A Decision Support System for real-time track assignment at railway yards

Master Thesis

for the

UNIVERSITY OF TWENTE.

written by

Bram B. W. Schasfoort
(born June 9th, 1994 in Hilversum, Netherlands)

at

ARCADIS Design & Consultancy for natural and built assets

Date of the public defense:
July 5th, 2019

Members of the Thesis Committee:
Prof. Dr. Ir. Eric van Berkum
Dr. Kostantinos Gkiotilitis
Vincent den Ouden, MSc

Abstract

English

In this thesis, we study the Real-Time Track Assignment Problem (RT-TAP), a real-time assignment problem that arises from the high percentage of stochastic arrivals of freight trains and the large quantity of last-minute parking requests at railway yards. We show that the RT-TAP is NP-Hard and provide a mixed integer program for solving the RT-TAP by minimizing the total weighted delay of trains. Because of its computational complexity, we develop a problem specific Genetic Algorithm (GA) and compare it with a First Scheduled First Served (FSFS) heuristic. Smaller instances show that there is no optimality gap between the Brute Force (BF) approach and the GA. The heuristic approaches are tested on two Real-Time simulations where we consider 74 inbound trains and 9 tracks. In order to define the effect of the input on the two models, we excluded in one of the simulations the track length i.e. all trains can be assigned to all tracks. Although we saw that the output of the GA remained the same, the FSFS heuristic was not able to show results as good as the GA. Therefore, we conclude that we developed a Decision Support System (DSS) that cannot only stabilize, but also improve the current decision-making process with regards to real-time track assignment at railway yards.

Keywords: Track Assignment; Real-Time; Rail Operations; Minimizing delays; Scheduling.

Nederlands

In deze scriptie bestuderen we het realtime spoor toewijzingsprobleem (RT-TAP), een probleem dat ontstaat door het hoge percentage van stochastische aankomsten van goederentreinen en het grote hoeveelheid last-minute parkeerverzoeken op emplacementen. We laten zien dat het RT-TAP NP-hard is en ontwikkelen een mixed integer program gemaakt voor het oplossen van het RT-TAP door de totale gewogen vertraging van treinen te minimaliseren. Vanwege de computationele complexiteit ontwikkelen we een probleemspecifiek Genetisch Algorithme (GA) en vergelijken deze met een First Scheduled First Served (FSFS) heuristiek. Bij het testen van kleinere scenario's tonen we aan dat er geen optimaliteitskloof is tussen een Brute Force (BF) benadering en het GA. De twee heuristieken worden getest op twee realtime simulaties met 74 inkomende treinen en 9 sporen. Om het effect van de invoer op de twee modellen te definiëren, hebben we in een van de simulaties de spoorlengte uitgesloten, d.w.z. alle treinen kunnen aan alle sporen worden toegewezen. Hoewel we zagen dat de resultaten van het GA hetzelfde bleven, had de FSFS heuristiek moeite met het produceren van even goede resultaten Daarom concluderen we dat we succesvol een beslissingsondersteuningssysteem (DSS) hebben ontwikkeld die niet alleen het huidige besluitvormingsproces met betrekking tot realtime spoortoewijzing kan stabiliseren, maar ook kan verbeteren.

Trefwoorden: Spoortoewijzing; Realtime; Spoorvervoer; Vertragingen minimaliseren; Planning.

Preface

The thesis on hand marks another milestone, which is the completion of the Master in Civil Engineering and Management. The research itself consisted of 7 months of hard work and dedication to the subjects of railway operations and mathematical optimization. The report is the result of a study about a Decision Support System for Real-Time track assignment at railway yards. Interestingly, when I started with this research at Arcadis, I can honestly say that my knowledge about railway operations and mathematical modeling was very limited. Nonetheless, the reason that everything was very new to me, was probably one of the biggest reasons that the subject remained interested, all the way to the end. I can therefore summarize this last period as one big learning experience, which I enjoyed a lot!!

Of course, achieving something like this was not without help. Therefore, I would hereby like to show my sincere gratitude to the following people whom without this project would not have been possible.

Konstantinos Gkiotsalitis,
My UT supervisor who without his dedications and the amount of energy he put in this project, I would never have been able to achieve the quality that I present you in this thesis! I highly appreciated the speed you provided me feedback and the time you took during the meetings to discuss every matter.

Vincent den Ouden,
My supervisor from Arcadis who always would find a moment to take the time and discuss everything that was going on with this project. Furthermore, I would like to thank you for giving me the freedom, that I probably also needed, during this research to use my own approaches. Heel erg bedankt!

I would like to give a special thank you to my brother ***Job Schasfoort***, who helped me greatly with learning and understanding the C++ programming language, ***Oskar Eikenbroek*** for helping me with the development of the methodology, and ***Andre van Es*** who arranged the opportunity to do my graduation at Arcadis. Further, I would like to thank my parents ***Adele Veldboer*** and ***Egbert Schasfoort*** for their support and encouragement during my whole study period, starting with the MBO, Bachelor, and eventually also this Master! Finally, a big thanks to all the nice colleagues from Arcadis including the football table on the third floor for all the inspirational moments, ProRail for making the data available for me to use during the numerical experiment, and everybody who I did not mention yet and supported me during the whole thesis process.

Hope you enjoy!

Bram Schasfoort,

Enschede, June 11th, 2019

Contents

List of Figures

List of Tables

List of acronyms

ARS	Automatic Route Setting
BF	Brute Force method
CBG	Centraal Bediend Gebied
DSS	Decision Support System
DY	Destination Yard
FIFO	First In First Out
FSFS	First Scheduled First Served
GA	Genetic Algorithm
HC	Hill Climbing
LIFO	Last In First Out
RIM	Rail Infrastructure Manager
RRT	Railcar Retrieval Problem
RSC	Rail Service Center
RT-TAP	Real-Time Track Assignment Problem
SA	Simulated Annealing
TAP	Train Assignment Problem
TD	Train Dispatcher
TS	Tabu Search
TUSP	Train Unit Shunting Problem
TY	Temporary Yard
VRP	Vehicle Routing Problem
Whz	Waalhaven Zuid

Chapter 1

Introduction

Since the use of the first railway in the early 19th century, rail freight transportation has always played an important role in the transportation sector (Network Rail, 2016). As discussed by Givoni and Banister (2013), after long and short haul maritime transport is, rail freight transport is the mode with the least CO2 emission per kg/tonne-km.

While the rail transportation sector is still growing, the increase in demand could cause capacity problems on the network. For example, US transportation officials already stated their concerns that future capacity limitations of the American railway system are likely to result in a degradation of speed and reliability of the network (U.S. Railroad, 2008). While a need for further expansion of railway networks is strongly desired around the globe, the most common solution would be an expansion of the railway network by constructing new infrastructure. However, constructing new infrastructure is generally decided by policymakers and involves a long-term decision process including large expenses (Boysen et al., 2012; Narayanaswami and Rangaraj, 2011). Because of high construction costs, it might be better to increase the occupancy and transit rate on the network in order to comply with the increase of traffic.

In order to comply with the demand for rail freight transporters, timetables of trains are determined one year in advance depending on the country. When a train is driving according to its schedule, the Automatic Route Setting (ARS) automatically assigns a train to a track (Pachl, 2004). Ideally, a Train Dispatcher (TD), who is responsible for the movement of trains in an assigned area (for example a railway yard), only needs to intervene when unplanned events occur. As discussed by Piner and Condry (2017), there is a possibility that unplanned events result in disturbances and/or disruptions. Now, when a delayed freight train arrives at a railway yard, the track needs to be assigned manually by the TD (D'Ariano, 2015; Josyula et al., 2018). Because of the large complexity of optimally assigning trains to tracks, one can imagine that planning delayed trains at real-time is very complicated especially when inexperienced TDs are involved (U.S. Department of Transportation, 1999).

When considering the arrival times of rail freight trains in Europe, it appears that 67% arrives within a 15-minute time frame of their original schedule. Furthermore, 19% of the trains were more than 3 hours delayed, and 4% arrived even after 24 hours or their original schedule (European Commission, 2014). Concerning the Netherlands, historical data from ProRail (2015) shows that 33% of the trains that are originally planned trains arrive within a $\pm$ 5-minute time frame of a train's original planned time at their destination yard. 48% of the trains arrive more than 5 minutes later than originally scheduled, where 26% and 15% of these trains arrive even more than one and two hours too late respectively (see table 1.1).

Table 1.1: Delays of freight trains; original compared to executed data at Waalhaven Zuid (ProRail, 2015)

	Too early (in min)			***On time***	***Too late (in min)***				***Totals***
	+60	***60 - 30***	***30 - 5***	***-5 to +5***	***5 - 30***	***30 - 60***	***60 - 120***	***+120***	
# of trains	16	8	25	85	40	18	28	40	260
Percentage	6%	3%	10%	33%	15%	7%	11%	15%	100%
Totals :	19% *too early*			33%	48% *too late*				

Large deviations in arrivals are likely to be related to the fact that rail freight is a part of a supply chain that also includes other modes such as ship and truck transportation. If one of these modes is delayed, it might cause a knock-on delay on the train that is waiting for its cargo. This could mean that the freight train leaves the yard already with a delay. Furthermore, when focusing on the distance traveled by freight trains, most of them are likely to cross borders of different countries resulting in long distance transport.

When we consider the performance of freight trains on any railway network, we also need to consider the other rail users. This means that if a train is delayed, it needs to drive in between other trains without influencing the other users' schedule. The complexity arises because this train has to perform on the mixed network including the different characteristics of other trains such as speed, acceleration, deceleration, and cargo. When a train is delayed, it can be temporarily stored on a side track so that the train that drives on time can pass. In addition, this later statement together with that trains need to travel long distances explains the large percentage of trains that have more than two or three hours of delay when arriving at their destination (Timmermans, 2018). As substantiated by Briggs and Beck (2007), delays typically follow an exponential distribution. Because of the high percentage and the high variance of delays, changing the assignment of freight trains at the execution phase is not a trivial task and is therefore related to the experience and adequate handling of the TD.

To add further complexity to the assignment process, high fluctuation on demand from the client side results in a late notice of trains requesting paths. For example, table 1.2 shows that 66% of the executed trains are planned one year in advance, i.e their arrival is known a year in advance. 90% of the executed trains is known when making the original planning, which is done 3 days before execution. This latter means that 10% of the trains will make a request at the last minute whether they can park at the yard.

Table 1.2: Executed trains from predetermined plan at Waalhaven Zuid (ProRail, 2015)

	Year plan	***Original plan***	***Scheduled plan***	***Executed***
# of trains	389	515	573	573
Percentage	66%	90%	100%	100%

The large delays on the network and the high percentage of last-minute requests contribute to the complexity of assigning trains to tracks at real-time, further referred to as the Real-Time Track Assignment Problem (RT-TAP). If a suitable fit is not possible, inbound trains might need to wait or be rerouted to another railway yard, which can result in larger delays. In addition, the large delays of trains, and the high percentage of trains scheduled with a small prior notice underscores the number of manual decisions made at railway yards and therefore highlights the need for a Decision Support System (DSS) for the allocation of trains to track at real-time.

To tackle this issue, this study models the RT-TAP and investigates mathematical optimization techniques that aim to minimize the total delays of outbound trains at a railway yard. The input of the model is the set of tracks and trains. The set of tracks includes the track number and its corresponding length. The set of trains incorporate the length of the train, planned arrival and departure time, the time that the train needs to spend at the yard, and a train weighting. The train weighting is related to the train's schedule in relation to the schedule of other trains, the type of train, or the type of cargo (U.S. Department of Transportation, 1999). The output of the model provides an optimal solution on which track each train should be assigned to in order to minimize the total weighted delay on outbound trains from the railway yard. The TD can make the final decision based on the output generated by the proposed DSS.

The remainder of this study is structured as follows. In chapter 2 we provide a summary of the relevant literature. After this, we identify the research gap, explain the research goal, and discuss the main contributions of this thesis with regards to the state-of-the-art in chapter 3. Chapter 4 describes the overall framework and introduces the mathematical program for solving the RT-TAP by using a CPLEX solver. Building upon this, we discuss three different solution methods for solving the RT-TAP in chapter 5. First, we will explain a Brute Force (BF) approach to solve the problem for smaller instances. For larger instances, we will introduce a problem specific Genetic Algorithm (GA), and a First Scheduled First Served (FSFS) heuristic solution method. In chapter 6, we provide a Numerical Experiment on smaller instances to determine the optimality gap between the three developed solution algorithms. Furthermore, we will show a real-time experiment on a railway yard in the western part of the Netherlands named Waalhaven Zuid based on historical data provided by the Rail Infrastructure Manager (RIM) of the Netherlands. Finally, discussions and conclusions are outlined in chapter 7 and 8 respectively.

Chapter 2

Related Studies

This section aims to describe the current literature available in the field of railway operations with regards to minimizing the total weighted delays, train rescheduling and rerouting, and decision-making processes in real-time control.

Train rescheduling & rerouting decision making

Scheduling is a decision-making process that involves identifying, assessing and making appropriate decisions to solve a problem (Josyula et al., 2018) with a goal to optimize one or more objectives (Pinedo, 2016). Mathematical approaches for scheduling problems in railway networks are extensively studied. With regards to track assignment problems, most works consider sorting and scheduling problems at marshaling or shunting yards, such as Hansmann and Zimmermann (2008). Shunting movements can be described as the process where a unit is driven to a depot track from a platform in the station and is induced whenever the train composition changes on successive train services (Boysen et al., 2012; Gatto et al., 2009; Haahr and Lusby, 2017).

Jaehn et al. (2018, 2015) describes the assignment problems including shunting operations as the Railcar Retrieval Problem (RRT). With this problem, each freight car receives a priority value which is linked with the due date of each outbound train. The main objective of this problem is to minimize the total shunting operation costs by minimizing the total weighted departure of all outbound trains at a shunting yard. Gestrelius et al. (2017) developed an integer programming model for scheduling shunting tasks as well as allocating arrival yard tracks and classification bowl tracks. More effective schedules can be found, and a variety of characteristics can be optimized, including shunting work effort, number or cost of tracks, and shunting task start times. Haahr et al. (2017) describes the Train Unit Shunting Problem (TUSP), which entails assigning train units from a depot or shunting yard to scheduled train service in such a way that the resulting operations are without conflict. An important constraint from the TUSP is that all tracks must be processed in Last In First Out (LIFO) order, which means that trains can only enter from one side of the yard. The Train Assignment Problem (TAP) is to determine the maximum number of trains that can be assigned to a yard according to the timetable and without the use of shunting. Gilg et al. (2018) shows that the TAP is NP-hard and presents two integer programming models for solving this problem. The approach integrates track lengths along with the three most common types of yard layouts: First In First Out (FIFO), LIFO, and FREE tracks, where FREE is a combination of both LIFO and FIFO.

Minimizing delays is a multidisciplinary problem that is widely discussed in many different fields (Schachtebeck and Schöbel, 2008, 2010; Schöbel, 2007). A survey on Problem models and solutions approached with regards to the rescheduling in railway networks is developed by Fang (2015). Several works such as Dollevoet et al. (2011) and Schöbel (2009) discuss minimizing the total (weighted) delays of trains, and use fast heuristic approaches for solving such problems (Dollevoet and Huisman, 2014).

Because the models presented above mostly include tactical planning, arrival times of trains can be modeled by including a stochastic and robust extension of a model in order to consider uncertainties in the optimization process (Boysen et al., 2012; Briggs and Beck, 2007; Gilg et al., 2018).

Real-time algorithms for rescheduling railway systems

In the previous section we discussed several assignment and retrieval problems, however, most of them are performed at the tactical level and might therefore not be valid when considering the stochastic arrival rates of trains. Hence, a problem solution method for solving assignment problems at real-time is therefore required (Cai and Goh, 1994). The works found in the literature mostly consist of real-time optimization models for solving railway networks when disruptions occur (Bettinelli et al., 2017). Cacchiani et al. (2014) presents an overview of recovery models and algorithms for real-time railway rescheduling. As discussed by Dollevoet et al. (2010), railway disruption management is a combination of three different aspects; timetables, rolling stock and crew. Real-time rescheduling of long-distance high-speed trains in a highly disrupted situation is discussed by Zhan et al. (2015). They developed a Decision Support System (DSS) which found solutions for real-time rescheduling within 10 minutes of computation time. Other models developed by Fischettia and Monaci (2017) and Corman et al. (2010) found practical solutions for real-time train rescheduling within seconds. Winter and Zimmermann (2000) developed a heuristic approach that assigns trams to tracks at real-time level considering the departure of the trams of the next day. If a global optimum is not found at real-time, trams can be reassigned again afterward in other with comply to the schedule of each tram for the next day.

The works described above all present models solved at real-time. The time limitation to solve a problem at real-time is not strongly specified. For example, while Fischettia and Monaci (2017) and Corman et al. (2010) tried to find practical solutions within seconds, Zhan et al. (2015) accepted finding solutions within 10 minutes of computation time. The real-time application for solving the Real-Time Track Assignment Problem (RT-TAP) is therefore dependent on the time between the last accurate arrival update and the actual arrival of a train.

Chapter 3

Research Gap, Aim & Relevance

This chapter highlights the research gap in current literature and states the aim that remains central in this research. Furthermore, we will justify the relevance of conducting research on this topic by discussing it on three different categories.

3.1 Research gap & context

In the previous chapter, we argued relevant literature related to the Real-Time Track Assignment Problem (RT-TAP). We discussed the many models related to Real-Time rescheduling methods and saw different approaches for track assignment. While the two models presented by Gilg et al. (2018) and Winter and Zimmermann (2000) were closely related to the RT-TAP, the models presented do not offer an exact solution method for this problem. Problems with the application of this model presented by Gilg et al. (2018) are that it is performed at the tactical level and considers the schedule as fixed. It includes an expected deviation, but at real-time, the arrival and departure times can still vary. The model presented by Winter and Zimmermann (2000) is close to a solution of the RT-TAP as it considers trams to be parked at a yard and includes the arrival and departure time at real-time. However, the model is developed on a single stud yard and focuses on the parking, and not the transit at a yard. Also, when assignments are not optimal, it considers the tram to easily exit the current track and rearrange a track assignment. Especially the latter is something that should be avoided because of the significant length and weight of freight trains. This makes the solution algorithms presented by Gilg et al. (2018) and Winter and Zimmermann (2000) not suitable as a solution method for the RT-TAP in trains operations.

In this work, we will investigate the RT-TAP and provide a mathematical approach for solving the problem at hand. Furthermore, we will assess the potentials of using mathematical optimization for allocating freight trains to tracks at any railway yard at real-time.

3.2 Research aim

In the previous section, we identified the research gap from existing literature. In this section, we translate this gap into a problem definition and state the aim that is central in this research.

Problem definition

Railway operations can deviate from original timetables because of events such as disruptions on the network. For this reason, arrival and departure sequences can change at real-time. Because of the size of railway yards, they could be perceived as unclear which makes fast decision making difficult and sensitive for mistakes. Furthermore, decisions at real-time are made by a Train Dispatcher (TD) and are dependent on the experience and adequate handling of a TD. Overlap in the schedule could cause conflict and resolve in more delays on other trains.

Research aim

To develop a fast and easy usable Decision Support System (DSS) that, if delays occur, reassigns inbound trains to tracks in order to minimize the total weighted delays of outbound trains at any railway yard at real-time.

3.3 Contribution of this Thesis

This section aims to justify the relevance to conduct a research on this topic. The relevance is discussed on three different categories.

Scientific relevance

Ideally, the best-case scenario is that all trains drive according to the original schedule and therefore a TD does not need to intervene at real-time. However, we saw that delayed freight trains performing on the network and last-minute path requests from transportation companies are not exceptional. Currently, the last-minute assignment of trains to tracks is done manually by a TD. Improving the manual allocation of delayed trains with a DSS is, therefore, an interesting field to further investigate. In order to be able to improve this manual decision making, mathematical approaches are already widely discussed in the literature. However, with regards to the research gap stated in section 3.1, we can conclude that the area of assigning delayed trains to tracks when they arrive at an already busy railway yard at real-time is still unexplored. This latter statement is also supported by Gilg et al. (2018), who indicates the limitation of literature with regards to the impact of a trains' delay on the planned depot schedules. This said, and considering the large computational complexity of similar existing track assignment problems, solving such complex problems within a real-time time frame would provide an incremental contribution to the state-of-the-art.

This research will provide insights on real-time track assignment of freight trains and the possibility of improving the assignment process using mathematical optimization. While manual track assignment is non-trivial, the new proposed DSS can be used as a support for a TD for real-time track assignment at railway yards. In this thesis, an important criterion will be the practical applicability of the proposed solution method.

Social relevance

As stated in the introduction, there is a need for increasing the capacity and transit on any railway network. However, railway networks are still triggered by large delays. Hoenders (2018) further addresses the relevance of improving network performance when the rail sector would like to continue competing with the other freight transportation modes. He points out that there are two options with regard to the large delays on the network; either we try to make sure that trains will drive on time, or we accept that delays are there and try to make the best of it. While a network without delays would be ideal for the performance because most freight trains cross several borders, and are therefore also dependent on the network performance of the other countries, reducing delays is not as easy as it sounds. Furthermore, sending out trains that are not fully loaded just to make sure that they drive on their path is not a good idea. Besides the fact that the transportation companies need to comply to the demand of the customer, as stated by several rail freight transporters, the profit margin of the trip is on the last wagon of a train (Spoorcafe, 2018). Improving the assignment process and therefore optimizing the total delay may, therefore, be a much better solution when addressing the problem within the borders of a country. With all these mentioned reasons above, we can conclude the importance of this research from the social point of view. This statement brings us to the next section; the Managerial relevance.

Managerial relevance

The managerial benefit of this research lies in the possibility for Rail Infrastructure Manager (RIM)s to take better-supported decisions concerning real-time track assignment of trains. In the previous section, we showed that the need for real-time track assignment is there, and is only likely to increase as more trains will drive on the network. With the large complexity of the current track assignment problem, a mathematical approach seems like a good approach in order to improve any inefficiencies at a railway yard with regards to track assignment. The end of the day, the RIM is responsible for scheduling and maintaining the assignments of trains to tracks. In the introduction, we discussed the complexity to ensure trains driving within their time slot. No further deterioration or better reducing delays is important for the overall performance of the supply chain and therefore also the responsibility of the RIM. For that reason, using a DSS for assigning trains to tracks at real-time seems like the only possible solution when considering the process optimization and excluding expensive construction of new infrastructure at railway yards. From a managerial point of view, this research, therefore, shows high relevance when countries, and more specifically the RIMs, want to carry on with improving the performance of the rail transportation sector and continue to compete with the other modes of freight transportation.

Chapter 4

Methodology

This chapter aims to describe the problem formulation and explains the mathematical program used for solving the Real-Time Track Assignment Problem (RT-TAP). In the overall framework, we discuss a generalized setup of the problem and explain what kind of input should be provided into the model in order to make it work properly. The model is eventually tested in the program AMPL by using a CPLEX solver. The three files necessary for executing the model (model, data and the run file) can be found in appendix A, B and C.

The current situation includes several trains with the same Destination Yard (DY). At this railway yard, each train needs to perform some sort of operation and therefore must be parked at the yard. These operations could vary from changing the train driver to switching the locomotives of the train. Different operations mean therefore that the minimum time interval a train needs to remain at the DY differs per train. The assignment schedule of the trains is shown in figure 4.1. If everything would go according to plan, the tactical schedule would be the same as the real-time schedule and no conflicts would occur. However, if the red train receives a delay of 30 minutes, and the green train is still operating according to plan, all tracks will be occupied the moment when the black train is planned to arrive at the yard.

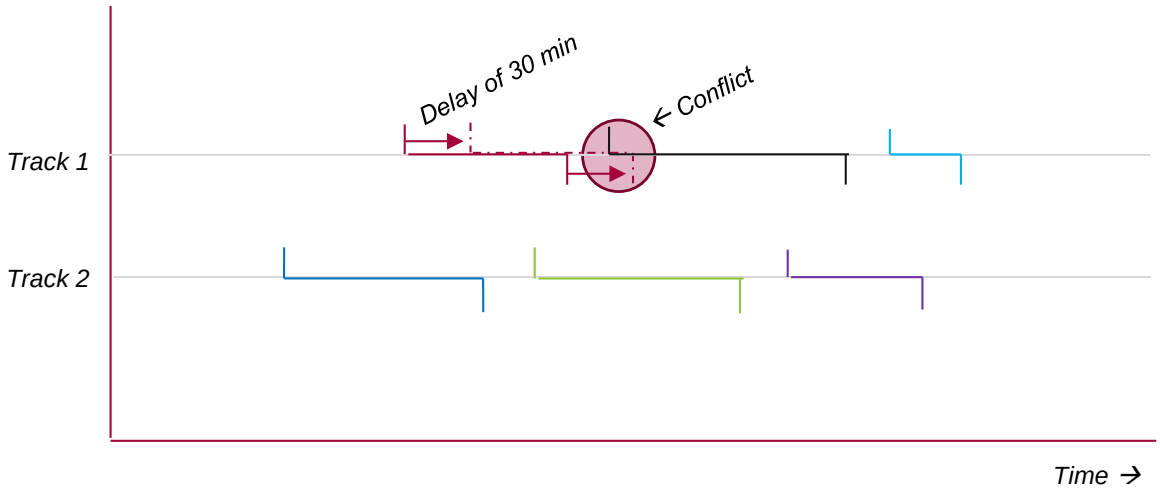

Figure 4.1: Visualization of the problem

As discussed in the introduction, delays on a network are very common, especially for freight trains. The complication in railway operations is that all trains are fixed by the rails, which makes overtaking rather difficult. However, it is possible to let another train pass by temporarily park the train on a side track. The sidetrack where trains can temporarily be parked at will be further referred to as the Temporary Yard (TY) (see figure 4.2). This procedure is currently already used to reduce knock-on delays on passenger and freight trains. In this way, the delays can be reduced by rearranging the arrival times and track assignment of trains. If, as described in the example, the red train would get a delay, it would be possible to let the black train temporarily park at the TY, assign the red train to the DY, and assign the black train again when the track has been cleared again at the DY. If the red train would arrive even later than planned, it could be more beneficial for the total delay to temporarily park the red train, let the black train pass, and assign the red train after the black train.

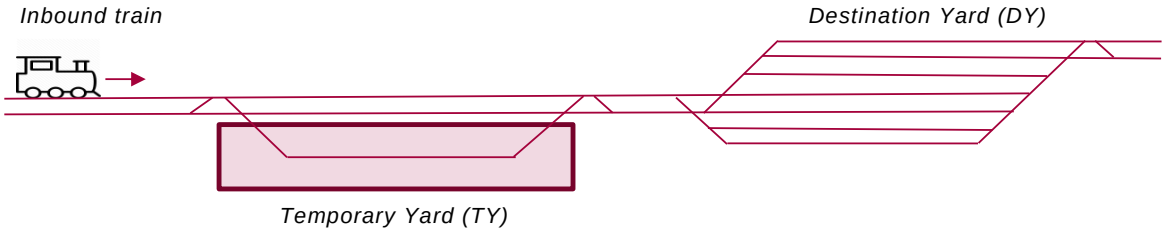

Figure 4.2: Schematic layout of the assignment situation

4.1 Overall framework

Consider a DY with a set of tracks $K = \{1, 2, ..., |K|\}$ where each track $k \in K$ has a fixed length m_k. At the yard, there is a set of trains $N = \{1, 2, ..., |N|\}$ assigned with each train $i \in N$ having its own original arrival time a_i, a original departure time d_i, a original time interval p_i that the train has to remain at the yard in order to execute its operations, and a length l_i. Because the assignment is dependent on the length of the train and track, train $i \in N$ can only be assigned to tracks that fit, and therefore $m_k \geq l_i$ when train $i \in N$ is assigned to track $k \in K$. At the tactical level, schedules are constructed so that all trains can be assigned at the yard if they drive according to their original schedule. However, as stated in the introduction, delays on freight trains are not exceptional, and therefore the arrival times of inbound trains are likely to change at real-time. When delays on trains occur, and no track is available at the yard, the trains should be re-assigned so that the total delay is minimized at the yard. Because some trains might have more priority than others, we consider minimizing the total weighted delay at the railway yard.

The model will determine a sequence of trains arriving at the yard by minimizing the total weighted delay. The total delay can be calculated by subtracting the original departure time (d_i) of each train $i \in N$ from the departure time $(\tilde{d}_i)$ determined by the model. Sequences are made by first checking all possible arrivals times of each train $i \in N$ at each track $k \in K$. The arrival time of each train at each track is called the latent arrival time of a train $(\mathbf{a}_i'^k)$. The latent departure time $(d_i'^k)$ is directly related to the latent arrival time by the minimum time a train has to spend at the yard (p_i). The model will then choose the best possible arrival time from all $\mathbf{a}_i'^k$ for each $i \in N$, and keeps the arrival time with the minimum possible delay, and denotes it as the determined arrival time $(\tilde{\mathbf{a}}_i)$. The determined departure time $(\tilde{d}_i)$ is again directly related to the determined arrival time by the time train $i \in N$ has to remain at the yard (p_i). A visualization of the sequential problem described above is presented in figure 4.3.

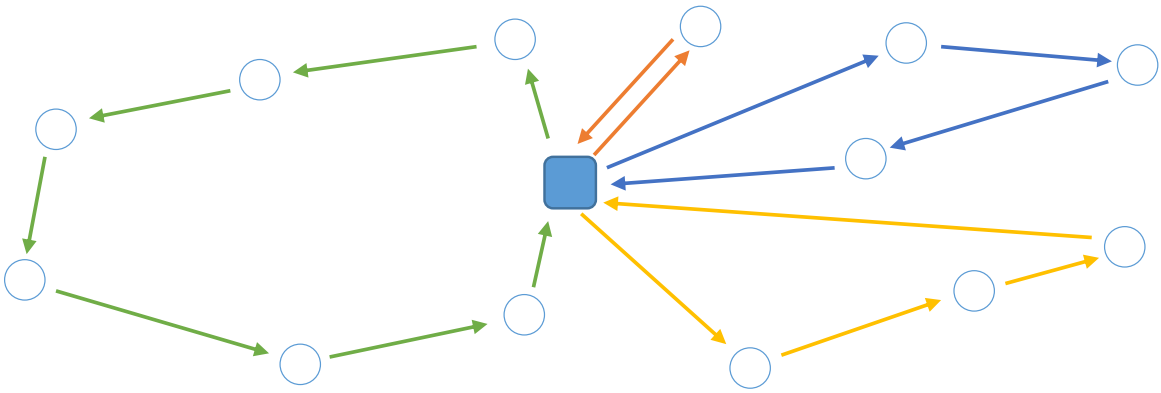

Figure 4.3: Visualization of the sequential problem where each sequence represents a track assignment, the square represents the beginning and end of each sequence, and each circle represents a train

In addition to the description above, the following assumptions are made in this model.

1. *There is unlimited storage at the TY:* From the moment that a new arrival time is known, it is assumed that the train will pass enough stops where it can temporarily park and wait until a track has been cleared at the DY.

2. *Trains cannot be shunted:* Trains arriving and departing at and from the DY only consists of block trains. This means that all cargo loaded on the train is obliged to arrive at its destination. For this reason, the train cannot be shunted, which makes the length of the train fixed.

3. *When new arrival times, departure times and/or delays are determined by the model, Automatic Route Setting (ARS) will always find a path on the network:* As discussed in the introduction, the possibility of freight trains running on the network is highly dependent on the possibility if a path has been cleared or not. Because path scheduling is also dependent on other trains running on the network, we assume that when the model determines new arrival, departure times, and/or delays, the ARS will always find a path on the network.

4. *No serial ordering is possible at same tracks:* Because of the significant length of all inbound trains, two trains cannot be parked at the same track at the same time. For this reason, it does not matter if the yard is a Last In First Out (LIFO), First In First Out (FIFO) or FREE yard (Gilg et al., 2018).

Constructing sequences with the Vehicle Routing Problem

Constructing sequences are already done in several problems. A well known sequential problem is the Vehicle Routing Problem (VRP). In the VRP, X number of trucks need to visit Y number of point, where each truck need to deliver of pick up some cargo. In a classical VRP, each point Y is known in advance and therefore the distance to each point is also known. The main objective in the VRP is to minimize to total distance that need to be traveled.

In the paper presented by Liong et al. (2008), a classical VRP is defined as follow: Let $G = (V, A)$ be a graph where $V = \{1, ..., n\}$ is a set of vertices representing cities with the depot is located at vertex 1, and A is the set of arcs. With every arc $(i, j) i \neq j$ is associated a non-negative distance matrix $C = (c_{i,j})$. In some contexts, $c_{i,j}$ can be interpreted as a travel cost or travel time. When C is symmetrical, it is often convenient to replace A by a set E of undirected edges. In addition, assume there are m available vehicles based at the depot, where $m_L < m < m_U$. When $m_L = m_U$, m is said to be fixed. When $m_L = 1$ and $m_U = n - 1$, m is said to be free. When m is not fixed, it often makes sense to associate a fixed cost f on the use of a vehicle. The VRP consists of designing a set of least-cost vehicle routes in such a way that:

I) each city in $V\backslash\{1\}$ is visited exactly once by exactly one vehicle;

II) all vehicle routes start and end at the depot;

III) some side constraints are satisfied.

Let $\mathbf{x}_{i,j}$ be an integer variable which may take value $\{0, 1\}, \forall\{i, j\} \in E\{\{0, j\} \colon j \in V\}$ and value $\{0, 1, 2\}, \forall\{0, j\} \in E, j \in V$. Note that $\mathbf{x}_{0,j} = 2$ when a route including the single customer j is selected in the solution.

The VRP described by Liong et al. (2008) can be formulated as the following integer program:

$$\min \sum_{i \neq j} d_{i,j} \mathbf{x}_{i,j} \tag{4.1}$$

subject to:

$$\sum_{j} \mathbf{x}_{i,j} = 1, \forall i \in V \tag{4.2}$$

$$\sum_{i} \mathbf{x}_{i,j} = 1, \forall j \in V \tag{4.3}$$

$$\sum_{i} \mathbf{x}_{i,j} \geq |S| - v(S), \{S \colon S \subseteq V\backslash\{1\}, |S| \geq 2\} \tag{4.4}$$

$$\mathbf{x}_{i,j} \in \{0, 1\}, \forall\{i, j\} \in E : i \neq J \tag{4.5}$$

In this formulation, 4.1, 4.2, 4.3 and 4.5 define a modified assignment problem (i.e. assignments on the main diagonal are prohibited). Constraint 4.4 is a sub-tour elimination constraint, where $v(S)$ is an appropriate lower bound on the number of vehicles required to visit all vertices of S in the optimal solution.

Note that in this formulation, the direction of the routes are not relevant. In the case of track assignment the sequences of the trains are because of the additional constraints of the arrival times.

Nomenclature

Tables 4.1, 4.2, 4.3 and 4.4 present the notations used for solving the RT-TAP.

Table 4.1: General subscripts

Notation	*Description*
$K = \{1, 2, ..., \|K\|\}$	Set of tracks at the railway yard.
$N = \{1, 2, ..., \|N\|\}$	Set of inbound trains in need of assignment at the railway yard.
$Q = \{1, 2, ..., \|K\|\}$	Set of dummy trains indicating the start of a sequence at each track $k \in K$.
$P = \{1, 2, ..., \|K\|\}$	Set of dummy trains indicating the end of a sequence at each track $k \in K$.
k	A track where $k \in K$.
i and j	A train where $i \in N$, $j \in N$, and $i \neq j$.
q	Dummy train where $q \in Q$ indicating the start of a sequence.
p	Dummy train where $p \in P$ indicating the end of a sequence.
T_S	Any possible sequential combination between trains i, j. $\{0, 1, ..., (2^N - 1)\}$.

Table 4.2: Parameters

Notation	*Description*
a_i	Original arrival time of train $i \in N$ at the yard at track k.
l_i	Length of each train $i \in N$.
p_i	Process time that train $i \in N$ must remain at the yard.
m_k	Length of each track $k \in K$.
w_i	Delay weighting of each train $i \in N$ indicating the importance of the train.
H	Minimum headway of two trains at the same track.
α	A large value (e.g. $1.000.000$).

Table 4.3: Variables

Notation	*Description*
d_i	Original departure time of train $i \in N$ from the yard.
$d_i'^k$	Latent departure time of train $i \in N$ from the yard at track $k \in K$.
$\tilde{d}_i$	Determined departure time by the model of train $i \in N$ from the yard.
S	A subset of T_S.

Table 4.4: Decision variables

Notation	*Description*
$\mathbf{x}_{i,j}^k$	A decision variable with $\mathbf{x}_{i;,j}^k \in \{0, 1\}$ where $i \in N$, $j \in N$, and $k \in K$.
$\mathbf{y}_{q,i}^k$	A decision variable with $\mathbf{y}_{q,i}^k \in \{0, 1\}$ where $q \in Q$, $i \in N$, and $k \in K$.
$\mathbf{z}_{i,p}^k$	A decision variable with $\mathbf{z}_{i,p}^k \in \{0, 1\}$ where $i \in N$, $p \in P$, and $k \in K$.
$\mathbf{a}'^k_i$	Latent arrival time of train $i \in N$ at the yard at track $k \in K$.
$\tilde{\mathbf{a}}_i$	Determined arrival time by the model of train $i \in N$ at the yard.

4.2 Feasibility set

As stated above, the main purpose of this mode is to make arrival sequences at tracks. Assignment sequences of trains include the following three different occasions:

$\mathbf{x}^k_{i,j}$ = train $i \in N$ is assigned at track $k \in K$ after train $j \in N$ $(\mathbf{x}^k_{i,j} = 1)$, or not $(\mathbf{x}^k_{i,j} = 0)$.
$\mathbf{y}^k_{q,i}$ = train $i \in N$ is the first in a sequence to arrive at track $k \in K$ $(\mathbf{y}^k_{q,i} = 1)$, or not $(\mathbf{y}^k_{q,i} = 0)$.
$\mathbf{z}^k_{i,p}$ = train $i \in N$ is the last in a sequence to arrive at track $k \in K$ $(\mathbf{z}^k_{i,p} = 1)$, or not $(\mathbf{z}^k_{i,p} = 0)$.

An example of an arrival sequence at a track where $\mathbf{x}^k_{i,j} = 1$ is presented below in figure 4.4.

Figure 4.4: Arrival sequence of train i and j at track k

The feasibility set for allocating trains to tracks can be divided within three different categories. (1) the assignment sequence, (2) the time, and (3) the length. The subjects to the sequence determine that a train can only be assigned once. If a train is assigned, it is either assigned first to a track, in sequence with other trains, or last as discussed above. The subjects to time determine the arrival and departure time of trains. The conditions with regards to the length make sure that a train cannot be assigned to a track when it cannot physically be assigned to. An extensive description of each set of constraints is discussed below.

Determine the assignment sequences of trains

As described in the previous section, a train will be assigned to a track in a sequence. Constructing sequences are already discussed in several problems (e.g. the VRP). Important to acknowledge is that in the RT-TAP, we are dealing with different tracks and trains with different characteristics. We therefore slightly adjust the VRP formulation. Each sequence can consist of zero, one, or more trains. In the RT-TAP, sequences need to be made where one track only visits each train once (so a train gets only assigned once) and the number of sequences (trucks used) cannot be more than the total number of tracks available.

At the yard, each train i can only be assigned once. When train i is assigned to track k, the train is either assigned after another train j, or the the first train to arrive at the track:

$$\sum_{k \in K} \left(\sum_{j \in N} \mathbf{x}^k_{j,i} + \sum_{q \in Q} \mathbf{y}^k_{q,i} \right) = 1, \forall i \in N \tag{4.6}$$

Further, train i is either parked before another train j, or is last train to arrive at track k:

$$\sum_{k \in K} \left(\sum_{j \in N} \mathbf{x}^k_{i,j} + \sum_{p \in P} \mathbf{z}^k_{i,p} \right) = 1, \forall i \in N \tag{4.7}$$

Also, each track k can only have maximum one train starting the sequence:

$$\sum_{i \in N} \mathbf{y}^k_{q,i} \leq 1, \forall k \in K, q \in Q \tag{4.8}$$

and, can only have maximum one train ending the sequence:

$$\sum_{i \in N} \mathbf{z}_{i,p}^{k} \leq 1, \forall k \in K, p \in P \tag{4.9}$$

In addition, the total number of trains $i \in N$ arriving before train $j \in N$ and $p \in P$ should always equal the total number of trains $j \in N$ and $q \in Q$ arriving before train $i \in N$.

$$\left(\sum_{j \in N} \mathbf{x}_{j,i}^{k} + \sum_{q \in Q} \mathbf{y}_{q,i}^{k} \right) - \left(\sum_{j \in N} \mathbf{x}_{i,j}^{k} + \sum_{p \in P} \mathbf{z}_{i,p}^{k} \right) = 0, \forall i \in N, k \in K \tag{4.10}$$

Including the above-mentioned conditions, sequences without a beginning or an end can still occur (see figure 4.5). This means that trains, in theory, have an assignment but are not physically assigned to a track, similar as the VRP as discussed in section 4.1. To make sure that each sequence has a beginning and an end, the following constraint is added to eliminated sub tours (Pferschy and Staněk, 2017). The constraint states that for each (nonempty) subset $S \subset T_S$, the total sum where train $i \in N$ arrives before train $j \in N$ ($\mathbf{x}_{i,j}^{k} = 1$) must be at most be $T_S - 1$.

$$\sum_{k \in K} \sum_{i \in T_S} \sum_{j \in T_S} \mathbf{x}_{i,j}^{k} \leq T_S - 1, \forall S \subset T_S, S \neq 0 \tag{4.11}$$

A visualization of the sub tour problem is presented in figure 4.5.

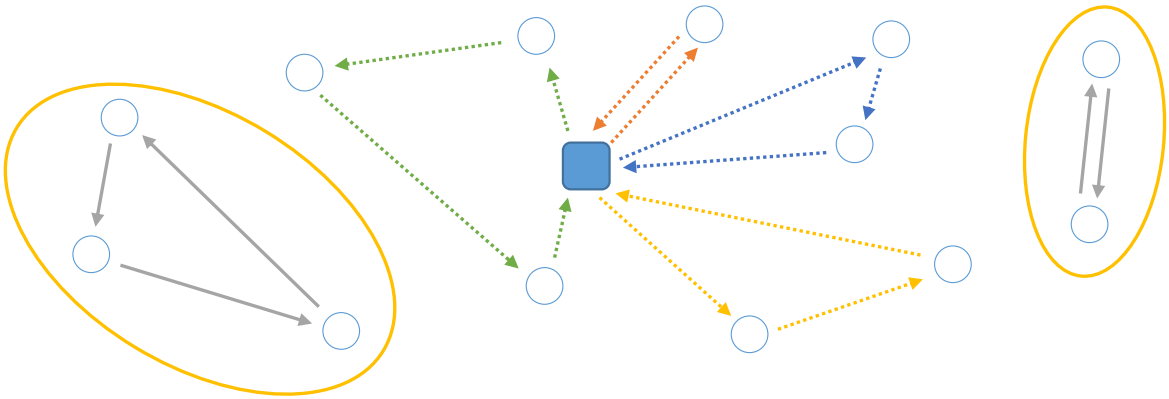

Figure 4.5: Example of sub tours that will be eliminated by using equation 4.11

Determine the arrival and departure time of trains

The first section discussed how to make the assignment sequences of each train to a track. In order to minimize the total weighted delay, we need to determine the arrival and departure times of each train $i \in N$. The relation of the arrival time of train j with the departure time of train i can be determined with the following equation:

$$d_i'^k - \alpha(1 - \mathbf{x}_{i,j}^k) + H \leq \mathbf{a}_j'^k, \forall i \in N, j \in N, k \in K \tag{4.12}$$

Equation (4.12) states that if train j arrives after train i, the arrival time of train j should always be larger than the departure time of the train i plus the minimum headway H. If train j is not arriving at the same track as train i, the two trains have no relation to each other, hence $\mathbf{x}_{i,j}^k = 0$. In this case, because α is a relatively large number, this equation makes the departure time of train i and the arrival time of train j independent from each other. If both trains i and j are assigned to track k, $(\mathbf{x}_{i,j}^k = 1)$, the equation makes sure that the arrival time of train j should always be later than the departure time of train i plus the minimum headway (H).

Because trains cannot be parked before the original arrival time, we defined the following constraint:

$$\mathbf{a}_i'^k \geq a_i, \forall i \in N, k \in K \tag{4.13}$$

Since the algorithm is minimizing the function, the model will always try to find a solutions where $\mathbf{a}_i'^k$ is as small as possible under the above-mentioned conditions. If the train does physically arrive at track $k \in K$, we determine the actual time of arrival $\tilde{\mathbf{a}}_i$ with the following equation:

$$\tilde{\mathbf{a}}_i \geq \mathbf{a}_i'^k, \forall i \in N, k \in K \tag{4.14}$$

Because the arrival and departure time of each train $i \in N$ has a direct relation with the time spend at the yard (p_i), the following equations can be applied to calculate the original, latent and determined departure of each train.

$$d_i = a_i + p_i, \forall i \in N \tag{4.15}$$

$$d_i'^k = \mathbf{a}_i'^k + p_i, \forall i \in N, k \in K \tag{4.16}$$

$$\tilde{d}_i = \tilde{\mathbf{a}}_i + p_i, \forall i \in N \tag{4.17}$$

Eliminating physical infeasible solutions with regards to the length

The final condition states that when the length of each train $i \in N$ exceeds the length of track $k \in K$, then:

i) the train cannot be assigned in sequence with another train at track $k \in K$,

$$\mathbf{x}_{i,j}^{k} = 0, \forall i \in N, j \in N, k \in K : l_i > m_k \tag{4.18}$$

ii) the train cannot be assigned first to any track $k \in K$,

$$\mathbf{y}_{q,i}^{k} = 0, \forall i \in N, q \in Q, k \in K : l_i > m_k \tag{4.19}$$

iii) and, the train cannot be assigned last in any sequence at track $k \in K$,

$$\mathbf{z}_{i,p}^{k} = 0, \forall i \in N, p \in P, k \in K : l_i > m_k \tag{4.20}$$

4.3 Computational complexity

Theorem 1 *(Complexity). The RT-TAP is $\mathcal{NP}$-Hard*
Proof: As discussed in section 4.1, the RT-TAP can be translated to a variation of the VRP. With regards to this problem, the trains can be translated to visiting points and need to be visited by a track (or as mentioned in the VRPs; a truck). The truck has to arrive at a visiting point at a given time and has to depart at a given time. The number of loops is the maximum trucks used (tracks used). The objective can be translated to visit each point only once, but we do not need to use all tracks (i.e trucks). In addition, within the condition that not all trucks can visit all points (not all trains can be assigned to all tracks).

This said, we show that the sequential part of the RT-TAP is done similar to the way that sequences are constructed as the VRP, which is a proven $\mathcal{NP}$-complete problem (Karp, 1972). We therefore proof that the VRP is reducible to the sequential part of the RT-TAP, which means that the RT-TAP is at least as hard as the VRP. In addition we showed that the RT-TAP is not in $\mathcal{NP}$ because of the additional arrival decision we have to consider. Including all above mentioned statements we proof the $\mathcal{NP}$-hardness of the RT-TAP.

4.4 Mathematical program

The objective is a summation of the total weighted delay of each train $i \in N$. The delay of each train can be calculated by subtracting the original departure time (d_i) of each train $i \in N$ from the determined departure time $(\tilde{d}_i)$ by the model. The delay of each train $i \in N$ will then be multiplied by its corresponding weighting (w_i). Including the above-mentioned statement and further notations of the previous chapter. The following objective function is established:

$$\min \sum_{i \in N} (\tilde{d}_i - d_i) w_i \tag{4.21}$$

subject to:

$$\sum_{k \in K} \left(\sum_{j \in N} \mathbf{x}_{j,i}^k + \sum_{q \in Q} \mathbf{y}_{q,i}^k \right) = 1, \forall i \in N \tag{4.6}$$

$$\sum_{k \in K} \left(\sum_{j \in N} \mathbf{x}_{i,j}^k + \sum_{p \in P} \mathbf{z}_{i,p}^k \right) = 1, \forall i \in N \tag{4.7}$$

$$\sum_{i \in N} \mathbf{y}_{q,i}^k \leq 1, \forall k \in K, q \in Q \tag{4.8}$$

$$\sum_{i \in N} \mathbf{z}_{i,p}^k \leq 1, \forall k \in K, p \in P \tag{4.9}$$

$$\left(\sum_{j \in N} \mathbf{x}_{j,i}^k + \sum_{q \in Q} \mathbf{y}_{q,i}^k \right) - \left(\sum_{j \in N} \mathbf{x}_{i,j}^k + \sum_{p \in P} \mathbf{z}_{i,p}^k \right) = 0, \forall i \in N, k \in K \tag{4.10}$$

$$\sum_{k \in K} \sum_{i \in T_S} \sum_{j \in T_S} \mathbf{x}_{i,j}^k \leq T_S - 1, \forall S \subset T_S, S \neq 0 \tag{4.11}$$

$$d_i'^k - \alpha(1 - \mathbf{x}_{i,j}^k) + H \leq \mathbf{a}_j'^k, \forall i \in N, j \in N, k \in K \tag{4.12}$$

$$\mathbf{a}_i'^k \geq a_i, \forall i \in N, k \in K \tag{4.13}$$

$$\tilde{\mathbf{a}}_i \geq \mathbf{a}_i'^k, \forall i \in N, k \in K \tag{4.14}$$

$$d_i = a_i + p_i, \forall i \in N \tag{4.15}$$

$$d_i'^k = \mathbf{a}_i'^k + p_i, \forall i \in N, k \in K \tag{4.16}$$

$$\tilde{d}_i = \tilde{\mathbf{a}}_i + p_i, \forall i \in N \tag{4.17}$$

$$\mathbf{x}_{i,j}^k = 0, \forall i \in N, j \in N, k \in K \colon l_i > m_k \tag{4.18}$$

$$\mathbf{y}_{q,i}^k = 0, \forall i \in N, q \in Q, k \in K \colon l_i > m_k \tag{4.19}$$

$$\mathbf{z}_{i,p}^k = 0, \forall i \in N, p \in P, k \in K \colon l_i > m_k \tag{4.20}$$

Chapter 5

Solution Method

This chapter will focus on the solution method that is used for finding the optimal solutions for the Real-Time Track Assignment Problem (RT-TAP) as described in chapter 4.

5.1 Brute Force solution method

The most obvious approach for solving computational problems, such as the RT-TAP, is to evaluate all possible solutions. Such a method is also described as a Brute Force (BF) attack and is mostly used to compute the global optimum from an entire solution space.

Solving the RT-TAP by using a BF attack is done in three different steps; (a) Constructing the initial assignment and determining assignment limits, (b) optimizing the assignment sequence of each track, and (c) minimizing the total weighted delay.

a) Constructing the initial assignment and determining track limits

The initial assignment is the first assignment that the algorithm considers. Because an algorithm needs a systematic problem-solving approach, the algorithm will only make one change each iteration. Initially, it is important to first determine the limitations of each train. Because different trains and tracks are considered with various lengths, it is highly probable that not all trains can be assigned to all tracks. In order to get an efficient algorithm, it is necessary to sort all tracks from long to short so that $k = 1$ is the longest track, and K is the shortest track at the yard. When sorted, the model will determine the maximum possible track that each train can be assigned on $(k_{max,i})$. Because in the previous step all tracks are sorted from long to short, it can be systematically checked if trains fit on a track or not. If the train fits the track, we can continue to check the next track until the train does not fit anymore. When the moment is reached that the train length exceeds the track length, we can claim that the maximum track the train can be assigned on is the previous track $(k - 1 = k_{max,i})$. If the train can be parked on all tracks, we determine that $K = k_{max,i}$ for the train.

It is important to check when train $i \in N$ exceeds the track limit of track $k \in K$ $(m_k \geq l_i)$ and then determine $k_{max,i}$ because there might be more tracks with the same length at the railway yard. When limits are determined, we assign all trains $i \in N$ to track $k = 1$. Then, we can continue optimizing the assignment as discussed in the next section.

Determining the initial assignment and train limits can be implemented as follows:

Begin function
For *every* $k \in K$: *sort so that* k_1 *is the longest track, and* K *the shortest*
For *every* $i \in N$
For *every* $k \in K$
If $m_k \geq l_i$: *continue with the next track* $k \in K$
Else *establish that* $k - 1 = k_{max,i}$ *for train* $i \in N$
Next *train* $i \in N$
For *every* $i \in N$: *assign each train* $i \in N$ *to track* $k = 1$
End function

b) Optimizing the assignment sequence of each track

After constructing the initial assignment, each track needs to be optimized in order to minimize the total weighted delay per track. Because tracks are independent of each other with regards to the assignment sequences, we can determine the delay per track and then calculate the total weighted delay by summarizing the total delay of all tracks. While calculating, we only need to consider the tracks that have more than one train assigned to it. In this algorithm, sequences are optimized based on a variation of the bubble sort algorithm. The algorithm works as follows: we consider a sequence of trains that are assigned to track k. We first compare the arrival sequences (i, j with j, i) where train i^k is the first to arrive at track k in the sequence, and train j^k the second train to arrive at track k, and choose the best option in terms of minimizing the total weighted delay. If the arrival sequence switches, train $i^k \Rightarrow j^k$ (first becomes second in the sequence) and $j^k \Rightarrow i^k$ (second becomes first in the sequence) and we start over. If there is no switch, we continue to compare i^k with $(j+1)^k$ (the third in the sequence) and continue until we compared the first with the last train. When established which should be the first train in the sequence, we have to check the arrival times of other trains. If the arrival time of any train $j^k \in N^k$ is less than the departure time of train i^k plus the minimum headway, we change the arrival time of train j^k to the departure time of the first train plus the headway ($\tilde{\mathbf{a}}_j^k = \tilde{d}_i^k + H$). Because we established the first train in the sequence, we can continue the same steps again with the second, third, etc. train in the sequence ($i^k = i^k + 1$ and $j^k = i^k + 1$), check the best option (i, j or j, i), and continue until we have considered all trains at track k. When this is done for track $k \in K$, we can calculate the total delay at this track, and repeat the same steps with every other track at the yard.

The optimization of the assignment sequences can be done as follows:

Begin function
For *every* $k \in K$ *where* $\sum_{i^k \in N^k} > 1$: *consider* $x = 1, i^k = x, j^k = i^k + 1$
For *every* $i^k \in N^k$
If $(\tilde{d}_j^k - \tilde{\mathbf{a}}_i^k) < (\tilde{d}_i^k - \tilde{\mathbf{a}}_j^k)$: $j^k \Rightarrow i^k$, $i^k \Rightarrow j^k$, $i^k = x, j^k = i^k + 1$
Else if $j^k < N^k$: $j^k = j^k + 1$
Else if $(i^k + 1) < N^k$: $i^k = i^k + 1$, $j^k = i^k + 1$
For $j^k \in N^k$
If $\tilde{\mathbf{a}}_j^k < (\tilde{d}_i^k + H)$: $\tilde{\mathbf{a}}_j^k \Rightarrow (\tilde{d}_i^k + H)$
Else $\tilde{\mathbf{a}}_j^k = \tilde{\mathbf{a}}_j^k$
continue with $x = x + 1, i^k = x$
Else *next track* k
End function

c) Minimizing the total weighted delay the yard

In order to minimize the objective stated in equation 4.21, the algorithm needs a systematic approach in order to check all possible solutions. In section (a) we started with an initial assignment of assigning all trains to the first track $(k = 1, \forall i \in N)$. The systematic approach can be done by changing one train at a time. In this case, we can take train $i = 1$ and assign to track $k_i = k_i + 1$. We can continue this as long as $k_i \leq k_{max,i}$. If $k_i > k_{max,i}$, we assign train i again to track $k = 1$, and assign train j to track $k_j = k_j + 1$ as long as $k_j \geq k_{max,j}$. We can again restart assigning train i to the next track as long as the limit allows it. If $k_j > k_{max,j}$, we assign train $j + 1$ to $k_{j+1} = k_{j+1} + 1$ and assign train i and j again to track $k = 1$. In every iteration we sort the tracks as discussed in section (b), and continue until $k_i = k_{max,i}, \forall i \in N$. During the iteration process, we check if the current assignment is better or worse in terms of minimizing the objective. When all possible solutions are considered, the minimized weighted delay can be determined with its corresponding assignment. This will then be the output of the model and advise from the Decision Support System (DSS). Important is that only one track change is done per iteration because else valid solutions might be skipped.

5.2 Problem specific Genetic Algorithm

With the increment of computer performance over the last decades, we are able to solve larger and more complex problems by using a BF approach. However, there will always be limits to what can be done even with the fastest computers. Because the RT-TAP needs to be solved at real-time, one might not have the time to use a BF approach. Especially because finding the global optimum might need days of computation before all options are considered.

A reasonable way of tackling this problem is perhaps just to satisfy oneself with a good, but not necessarily the best solution. For example, in many practical circumstances when limited time is available i.e. real-time optimization, it might be preferable to get a quick 'reasonably good' solution rather than to wait for much longer and get a marginally better one. Approximation algorithms i.e. heuristics, compute solutions based on a solving method and determine the solutions by not considering all possible solutions, unlike the BF method. Several heuristic approaches used for solving real-time problems or problems with an NP-Complete or NP-Hard computational complexity are discussed in literature. Heuristic approaches include Greedy algorithms (Törnquist, 2010, 2012), Hill Climbing (Rahim et al., 2013), Simulated Annealing (Törnquist and Persson, 2005), Genetic Algorithm (Holland, 1975) and Tabu Search (Glover, 1986).

Interestingly, while Gkiotsalitis et al. (2019) noted after computational results that the Genetic Algorithm (GA) method outperformed both the Simulated Annealing (SA) and the Hill Climbing (HC) algorithms in their model for cost minimization for bus fleet allocation, Rahim et al. (2013) concluded that after comparing the HC with the GA that the HC had better computational results and therefore suggested to incorporate the HC optimization rather than GA in their work. Furthermore, while Törnquist and Persson (2005) showed that the Tabu Search (TS) outperformed the SA, Ho and Yeung (2001) stated that the GA, SA and the TS provide a similar balance between computation time and optimality. From these contradictions in different researches with regards to meta-heuristics used, it can be concluded that it is not determined that one heuristic approach performs better or worse, or faster than another and therefore the performance of the algorithm might be completely dependent on the research and solution method itself.

In this study, similar to the work from Gkiotsalitis et al. (2019), we will use a problem specific GA in order to find a reasonably good solution for larger instances. A GA is based on Darwins theory of evolution and tries to find a solution based on natural selection and survival of the fittest and is known for considers a pool of solutions rather than a single solution at each iteration. One of the first works including a GA is the book of Holland (1975), which detailed describes the stages of a GA. The principal stages consider (1) encoding the initial population, (2) evaluating the fittest of each population, (3) parent selection for offspring generation, (4) crossover and (5) mutation (Gkiotsalitis et al., 2019). Important to acknowledge is that different (meta-)heuristic methods may also be used for solving this problem as discussed above.

Encoding the Genetic Algorithm

The problem specific GA is based on the BF approach described in section 5.1. The algorithm consists of three different parts. (1) Constructing the initial population, (2) the assignment of trains to tracks, and (3) optimizing the assignment sequence of each track. The GA will only be applied on the assignment of trains to tracks. Optimizing the assignment sequence of each track will still be done by the BF method as discussed in section 5.1.b. This is because sorting each track is relatively less work, but can give a large impact on the result. For example, scheduling train i before j would make no sense as the delay would be zero if j is scheduled before i.

Constructing the initial population

The first step that needs to be made is initializing the first population, which consists of a random track assignment for each train. The population is defined as P with $\{m_1, m_2, ..., M\}$ population members. Each population member $m \in P$ denotes an assignment of all trains to a track. Each $m \in P$ consists of $\{g_1, g_2, ..., G\}$ genes where each $g \in m$ represents an assignment of one train to a track, and should therefore always take in integer value from $\{k_1, ..., k_{lim,i}\}$. Each population member should always have as many genes as inbound trains.

Because we are currently dealing with large solution spaces, it is preferred to reduce it as much as possible so that fewer solutions have to be considered, and therefore increases the performance of the model.

As starters, all trains and track will be sorted based on their length so that $i = 1$ is the longest train and N the shortest, and $k = 1$ is the shortest track and K the longest. Similar to the BF method, we determine a track maximum, which is the maximum track a train can be physically assigned to ($k_{max,i}$). In addition, we apply formulas 5.1 and 5.2 to determine a track limitation ($k_{lim,i}$), which is developed in order to delete double members from the solution space.

$$k_{lim,(i=0)} = 0 \tag{5.1}$$

$$k_{lim,i} = \begin{cases} k_{max,i} & \textit{if} k_{max,i} \leq k_{lim,i-1} + 1 \\ k_{lim,i-1} + 1 & \textit{else} \end{cases}, \forall i \in N \tag{5.2}$$

Table 5.1 shows an example on a simple case with 4 trains and 3 tracks to elaborate and proof that equations 5.1 and 5.2 work.

Table 5.1: Removing double member from the solution space.

Train number	$i = 1$	$i = 2$	$i = 3$	$i = 4$
Track maximum	$k_{max,1} = 1$	$k_{max,2} = 1$	$k_{max,3} = 3$	$k_{max,4} = 3$
Track limitation	$k_{lim,1} = 1$	$k_{lim,2} = 1$	$k_{lim,3} = 2$	$k_{lim,4} = 3$

When we apply equation 5.1 and 5.2, we can reduce the track maximum of train $i = 3$ from $k_3 = 3$ to $k_3 = 2$. This statement is valid because in practice, we never have to test an assignment with trains $k_{1,2} = 3$ and $k_{1,2} = 1$, because the assignment would exactly be the same as when $k_{1,2} = 1$ and $k_{3,4} = 2$. In addition, an assignment where $k_{1,2,4} = 1$ and $k_3 = 2$ is exactly the same as when $k_{1,2,4} = 1$ and $k_3 = 3$ because the delays are calculated with the sequence at track 1, and in both cases this sequence does not change. In this simple case, we actually reduced the solution space by $\frac{1}{3}$ because we have applied these limitations.

The initial population is randomly generated as follows:

Begin function
For *every* $m \in M$
For *every* $g \in G$: *Generate random genes where* $g \in \{k_1, ..., k_{lim,i}\}$
Minimize weighted delay for each track $k \in K$
Calculate total weighted delay by summing up all delays of tracks $k \in K$
End function

After initializing the first random population, an evaluation of the fittest is made. The fittest members will be selected as parents in the next stage of the algorithm. In this study, the fittest population members will be those with the smallest delays.

Constructing next generations

The next population consists of two different parts: (1) The fittest members of the previous generation, and (2) offsprings of the previous fittest members. Keeping the fittest $\mu\%$ of the previous generation is to make sure that good assignments are not getting lost during the iteration process, and that they can always be used for constructing the next generations. The remaining members $(100\% - \mu\%)$ will be the offspring of the fittest members of the previous population. A gene of an offspring consists of one of the following three options: (a) take gene of parent 1, (b) take gene of parent 2, or (c) mutate.

The remainder population P of the new generation is constructed as follows:

Begin function
For *every remaining* $m \in P$
For *every* $g \in G$: *Generate random number from* 0 *to* 100%
If *random nr.* $< P(p1)$: *Take gene of parent 1*
Else if *random nr.* $< (P(p1) + P(p2))$: *Take gene of parent 2*
Else *generate random genes where* $g \in \{k_1, ..., k_{lim,i}\}$
Minimize weighted delay for each track $k \in K$
Calculate total weighted delay at the yard by summing up all tracks $k \in K$
End function

The model will continue with constructing and testing new populations until a total weighted delay of 0 seconds is achieved, or after a number of predetermined iterations. One iteration equals constructing and calculating the delays of one population.

The number of population members $m \in P$, the recombination rate μ, probabilities of taking genes of parent 1 $P(p1)$, taking genes of parent 2 $P(p2)$, and the mutation rate $(1 - (P(p1) + P(p1)))$ are hyper-parameters of the GA which can affect the performance of the algorithm and might be case specific. For this reason, different scenarios should be performed with different parameters in order to increase the probability of finding a solution which is as close as possible to the global optimum.

5.3 First Scheduled First Served approach

The First Scheduled First Served (FSFS) algorithm assigns trains to tracks based on who first arrives at the yard. Several papers already discussed the FSFS approach in different rescheduling based problem (Schachtebeck and Schöbel, 2010; Schöbel, 2009). In this study, the model will schedule the inbound trains after sorting tracks from short to long and trains based on arrival. The model will then always first check if the trains fit on the empty shortest track. If an assignment is not possible because the track occupied by another train, the algorithm will compare the inbound train with all trains that are parked, and will park the inbound train on the track were the parked train is first to leave.

Chapter 6

Numerical Experiment

The following chapter aims to describe the application of the model in a numerical case study. The proposed solution method for efficient allocation of trains to tracks while reducing the delay is applied on the railway yard Waalhaven Zuid (Whz) which is located in the western part of the Netherlands.

This chapter is structured as followed. The first section provides a general description of the railway yard. This section is followed by a discussion about which input data was used in the case study, and which assumptions were made. The third and fourth sections discuss the sensitivity analysis on the different hyper-parameters and the benchmark for conducting the experiments respectively. The computational results are discussed in section five. The chapter is finalized by a discussion about the exploitation of the local optima.

6.1 Case study description

The Whz railway yard consists of about 100 tracks where each track has its own function. The function for the tracks vary between processing, renting, passing through, shunting, repair and/or parking. At Whz, only the process tracks consist of electrical wiring and are Central Operated (in Dutch; Centraal Bediend Gebied (CBG)) and therefore can be controlled by the Train Dispatcher (TD). Because main capacity problems related to the Real-Time Track Assignment Problem (RT-TAP) only relate to the process tracks, only these tracks are considered for the case study. A schematic overview of the process tracks can be found in figure 6.1. A complete overview drawing of Whz is presented in appendix D.

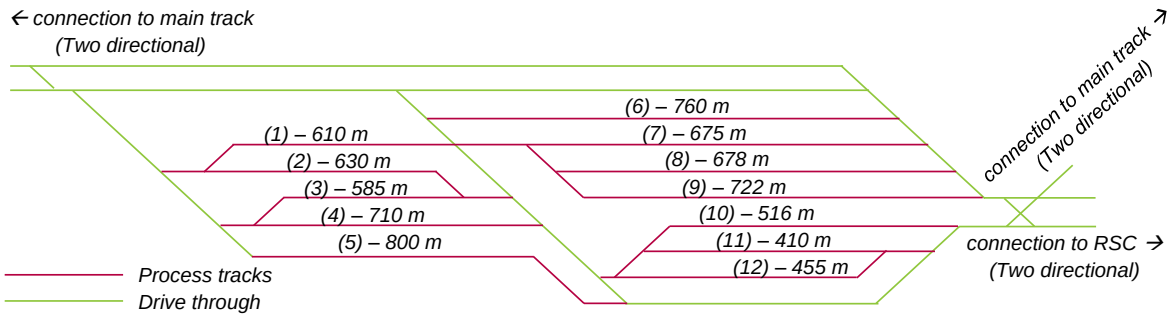

Figure 6.1: Schematic overview of Waalhaven Zuid process tracks

The characteristics of the process tracks at Whz can be divided into two different categories. (1) the drive through (green), and (2) process tracks (red). The drive through tracks need to remain clear all time as they are purposed for entering and exiting the process tracks, and therefore the yard. The main purpose of trains arriving at Whz has to do with the container terminal (Rail Service Center (RSC)) located on the right side of the yard where trains are getting loaded and unloaded from their cargo. At Whz, trains first arrive from one of the two sides (dependent on the trains origin) and are then assigned to one of the 12 process tracks. After the necessary operations, the train gets assigned to another track at RSC and the unloading and loading operations start there. When the necessary operations at RSC are done, the train continues again to one of the 12 tracks at Whz. After again a sequence of operations, the train exits the railway yard from one of the two exit tracks (dependent on the trains destination). The process tracks can be used for temporary parking trains to perform the necessary operations. A more extensive description about the operations after arriving Whz and before leaving are described in table E.1 and E.2.

6.2 Numerical input received and assumptions made

The numerical experiment is developed based on data provided by ProRail. The model is executed based on one day of historical data between the 1st of October 2015 and the 1st of December 2015. As discussed in chapter 4, the input for the model needs to consist of the following element with regards to the inbound trains, and the tracks at the yard.

Table 6.1: Data necessary for the model based on the mathematical model

Description
(m_k) Length of each track $k \in K$. (see section 6.1)
(a_i) Original arrival time of train $i \in N$ at the yard at track k.
(p_i) Process time that train $i \in N$ must remain at the yard.
(d_i) Original departure time of train $i \in N$ from the yard at track k.
(w_i) Delay weighting of each train $i \in N$ indicating the importance of the delay of a train.
(l_i) Length of each train $i \in N$.
(H) Minimum headway of two trains at the same track.

The following sections explain the data provided by ProRail and shows how the data is used in the model including the assumptions made.

Capacity planning

Capacity on a railway yard for freight transportation is strongly related to the capacity on the network. The Rail Infrastructure Manager (RIM) constructs the capacity planning on the network in four different stages:

- Year plan
- Original plan
- Scheduled plan
- Executed

Year plan *(± 1,5 year in advance)*: The making of a year plan starts about 1,5 years in advance. The RIM starts with making a capacity planning where it determines where and when trains can perform on the network. After the concept capacity planning is made, all transporters (freight and passengers) can submit their preferred schedule. The RIM then tries to implement it in a concept schedule. The concept plan will be sent back to the transporters where they can decide if adjustments are preferred, or if they concur with the schedule. When all parties agree, a permanent schedule is made for the whole year. During the year changes can still be made and transporters can still request times when they want to drive on the network, however, the transporter that made the request at the concept stage will always have a claim on the time that is given to the transporter.

Original plan *(± 3 days in advance)*: Three days before execution, a so-called original plan is made where the RIM checks if the trains planned in the year-plan will still be executed, if there are some trains that canceled their request, or if there are additional trains that requested a path on the network. Canceling a path requested in advance is no problem, however, if done frequently it could have consequences when requesting paths in the year-plan for the next year.

Scheduled plan *(assumed 1,5 hours in advance)*: A scheduled arrival is communicated 1,5 hours in advance and is needed if a train wants to park at a railway yard. It serves as a final call to the TD to let them know the train is coming. The TD can then determine if the train can still be assigned at the yard. In this study, the scheduled arrival is assumed 1,5 hours before executed

Executed : The executed time is the actual time that a train arrived or departed at or from a yard.

Because a final check on network capacity is determined at the original plan, the schedule of the year plan is not relevant and therefore not considered in this study.

Selection of one day from two months of data

In order to measure the performance of the model, one arbitrary day will be chosen from the two months data received from ProRail (2015) between the 1st of October 2015 and the 1st of December 2015. To know which day will be normative for the week, the busiest day in a week will be chosen for the experiment. Table 6.2 shows the arrival distribution over two arbitrary weeks. On request of ProRail, the week numbers are not mentioned in this report.

Table 6.2: Arrival distribution over the week (ProRail, 2015)

	Monday	***Tuesday***	***Wednesday***	***Thursday***	***Friday***	***Saturday***	***Sunday***	***Total***
week 1	30	44	40	48	42	30	21	255
week 2	34	36	36	41	40	30	30	247
week 1+2	64	80	76	89	82	60	51	502
Percentage	13%	16%	15%	18%	16%	12%	10%	100%

From table 6.2 can be concluded based on these two weeks the Thursdays are the busiest days during the week. For this reason, the Thursday of the first week will be chosen as input data for the case study.

Process times of trains

Because from the received data could not be determined for all trains how long a was spend at the yard, additional data regarding the processes at Whz were required. Based on the documentation received from Keyrail (2014), the total process time needed when arriving at Whz is 1h 35'. The total time needed when departing from Whz is 2h 15'. An extensive description of the assumptions made can be found in appendix E. The process times of each train is visualized in figure 6.2.

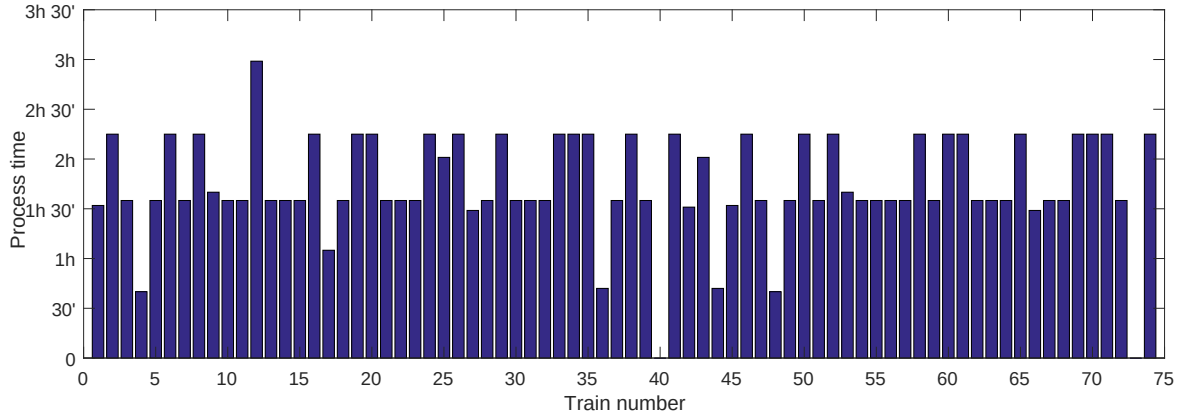

Figure 6.2: Visualization of the process times of each train

Initial train schedule

Including the arrival times and assumed process times of each train, we developed an original train schedule. In this study, we slightly adjusted the data that we received because if we would consider only the input data provided by ProRail, there would always be an empty track at the yard at any time, and therefore never occur a delay. For that reason, we took the 6 busiest tracks and copied the first 3 tracks, giving input of trains 74 and 9 tracks in total. The original train schedule is visualized in appendix E, figure E.2. The adjusted assignment used during the experiments is visualized in figure 6.3.

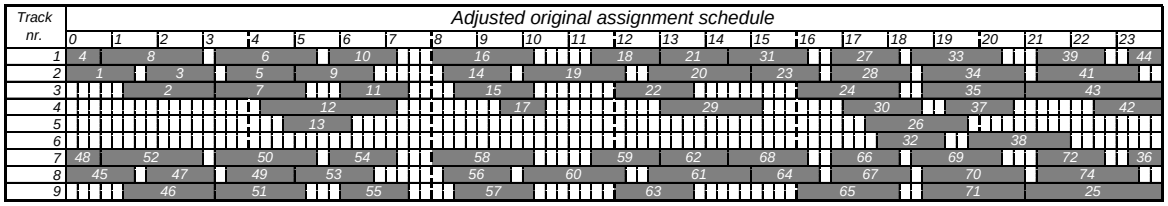

Figure 6.3: Visualization of the adjusted original train scheduled used in the experiments

Applied train weighting

In the Netherlands, the RIM (i.e. ProRail) is not allowed to give priorities or weightings to trains. For this reason the train weighting is not considered in these experiments.

Applied delays for the scheduled train plan

As discussed above, three different stages with regards to capacity planning are used. Above we visualized the initial train schedule based on the original data received from ProRail. Based on the data from ProRail, we applied the following delays on top of the original arrival for calculating the scheduled arrival of each trains.

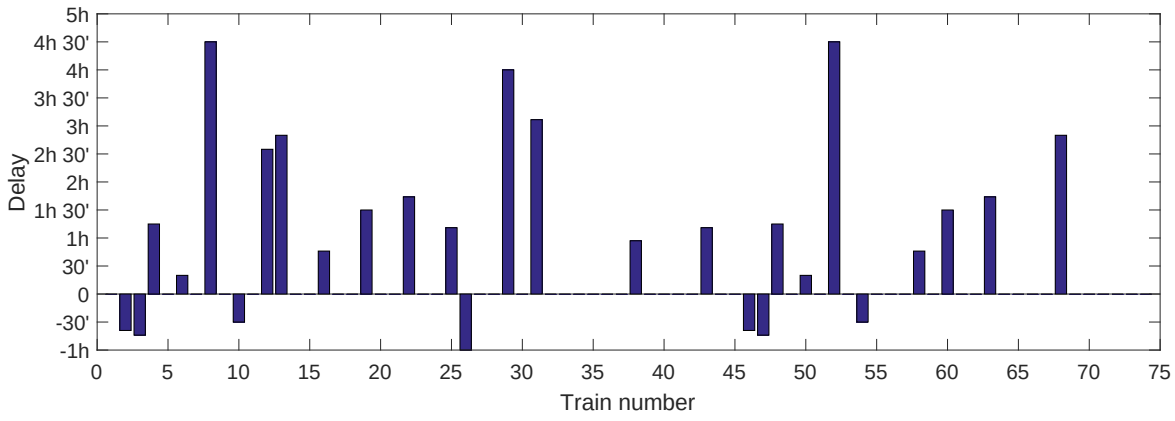

Figure 6.4: Applied delays from the Original plan to Scheduled plan

Applied delays for the executed train plan

Based on the data provided by ProRail, we used the following delays on top of the original arrival for calculating the executed arrival times of each trains.

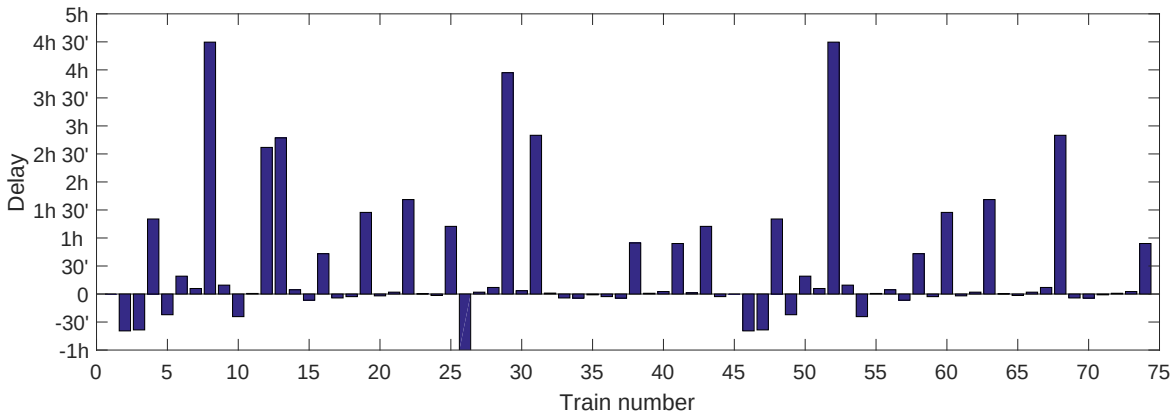

Figure 6.5: Applied delays from the Original plan to Executed

Headway between trains

The minimum headway (H) of two trains is assumed to be included in the process as discussed in appendix E and is therefore assumed to be 0.

Used train lengths

Because not all train lengths were known from the data provided, we applied our own distribution on the lengths of the trains. An extensive explanation of how the train lengths are determined can be found in appendix F. The used length of each train is visualized in figure 6.6. When a length of 400m is considered, it means that the train can be assigned to every track.

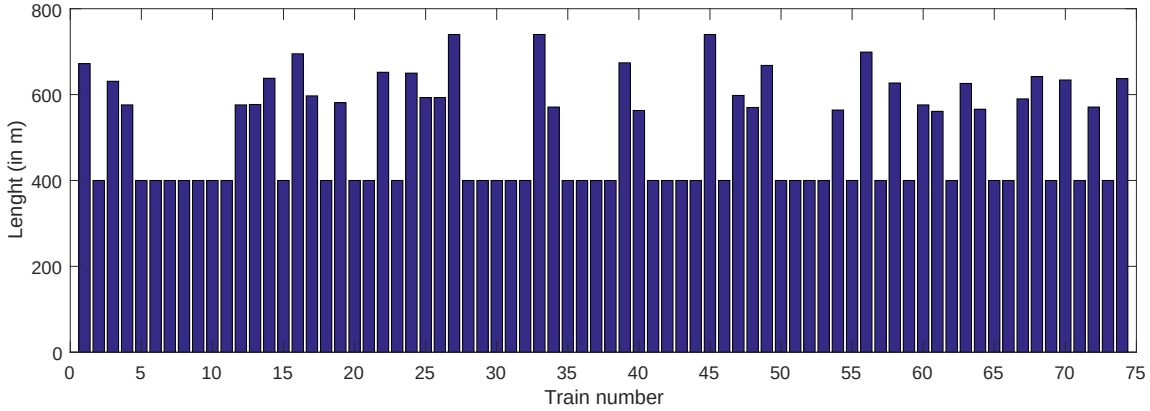

Figure 6.6: Visualization of the length of each train

6.3 Sensitivity analysis on hyper-parameters

As discussed in section 5.2, we stated that the performance of the Genetic Algorithm (GA) depends on the different hyper-parameters. The different hyper-parameters that can influence the result and performance of the model are the recombination rate $\mu\%$, the mutation rate, and the number of population members in one population $(m \in P)$.

Recombination rate

The recombination rate is the percentage of fittest members which will be used to construct the new generation. In the algorithm, all sequences will be sorted based on their fitness. The best $\mu\%$ will become automatic input for the next generation in order to make sure that good assignment is not getting lost. These same members will also become the parents for the next generation members. In the following test four different $\mu\%$ were used. Table 6.3 and 6.7 show the results of the tests with the different recombination rates.

After executing the tests with the different rates, we can see that with regards to the optimality gap, the average differences are very small. However, with regards to the number of populations needed to find the best solution we see that with a 10% recombination rate the model performs fastest when compared with the other rates. For this reason, a recombination rate of 10% will be further used in the execution of the experiment.

Table 6.3: Results when applying different recombination rates

Recombination rate $(\mu\%)$	***Delay low***	***Delay high***	***Difference***	***Average***	***Populations needed*** $(\times 10^2)$
$\mu = 5\%$	$7:34:32$	$7:34:32$	0 ***sec.***	$7:34:32$	80
$\mu = 10\%$	$7:34:32$	$7:34:32$	0 ***sec.***	$7:34:32$	55
$\mu = 15\%$	$7:34:32$	$7:34:57$	25 ***sec.***	$7:34:37$	93
$\mu = 20\%$	$7:34:32$	$7:34:57$	25 ***sec.***	$7:34:47$	156

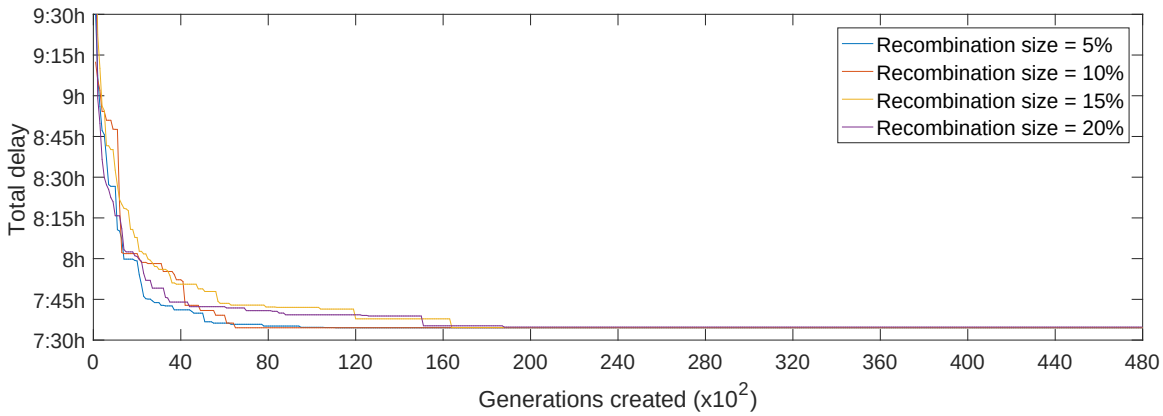

Figure 6.7: Four different recombination rates $(\mu\%)$ (average over five tests)

Crossover & mutation rate

In this study, the crossover and mutation are determined per gene. During the algorithm, a random number from 0% to 100% is generated. If the number is smaller than $P(p_1)$, we use the gene of parent one, if the number is smaller than $(P(p_1) + P(p_2))$, we use the genes of parent two. Else, we will generate a random number which will be the mutation of the gene. During multiple tests with different probabilities of $P(p_1)$ and $P(p_2)$, we can conclude that there was no difference in the performance of the model if these numbers are different from each other. For this reason, the probabilities of the different parents are kept equal when possible. The mutation rate, however, had significant changes with the performance of the model. Each mutation is tested five times. The results of the different mutation rates are presented in table 6.4 and figure 6.8.

From these results can be concluded that if we use a 2% mutation rate, on average the model performs best. Besides that the model gives the best average performance, the gap between the best and worst out of the five tests is by far best compared to the other mutations rates. If we consider too much mutation, we can see that too many random numbers are generated and the model will therefore not converge to a reasonably good result. If we consider 0% mutation, we can see that although the model sometimes does converge to the best-generated solution, the model does also sometimes gets stuck in a local optimum. A valid reason for this could be that good solutions must consist of genes from either parent one or two. If the gene of this good solution is not within one of the two, the model might not be able to further optimize the assignment sequence and minimize the delays. In table 6.4, delay low is the best-minimized delay out of the five tests. Delay high is the worst optimization of the five tests.

Table 6.4: Results of the five test for each different mutation rates

Parenting rate	*Mutation rate*	*Delay low*	*Delay high*	*Difference*	*Average*
50% / 50%	0%	$7:34:32$	$8:00:00$	1528 *sec.*	$7:47:10$
50% / 49%	1%	$7:34:32$	$8:00:00$	1528 *sec.*	$7:39:59$
49% / 49%	2%	$7:34:32$	$7:36:03$	91 *sec.*	$7:34:55$
48% / 48%	4%	$7:35:38$	$8:14:47$	2349 *sec.*	$7:53:19$
47% / 47%	6%	$7:40:32$	$8:30:09$	2977 *sec.*	$8:02:35$
45% / 45%	10%	$8:43:51$	$10:51:26$	7655 *sec.*	$9:40:50$

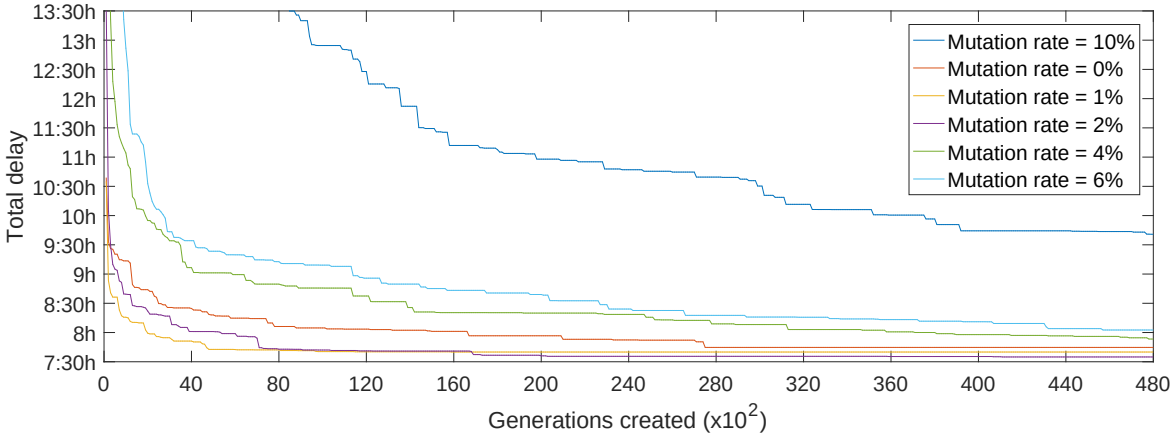

Figure 6.8: Six different mutation rates (average over five tests)

Population size

The final hyper-parameter that might affect the performance of the model is the population size. Below, we tested the model four different times with population sizes of: $P = 90$, $P = 100$, $P = 105$, and $P = 110$. Each population size is tested five times.

After the tests, we see that all population sizes perform similarly. Although we see that the populations of 100 and 105 perform best, it does not mean that they are always better than the others. During one of the five tests, it could be a coincidence that the population sizes of 90 and 110 get stuck in a local optimum as the average of all tests is almost the same.

Because of the relatively small differences in the results, further in this study, we will use a population size of $P = 100$.

Table 6.5: Results with different population sizes

Population size	***Delay low***	***Delay high***	***Difference***	***Average***	***Populations needed*** $(\times 10^2)$
$P = 90$	$7:34:32$	$7:37:57$	205 *sec.*	$7:35:23$	270
$P = 100$	$7:34:32$	$7:34:32$	0 *sec.*	$7:34:32$	249
$P = 105$	$7:34:32$	$7:34:32$	0 *sec.*	$7:34:32$	287
$P = 110$	$7:34:32$	$7:35:38$	66 *sec.*	$7:34:45$	163

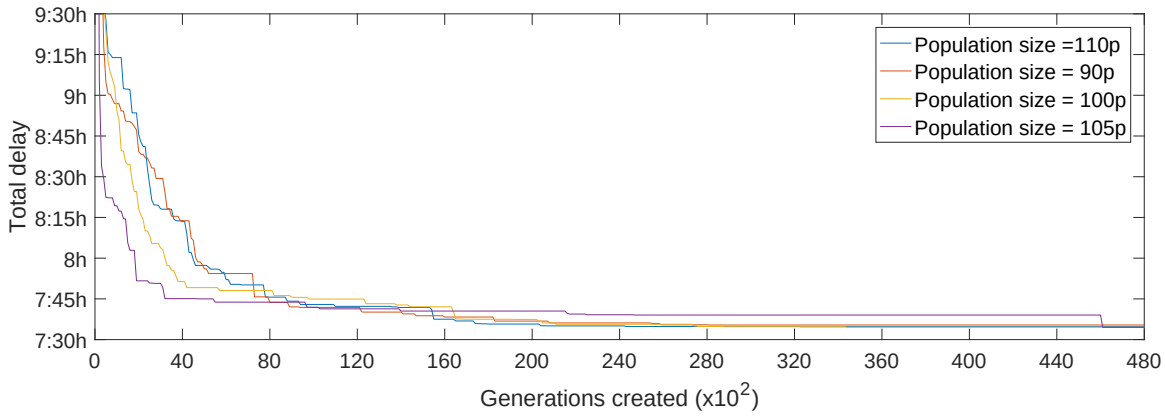

Figure 6.9: Four different population sizes (average over five tests)

Because the model might perform better when using a different mutation rate in combination with the different recombination rates and/or population sizes, we did several tests again with the different rates. After the tests, the 2% mutation rate still performs best in combination with a population size of $p = 100$, and a recombination rate of 10%, and therefore is further used in the model.

6.4 Description of the benchmark

In order to define the performance of the different solution methods, we will compare the GA with a Brute Force (BF) attack and a First Scheduled First Served (FSFS) algorithm on three different problem scenarios. The global optimum will be found by using the mathematical program as presented in section 4.4 in the AMPL program by using a CPLEX solver. The first two scenarios consider 13 inbound trains on a railway yard of 3 tracks and 4 tracks, and the third with 74 inbound trains and 9 tracks. Because the BF approach is not able to perform on larger instances, we will exclude this method when comparing the algorithms on the larger scenario. Because, as discussed in the introduction, constructing new infrastructure consists of a long decision-making process including large expenses, it is also interesting to see the effect when we assume that we made these investments (i.e. consider that each train can be assigned to all tracks) and compare to when the yard experiences track limitations.

The two smaller railway yards will be tested with input based on arrival data of ProRail (2015) including a random distribution on the train lengths and includes delays. The main purpose of the smaller instances is to define the optimality gap of the FSFS and the GA when comparing the result with the global optimum. Both scenarios will be tested two times when considering and not considering the track lengths limitations in four different experiments.

For larger instances, we will compare the performance of the GA with a FSFS algorithm, similar to Zhan et al. (2015). We will simulate a 24h day including all real-time changes that occur with regards to the arrival times of the trains. The input data for the larger instance will be the data defined in section 6.2. The simulation will be conducted two times when considering and when not considering the track length limitations in order to define the effect of the input on the two models.

Results of the smaller instances are shown in section 6.5.1. Results for the larger instances are shown in section 6.5.2 for the instance excluding the train lengths, and section 6.5.3 when considering the track length determined in section 6.1.

6.5 Computational results

For all computations in this report, we used a macOS High Sierra computer with processor 3,4 GHz Intel Core i5 and memory 8 GB 2400 MHz DDR4. In all cases, the FSFS algorithm provided solutions within a second. With regards to the performance of the GA, we measured an average computational speed of about 150 populations per second.

6.5.1 Performance on smaller instances

As discussed in section 6.4, the performance will be tested in four different experiments. In the first two experiments we consider a railway yard with 4 tracks and 13 inbound trains. In the first two experiments we consider the following track lengths: $m_1 = 800$ meter $m_2 = 700$ meter, $m_3 = 600$ meter, and $m_4 = 500$ meter.

In order to increase the complexity of the model, we will also test two scenarios by decreasing the number of tracks from four to three tracks while still considering the same 13 inbound trains. This is again tested for the experiments where we consider sufficient track length, and not. The three tracks considered for the third and fourth scenario are tracks $k = 1$, $k = 2$, and $k = 3$.

These four experiments are executed each four times with different distributions on length and delays. In table 6.6 we present the preliminary result of one of the four experiments of the smaller instances.

Table 6.6: Comparison of the CPLEX optimization, the First Scheduled First Served algorithm, and the Genetic Algorithm

Test *(4 tracks without length)*	***Total delay*** $(h:mm:ss)$	***Running time***
CPLEX	$0:21:10$	$50\ sec$
FSFS	$0:21:10$	$< 1\ sec$
GA	$0:21:10$	$< 1\ sec$

Test *(4 tracks with length)*	***Total delay*** $(h:mm:ss)$	***Running time***
CPLEX	$1:48:36$	$121\ sec$
FSFS	$3:29:10$	$< 1\ sec$
GA	$1:48:36$	$< 1\ sec$

Test *(3 tracks without length)*	***Total delay*** $(h:mm:ss)$	***Running time***
CPLEX	$2:10:13$	$285\ sec$
FSFS	$2:11:26$	$< 1\ sec$
GA	$2:10:13$	$< 1\ sec$

Test *(3 tracks with length)*	***Total delay*** $(h:mm:ss)$	***Running time***
CPLEX	$4:41:48$	$255\ sec$
FSFS	$6:24:45$	$< 1\ sec$
GA	$4:41:48$	$< 1\ sec$

When comparing the GA and the FSFS algorithms with the CPLEX optimization, we can conclude that the optimality gap between the GA and CPLEX has zero difference in terms of delay for smaller yards. If we consider a railway yard with 4 tracks and assume that the length of the tracks is always sufficient for all trains, we can see that the FSFS performs equal to the CPLEX and the GA. However, if we increase the complexity by adding length constraints on trains and tracks, or by removing the availability of a track, we see that the FSFS performs rather poorly, while the GA still converges to the global optimum.

6.5.2 Real-time simulation of the model without considering train lengths

Because train lengths could not be determined by the input data from ProRail, we do not consider the length of trains in the first scenario. In this case, we can simulate the performance of the model if all yards would be build conform to European standards, and therefore have sufficient length for each train. In these tests, we considered the original schedule without delays. The results of the first real-time performance of the model are presented in figure 6.10.

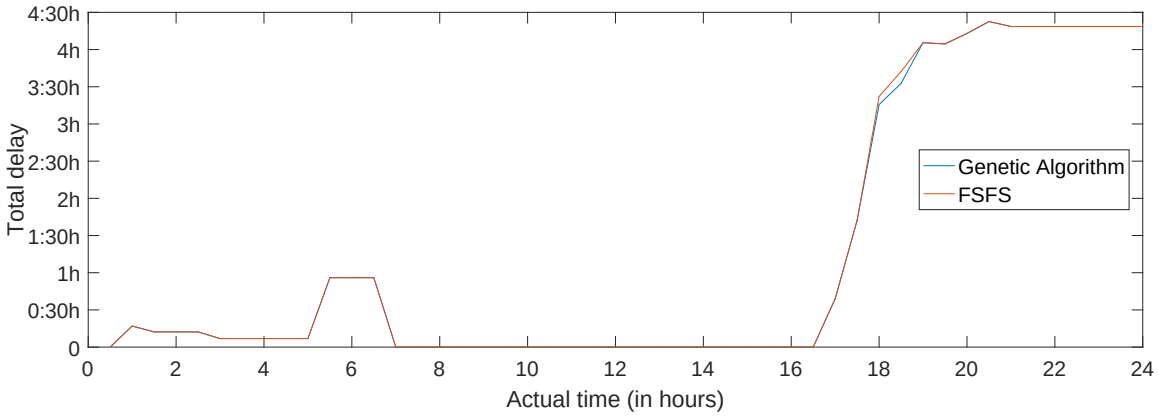

Figure 6.10: Real-time performance (excluding track length)

As expected and proven in section 6.5.1 for smaller cases, we see that when we do not consider train lengths, the performance of the GA and the FSFS algorithm performs almost the same. At around 18 pm we see that the GA performs slightly better than the FSFS algorithm with a gap of 3 minutes and 29 seconds, but the final result at the end of the simulation is exactly the same.

At the beginning of the day, we see that a delay starts to occur. Interestingly is that these delays disappear when the time exceeds. In order to understand where this delay comes from, we must check on which train this delay occurs. If we look in the results we see that a train is originally planned at a certain time, and therefore the assignment is reserved in the model. Later, we see that this train is delayed and can be parked at a moment when there is room available in the schedule. At this moment we do not need to delay trains anymore and optimize our schedule without delays. from 16 pm on wards, we see that by updating the arrival times of trains, conflicts occur in the schedule and therefore more trains receive delays. By getting more and more information about the updated arrival times, we see that more and more delays occur in the schedule.

6.5.3 Real-time simulation of the model with track length limitations

In the second simulation we consider trains lengths as determined in section 6.2. The results are shown in figure 6.11.

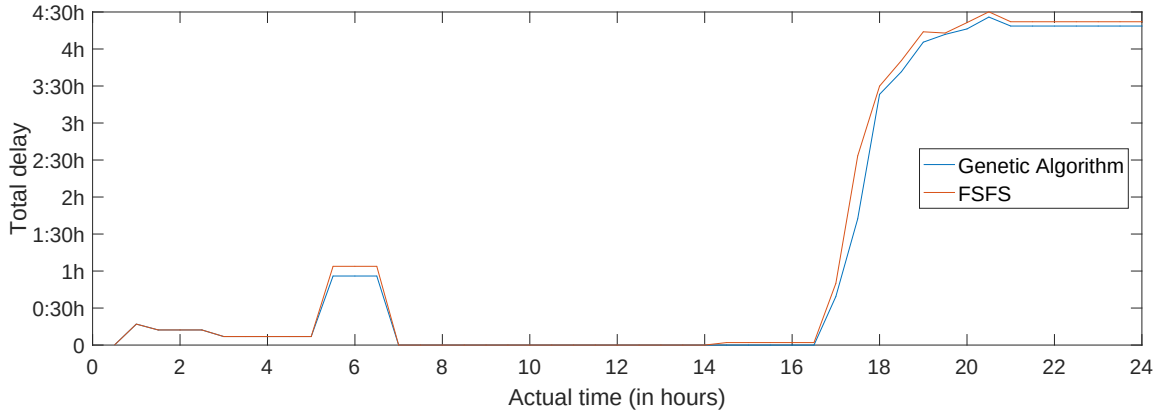

Figure 6.11: Real-time performance (including track length limitations)

During the day we see a similar optimization of the GA compared to the previous simulation. However, one can observe that the GA outperforms the FSFS algorithm at almost all points at moments when delays occur. At the end of the day, the optimization gap between the GA and the FSFS is a gap of 9 minutes and 35 seconds on top of a total delay of 4h 18' 36" in favor of the GA. The final optimization of the GA is exactly the same as in the simulation of the previous section (see section 6.5.2).

Also in this simulation, we see that at the beginning of the day a delay starts to occur. Similar as in the previous simulation, we see that this delay disappears when the time exceeds. The reasons for this delay also has to do with the reservation of tracks of originally planned trains. The delays disappear when the train time is updated and the originally planned train is scheduled at a less critical time.

6.5.4 Estimating the trains' arrival

In addition to the experiments presented above, we tried to provide an estimate of the executed data by adding a delay probability distribution on the original and planned arrival based on the data provided by ProRail. Nonetheless, the applied distributions did not reflect reality at all. For this reason, this section is not further discussed in this research. The experiments with the applied distributions can be found in appendix G.

6.6 Exploitation of the local optima

When assigning trains to tracks by using the GA, there can be more than one solution that we found that there are results in (quite) similar total delays. For example, if we consider the real-time performance of the model from 22 pm till 24 pm as shown in figure 6.10 and 6.11, there was no change in the minimized weighted delay, but all local optima found were different.

There could be several reasons for this. With regards to general railway operations, we see that some tracks are interchangeable (i.e with the same length). This means that while arrival sequences of trains at an arbitrary track remains the same, it does not matter to which track these trains are assigned too. Furthermore, within the assignment schedule, some trains are not related to the optimization of the model. For example, if a train is alone to arrive at 22 pm, and all tracks are available, several solutions are possible which would affect the assignment sequences of the model, however, it will not affect the total minimized weighted delay by the model. With regards to the assumptions made, in section 6.2 we determined that some of the process times of trains are the same and copied three of the most busiest tracks, which resulted in trains having the exact same characteristics. It could therefore be that if train x arrives after train y, it would give the exact same minimized delay as when train z is parked after y as train x and z have the exact same properties.

To further substantiate, figure 6.12 visualizes some of the local optima of four instances from the 24h simulation of 74 trains and 9 tracks. Each color corresponds to a track assignment ($k = \{0, 1, ..., 8\}$) of one train. Although all optimizations of the given data converted to a delay of 4h 18' 36", the optimization still provided different track assignments and sequences for some of the trains.

Note that during the modeling, the trains are sorted based on their lengths. For this reason the train numbers in figure 6.12 do not equal the train number in the figures presented in previous sections and chapters.

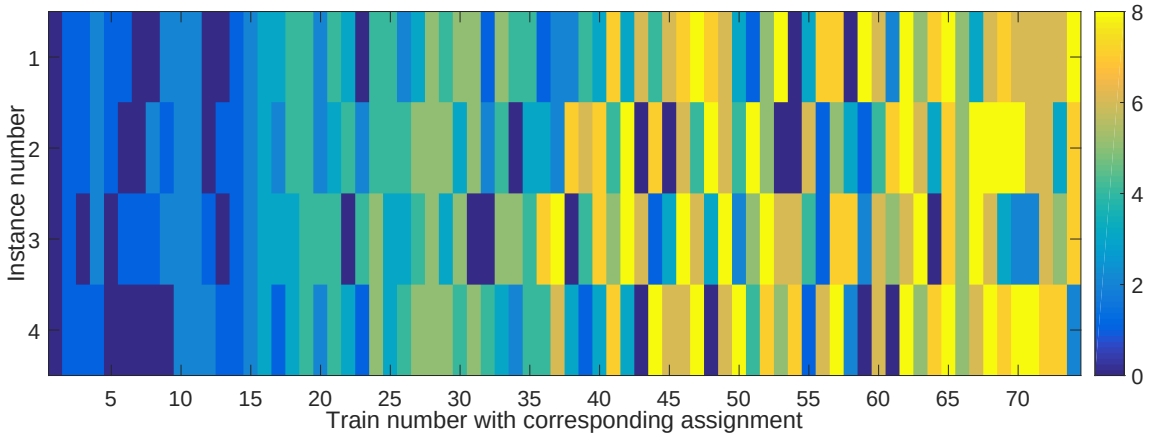

Figure 6.12: Visualization of train assignments that yield an equivalent performance

Chapter 7

Discussion

In chapter 3 we identified the Real-Time Track Assignment Problem (RT-TAP) as an assignment problem that resulted from deviations in the original schedule because of disruptions and last-minute requests of train operators. In order to provide a solution for the RT-TAP, we developed the following aim that stayed central in this research:

To develop a fast and easy usable Decision Support System (DSS) that, if delays occur, reassigns inbound trains to tracks in order to minimize the total weighted delays of outbound trains at any railway yard at real-time.

After formulating the model as a mixed integer program, we successfully developed a DSS that reassigns inbound trains by minimizing the total weighted delays at a railway yard by using a Brute Force (BF) attack. Because of the computational complexity of the problem, a BF attack would not work when considering larger instances. For this reason, we developed a problem specific Genetic Algorithm (GA) that tries to find a ”reasonable good” solution based on natural selection and survival of the fittest and a First Scheduled First Served (FSFS) heuristic for fast computation of solutions.

To validate the solutions methods, we first performed numerical experiments on a smaller instance in order to determine the optimality gap between the GA, FSFS, and the BF approach. To validate the performance of the GA for an actual case including real-time changes in arrival times of trains, we executed the model in a 24h simulation based on historical data, provided by the Rail Infrastructure Manager (RIM), of a Dutch railway yard named Waalhaven Zuid (Whz), and compared it with a FSFS heuristic approach.

When considering the smaller instances, we saw that when we don’t consider the train lengths, there is no significant difference between the computational performance of the three solution algorithms. However, when considering the variation of length of trains, we noticed that the GA performed up to 48% better than the FSFS algorithm in one of the instances. In another case, while the FSFS reduced the total delay to 6h 24’ 45”, the GA was able to minimize the same input to a total delay of 4h 41’ 48”. In all instances, the GA equaled the solution provided by the BF approach.

When it comes to minimizing the total weighted delay for larger instances, the model provided optimal solutions within 1,5 to 2 minutes of computation for a railway yard with 9 tracks and 74 inbound trains. When comparing the GA with the FSFS algorithm, we showed that there was again no significant difference in delay when excluding the train length from the input. However, when we include different train lengths (i.e. not all trains can be parked at all tracks), we saw that the GA performed slightly better than the FSFS assignment method, with a delay gap of 9 minutes and 35 seconds on top of a total delay of 4h 18’ 36”. An important finding was that at the end of the real-time

simulation, the minimized delay determined by the GA when including and excluding the track length was exactly the same. The FSFS however perceived some difficulties and was not able to convert to the same delay. This proves that the FSFS approach was actually dependent on the input provided.

The reasons for these extreme results in the smaller instances are related to how the heuristic approach works. Because the heuristic assigns the train that is first to arrive at the yard first, the approach might assign short trains to long tracks. Of course, this latter is not a direct problem, however if the processing time of a short train is significantly longer than that of the longer train, the long train has no other option than to wait before the long track has been cleared, resulting in a large delay. When we consider the larger instance, based on the finding of the smaller instances, we would have expected the optimality gap between the FSFS and the GA to be much higher than actually modeled. There are several possibilities of why this was not the case. Firstly, because the yard is consists of more tracks, it is less likely that all short tracks were occupied at the point when a long train arrives. Secondly, in the assumptions, we assumed similar process times of trains. This means that if we consider a FSFS approach, the probability of long trains with short processes having to wait on trains with long processes occupying the tracks is much less then when considering similar process times. With a large variation in process times, as in the smaller instance, we expect the difference of the FSFS approach to worsen compared to the GA.

Because of the many local optima, we noticed that when rerunning the model again, although the total minimized delay of the trains remained the same, the assignment was completely knocked over. Although this could be considered as a large limitation of this model, it can easily be fixed by e.g. adding an additional soft objective by minimizing the total changes made with regards to the previous schedule. When including an extension, the number of local optima would actually be beneficial for the model. This is because where heuristics and meta-heuristic approaches are known to sometimes get stuck in a local optimum, with a high amount of possible solutions it is easy to convert to another one. Especially this latter statement contributes to the stability of the model.

During the execution of the 24h simulation, we saw that at a certain point the FSFS and the GA both calculated some delay. Interestingly, later this delay was again reduced to 0. The train who got the delay was one close to the yard. This delay was needed because another train was originally planned at a time instance with limited capacity. Because the original train was earlier to arrive, and the total delay would be more if this train would be delayed, the model decided that it was better to delay the train close to the yard instead of eliminating the schedule of the originally planned train. To tackle this issue, we tried to estimate the arrival of trains by using the delay distribution of the two months of historical data received by ProRail. Although we saw that the total delay could actually be reduced to 0, the distributions did not reflect the reality at all. The reason why the distributions were able to reduce the delays, was because the arrival of originally planned trains was not considered anymore at a critical point because. A reasonable solution approach would be to apply a train weighting for trains closer to the yard. This would mean that although the total delay of the GA would be larger than the FSFS, the total weighted delay would be much less. This latter statement shows the relevance of applying a train weighting again and further highlights the performance gap between the GA and the FSFS approach.

Main contributions

To sum up the main contribution of this research:

- we provided new insights in the (computational) complexity of Real-Time track assignment;
- developed a mathematical program that is able to solve the RT-TAP;
- developed a problem specific GA that solves the RT-TAP within 1,5 to 2 minutes without, to best of our knowledge, compromising the global optimum;
- created new insights into the relation between the complexity of the assignment and the input of the model, which were not earlier discussed in the literature.

Limitations

Concerning the limitations to the approach:

- rail freight transportation has many uncertainties about the future. E.g. next to Whz was the Rail Service Center (RSC). If a train would finish its operations at RSC before or after scheduled, it would need a immediate assignment at the yard. It is uncertain what the effects are with this model on the assignment when this would happen;
- the model successfully allocated trains to track. However, we found many local optima in the solution space. While the minimized delays remained the same, the assignment was completely knocked over every time when running the model again;
- we can apply large weightings when a train is parked, and therefore fix the delay of a train, however, because we were not able to fix tracks, the model can still reassign a train to another track even if the train is already parked;
- we compared the GA with a FSFS approach. Although this latter approach is widely used in literature, it is unknown how this approach reflects the actual real-time assignments of a Train Dispatcher (TD);
- because visiting an actually traffic control post was not possible, we could not compare the proposed method with an actual real-time assignment of a TD. It is therefore unknown how this model will improve the current assignment process;

To sum up the main limitation with regards to the numerical experiment:

- the Numerical Experiment was only limited to the railway yard Whz. The performance of the model at other railway yards is therefore unknown and could differ;
- the data provided by ProRail showed many gaps with regards to the data needed for the model to perform. For example, the data did not include all lengths of all trains, did not show an arrival time of trains with a corresponding departure, and therefore we also assumed the process times of each train at the yard. Especially the assumption with regards to the process times may have improved the performance of the FSFS algorithm. As we saw from our own made scenario, when considering a large deviation in process times and I the gap between the GA and the FSFS algorithm is much higher than with similar processes;
- the numerical experiments are tested based on executed historical data. This means that the data could give a distorted picture as the problems that occurred at real-time problems might already be solved in this data;
- we made many assumptions with regards to length, process times and arrival times of trains which where needed for the model to perform. If these assumptions are relaxed, the results could differ than presented in this thesis.

Future research directions

Concerning the limitation of the approach, future research direction may consider extending the model with regards to the number of changes made in the schedule and be able to fix track assignments of a train. This could be done for example adding soft constraints or by applying 'section optimization', where we only consider optimizing the part which includes a delayed train. Moreover, more detailed observations can be performed to better understand the task and decisions made by a TD, and show a better comparison between the model and the actual real-time track assignments.

When considering an actual implementation of the model, it is important to first have a better information exchange so more accurate data is known. For example, the data of ProRail showed a high stochastic arrival of trains. Trains might just pop-up when finished with operations at RSC, which made it very difficult to get a stable assignment. It might therefore be interested to perform research for an integrated approach for data exchange to gain better insights in when trains arrive, depart, have delays, etc.

We tried to make a better expectation of the arrival of trains by applying a delay distribution. Although this method did not reflect the reality at all, a stochastic optimization (i.e. Monte Carlo), could be tried to make a better approximation of the arrivals of trains. Furthermore, to increase the robustness of the assignment, it might be an idea to incorporate an buffer of trains as a soft constraint to the mathematical model.

With regards to additional application of the model, because of the fast and accurate performance, similar as discussed by Gilg et al. (2018), we see the possibility to enhance the proposed model for the task of designing railway yards, assessing the effects of putting tracks out of service for maintenance, or for a sensitivity analysis of bottlenecks on railway yards in the overall network.

When considering improving the performance of the solution method (i.e. of the GA), we saw that the solutions provided by the FSFS algorithm were already generally good solutions. As van Hove (2019) also suggested in her thesis, running the model with a warm start, i.e. starting with an initial solution generated by a quick method, could contribute to the computational performance and stability of the assignments of the model.

Chapter 8

Conclusion

The thesis at hand aimed in creating a Decision Support System (DSS) that assists a Train Dispatcher (TD) in finding optimal track assignments for each inbound train by minimizing the total weighted delay at any railway yard. Real-Time assignment problems resulting from large stochastic arrival of freight trains is formulated as the Real-Time Track Assignment Problem (RT-TAP). We presented the model as a mixed integer program and proved the general case to be NP-hard because of its strong relation with the Vehicle Routing Problem (VRP). By reason of the computational complexity of the problem, a Brute Force (BF) approach would not work to solve the RT-TAP when considering larger instances at real-time. Therefore, a problem specific Genetic Algorithm (GA) was employed for finding 'reasonable good' solutions to the problem. After the execution of several tests, results showed that there is no optimality gap between the performance of the BF method and the GA for smaller instances.

Application of the model shows that when excluding the lengths from the input, the results indicate a small optimality gap between the overall performance of the GA and the First Scheduled First Served (FSFS) approach for both smaller and larger instances in favor of the GA. When considering track limitations of trains (i.e. not all trains can be parked at all tracks), the results showed for smaller instances that the GA performed up to 48% better than the FSFS. Better performance of the GA also accounted on the larger instances as the FSFS was not able to provide solutions as good as the GA.

To conclude, the obtained results demonstrated that while the GA gave a stable performance, the performance of the FSFS approach was strongly related with the input, such as the length and arrival times of the trains. This latter showed that there is a relation between the complexity of the problem and the input that is provided. The use of this DSS can therefore not only stabilize, but also improve current decisions making with regards to real-time track assignment, and should be considered especially before e.g. investing in additional infrastructure at railway yards.

Bibliography

Bettinelli, A., Santini, A., and Vigo, D. (2017). A real-time conflict solution algorithm for the train rescheduling problem. *Transportation Research part B*, vol 106:237–265.

Boysen, N., Fiedner, M., Jaehn, F., and Pesch, E. (2012). Shunting yard operations: Theoretical aspects and applications. *European Journal of Operational Research*, vol 220(1):1–14.

Briggs, N. and Beck, C. (2007). Modeling train delays with q-exponential functions. *Physica A: Statistical Mechanics and its Applications*, vol 378(2):498–504.

Cacchiani, V., Huisman, D., Kidd, M., Kroon, L., Toth, P., Veelenturf, L., and Wagenaar, J. (2014). An overview of recovery models and algorithms for real-time railway rescheduling. *Transportation Research Part B*, vol 63:15–37.

Cai, X. and Goh, C. (1994). A fast heuristic for the train scheduling problem. *Computers & Operations Research*, vol 21(5):499–510.

Corman, F., D'Ariano, A., Pacciarelli, D., and Pranzo, M. (2010). A tabu search algorithm for rerouting trains during rail operations. *Transportation Research Part B*, vol 44:175–192.

D'Ariano, A. (2015). Improving real-time train dispatching: Models, algorithms and applications. [PhD Thesis] `https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=2ahUKEwjhr4e-3JviAhUSDuwKHTImBHIQFjABegQIAhAC&url=https%3A%2F%2Frepository.tudelft.nl%2Fislandora%2Fobject%2Fuuid%3A178b886e-d6c8-4d39-be5d-03d9fa3a680f%2Fdatastream%2FOBJ%2Fdownload&usg=AOvVaw3PKpXVxJTtJ-NtvKMWgkWx`. Accessed on 14 May 2019.

Dollevoet, T. and Huisman, D. (2014). Fast heuristics for delay management with passenger rerouting. *Public Transport*, 6(1):67–84.

Dollevoet, T., Huisman, D., Kroon, L. G., Veelenturf, L. P., and Wagenaar, J. C. (2010). Application of an iterative framework for real-time railway rescheduling. *Computers Operations Research*, vol 78:203–217.

Dollevoet, T., Schmidt, M., and Schöbel, A. (2011). Delay management including capacities of stations. In *ATMOS*.

European Commission (2014). Fourth report on monitoring development of the rail market. [PDF] `https://eur-lex.europa.eu/resource.html?uri=cellar:d261b1f8-f5f4-11e3-831f-01aa75ed71a1.0002.01/DOC_2&format=PDF`. Accessed on 14 May 2019.

Fang, W. (2015). A survey on problem models and solution approaches to rescheduling in railway networks. *IEEE Transactions on Intelligent Transportation Systems*, vol 16(6):2997–3016.

Fischettia, M. and Monaci, M. (2017). Using a general-purpose mixed-integer linear programming solver for the practical solution of real-time train rescheduling. *European Journal of Operational Research*, vol 263:258–264.

Gatto, M., Maue, J., Mihalak, M., and Widmayer, P. (2009). *Shunting for dummies: an introductory algorithmic survey*. Springer.

Gestrelius, S., Aronsson, M., Joborn, M., and Bohlin, M. (2017). Towards a comprehensive model for track allocation and rolltime scheduling at marshalling yards. *Journal of Rail Transport Planning Management*, vol 7:157–170.

Gilg, B., Klug, T., Martienssen, R., Paat, J., Schlechte, T., Schulz, C., Seymen, S., and Tesch, A. (2018). Conflict-free railway track assignment at depots. *Rail Transport Planning & Management*, vol 8:16–28.

Givoni, M. and Banister, D. (2013). *Moving Towards Low Carbon Mobility*. Edward Elgar.

Gkiotsalitis, K., Wu, Z., and Cats, O. (2019). A cost-minimization model for bus fleet allocation featuring the tactical generation of short-turning and interlining options. *Transportation research. Part C: Emerging technologies*, vol 98:14–36.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, vol 13(5):533–549.

Haahr, J. and Lusby, R. (2017). Integrating rolling stock scheduling with train unit. *European Journal of Operational Research*, vol 259(2):452–468.

Haahr, J., Lusby, R., and Wagenaar, J. (2017). Optimization methods for the train unit shunting problem. *European Journal of Operational Research*, vol 262(3):981–995.

Hansmann, R. S. and Zimmermann, U. T. (2008). *Optimal Sorting of Rolling Stock at Hump Yards*, pages 189–203. Springer Berlin Heidelberg, Berlin, Heidelberg.

Ho, T. and Yeung, T. (2001). Railway junction traffic control by heuristic methods. *Electric Power Applications, IEE Proceedings*, 148:77–84.

Hoenders, C. (2018). Interview with Cor Hoenders from Rail Cargo, 6 november 2018.

Holland, J. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press, Michigan.

Jaehn, F., Otto, A., and Seifried, K. (2018). Shunting operations at flat yards: retrieving freight railcars from storage tracks. *OR Spectrum*, vol 40(2):367–393.

Jaehn, F., Rieder, J., and Wiehl, A. (2015). Single-stage shunting minimizing weighted departure times. *Omega*, vol 52:133–141.

Josyula, S. P., Krasemann, J. T., and Lundberg, L. (2018). A parallel algorithm for train rescheduling. *Transportation Research Part C*, vol 95:545–569.

Karp, R. M. (1972). Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103.

Keyrail (2014). Process reports of trains at Waalhaven Zuid. [report].

Liong, C.-Y., Wan, I., and Omar, K. (2008). Vehicle routing problem: Models and solutions. *Journal of Quality Measurement and Analysis*, vol 4:205–218.

Narayanaswami, S. and Rangaraj, N. (2011). Scheduling and rescheduling of railway operations: A review and expository analysis. *Technology Operation Management*, vol 2(2):102–122.

Network Rail (2016). Value and importance of rail freight. [PDF] `http://www.networkrail.co.uk/wp-content/uploads/2016/11/The-Value-and-Importance-of-rail-Freight-summary-report.pdf`. Accessed on 07 February 2019.

Pachl, J. (2004). *Railway operations and Control*. Vtd Rail Pub.

Pferschy, U. and Staněk, R. (2017). Generating subtour elimination constraints for the TSP from pure integer solutions. *Central European Journal of Operations Research*, vol 25(1):231–260.

Pinedo, M. L. (2016). *Scheduling: Theory, Algorithms, and Systems*. Springer, Cham.

Piner, D. and Condry, D. (2017). International best practices in managing unplanned disruption to suburban rail services. *Transportation Research Procedia*, vol 25:4403–4410.

ProRail (2015). Two months of historical data about train arrivals and departures at Waalhaven Zuid.

Rahim, S. K. N. A., Bargiela, A., and Qu, R. (2013). Hill climbing versus genetic algorithm optimization in solving the examination timetabling problem. *Proceedings of the 2nd International Conference on Operations Research and Enterprise Systems*, vol 1:43–52.

Roland Rail (2015). Dienstregeling goederentreinen from Roland Rail schedule database. [Report].

Schachtebeck, M. and Schöbel, A. (2008). Ip-based techniques for delay management with priority decisions. In *ATMOS*.

Schachtebeck, M. and Schöbel, A. (2010). To wait or not to wait and who goes first? Delay management with priority decisions. *Transportation Science*, vol 44(3):307–321.

Schöbel, A. (2007). *Integer Programming Approaches for Solving the Delay Management Problem*. Springer, Berlin, Heidelberg. In: Geraets F., Kroon L., Schoebel A., Wagner D., Zaroliagis C.D. (eds) Algorithmic Methods for Railway Optimization. Lecture Notes in Computer Science, vol 4359.

Schöbel, A. (2009). Capacity constraints in delay management. *Public Transport*, vol 1(2):135–154.

Spoorcafe (2018). Network gathering with several rail freight transportation companies, 29 november 2018.

Sporenplan (2016). Drawings of Waalhaven Zuid. [drawings] `www.sporenplan.nl`. Accessed on 27 January 2019.

Timmermans, P. (2018). Interview with Patrick Timmermans from ProRail, 12 october 2018.

Törnquist, J. (2010). Greedy algorithm for railway traffic re-scheduling during disturbances: A swedish case. *IET Intelligent Transport Systems*, vol 4(4):75–386.

Törnquist, J. (2012). Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances. *Transportation research. Part C: Emerging technologies*, vol 20(1):62–78.

Törnquist, J. and Persson, J. A. (2005). Train traffic deviation handling using tabu search and simulated annealing. *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, pages 73a–73a.

U.S. Department of Transportation (1999). Understanding how train dispatchers manage and control trains: A cognitive task analysis of a distributed team planning task. [PDF] `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.876.1554&rep=rep1&type=pdf`. Accessed on 14 May 2019.

U.S. Railroad (2008). A review of capacity and performance data. [online] . Accessed on 07 January 2019.

van Hove, E. (2019). Train routing at the shunt yard : a disjoint paths approach. [Master Thesis] `http://essay.utwente.nl/77267/`. Accessed on 08 May 2019.

Winter, T. and Zimmermann, U. (2000). Real-time dispatch of trams in storage yards. *Annals of Operational Research*, vol 96:287–315.

Zhan, S., Kroon, L. G., Veelenturf, L. P., and Wagenaar, J. C. (2015). Real-time high-speed train rescheduling in case of a complete blockage. *Transportation Research Part B*, vol 78:182–201.

Appendix A

MOD file of the AMPL model

The following shows the .MOD file necessary for running the AMPL model. The model is based on the mathematical formulation described in chapter 4.

```
reset;
#sets
set N ordered;
set K;
set z_b;
set z_e;

#input for trains
param arrival_n {i in N};
param departure_n {i in N};
param length_n {i in N};
param weighting_n {i in N};
param beta; #followup time of trains
var process {i in N};

#input for tracks
param length_k {k in K};

#other parameters
param C;
param g := card(N);

#defining the subtour elimination
set S := 0 .. (2**g-1);
set POW {s in S} := {n in N: (s div 2**(ord(n,N)-1)) mod 2 == 1};

#binary varibales, assigned or not
var x {i in N , j in N, k in K} binary;
var y {q in z_b , i in N, k in K} binary;
var z {i in N , p in z_e, k in K} binary;

#new determined
var aks_arrival_n {i in N, k in K};
var aks_departure_n {i in N, k in K};
var tilde_arrival_n {i in N};
```

```
var tilde_departure_n {i in N};

#result
var differ {i in N};
var W_differ {i in N};

### SEQUENCE ###
subject to seq1 {i in N}:
    sum {k in K} (sum {j in N} x[i,j,k] + sum {p in z_e} z[i,p,k]) = 1;
subject to seq2 {j in N}:
    sum {k in K} (sum {i in N} x[i,j,k] + sum {q in z_b} y[q,j,k]) = 1;
subject to seq3 {k in K, q in z_b}:
    sum {i in N} y[q,i,k] <= 1;
subject to  seq4 {k in K, p in z_e}:
    sum {i in N} z[i,p,k] <= 1;
subject to seq5 {i in N, k in K}:
    (sum {j in N} x[j,i,k] + sum {q in z_b} y[q,i,k]) - (sum {j in N} x[i,j,k] + sum {p in z_e} z[i,p,k]) = 0;
subject to SUBS {s in S: card(POW[s]) >= 1}:
    sum {i in POW[s], j in POW[s], k in K} x[i,j,k] <= card(POW[s]) - 1;

### TIME ###
subject to time1 {i in N, j in N, k in K}:
    aks_departure_n[i,k] - C*(1 - x[i,j,k]) + beta <= aks_arrival_n[j,k];
subject to time2 {i in N, k in K}:
     aks_arrival_n[i,k] >= arrival_n[i];
subject to time3 {i in N, k in K}:
     tilde_arrival_n[i] >= aks_arrival_n[i,k];
subject to time4 {i in N, k in K}:
    aks_departure_n[i,k] = aks_arrival_n[i,k] + process[i];
subject to time5 {i in N}:
    tilde_departure_n[i] = tilde_arrival_n[i] + process[i];
subject to ProcessTime {i in N}:
    departure_n[i] - arrival_n[i] = process[i];

### LENGTH ###
subject to len1 {i in N, j in N, k in K: length_n[i] > length_k[k]}:
    x[i,j,k] = 0;
subject to len2 {q in z_b, i in N, k in K: length_n[i] > length_k[k]}:
    y[q,i,k] = 0;
subject to len3 {i in N, p in z_e, k in K: length_n[i] > length_k[k]}:
    z[i,p,k] = 0;

### RESULT ###
subject to result {i in N}:
    differ[i] = tilde_departure_n[i] - departure_n[i];
subject to Weigthed_result {i in N}:
    W_differ[i] = ((tilde_departure_n[i] - departure_n[i]) * weighting_n[i]);

### MINIMIZED FUNCTION ###
minimize f:
    sum {i in N} ((tilde_departure_n[i] - departure_n[i]) * weighting_n[i]);
```

Appendix B

DATA file of the AMPL model

The following shows an example of a .DATA file necessary for the AMPL model to run.

```
### SCENARIO 2 ###
set N := train1 train2 train3 train4 train5 train6 train7 train8 train9 train10 train11 train12 train13;
set K := track1 track2 track3 track4;
set z_b:= Begin;
set z_e:= End;

param C := 10000000;  ### DO NOT CHANGE ###
param beta := 300; #follow up time of trains is: 300 seconds = 5 min

### INPUT TRACKS ###
param length_k :=
track1 350
track2 500
track3 200
track4 800
;

### INPUT TRAINS ###
param arrival_n :=
train1 15300
train2 29700
train3 29700
train4 30900
train5 28900
train6 33900
train7 41100
train8 35940
train9 45540
train10 46800
train11 36000
train12 53700
train13 57780
;

param departure_n :=
```

```
train1 26040
train2 35400
train3 35400
train4 36600
train5 75000
train6 37800
train7 46800
train8 44040
train9 53640
train10 52500
train11 49200
train12 59400
train13 65880
;

param length_n:=
train1 750
train2 200
train3 750
train4 400
train5 350
train6 600
train7 800
train8 200
train9 100
train10 50
train11 600
train12 300
train13 650
;

param weighting_n:=
train1 1
train2 1
train3 1
train4 1
train5 1
train6 1
train7 1
train8 1
train9 1
train10 1
train11 1
train12 1
train13 1
;
```

Appendix C

RUN file of the AMPL model

The following shows the .RUN file of the AMPL model.

```
reset;

model '/Users/bramschasfoort/Desktop/amplide/Thesis/RTTAP.mod';
data '/Users/bramschasfoort/Desktop/amplide/Thesis/RTTAP.dat';

option solver cplex;

solve;

display x;
display y;
display z;


display arrival_n;
display tilde_arrival_n;
display departure_n;
display tilde_departure_n;
display differ;
#display W_differ;
#display length_n;
#display length_k;

display _total_solve_time;
```

Appendix D

Overview drawing of Waalhaven Zuid

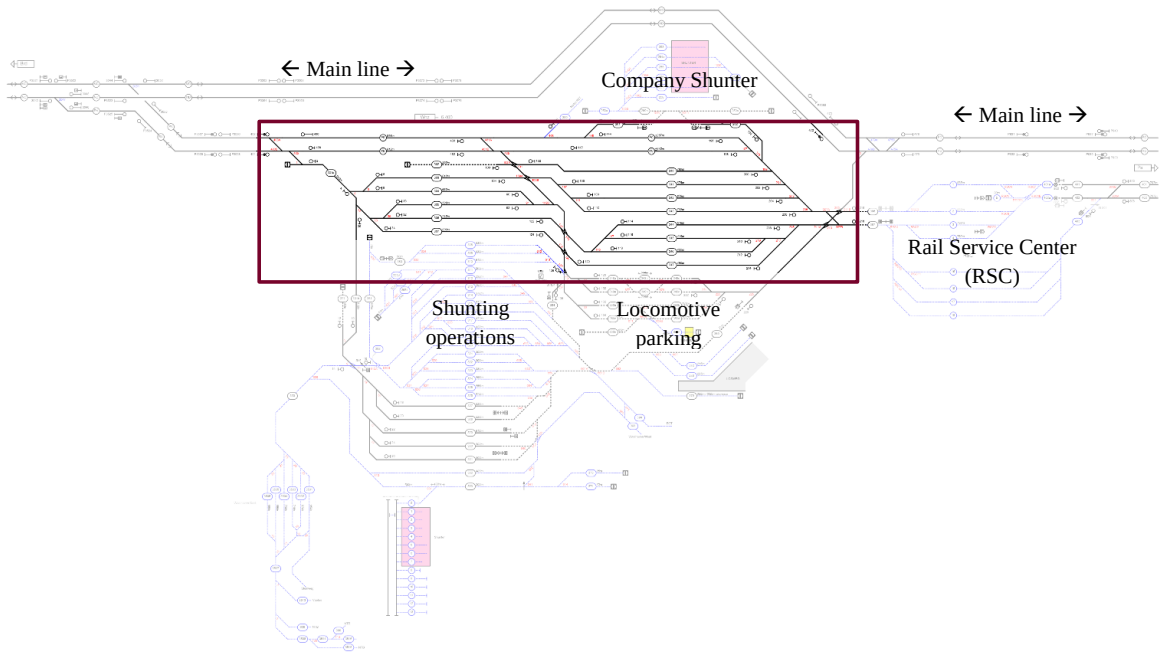

Figure D.1: Overview drawing of Waalhaven Zuid (Sporenplan, 2016)

Appendix E

Determining process times of trains

Problems in the received data from ProRail is that the data did not show an arrival time with its corresponding departure time. The data received consisted of a list of train number including if the train is ether arriving, departing or driving though the yard, and the times as discussed in previous section. This seemed like a problem however, all trains operating on the network in the Netherlands are linked with a train number.

The train number is linked with its origin and destination and with the road travelled by a train (see figure E.1). If for example a train departs from A and arrives at B, it could receive an even train number 40000. If the same train would again depart from B in the direction of A, it will receive an uneven train number in the same sequence (for example 40001). With this knowledge, a train departing from B in the direction of destination C, could never receive a train number in the sequence of 40000, but will get a number of for example 40300. The same train then departing again from C in the direction of B would then receive the number 40301. Important to know is that all train numbers will be reset at midnight.

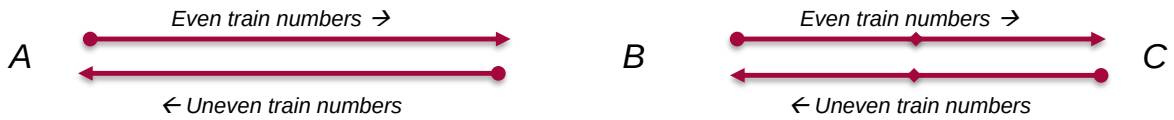

Figure E.1: Train number setup of the Dutch railway network

By knowing the train number setup, we could link the trains in the list with the same train number on the same day. Train numbers that had a number arriving and the same number departing at the same day can be assumed that the train was driving through the yard and for example needed to wait for a path. The trains with a number when arriving and a number in the same sequence when departing where assumed to have a destination at Waalhaven Zuid (Whz) and therefore Rail Service Center (RSC). To further conform the data, many trains with data from the year plan including train numbers, origin, destination and in-between stops were also provided by Roland Rail (2015), and could therefore be double checked with the data received from ProRail. The trains arriving and departing from the yard can therefore be divided in two different categories.

1. Trains passing through Whz.
2. Trains with origin or destination Whz (RSC).

With the above knowledge, trains arriving and departing with the same train numbers on the same day could be linked with each other and gave a trains' arrival and corresponding departure time which then also gives the time a train remained at the yard. The problem with the other category is that because the origin or destination is Whz, it could not be determined from the data provided how long a train had to remain at the yard, or more specific, a track.

Process times at Waalhaven Zuid

Because from the received data from ProRail could not be determined how long a train from the second category was occupying the track, additional data regarding the processes at the railway yard was required. According to documentation from Keyrail (2014), trains from the second category must go through some operations. When arriving at Whz from the network, the process described in table E.1 should be used. When the train arrives at Whz from RSC and is prepared for the network, table E.2 is applied.

Table E.1: Processes when arriving at Waalhaven Zuid from the network and departing to Rail Service Center

Step	*Time of Operations*	*Time needed*
1.	Block train arrives at track $k = \{1, 2, ..., K\}$ (incl. electrical locomotive)	5 min
2.	Electrical locomotive is detached and parked	15 min
3.	Train receives arrival check	30 min
4.	Optional buffer	0 min
5.	Diesel locomotive is attached (ether front of back of the train)	20 min
6.	Train pushed or pulled to RSC	20 min
7.	Locomotive is detached. Parking diesel locomotive	5 min
	Total time needed at the yard after arriving at Whz : 95 min	

Table E.2: Processes when departing from Waalhaven Zuid after visiting Rail Service Center

Step	*Time of Operations*	*Time needed*
1.	Diesel locomotive is moved to RSC and attached to the train	10 min
2.	Block train moved to track $k = \{1, 2, ..., K\}$ (incl. diesel loc).	20 min
3.	Diesel locomotive is detached and parked	20 min
4.	Electrical locomotive is attached	25 min
5.	Large break test	45 min
6.	Waiting for network path to be cleared	10 min
7.	Departure from Whz	5 min
	Total time needed at the yard before departing from Whz : 135 min	

Considering the above, the following process times are assumed:

- Trains arriving at Whz have a process operation time of 1 hour and 35 min.
- Trains departing from Whz have a process operation time of 2 hours and 15 min.

Including the arrival time of the trains and the process times determined above, we determined the following original train schedule:

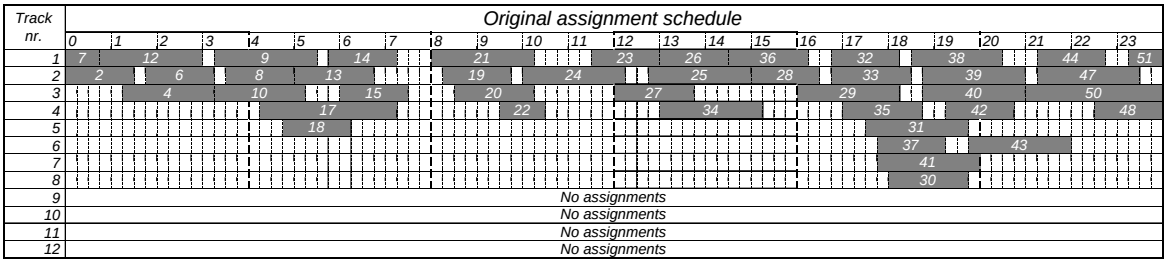

Figure E.2: Visualization of the original train scheduled

Appendix F

Generating Length

The track assignment and the length of the trains was received in the same document. However, the problems considering the data was that it was incomplete. In the overview only a small percentage of the lengths and executed track assignments were shown. Because the data might miss the normative lengths, the received lengths of trains are not considered in the model. Because ProRail was not able to provide the train lengths we generate a random length for each train.

Constructing the train length for each train is done as follows:

Begin function

For *every* $i \in N$: *Generate random number from* 0 *to* 100%

If *random nr.* $> 95\%$: $l_i = 740$ *meter*

Else if *random nr.* $> 80\%$: $l_i =$ *random nr. between* $(620 - 700)$ *meter*

Else if *random nr.* $> 50\%$: $l_i =$ *random nr. between* $(550 - 600)$ *meter*

Else $l_i = 400$ *meter*

End function

The only condition with regards to the length was that if we execute the model based on original data, the total delay has to be 0 seconds. This is because also in practice no schedule will be made in the beginning of the day which starts with a delay.

Appendix G

Delay distribution of trains

We saw that the optimization of the total delays is highly dependent on the original and planned schedule of other trains. For example, we saw that because a train was delayed we reduced the expected total delay from 55 minutes and 58 seconds at 6 am, to a 0 minute delay at 10 am. The main reason for applying a stochastic optimization is to see what happens to the total weighted delay when including different distributions on the arrival times of trains.

The stochastic optimization is done at two different moments on both scenarios where we consider the track limitations and at two different moments where we do not consider the lengths. In the stochastic model we compute solutions based on the trains expected delays with four different probability ranges (50%, 30% to 70%, 10% to 90%, 5% to 95%). Each probability rage is computed five times.

Based on the data received from ProRail, we determined the following mean values and standard deviations (See table G.1).

Table G.1: Trains' delay, mean values and variances

	Mean value	***Standard deviation***	***Sample size***
Original to executed (arrival)	$00:50:35$	$02:19:03$	260
Planned to executed (arrival)	$00:01:18$	$00:19:07$	289

Figure G.1 and G.2 visualize the distributions and the histograms of the delays on the trains' arrivals from the original data compared with the executed data, and the planned data compared with the executed data respectively.

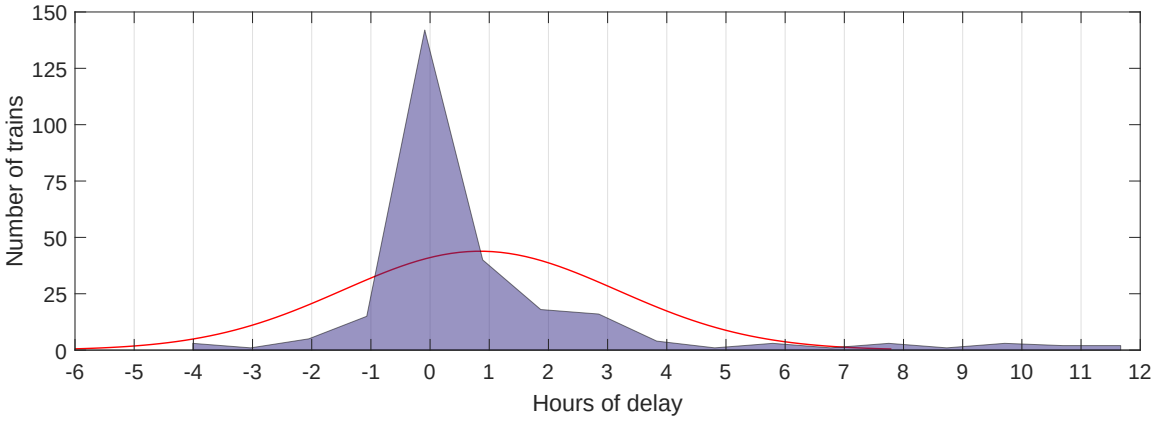

Figure G.1: Delay distribution on arrivals of trains (original compared with executed data)

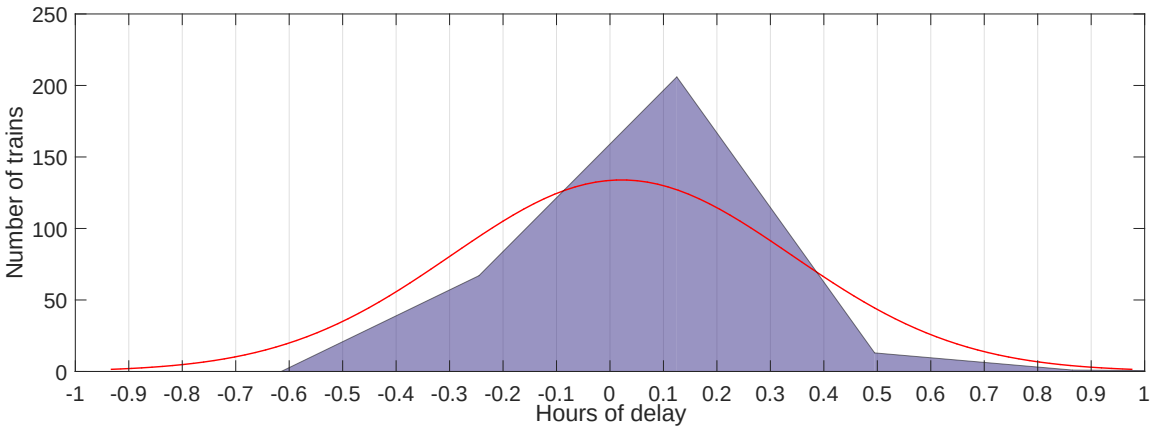

Figure G.2: Delay distribution on arrivals of trains (planned compared with executed data)

Results of stochastic optimization

Including the distributions determined above, we executed the model five different times for each different range at two different times on the two different scenarios.

During different tests we saw that at 46 am the model even optimized to a 0 second delay when the distributions were applied. On average, we see that the 50% distribution performs best for all scenarios. Figure G.3 shows the results when adding the different distributions on arrival times at two time instances for the scenario where the train length is not considered. From this figure one can observe that the total range of delays lays on average between 1h 17' 59" when including a distribution range of 10% to 90%, and a 0h 22' 33" delay when using the 40% distribution for the time instance of 6 am. At the time instance of 10 am, one can observe that the total range of delays lays on average between 0h 43' 24" when including a 90% distribution, and that the delay converges to 0 seconds when excluding the distributions. Important to note is that the distribution ranges of 30% to 70%, 10% to 90%, and 5% to 95%, all have at least one result where the total weighted delay converged to 0 seconds.

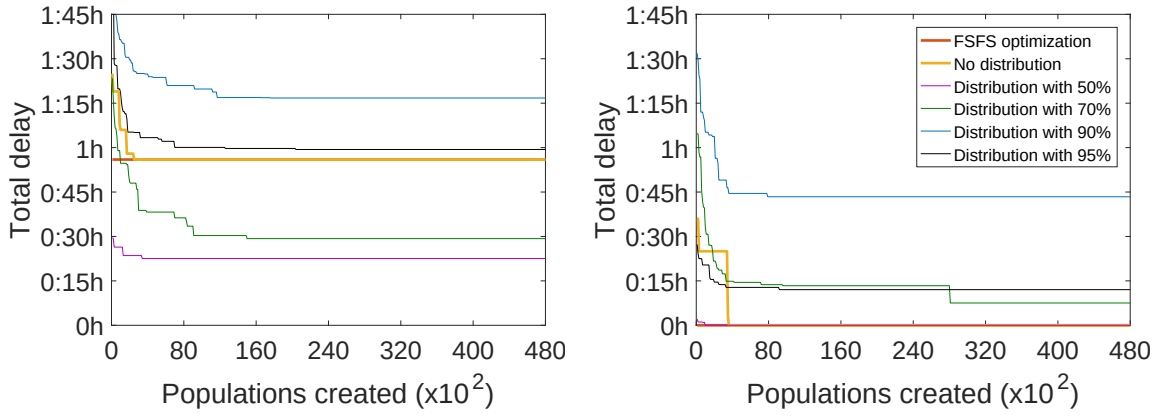

Figure G.3: Stochastic optimization excluding the length of trains for 6 am (left) and 10 am (right)

Figure G.4 visualizes the optimizations when we include the length of the trains for time instance 6 am and 10 am. From this figure one can observe that at 6 am the Genetic Algorithm (GA) performed slightly better than the First Scheduled First Served (FSFS) scenario. When adding the distributions we can conclude that on average all tests perform better than the FSFS algorithm. When applying a distribution of 50%, the total weighted delay again converges to 0h 22' 33" delay. At 10 am we see that because we apply distributions on arrival of original data delays occur further in the schedule.

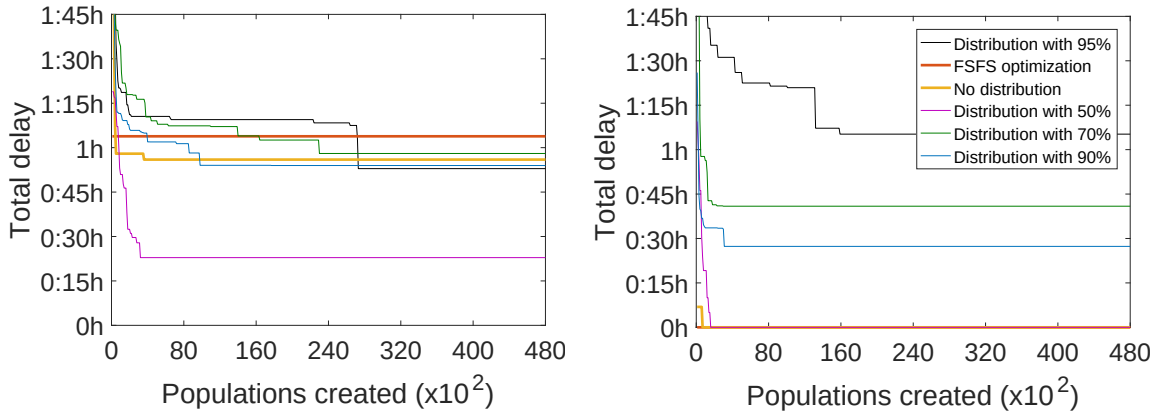

Figure G.4: Stochastic optimization including the length of trains for 6 am (left) and 10 am (right)

On average the 50% distribution performs best in all four scenarios. Reason for the reduction could be that in both cases the calculated delays at 6 am is based on the delay on trains that cannot be assigned because executed trains are parked at that point. The original place was reserved but the train was eventually not assigned there. If we include the distributions, we shift the original planned train that had trouble with assignment at 6 am, to an other moment where their assignment becomes less relevant. Hence, the total delays can be further optimized, even to a point where they receive a 0 second delay for some of the instances.

Another important finding is that the solutions provided by the model when including a stochastic optimization have no relation with the executed data of the model. Because the delay of each train is very specific the distribution do not give a valid estimation on which train does get the actual delay when the model is performed. For this reason the delays of trains determined by the model should always be decided as late as possible.