Contractors starting location

The contractors repair crew could be starting at any given location within the contract area, unrelated to switch locations. This is because it is not known to us where the repairmen live or start their day. Therefore we set the start location for each contract area as point of the average x and y of all switches in that area. Table 5 shows the x and y coordinate of this location for each contract area.

Contract area	X	Y
Amsterdam	*120119*	*488960*
Kennemerland	*105907*	*484158*
Amstelpoort	*136471*	*486629*
Midden-Nederland	*140349*	*456655*

Table 5: Coordinates of starting locations

5.5. Travelling speed

In the travel time calculations we used a constant speed while in practice the traveling speed is stochastic and differs between trips. Therefore we conduct all tests for 4 different constant values of speed: 15 km/h 20 km/h 25km/h and 30km/h. These values are based on the fact that the repair crew is driving in crowded urban areas with traffic impacted by snowfall. As the results confirm, any practical value for the traveling speed does not affect the optimal order of performing repairs. However if we take really high or really low values this does significantly impact the relative performances. This is intuitive as a high speed, drastically reduces the travel time, making it a lot less relevant while taking a really low speed, below 1km/h, has the opposite effect.

6. Testing the solution

In this section, we discuss how we tested the proposed solutions.

6.1. The simulation model

In order to determine whether and to what extent the proposed solutions can reduce the passenger hindrance we created a Monte Carlo simulation model. This model takes a list of failures and returns the total passenger hindrance for each of the proposed ways of scheduling repairs. The only variable input other than the list of failures is the traveling speed as explained in section 5.5. The simulation model performs 1000 runs for each of the traveling speeds. (see section 6.3) In each of these runs a new string of randomly generated repair times is used. These are drawn from the gamma distribution with $k = 3.18$ and $\theta = 0.47$ as explained in section 5.2. The model was coded from scratch in Python. The output of the simulation is analysed in the results section of the report section 7. This serves two purposes, in the first place we want to determine if and to what extend we can reduce passenger hindrance. Secondly we want to determine which of the alternatives performs the best, i.e. leads to the least amount of passenger hindrance. All input and data that the model makes use of is discussed in the data analysis section (section 5).

6.2. Output

The output of the simulation model is of course the one thing that this study resolves around: the total passenger hindrance. After the last repair has been completed the model looks at how many trains had to be cancelled because of the failures. For every train that has been cancelled it counts 30 minutes. The result of this sum is the total passenger hindrance.

A single run of the simulation outputs 4 total passenger hindrances, one for each of the methods of scheduling. I.e. Random, weight based, routing based, and our custom greedy constructive heuristic.

We don't just perform a single run, we run the model 1000 times for every day and for each of the four travelling speeds are described in section 5.5. The results as can be seen in section 7 are the averages of the 1000 runs.

6.3. Calculating the total passenger hindrance

Formally notated the total passenger hindrance is calculated using the following formula:

$$total\ passenger\ hindrance = 30 * \sum_{i=1}^{k} \sum_{j=1}^{n_k} t_j w_j$$

Where t_j is the time it took in hours for the functionality of switch j to be fully restored, measured from the start of the simulation and w_j is the weight of switch j (number of cancelled trains per hour) With k number of contract areas and n_k number of failures in contract area k.

6.4. The random order

The random order represents the order that would be obtained in a situation in which the contractors are left to determine their own order of repairs, resulting in a seemingly arbitrary order. This what we compare the alternatives to as without ProRail's intervention, we do not expect the contractors to follow any particular order.

6.5. Method

We start by running the simulation for the two historic days and unlike in the simulation runs for the randomly generated days, we also calculate a 5th total passenger hindrance. This passenger hindrance is calculated for a manually set order. This order is the order in which the repairs were actually performed on the two historic days according to the failure data. These are what we call the realities. This is done to actually measure the problem as explained in the problem analysis. See section 2.4.

This allows us to compare the performance of the proposed scheduling methods not only to the performance of the average random order, but also to the actual historic performance.

After we run the simulation model as described above for the two historic days, we run it for the four days (sets) of randomly generated failures. Two of which have a roughly equal number of failures as the observed historic days and two that have a strongly increased number of failures. The exact build up of the failures on the historic days as well as the randomly generated days is described in section 5.1. The results of these simulation runs are analyzed in the results section. (section 7)

6.6. Number of replications

As mentioned above we run the simulation 1000 times for each travelling speed, for each day / each set of failures, using a different stream of random numbers in every run. In calculating the required number of replications we use the significance level of a = 0.05 and a relative error of d = 0.05. The relative error is the deviation of the confidence interval about the mean.

$$n^* = \left(\frac{t_{n^*-1,1-\alpha/2} \sqrt{S_n^2}}{d|\bar{X}_n|} \right)^2$$

We used the above formula to determine the lowest number of replications for which the estimated relative error (d) is equal or lower than 0.05 for all four of the alternatives. We've calculated this for every day and every speed. Table 6 shows these minimal number of replications.

	day					
speed	*11-12-2017*	*16-12-2018*	*Random 1*	*Random 2*	*Random 3*	*Random 4*
15	*57*	*88*	*52*	*61*	*41*	*27*
20	*62*	*96*	*58*	*55*	*38*	*25*
25	*62*	*98*	*63*	*61*	*33*	*24*
30	*67*	*95*	*66*	*66*	*34*	*24*

α = 0.05

Table 6: Minimal number of replications for which relative error (d) $\leq$ 0.05 for all alternatives

Clearly a 1000 runs is more than enough, but as the runtime is short enough that we don't have to reduce the number of runs. Doing a 1000 runs allows us achieve much lower levels of estimated relative error than 0.05. Table 7 shows the estimated relative errors for every day and speed for the random order. These have actually been calculated for all four of the alternatives as well. These can be found in appendix A.

	day					
speed	*11-12-2017*	*16-12-2018*	*Random 1*	*Random 2*	*Random 3*	*Random 4*
15	0.0116	0.0146	0.0111	0.0121	0.0097	0.0078
20	0.0122	0.0152	0.0117	0.0114	0.0094	0.0074
25	0.0120	0.0155	0.0122	0.0121	0.0087	0.0073
30	0.0126	0.0152	0.0126	0.0125	0.0088	0.0072

α = 0.05

Table 7: estimated relative error for all random orders, for every speed and every day

6.7. Validity concerns

The way we measure the total passenger hindrance has two notable drawbacks. First of all it does not take into account the expected number of passengers in the trains. We just count 30 minutes for every cancelled train, regardless of how many passengers are expected to be in it. This is because we did not have access to any passenger data as this is not measured by ProRail but rather by NS. However we're only looking at the handshake area, which in its entirety is a super busy and densely populated area in which almost all trains are at least pretty full. If we had included areas that are less busy this would have mattered more. The actual measure for passenger hindrance that is used within ProRail also doesn't take this into account either so it could be argued that they do not find it all that relevant. Secondly it only takes into account cancelled trains and not delays as there was no way for us to simulate this without the simulation becoming overly complex. We would have to take into account the schedules of all trains and place everything within a timeframe and even then we could only speculate about when the operators would decide to cancel a train and when they would let it depart with a delay. We also wouldn't be able to account for trains that make up for their delays by shortening stops. This cannot influence the total passenger hindrance to a great extent as this can never be more than 30 minutes per switch since otherwise the train would just be cancelled. So there are some validity concerns regarding the measurement of the passenger hindrance. It was however not possible to use a more "valid" measurement as this would be overly complex considering the limited time or required data that we have no access to.

The model does however validly measure the passenger hindrance for the way we use it. There is very little that can go wrong here, as long as the critical switch list from which we obtain the weights is correct.

6.8. Model verification

It goes without saying that it is of essential importance that the simulation model correctly performs the tasks that we want it to do, otherwise the results are worthless. During the coding of the simulation model and the included heuristics the code has continuously been verified.

Every part / function in the code has been tested carefully by debugging the code every step of the way. This means every single function and calculation has been thoroughly tested by comparing the output with what we know is the correct value. Take for example the simple function that finds a switch in the list of switches by its equipmentID. The code below is supposed to return the index of the switch in the switch list.

```
'''Find element in finalData by switchID'''
def findSwitch(switchID):
    switchList = [i[0] for i in finalData]
    location = switchList.index(switchID)
    return location
```

We test this by printing the result of the function for a switch of which we know the index, say switch 11560524, which has index 2.

```
print(findSwitch(11560524))
```

Running the code above should then print the value 2, which of course, it does.

7. Results

In this section the results of the simulation runs are shown and discussed. The alternatives are compared to each other and for the historic days, also to the measured reality and norm.

7.1. Reality and norm

We start by determining the realities and corresponding norms. By reality we mean the total passenger hindrance as it would have occurred on the historic day, according to our model. This is a quantification of our problem: “too much passenger hindrance”. (see section 2.4) This is done by taking the actual orders in which repairs were performed and then using the simulation model to calculate the total passenger hindrance. We take the averages of 1000 simulation runs for each of the different travelling speeds. Of course we use the same stream of random numbers in obtaining these values and in testing the alternatives. Table 8 shows the resulting realities and corresponding norms for both of the historic days, for each speed. As described in section 2.1.4, the norms are a 10% reduction of the realities.

	11-12-2017		16-12-2018	
speed	*reality*	*norm*	*Reality*	*norm*
15	14615	*13154*	*10389*	*9350*
20	*13819*	*12437*	*9700*	*8730*
25	*13342*	*12008*	*9371*	*8434*
30	*13022*	*11720*	*9068*	*8161*
average	*13700*	*12330*	*9632*	*8669*

Table 8 - Measured realities and corresponding norms

7.2. Simulation results historic data

These are the simulation results for both of the historic days as described in section 6. Table 9 shows the average total passenger hindrance over the 1000 runs for both historic days given the static travelling speed. Where Random is the randomly generated order, Weight is the WSEPT heuristic, Travel is the nearest neighbor algorithm and Heuristic is the greedy constructive heuristic as proposed in section 4.6.

	11-12-2017				16-12-2018			
speed	*random*	*weight*	*travel*	*heuristic*	*random*	*weight*	*travel*	*heuristic*
15	*14446*	*11650*	*14138*	*11649*	8341	6567	6638	6567
20	*13587*	*11142*	*13462*	*11141*	7892	6182	6279	6181
25	*13269*	*10862*	*13137*	*10861*	7704	5978	6093	5978
30	*13043*	*10682*	*12949*	*10681*	7393	5795	5919	5795
average	*13586*	*11084*	*13421*	*11083*	7833	6130	6232	6130

Table 9: Simulated passenger hindrance results for actual historic days

The simulation has made clear that for any reasonable speed the weight based methods performs nearly identically to our proposed greedy constructive heuristic on both of the historic days. They both clearly perform a lot better than randomly scheduling repairs and they also perform better than the nearest neighbor algorithm. On 16-12-2018 this effect was much smaller than on 11-12-2017, which means that the way failures were spread out over the areas on 16-12-2018 made the travel time more important.

	11-12-2017				16-12-2018			
speed	*random*	*weight*	*travel*	*heuristic*	*random*	*weight*	*travel*	*heuristic*
15	.988	.797	.967	.797	.803	.632	.639	.632
20	.983	.806	.974	.806	.814	.637	.647	.637
25	.995	.814	.985	.814	.822	.638	.650	.638
30	1.002	.820	.994	.820	.815	.639	.653	.639
average	.992	.809	.980	.809	.813	.636	.647	.636

Table 10: Percentage of total passenger hindrance, simulation results vs measured reality

Our proposed heuristic performs a lot better than the orders in which repairs where actually performed and even perform better than the norms we have set. On 11-12-2017 it on average results in a 19.1% reduction in comparison to the actual used order of repairs. On 16-12-2018 it on average results in a 36.4% reduction compared to the actual used order. Table 10 shows the resulting percentage of total passenger hindrance as compared to the measured realities for each of the methods and travelling speeds.

Notable is that the order in which repairs were performed on 16-12-2018 results in a really high passenger hindrance, even the average random order performed roughly 19% better. On 11-12-2017 the actual order of repairs performs almost identical to the average random order. This means that using our heuristic, we can comfortably exceed our norm of a 10% reduction. To verify that the differences are significant we performed t-tests in which we construct confidence intervals for the difference of the means. To do so we use the following formula:

$$CI = \bar{X} - \bar{Y} \pm t_{2n-2,1-\alpha/2}\sqrt{\frac{S_X^2+S_Y^2}{n}}$$

If we take for example 16-12-2018, speed 30 and level of significance a = 0.05, we obtain the confidence interval of [3132, 3414]. Clearly 0 does not lie within this interval, therefore we can say that the difference is significant for a = 0.05. In doing so we found that the difference is significant for all speeds. See appendix A for all the confidence intervals.

	11-12-2017				16-12-2018			
speed	*random*	*weight*	*travel*	*heuristic*	*random*	*weight*	*travel*	*heuristic*
15	1	.806	.979	.806	1	.787	.796	.787
20	1	.820	.991	.820	1	.783	.796	.783
25	1	.819	.990	.819	1	.776	.791	.776
30	1	.819	.993	.819	1	.784	.801	.784
average	1	.816	.988	.816	1	.783	.796	.783

Table 11: Performance of the different heuristics, compared to the average random order (historic)

Table 11 again shows the percentage of total passenger hindrance each of the methods results in, this time compared to the average random order. So for the travelling speed of 15 km/h the proposed heuristic only results in 80,6% of the total passenger hindrance of the average random order.

In other words it performed 19,4% better. This also again shows that on both historic days the weight based method performs the same as our proposed heuristic.

7.3. Simulation results generated data

These are the simulation results for the randomly generated days. Unlike in the test with the historic days we have no “reality” to compare the different results to, therefore we compare them to the results of the random order.

7.3.1. Results comparable number of random failures

Table 12 shows the results for the randomly generated days with a number of failures that is roughly the same as on the observed historic days. For the exact division of failures over the contract areas see section 5.1.2.

	Random day 1				Random day 2			
speed	*random*	*weight*	*travel*	*heuristic*	*random*	*weight*	*travel*	*heuristic*
15	7299	6213	7122	6081	22575	15761	16949	15542
20	6837	5745	6736	5643	20961	14842	16184	14670
25	6587	5486	6523	5404	19472	14101	15490	13961
30	6403	5307	6370	5238	18937	13797	15267	13680
average	6781	5688	6688	5591	20486	14625	15973	14463

Table 12: Simulation results - randomly generated days with comparable number of failures

	Random day 1				Random day 2			
speed	*random*	*weight*	*travel*	*heuristic*	*random*	*weight*	*travel*	*heuristic*
15	1	.851	.976	.833	1	.698	.751	.688
20	1	.840	.985	.825	1	.708	.772	.700
25	1	.833	.990	.820	1	.724	.796	.717
30	1	.829	.995	.818	1	.729	.806	.722
average	1	.839	.986	.825	1	.714	.780	.706

Table 13: performance of the different heuristics, compared to the average random order (equal)

These results show that again the weight based method and our proposed heuristic perform the best. This time however they do not perform identical, our proposed heuristic performs slightly better. Compared to the random average order it on average performs 1.4% better on random day 1 and 0.8% better on random day 2. Again performing t-test shows us that these differences are actually not significant, therefore we say that they perform roughly the same. Again see appendix A for all the calculated confidence intervals.

On average our proposed heuristic performed 17,5% better than the average random order on random day 1 and even 29,4% on random day 2.

7.3.2. Results strongly increased number of random failures

Table 14 shows the results for the randomly generated days with a strongly increase number of failures. Again, for the exact division of failures over the contract areas see section 5.1.2.

	Random day 3				Random day 4			
speed	*random*	*weight*	*travel*	*heuristic*	*random*	*weight*	*travel*	*heuristic*
15	130046	111551	76051	67612	251349	212476	171429	131914
20	113993	95443	71136	62730	224130	182468	163564	122793
25	105068	85887	68274	60041	208088	165409	159842	118191
30	97706	79363	66268	57998	195626	153248	156257	114570
average	111703	93061	70432	62095	219798	178400	162773	121867

Table 14: Simulation results - randomly generated days with strongly increased number of failures

	Random day 3				Random day 4			
speed	*random*	*weight*	*travel*	*heuristic*	*random*	*weight*	*travel*	*heuristic*
15	1	.858	.585	.520	1	.845	.682	.525
20	1	.837	.624	.550	1	.814	.730	.548
25	1	.817	.650	.571	1	.795	.768	.568
30	1	.812	.678	.594	1	.783	.799	.586
average	1	.833	.631	.556	1	.812	.741	.554

Table 15: performance of the different heuristics, compared to the average random order (increased)

These results show that on the randomly generated days with a strong increased number of failures our proposed heuristic really performs a lot better than the others. On both days it on average performed roughly 44,5% better than the average random order. This is a massive decrease in passenger hindrance. We also see that on these days the weight based method really perform a lot worse than our proposed heuristic. Compared to the average random order we're talking about a difference of 27,7% on random day 3 and 25,8% on random day 4. Also notably the routing based method performs better than the weight based on both days. It's performance falls between the performance of the proposed heuristic and weight based method.

7.4. Other findings and conclusion of the results

In not a single of the 24000 simulation runs that were performed did the average random order result in a lower passenger hindrance than the greedy constructive heuristic proposed in section 4.6. Therefore we performed some of the tests again, this time with 100000 runs. Take for example 16-12-2018, speed 25. There are 2! * 4! * 1! = 48 possible orders, all equally likely to be drawn by the random order. If out heuristic does not result in the optimal order of performing repairs, then with that many runs there should have at least been a few or even one run in which the random order resulted in a lower passenger hindrance. Because if we say there's 1 out of 48 possibilities that is better, than the chance of the random order not choosing that order a single time is $(\frac{47}{48})^{100000}$, which is a number so small we can say it is equal to 0. To give another example, on 11-12-2017, speed 30, there were 3! * 4! * 2! *1! = 288 possibilities, but even then $(\frac{287}{288})^{100000}$ becomes such a small number that we can say it is roughly equal to 0. Therefore in combination with the rest of the simulation results we can conclude that the heuristic we proposed in section 4.6 does indeed find the optimal order of performing repairs for the observed cases.

We also find that as the number of failures with equal weights increases the more important the traveling time becomes and therefore the proposed heuristic starts to perform better than the weight based (traditional WSEPT) heuristic. This is because thee weight based alternative just chooses a random order for failures with the same weight where our heuristic then orders them by travel time.

Furthermore the importance of the travel time decreases as the travelling speed increases, this is pretty intuitive as increasing the speed reduces the travel time and therefore makes it less relevant to the static 1.5 hours of repair time.

Concludingly we find that we can indeed decrease the passenger hindrance by efficiently scheduling the order in which switch failure repairs are performed. This reduction ranges from 16,8% to 31,2%, compared to the average random order, for situations with a number of failures comparable to that of the two worst historic days. The reduction becomes significantly larger when the number of failures increases. We also constructed the confidence interval for the difference between the average random order and the proposed heuristic for every day and every speed and we found the difference to be significant for all cases. See appendix A for these confidence intervals.

8. Tool

As a result of this research we've created a tool that ProRail can use to determine the preferred order of repairs for each of the four contract areas that lie in the handshake area. We considered building a simple to use tool to be the easiest way to implement the proposed solution within ProRail.

At first we considered making the tool a .exe file as to make it a standalone app, however after testing an early prototype we quickly realized that the tool would actually function much better as a excel based tool. This would make it much easier to use, mainly because of the convenient way excel allows the user to input the failures.

The tool is really straight forward to use, you input all your failures in one column (equipmentID), the direction in which the switch is stuck (LL, RL or T) in the column next to it and then you press the "Calculate optimal order of repairs" button. The tool then automatically writes the optimal order for each of the contract areas, according to our proposed heuristic, in the designated columns. This order is written down top to bottom, so the first switch that should be repaired will be written in row 2, the second in row 3 and so on.

	A	B	C	D	E	F	G	H	I	J
1	**Failures to schedule (equipmentID)**	**direction (LL, RL, T)**		Calculate optimal order of repairs			**Amsterdam**	**Amstelpoort**	**Kennemerland**	**Midden-Nederland**
2	10265699	LL					10265699	10657137	10265790	
3	10657137	LL					10608485		10689078	
4	10265790	T							10689079	
5	10689078	RL							11346443	
6	10608485	T	1. insert all failures in column A							
7	11346443	LL	2. insert the directions is coulumn B							
8	10689079	RL	3. press the button							
9										

Figure 9: screenshot of the excel tool

This tool can easily be expanded to cover the entire nation instead of just the handshake area. The next time one of the contractors calls ProRail to ask them what order of repairs they would prefer, the employee that answers the phone can just quickly use the tool to determine the optimal order. As explained above the tool is super simple to use, the only thing needed to implement it within ProRail is to make sure the employees are aware of its existence and that they know where to find it.

The bigger question is whether the contractors will every really ask ProRail in what order to perform their repairs as the PGO contract form make the contractors responsible for everything that happens in their contract area.

9. Conclusion

In this section we draw a conclusion based on the results, we discuss the study as a whole and we formulate our recommendations.

9.1. Conclusion

First off all based on the failure data we analyzed to determine which days to use for the historic data portion of the simulation, we conclude that the scheduling problem as we defined it is a lot less relevant than we originally thought. We barely managed to find two historic days on which there were enough switch failures in a single contract area, with a weight bigger than zero, for us to be able to apply any form of scheduling to. The number of failures in these areas was also so low that instead of using a heuristic to approximate an optimal solution we could have just exactly worked out all possibilities and then just pick the solution that leads to the lowest level of passenger hindrance.

Perhaps most importantly, based on the simulation results we conclude that the passenger hindrance can indeed be reduced by optimizing the order in which repairs are performed. Furthermore we find that we can reduce the passenger hindrance to below our norm of a 10% reduction of the historic performance. This does however require the all of the contractors to let ProRail tell them in which order to perform repairs, which in practice is very unlikely to happen.

Finally the fact that the random order did not even in a single run perform better than our proposed greedy constructive heuristic leads us to believe that this solution actually finds a nearly optimal if not optimal solution to the scheduling problem. With that many runs the random order should have at least once resulted in less passenger hindrance if there exists a more optimal way of scheduling switch failure repairs than the heuristic that we proposed in section 4.6. This means that the tool we've created, which applies our proposed heuristic, can be used to determine a close to optimal order of repairs the next time a contractor asks ProRail for help.

9.2. Discussion

The way we measure the total passenger hindrance has two notable drawbacks. First of all it does not take into account the expected number of passengers in the trains. We just count 30 minutes for every cancelled train, regardless of how many passengers are expected to be in it. This is because we did not have access to any passenger data as this is not measured by ProRail with rather by NS. However we're only looking at the handshake area, which in its entirety is a super busy and densely populated area in which al most all trains are at least pretty full. If we had included areas that are less busy this would have mattered more. The actual measure for passenger hindrance that is used within ProRail also doesn't take this into account either so it could be argued that they do not find it all that relevant. Secondly it only takes into account cancelled trains and not delays as there was no way for us to simulate this without the simulation becoming overly complex. We would have to take into account the schedules of all trains and place everything within a timeframe and even then we could only speculate about when the operators would decide to cancel a train and when they would let it depart with a delay. We also wouldn't be able to account for trains that makeup for their delays by shortening stops. This does not influence the total passenger hindrance to a great extent as this can never be more than 30 minutes per switch since otherwise the train would just be cancelled.
Realistically the way we chose to measure the passenger hindrance is the only way that was feasible given the limited time and the lack of passenger data.

The contractors' starting locations are determined by taking the average of the coordinates of all switches in that contract area. In practice the contractor's crew could be starting their day at any point inside the area, or perhaps even slightly outside the area.

The employees of the contractor could go to work immediately from their houses or perhaps they first gather at some kind of office. This could very well differ per contractor as well. In theory this could have a slight impact on the optimal order of repairs.

Furthermore we assume that the contractors crew continuously works on repairing the switches without taking breaks, detours or having to perform other repair and maintenance work unrelated to switch failures. This could very well not be the case in practice, however there is no feasible way for us to take this into account.

Finally we use 4 different static travelling speeds in our model where as in practice the travelling speed is not constant nor deterministic. However as we have shown no practical value that the travelling speed can assume significantly affects the relative performances of the alternatives. Only when working with extremely low travelling speeds, say less than 1km/h, do the traveling times become significantly more important, affecting the relative performances of the alternatives.

9.3. Recommendations

As it should require relatively little effort to expand the tool to cover the entire nation rather than just the handshake area, we would recommend ProRail to do so. This because based on the results we believe that the tool finds an (nearly) optimal solution which does actually lead to a lower total passenger hindrance than when contractors just perform repairs in a seemingly random order. Even though the chance is slim that the situation in which the tool can actually be used in practice ever occurs. This recommendation is based on the notion that it can't hurt to have the tool as in the worst case scenario it is simply never used. If it does not take too much effort we would also recommend somewhat regularly updating the critical switch list that is used in the tool, as this is vital for the accuracy of the results.

In an ideal situation the tool would be changed to include more, if not all kinds of failures which would great increase the usability of the tool. However it is highly unlikely this would be worth the time and effort as even if this would be feasible, the way the PGO contracts work really limit, if not completely take away, the influence ProRail has on the order in which repair are performed. However if this ever changes in the future then it could considered.

9.4. Suggestions for further research

As it stands we would not recommend putting too much effort in further optimizing the order in which urgent repairs are performed because of the afore mentioned issues. However if for some reason some one decides to further research the optimization of urgent repair scheduling within ProRail, we would highly recommend looking to include actual passenger data in the model as this is the one thing that could really drastically impact the results on a practical level.

Another suggestions would be to look at the possible impact of traffic on the optimal solution. This is something we considered incorporating in our model but was eventually left out due to time restraints.

10. Bibliography

1. Chaos op spoor door sneeuw (2012, February 3), rtl nieuws, Obtained from: https://www.rtlnieuws.nl/nieuws/artikel/2983806/chaos-op-spoor-door-sneeuw
2. Heerkens, J. M. G., & van Winden, A., (2012), *Geen probleem, een aanpak voor alle bedrijfskundige vragen en mysteries.* Buren: Business School Nederland.
3. Pinedo, M. L., (2008), New York, NY, Springer Science+Business media
4. Shahid, R., Bertazzon, S., Knudtson, M.L., Ghali, W.A. (2009), Comparison of distance measures in spatial analytical modeling for health service planning.,BMC Health Serv Res 9, 200.
5. Moghaddam, A., Teghem, J., Tuyttens, D., Yalaoui, F., Amodeo, L., (2019), *Toward an Efficient Resolution for a Single-machine Bi-objective Scheduling Problem with Rejection,* Foundations of Computing and Decision Sciences, 44 (2), pp. 179-211.
6. Zhang, X., (2017), *Single machine and flowshop scheduling problems with sum-of-processing time based learning phenomenon,* Journal of Industrial and Management Optimization, 13 (5), pp. 1-14.
7. Lee, W.-C., Wang, W.-J., Shiau, Y.-R., Wu, C.-C., (2010), *A single-machine scheduling problem with two-agent and deteriorating jobs,* Applied Mathematical Modelling, 34 (10), pp. 3098-3107.
8. Mosheiov, G., Sarig, A., (2009), *Scheduling a maintenance activity to minimize total weighted completion-time,* Computers and Mathematics with Applications, 57 (4), pp. 619-623.
9. Arroyo, J.E.C., Dos Santos Ottoni, R., De Paiva Oliveira, A., (2011), *Multi-objective variable neighborhood search algorithms for a single machine scheduling problem with distinct due windows,* Electronic Notes in Theoretical Computer Science, 281, pp. 5-19.
10. Ángel-Bello, F., Álvarez, A., Pacheco, J., Martínez, I., (2011), *A heuristic approach for a scheduling problem with periodic maintenance and sequence-dependent setup times,* Computers and Mathematics with Applications, 61 (4), pp. 797-808.
11. Lin, M.-Y., Kuo, Y., (2017), *Efficient mixed integer programming models for family scheduling problems,* Operations Research Perspectives, 4, pp. 49-55.
12. Wang, X.-Y., Wang, J.-J., (2013), *Scheduling problems with past-sequence-dependent setup times and general effects of deterioration and learning,* Applied Mathematical Modelling, 37 (7), pp. 4905-4914.
13. Lee, W.-C., (2011), *A note on single-machine scheduling with general learning effect and past-sequence-dependent setup time,* Computers and Mathematics with Applications, 62 (4), pp. 2095-2100.
14. Wang, J.-B., Li, J.-X., (2011), *Single machine past-sequence-dependent setup times scheduling with general position-dependent and time-dependent learning effects,* Applied Mathematical Modelling, 35 (3), pp. 1388-1395.
15. Wang, J.-B., Wang, D., Wang, L.-Y., Lin, L., Yin, N., Wang, W.-W., (2009), *Single machine scheduling with exponential time-dependent learning effect and past-sequence-dependent setup times,* Computers and Mathematics with Applications, 57 (1), pp. 9-16.

Appendix A – confidence intervals and relative errors

	16-12-2018		16-12-2018	
speed	*Lower bound*	*Upper bound*	*Lower bound*	*Upper bound*
15	*3689*	*3956*	*3689*	*3956*
20	*3385*	*3651*	*3385*	*3651*
25	*3254*	*3533*	*3254*	*3533*
30	*3132*	*3414*	*3132*	*3414*
a = 0.05				

Table A1: Confidence intervals for the difference between the means of the reality and the proposed heuristic

	11-12-2017		16-12-2018		Random day 1	
speed	*Lower bound*	*Upper bound*	*Lower bound*	*Upper bound*	*Lower bound*	*Upper bound*
15	2583	3012	1676	1975	1115	1321
20	2232	2660	1547	1838	1092	1296
25	2201	2615	1509	1796	1081	1285
30	2149	2575	1477	1753	1062	1267
	Random day 2		Random day 3		Random day 4	
speed	*Lower bound*	*Upper bound*	*Lower bound*	*Upper bound*	*Lower bound*	*Upper bound*
15	6723	7343	61103	63765	117357	121513
20	6010	6572	50110	52417	99521	103153
25	5230	5790	44020	46035	88225	91568
30	4972	5542	38757	40658	79479	82635
a = 0.05						

Table A2: Confidence intervals for the difference between the means of the average random order and the proposed heuristic

	11-12-2017		16-12-2018		Random day 1	
speed	*Lower bound*	*Upper bound*	*Lower bound*	*Upper bound*	*Lower bound*	*Upper bound*
15	2583	3012	1676	1975	41	223
20	2232	2660	1547	1838	13	191
25	2201	2615	1509	1796	-7	171
30	2149	2575	1477	1753	-21	158
	Random day 2		Random day 3		Random day 4	
speed	*Lower bound*	*Upper bound*	*Lower bound*	*Upper bound*	*Lower bound*	*Upper bound*
15	11	427	43357	44521	79592	81531
20	-35	379	32102	33325	58649	60701
25	-74	353	25260	26432	46216	48219
30	-105	340	20786	21944	37681	39676
a = 0.05						

Table A3: Confidence intervals for the difference between the means of the weight based order and the proposed heuristic

	11-12-2017				16-12-2018			
speed	*random*	*weight*	*travel*	*heuristic*	*random*	*weight*	*travel*	*heuristic*
15	.0116	.0115	.0110	.0115	.0146	.0129	.0131	.0129
20	.0122	.0122	.0118	.0122	.0152	.0134	.0136	.0134
25	.0120	.0122	.0117	.0122	.0155	.0141	.0143	.0141
30	.0126	.0126	.0122	.0126	.0152	.0142	.0144	.0142
	Random day 1				Random day 2			
speed	*random*	*weight*	*travel*	*heuristic*	*random*	*weight*	*travel*	*heuristic*
15	.0111	.0104	.0105	.0105	.0121	.0093	.0096	.0094
20	.0117	.0110	.0112	.0111	.0114	.0099	.0101	.0100
25	.0122	.0115	.0116	.0116	.0121	.0107	.0108	.0108
30	.0126	.0119	.0117	.0121	.0125	.0114	.0114	.0115
	Random day 3				Random day 4			
speed	*random*	*weight*	*travel*	*heuristic*	*random*	*weight*	*travel*	*heuristic*
15	.0097	.0037	.0059	.0061	.0078	.0033	.0052	.0051
20	.0094	.0045	.0066	.0069	.0074	.0040	.0058	.0058
25	.0087	.0048	.0066	.0069	.0073	.0043	.0058	.0060
30	.0088	.0051	.0068	.0071	.0072	.0046	.0058	.0061
a = 0.05								

Table A4: estimated relative error for all alternatives, for every speed and every day

Appendix B – Literature research

Research question

As described above, we wish to learn how scheduling problems similar to ours, are solved in literature. Therefore we've formulated the following research question:
"What ways of solving single machine scheduling problems with the objective of minimizing the total weighted completion time and given sequence dependent setup times exist in literature?"

Key theoretical concepts

The following is a list of the key theoretical concepts that are involved in this literature review:

- ***setup time,*** the time that is incurred between the processing of two jobs.
- ***Sequence dependent setup times,*** setup times depending on the sequence in which the jobs are executed on the machine i.e. the setup time between job *n* and job *n* + 1 depends on the type of job *n*.
- **Total weighted completion time**, the sum of the weighted completion times of all jobs sum. Often referred to as the weighted flow time.
- **Single machine scheduling problem**, scheduling problem in which there is only a single machine available for the processing of all jobs.

Exclusion criteria

The following exclusion criteria are used to exclude texts deemed irrelevant for this review:

- *Non-English (or dutch)*
  I do not understand other languages well enough to include texts that are not written in either English or Dutch in this review.
- *Non-Open acces*
  For obvious reasons I have to be able to read the full text to be able to include it in this review.
- *Not applicable to single machine problems*
  As the scheduling problem in my assignment concerns a single machine problem, the proposed solution needs to be applicable to the single machine problem in order to be relevant to my research.

Search terms & strategy

The following strings were used as search terms in this review:

1. heuristic AND scheduling AND "weighted completion time"
2. heuristic AND scheduling AND "sequence dependent setup times"
3. weighted completion time" AND "sequence dependent setup times"

We first search Scopus using the strings as formulated above, we then reduce the number of results by applying the exclusion criteria. We then read the abstracts of all the texts that are left over to determine whether the text actually is relevant and therefore needs be included or is irrelevant and can be excluded. Finally we study the full texts to obtain the heuristics used, determine the characteristics of the problem and determine the applicability of the heuristic to the scheduling problem at ProRail. We then use this to construct a conceptual matrix.
The table below (table 1) deconstructs the exact search queries used. The queries are listed in full in the literature review appendix (appendix A - systematic literature review)

TITLE-ABS-KEY	LIMIT-TO	
	ACCESSTYPE	EXACTKEYWORD (OR)
heuristic AND scheduling AND "weighted completion time"	*ACCESSTYPE(OA)*	*"Single Machine"* *"Single Machine Scheduling Problems"* *"Single Machines"* *"Single- Machines"* *"Single-machine"* *"Single-machine Scheduling"*
heuristic AND scheduling AND "sequence dependent setup times"	*ACCESSTYPE(OA)*	*"Single Machine Scheduling"* *"Single Machine Scheduling Problems"* *"Single-Machine"*
scheduling AND heuristic AND "single machine"	*ACCESSTYPE(OA)*	*"Sequence Dependent"* *"Sequence Dependent Setup"* *"Sequence Dependent Setup Time"* *"Sequence Dependent Setup Times"* *"Sequence Dependent Setups"* *"Sequence Independent Setup"* *"Sequence-dependent"* *"Sequence-dependent Setup"* *"Sequence-dependent Setup Costs"* *"Sequence-dependent Setup Time"* *"Sequence-dependent Setup Times"* *"Sequence-dependent Setups"* ***AND*** *"Total Completion Time"* *"Total Weighted Completion Time"*
"weighted completion time" AND "sequence dependent setup times"	*ACCESSTYPE(OA)*	*"Single Machine"* *"Single Machine Scheduling"* *"Single Machine Scheduling Problems"* *"Single Machines"* *"Single-machine"* *"Single-machine Scheduling"*

Table B1: Deconstruction of the exact search strings

Search results

Search string	Date of search	Number of entries before exclusion	Number of entries after exclusion
heuristic AND scheduling AND "weighted completion time"	21/01/2020	210	3
heuristic AND scheduling AND "sequence dependent setup times"	21/01/2020	534	2
scheduling AND heuristic AND "single machine"	21/01/2020	8743	1
"weighted completion time" AND "sequence dependent setup times"	21/01/2020	42	4
total		**9525**	**10**

Table B2: systematic literature review search results

Conceptual matrix

The following table (table B3) displays the conceptual matrix constructed as a result of this literature review. It notes per article the heuristic(s) used, the specific characteristics that applied to the problem and the applicability of the solution to the problem at ProRail which is a combination of the complexity, the relevance and the proven efficiency. The applicability is also rated 1 to 5, 1 being the lowest and 5 being the highest.

Article	Heuristic	Characteristics	Applicability
Moghaddam, A., Teghem, J., Tuyttens, D., Yalaoui, F., Amodeo, L.	*4 types of Implicit enumeration following a branching scheme*	*Bi-objective, rejection*	*medium-low (2)*
Lee, W.-C., Wang, W.-J., Shiau, Y.-R., Wu, C.-C.	*Branch-and-bound based on a combination of WSPT and EDD*	*Deteriorating jobs, multi-agent common resource*	*Low (1)*
Mosheiov, G., Sarig, A.	*Dynamic programming with a WSPT based heuristic*	*Due dates*	*High (5)*
Arroyo, J.E.C., Dos Santos Ottoni, R., De Paiva Oliveira, A.	*Multi-objective algorithm based on Variable neighbourhood search (VNS)*	*Multi-objective, Due windows, Sequence dependent setup times*	*Medium (3)*
Ángel-Bello, F., Álvarez, A., Pacheco, J., Martínez, I.	*GRASP with Tabu search*	*Preventive maintenance, Sequence dependent setup times*	*Low (1)*
Lin, M.-Y., Kuo, Y	*Mixed integer programming with a WSPT based heuristic*	*Sequence dependent setup times with job families*	*Medium-high (4)*
Wang, X.-Y., Wang, J.-J.	*WSPT*	*Deterioration and learning effects, past sequence dependent setup times*	*medium-low (2)*
Lee, W.-C.	*WSPT*	*Learning effects, past sequence dependent setup times*	*medium-low (2)*
Wang, J.-B., Li, J.-X.	*WSPT*	*Learning effects, past sequence dependent setup times*	*medium-low (2)*
Wang, J.-B., Wang, D., Wang, L.-Y., Lin, L., Yin, N., Wang, W.-W.	*WSPT*	*Exponential learning effects, past sequence dependent setup times*	*low (1)*

Table B3: conceptual matrix

6.1. Conclusion

The first thing we notice is that we haven't found a single document about a scheduling problem with the exact same characteristics as our problem at hand at ProRail. They either don't have all characteristics or have extra properties that change the nature of the scheduling problem. e.g. deteriorating or learning effects.
Secondly we want to note that the last 4 reviewed documents, which are very similar in nature, don't concern the performance of the proposed heuristics. They simply show that the WSPT rule is optimal under specific circumstances.
Finally almost all, i.e. nine out of eleven, based their method on the WSPT heuristic, often referred to as the WSPT-rule. This is in line with what we already knew about simplistic scheduling problems with the objective of minimizing the total weighted completion time. Which is that the WSPT-rule is proven to result in an optimal solution for those more simplistic variants of the problem. (Pinedo, 2008) It therefore seems clear that we should also look to apply a WSPT-rule based heuristic to our scheduling problem at ProRail.

Exact search queries

1. TITLE-ABS-KEY (heuristic AND scheduling AND "weighted completion time") AND (LIMIT-TO (ACCESSTYPE(OA))) AND (LIMIT-TO (EXACTKEYWORD , "Single Machine") OR LIMIT-TO (EXACTKEYWORD , "Single Machine Scheduling Problems") OR LIMIT-TO (EXACTKEYWORD , "Single Machines") OR LIMIT-TO (EXACTKEYWORD , "Single- Machines") OR LIMIT-TO (EXACTKEYWORD , "Single-machine") OR LIMIT-TO (EXACTKEYWORD , "Single-machine Scheduling"))
2. TITLE-ABS-KEY (heuristic AND scheduling AND "sequence dependent setup times") AND (LIMIT-TO (ACCESSTYPE(OA))) AND (LIMIT-TO (EXACTKEYWORD , "Single Machine Scheduling") OR LIMIT-TO (EXACTKEYWORD , "Single Machine Scheduling Problems") OR LIMIT-TO (EXACTKEYWORD , "Single-Machine"))
3. scheduling AND heuristic AND "single machine" AND (LIMIT-TO (ACCESSTYPE(OA))) AND (LIMIT-TO (EXACTKEYWORD , "Sequence Dependent") OR LIMIT-TO (EXACTKEYWORD , "Sequence Dependent Setup") OR LIMIT-TO (EXACTKEYWORD , "Sequence Dependent Setup Time") OR LIMIT-TO (EXACTKEYWORD , "Sequence Dependent Setup Times") OR LIMIT-TO (EXACTKEYWORD , "Sequence Dependent Setups") OR LIMIT-TO (EXACTKEYWORD , "Sequence Independent Setup") OR LIMIT-TO (EXACTKEYWORD , "Sequence-dependent") OR LIMIT-TO (EXACTKEYWORD , "Sequence-dependent Setup") OR LIMIT-TO (EXACTKEYWORD , "Sequence-dependent Setup Costs") OR LIMIT-TO (EXACTKEYWORD , "Sequence-dependent Setup Time") OR LIMIT-TO (EXACTKEYWORD , "Sequence-dependent Setup Times") OR LIMIT-TO (EXACTKEYWORD , "Sequence-dependent Setups")) AND (LIMIT-TO (EXACTKEYWORD , "Total Completion Time") OR LIMIT-TO (EXACTKEYWORD , "Total Weighted Completion Time"))
4. TITLE-ABS-KEY ("weighted completion time" AND "sequence dependent setup times") AND (LIMIT-TO (ACCESSTYPE(OA))) AND (LIMIT-TO (EXACTKEYWORD , "Single Machine") OR LIMIT-TO (EXACTKEYWORD , "Single Machine Scheduling") OR LIMIT-TO (EXACTKEYWORD , "Single Machine Scheduling Problems") OR LIMIT-TO (EXACTKEYWORD , "Single Machines") OR LIMIT-TO (EXACTKEYWORD , "Single-machine") OR LIMIT-TO (EXACTKEYWORD , "Single-machine Scheduling"))

Scopus source exports per query

1. Moghaddam, A., Teghem, J., Tuyttens, D., Yalaoui, F., Amodeo, L.
56219133500;6701582057;6603225959;6507073882;19638286300;
Toward an Efficient Resolution for a Single-machine Bi-objective Scheduling Problem with Rejection
(2019) Foundations of Computing and Decision Sciences, 44 (2), pp. 179-211.
https://www.scopus.com/inward/record.uri?eid=2-s2.0-85067407391&doi=10.2478%2ffcds-2019-0010&partnerID=40&md5=194d0d0c21f8ed0f0704d6c0b39a1dd7
DOI: 10.2478/fcds-2019-0010
DOCUMENT TYPE: Article
PUBLICATION STAGE: Final
ACCESS TYPE: Open Access
SOURCE: Scopus

Zhang, X.
34877996400;
Single machine and flowshop scheduling problems with sum-of-processing time based learning phenomenon
(2017) Journal of Industrial and Management Optimization, 13 (5), pp. 1-14.
https://www.scopus.com/inward/record.uri?eid=2-s2.0-85063214664&doi=10.3934%2fJIMO.2018148&partnerID=40&md5=63586ce3c324cb5f4334ed29a64e74be
DOI: 10.3934/JIMO.2018148
DOCUMENT TYPE: Article
PUBLICATION STAGE: Final
ACCESS TYPE: Open Access
SOURCE: Scopus

Lee, W.-C., Wang, W.-J., Shiau, Y.-R., Wu, C.-C.
8593735700;36119383200;7102802668;35202431900;
A single-machine scheduling problem with two-agent and deteriorating jobs
(2010) Applied Mathematical Modelling, 34 (10), pp. 3098-3107. Cited 42 times.
https://www.scopus.com/inward/record.uri?eid=2-s2.0-77952888252&doi=10.1016%2fj.apm.2010.01.015&partnerID=40&md5=730f72d9d979a50dd2f7edc310363327
DOI: 10.1016/j.apm.2010.01.015
DOCUMENT TYPE: Article
PUBLICATION STAGE: Final
ACCESS TYPE: Open Access
SOURCE: Scopus

Mosheiov, G., Sarig, A.
7004682254;23036674700;
Scheduling a maintenance activity to minimize total weighted completion-time
(2009) Computers and Mathematics with Applications, 57 (4), pp. 619-623. Cited 41 times.
https://www.scopus.com/inward/record.uri?eid=2-s2.0-58349089245&doi=10.1016%2fj.camwa.2008.11.008&partnerID=40&md5=9a96af8135395bfda0e1876c0f83cd5f
DOI: 10.1016/j.camwa.2008.11.008
DOCUMENT TYPE: Article
PUBLICATION STAGE: Final

ACCESS TYPE: Open Access
SOURCE: Scopus

2. Arroyo, J.E.C., Dos Santos Ottoni, R., De Paiva Oliveira, A.
7103143879;54881279700;8651950400;
Multi-objective variable neighborhood search algorithms for a single machine scheduling problem with distinct due windows
(2011) Electronic Notes in Theoretical Computer Science, 281, pp. 5-19. Cited 30 times.
https://www.scopus.com/inward/record.uri?eid=2-s2.0-84855280955&doi=10.1016%2fj.entcs.2011.11.022&partnerID=40&md5=10bb33a0dfd4d87100875b1374db80ad
DOI: 10.1016/j.entcs.2011.11.022
DOCUMENT TYPE: Conference Paper
PUBLICATION STAGE: Final
ACCESS TYPE: Open Access
SOURCE: Scopus

Ángel-Bello, F., Álvarez, A., Pacheco, J., Martínez, I.
56013683700;56817717600;7101665102;7101802806;
A heuristic approach for a scheduling problem with periodic maintenance and sequence-dependent setup times
(2011) Computers and Mathematics with Applications, 61 (4), pp. 797-808. Cited 21 times.
https://www.scopus.com/inward/record.uri?eid=2-s2.0-79651469503&doi=10.1016%2fj.camwa.2010.12.028&partnerID=40&md5=76f609f335135ca82fa7a29fc00ac3ff
DOI: 10.1016/j.camwa.2010.12.028
DOCUMENT TYPE: Article
PUBLICATION STAGE: Final
ACCESS TYPE: Open Access
SOURCE: Scopus

3. Lin, M.-Y., Kuo, Y.
57193802631;9633017500;
Efficient mixed integer programming models for family scheduling problems
(2017) Operations Research Perspectives, 4, pp. 49-55.
https://www.scopus.com/inward/record.uri?eid=2-s2.0-85016757814&doi=10.1016%2fj.orp.2017.03.001&partnerID=40&md5=35c222e99d3db10f078f2c142a270328
DOI: 10.1016/j.orp.2017.03.001
DOCUMENT TYPE: Article
PUBLICATION STAGE: Final
ACCESS TYPE: Open Access
SOURCE: Scopus

4. Wang, X.-Y., Wang, J.-J.
56388792000;55917088400;
Scheduling problems with past-sequence-dependent setup times and general effects of deterioration and learning
(2013) Applied Mathematical Modelling, 37 (7), pp. 4905-4914. Cited 25 times.
https://www.scopus.com/inward/record.uri?eid=2-s2.0-84872603704&doi=10.1016%2fj.apm.2012.09.044&partnerID=40&md5=5e260e5d14697775b63a4c425f183303
DOI: 10.1016/j.apm.2012.09.044
DOCUMENT TYPE: Article
PUBLICATION STAGE: Final
ACCESS TYPE: Open Access
SOURCE: Scopus

Lee, W.-C.
8593735700;
A note on single-machine scheduling with general learning effect and past-sequence-dependent setup time
(2011) Computers and Mathematics with Applications, 62 (4), pp. 2095-2100. Cited 17 times.
https://www.scopus.com/inward/record.uri?eid=2-s2.0-80051796890&doi=10.1016%2fj.camwa.2011.06.057&partnerID=40&md5=9337ce2a597988951bbe3b2e77640f95
DOI: 10.1016/j.camwa.2011.06.057
DOCUMENT TYPE: Article
PUBLICATION STAGE: Final
ACCESS TYPE: Open Access
SOURCE: Scopus

Wang, J.-B., Li, J.-X.
8449058700;36664885200;
Single machine past-sequence-dependent setup times scheduling with general position-dependent and time-dependent learning effects
(2011) Applied Mathematical Modelling, 35 (3), pp. 1388-1395. Cited 40 times.
https://www.scopus.com/inward/record.uri?eid=2-s2.0-78649329148&doi=10.1016%2fj.apm.2010.09.017&partnerID=40&md5=1f1e85d356e9343fd917dab53bdf1434
DOI: 10.1016/j.apm.2010.09.017
DOCUMENT TYPE: Article
PUBLICATION STAGE: Final
ACCESS TYPE: Open Access
SOURCE: Scopus

Wang, J.-B., Wang, D., Wang, L.-Y., Lin, L., Yin, N., Wang, W.-W.
8449058700;57198728903;55721459900;36014583600;35249419900;55714241400;
Single machine scheduling with exponential time-dependent learning effect and past-sequence-dependent setup times
(2009) Computers and Mathematics with Applications, 57 (1), pp. 9-16. Cited 63 times.
https://www.scopus.com/inward/record.uri?eid=2-s2.0-57349083985&doi=10.1016%2fj.camwa.2008.09.025&partnerID=40&md5=baa8e519cb5f81299e0fb4fa74fc7c83

DOI: 10.1016/j.camwa.2008.09.025
DOCUMENT TYPE: Article
PUBLICATION STAGE: Final
ACCESS TYPE: Open Access
SOURCE: Scopus