

Jesse Jaspers



Optimising railway  
timetable robustness  
using random search  
metaheuristics

26

# Optimising railway timetable robustness using random search metaheuristics

Jesse Peter Jaspers

To obtain the degree of Master of Science in Applied Mathematics  
at Delft University of Technology (faculty of EEMCS),  
to be defended publicly on Monday August 24, 2026 at 13:00.

|                   |                             |
|-------------------|-----------------------------|
| Student number:   | 5645867                     |
| Project duration: | January – August, 2026      |
| Specialisation:   | Mathematics of Data Science |

Graduation committee:

|                           |                        |
|---------------------------|------------------------|
| Prof.dr. R.M.P. Goverde,  | daily supervisor       |
| Dr.ir. T.M.L. Janssen,    | daily supervisor       |
| Dr.ir. L.J.J. van Iersel, | responsible supervisor |
| Dr.ir. E. Lorist,         | independent member     |

An electronic version of this thesis is available at: <https://repository.tudelft.nl/>

# Acknowledgements

This thesis was written to complete the Master of Science Applied Mathematics at Delft University of Technology. I was supervised by Teun Janssen, from the Discrete Mathematics & Optimisation research group of the Delft Institute of Applied Mathematics, which is part of the Faculty of Electrical Engineering, Mathematics, and Computer Science. Additionally, I was supervised by Rob Goverde, from the Digital Rail Traffic Lab, which is part of the faculty of Civil Engineering and Geosciences. This thesis was written between January and August 2026.

I have always been interested in public transport. As my father once said, 'Jesse is probably the only four year old to know the meaning of a fare zone boundary'. Therefore, I decided to do a minor in mobility and transport during my bachelor programme in Applied Mathematics. This minor gave rise to this graduation project, where I could combine my interests of transport and mathematics. This project has allowed me to use both stochastics and optimisation knowledge obtained during my course of study. I have very much enjoyed applying this knowledge to the world of railways.

During this thesis, I used the Google search engine, which does contain an AI overview which has redirected me to websites and suggested translations. Apart from this, no artificial intelligence has been used during this project. So all ideas presented and words written were created by myself with the help of my supervisors.

I would like to thank both my supervisors for their advice. Rob Goverde, I would like to thank for accepting this project, correcting me whenever I got something wrong about railways and his advice on writing. Teun Janssen, I would like to thank for always listening carefully and for his suggestions on design options for metaheuristics. Additionally, I would like to thank Leo van Iersel and Emiel Lorist, for being part of my graduation committee.

This thesis concludes my five year period of studying in Delft. Therefore, I would like to thank my parents, Frans and Annemieke, for their continued support. My father should also be acknowledged as the creator of the cover image of this thesis. Finally, for giving me a memorable study time, I would like to thank: Dieter, Johan, Lars, Lorenzo, Lucas and Thijs.

*Jesse Jaspers  
The Hague, 2026*

# Layman's summary

## English version

In the Netherlands, each day there are over one million journeys made by train. Passengers hope to reach their destination on time. However, frequently small delays can occur. This can be caused by passenger boarding and alighting at a station taking longer than expected. Also bad weather or subpar performing trains can lead to delay. When a timetable is designed, these delays are taken into account. This is done by planning more time than strictly necessary for a train to complete its journey, we call this time supplement.

If you want to find the guaranteed best possible allocation of time supplement, then this takes a computer a long time to find. In this thesis, we instead use a certain kind of algorithms, which are called metaheuristics. Metaheuristics try to make modifications to a timetable, hoping to find a better solution. They are not guaranteed to find the best possible solution. However, in this thesis, it was discovered that they do manage to find good solutions within limited time. So metaheuristics can be used to adjust a timetable, such that small delays are better taken into account.

## Nederlandse versie

In Nederland worden dagelijks meer dan een miljoen reizen per trein gemaakt. Passagiers hopen hun bestemming op tijd te bereiken, maar worden vaak gehinderd door kleine vertragingen. Dit kan komen doordat in- en uitstappen op een station langer duurt dan verwacht. Ook slecht weer of een ondermaats presterende trein kan zorgen voor kleine vertragingen. Bij het maken van een dienstregeling wordt er rekening mee gehouden, dat zulke vertragingen kunnen optreden. Extra marge wordt gepland in de dienstregeling om deze vertragingen op te vangen.

Een computer doet er lang over om te bepalen wat de best mogelijke plekken zijn om marge toe te voegen in een dienstregeling. Daarom wordt in deze scriptie gebruik gemaakt van een groep algoritmes, die metaheuristieken worden genoemd. Metaheuristieken proberen steeds een aanpassing te maken aan een dienstregeling, in de hoop een betere oplossing te vinden. Het is niet gegarandeerd dat de best mogelijke oplossing wordt gevonden, maar in deze scriptie is wel bepaald dat metaheuristieken uitermate geschikt zijn om binnen beperkte tijd een goede dienstregeling te vinden.

# Abstract

When designing a railway timetable, it is important to consider that small delays can frequently occur. If a timetable can recover from small delays and this leads to little hindrance for other trains, we call this a robust timetable. Time supplement is planned in a timetable to improve robustness. Finding the best possible allocation of time supplement is computationally expensive. In this thesis, we instead use a group of algorithms called metaheuristics, which try to efficiently explore a search space to find near-optimal solutions. These metaheuristics are used to reallocate time supplement in a timetable. The timetable is only modified within a network, without requiring changes to the timetable outside of this network.

A graph-based model is used to evaluate timetables created by the metaheuristics based on a set of delay scenarios. These delay scenarios first consisted of sampled entrance delays and dwell times, later also fluctuations in running time were added. A computational study on the Utrecht-Arnhem corridor showed that metaheuristics are suitable for improving allocation of time supplement within limited time. Therefore, metaheuristics deserve more attention in future research to improve the robustness of railway timetables.

# Contents

|                                           |           |
|-------------------------------------------|-----------|
| <b>Acknowledgements</b>                   | <b>3</b>  |
| <b>Layman's summary</b>                   | <b>4</b>  |
| <b>Abstract</b>                           | <b>5</b>  |
| <b>1 Introduction</b>                     | <b>8</b>  |
| <b>2 Literature</b>                       | <b>12</b> |
| 2.1 Timetable preliminaries . . . . .     | 12        |
| 2.2 Robust timetabling . . . . .          | 15        |
| 2.3 Research goal . . . . .               | 22        |
| <b>3 Framework</b>                        | <b>23</b> |
| 3.1 Overview of notation . . . . .        | 24        |
| 3.2 Directed graph . . . . .              | 24        |
| 3.3 Feasibility check . . . . .           | 30        |
| 3.4 Delay propagation algorithm . . . . . | 31        |
| 3.5 Objective function . . . . .          | 33        |
| 3.6 Combining algorithms . . . . .        | 35        |
| 3.7 Timetable performance . . . . .       | 37        |
| <b>4 Delay scenarios</b>                  | <b>39</b> |
| 4.1 Validity of sampling . . . . .        | 39        |
| 4.2 Number of scenarios . . . . .         | 42        |
| 4.3 Dwell times . . . . .                 | 43        |
| 4.4 Entrance delays . . . . .             | 47        |
| <b>5 Random search metaheuristics</b>     | <b>49</b> |
| 5.1 Overview of metaheuristics . . . . .  | 49        |
| 5.2 Initial solution generation . . . . . | 50        |
| 5.3 Simulated Annealing . . . . .         | 51        |
| 5.4 Genetic Algorithms . . . . .          | 53        |
| 5.5 Differential Evolution . . . . .      | 57        |
| 5.6 Particle Swarm Optimisation . . . . . | 60        |
| 5.7 Convex Combinations . . . . .         | 60        |
| 5.8 Conclusion . . . . .                  | 61        |
| <b>6 Test case</b>                        | <b>62</b> |
| 6.1 Network . . . . .                     | 62        |
| 6.2 Parameter choices . . . . .           | 65        |
| 6.3 Original timetable . . . . .          | 67        |

|                                                         |            |
|---------------------------------------------------------|------------|
| <b>7 Experiments</b>                                    | <b>70</b>  |
| 7.1 Setup . . . . .                                     | 70         |
| 7.2 Local Search . . . . .                              | 71         |
| 7.3 Simulated Annealing . . . . .                       | 74         |
| 7.4 Genetic Algorithm . . . . .                         | 78         |
| 7.5 Differential Evolution . . . . .                    | 81         |
| 7.6 Convex Combinations . . . . .                       | 85         |
| 7.7 Comparing results . . . . .                         | 86         |
| <b>8 Analysing obtained timetable</b>                   | <b>90</b>  |
| 8.1 Adjust objective function . . . . .                 | 90         |
| 8.2 Obtained timetable . . . . .                        | 90         |
| 8.3 Delay breakdown . . . . .                           | 92         |
| 8.4 Impact of adjustments . . . . .                     | 94         |
| <b>9 Model adjustments</b>                              | <b>96</b>  |
| 9.1 Different objective function . . . . .              | 96         |
| 9.2 Change dwell times at Driebergen-Zeist . . . . .    | 96         |
| 9.3 Fixed departure time from initial station . . . . . | 97         |
| 9.4 Varying train running times . . . . .               | 97         |
| <b>10 Conclusion &amp; Discussion</b>                   | <b>102</b> |
| <b>Bibliography</b>                                     | <b>105</b> |
| <b>A Complexity class</b>                               | <b>111</b> |
| <b>B Alternative graphs</b>                             | <b>112</b> |
| <b>C Implementation</b>                                 | <b>117</b> |
| <b>D Particle Swarm Optimisation</b>                    | <b>122</b> |
| <b>E Grid search tables</b>                             | <b>123</b> |
| <b>F Figures</b>                                        | <b>130</b> |

# Chapter 1

## Introduction

In the Netherlands, each day there are over one million journeys made by train (CBS, 2026). This includes people getting to work, visiting a hospital, going to a concert or travelling to family. All these people rely on trains to reach their destination on time. To ensure that railways stay attractive to passengers, it is important that a reliable timetable is available to plan journeys.

In the Netherlands, most passenger transport is operated by NS (Nederlandse Spoorwegen). This company has the sole right to operate passenger trains on the ‘main rail network’ (in Dutch: hoofdrailnet, see Figure 1.1), which consists of the most important railway corridors in the country. In 2025, 85.5% of passengers travelling across the main rail network reached their destination at most 3 minutes after their scheduled arrival time (NS, 2026). It is known that delays influence passenger satisfaction (Monssuur et al., 2021). Therefore, the NS wants to improve the 3-minute punctuality to 86% by 2029 (NS, 2026).

An agreement has been made between NS and the Dutch government. The yearly 3-minute punctuality may not drop below 84.4% for each year until 2029 (NS, 2026). This norm must ensure that a basic level of service is guaranteed for people who depend on travelling by train. Furthermore, the Netherlands is striving for climate neutrality in 2050 (Government of the Netherlands, n.d.), while cars lead to more emissions than trains (Van Seters & Van den Berg, 2026). Therefore, it is desired to achieve a modal shift from cars to trains. The number of kilometres travelled is also expected to increase by 22-41% between 2022 and 2040 (Rijkswaterstaat, 2025). Since trains are very space-efficient compared to cars (Hensher et al., 2020), trains can play an important role in satisfying increasing demand for transport. In conclusion, trains are an important mode of transport, which attractiveness is influenced by the level of punctuality.



**Figure 1.1:** Depiction of the Dutch railway network. In red, the main railway network (hoofdrailnet) is depicted, that NS has the sole right to operate on until 2033. Passenger transport, which is not part of the hoofdrailnet is depicted with dotted blue lines. Figure has been adapted from Kompeer (2023) based on Rijksoverheid (2023).

Railways face disruptions which can lead to delays. These are malfunctions which disrupt traffic, like switch failures, signal failures and broken overhead wires. To decrease the risk of disruptions, infrastructure needs to be improved. However, there is currently a lack of funding available in the Netherlands for improvements in infrastructure (Rijksoverheid, 2026). Therefore, this thesis will not focus on disruptions, instead we will deal with disturbances. Disturbances are frequently occurring small delays. This includes boarding and alighting of passengers at a station taking longer than expected. Differences in train performance, train driver behaviour or weather conditions can also lead to fluctuations in running times (Goverde & Scheepmaker, 2023).

The ability of a timetable to deal with disturbances is called robustness. The punctuality of railways can be improved by designing a more robust timetable. This is much cheaper than creating adjustments to infrastructure. A timetable can be designed to be robust by using the following guidelines. A timetable should not assume that trains can always travel at maximum speed. It should also be taken into account that passenger boarding and alighting can

take longer than expected. Therefore, extra time needs to be incorporated in the timetable, which can be used to absorb small delays. This is known as time supplement. If a train does suffer delay, it is also important that this leads to as little hindrance for other trains as possible. Trains need to keep headway for safety purposes. If two trains are planned at minimum headway, delay of the first train immediately leads to a knock-on delay. To limit the amount of knock-on delay, extra buffer time should be added on top of the minimum headway. The robustness of a timetable can be improved by finding a better allocation of time supplement and buffer times.

Creating robust timetables has become an increasingly popular topic for research. Kroon et al. (2008) tried to find a better allocation of running time supplement within a part of the Dutch railway network by using an ILP solver. The delay was estimated based on 20 generated delay scenarios, where each delay scenario was one day. The optimisation showed promising results when the optimised timetable was tested in practice by NS on a railway corridor. The 3-minute punctuality improved from 79.4% to 85.4%. But the downside of the optimisation model used by Kroon et al. (2008) is that long computation times are needed to solve it with an ILP solver. This makes it difficult to compute timetables on large networks or with a large number of delay scenarios.

Instead of using an ILP solver to find an optimal solution, heuristics have been proposed to improve timetable robustness. However, these heuristics do have certain limitations. Some heuristics create a new timetable completely from scratch, including determining the train order. However, if a network of limited size is considered, trains will often travel outside of the network too. So alignment with the timetable on the rest of the route would be required, which makes it harder to apply the heuristic on a network of limited size. This is a problem faced by e.g. Chow and Pavlides (2018) and Bešinović et al. (2016). Additionally, it is rare that heuristics focus on allocating time supplement to improve robustness. Usually, allocation of supplement is done based on a different objective. Subsequently, the robustness of created timetables is analysed and considered when selecting a timetable, but it is not investigated whether further adjustments to the supplement allocation could improve the robustness. An exception to this is Maróti (2017), who designed a branch and bound heuristic that adjusted the timetable based on a weighted delay function. However, this heuristic could only modify times in the timetable by one full minute, which did limit the search space.

During this thesis, we will utilise metaheuristics to improve the robustness of an existing timetable. Metaheuristics are algorithms which are applicable to a large variety of optimisation problems and try to efficiently explore a search space to find near-optimal solutions. Metaheuristics are often used for solving large problems within a limited computation time (Talbi, 2009). The metaheuristics discussed in this thesis all utilise randomness and are therefore random search metaheuristics (Zabinsky, 2011). Well-known examples include simulated annealing and genetic algorithms. There do exist some studies which use metaheuristics to adjust a timetable, e.g. Liebchen et al. (2008) and Zhong et al. (2012). However, these studies did not consider robustness during optimisation. There has been little research which did utilise metaheuristics to create a more robust timetable. Since metaheuristics are often useful for solving large problems, they could provide an interesting alternative to using an ILP Solver. Therefore, during this thesis, we will test the performance of multiple random search metaheuristics to allocate time supplements.

In this thesis, a timetable is improved on a network. We want to be able to choose a network of arbitrary size. Therefore, we fix the scheduled network entrance and exit time to make sure that the trajectories of trains outside of the network can be maintained. We reallocate the running and dwell time supplement from the original timetable with the help of metaheuristics. Reallocating time supplement does also affect the buffer times present between trains. A graph based algorithm will be used to evaluate timetables. Firstly, we check if the

entire timetable is attainable if trains run as expected. If this is the case, we check the delay which arises based on a set of delay scenarios. These delay scenarios consist of sampled entrance delays and dwell times. A delay-based objective function is used to quantify the robustness, consisting of weighted average delay and a bonus term punishing delayed network exit.

The thesis is structured as follows. Chapter 2 contains background knowledge and literature about creating robust timetables, which is used to formulate a research goal. Then, in Chapter 3, we determine how a timetable may be modified and how the robustness of timetables is evaluated. Subsequently, we design delay scenarios consisting of sampled entrance delays and dwell times in Chapter 4. Chapter 5 describes multiple random search metaheuristics. In Chapter 7, we will test design options for four metaheuristics. We also compare the performance of these metaheuristics. This is done on a railway corridor between Utrecht and Arnhem, which we introduce in Chapter 6. The best found timetable will be analysed in Chapter 8. In Chapter 9, we test some modifications to the used approach. Most importantly, fluctuations in running times are considered. Finally, in Chapter 10, we look back on the research and do suggestions for further research.

# Chapter 2

## Literature

We will start this chapter with important background knowledge about timetabling in Section 2.1. Then, in Section 2.2, we will discuss literature about creating robust timetables. Finally, in Section 2.3, the research goal of this thesis is formulated.

### 2.1 Timetable preliminaries

We will start this section with information on the different design phases that a timetable needs to undergo.

#### 2.1.1 Creating timetables

There exist multiple levels of railway transport planning that need to be carried out. First, there is the strategic level, which discusses adjustments to infrastructure and creating the line plan. Adjustments to infrastructure can include investments in new railway lines, adding extra platforms at stations or modifying the signalling system. Based on the available infrastructure, a line plan is created. This describes all train routes in the network, the served stops and their desired frequencies. Based on the line plan, a timetable on the tactical level can be made. Here the departure and arrival times of every train are determined, along with the detailed capacity allocation of routes and platforms. On a tactical level a new design is made every year. Then on an operational level, this timetable is modified to make temporary adjustments. This can include restrictions due to maintenance being executed or special events requiring the deployment of additional trains. Once the operational timetable has been established, the assignment of rolling stock and crew to the trains can be planned (Duarte et al., 2025). Once a timetable is being carried out, delays can cause planned departure or arrival times to become unattainable. Then a train dispatcher needs to make changes to the timetable. This can be done by retiming, reordering or rerouting trains. This is known as traffic management or real-time rescheduling (Cacchiani et al., 2014).

This thesis focuses on improving timetables on a tactical level. We will focus on planning the departure and arrival times of trains, while not making adjustments to routing or platform allocation. A lot of factors need to be taken into account when creating a timetable. To ensure a safe journey, there must be sufficient distance between all trains. It is also important that trains travel fast between stations, to keep journey times attractive for passengers. However, it is also necessary that there is room for trains to catch up delay if they face a disturbance (Goverde et al., 2016). This will be the main focus of this thesis.

In the Netherlands, it is common to plan hourly cyclic timetables. The idea is that the same timetable is replicated for multiple consecutive hours, which provides a regular train service. Although it should be noted that additional international, freight and peak-only trains do lead to a timetable that is not completely replicated each hour. In hourly cyclic

timetables, the train service on a line is repeated after an hour. It is desirable that the service is also regular within the hour. For a train line with four trains an hour, it is favourable for passengers if the trains are exactly 15 minutes apart. This limits the maximum time that passengers need to wait for their train. Also different train lines that share part of their route can be aligned to improve the regularity of service and limit the waiting times of passengers. Regularity is an extra aspect that can be considered when creating a timetable. Additionally, it can be important that there is a fast transfer connection between certain train lines.

### 2.1.2 Signalling system

As mentioned in the previous section, trains need to keep a safe distance to each other. In this section, we will explain how signals are used to ensure this.

The signalling system in the Netherlands is based on three-aspect fixed block signalling (see Figure 2.1). In this system, there are signals next to the track, which indicate the allowed driver behaviour until the next signal. The track section in between two signals is often called a block section. At most one train is allowed to be in a block section at the same time. A signal can show three colours: red, yellow and green. If a signal is red, then a train must wait in front of the signal. This means that the block section ahead is still occupied by another train or not available. If the block section ahead of the signal is available, but the one thereafter is not, a yellow signal is shown instead. This instructs train drivers to slow down to approach the next signal with a maximum speed of 40 km/h. This is sufficient for a train to come to a standstill in time if the next signal is red. Lastly, a green signal is shown when the next two block sections are free. A train is then allowed to travel the next block section at their scheduled speed (Corman et al., 2009).

In the Netherlands, there are also short blocks close to stations. These block sections are shorter than the maximum breaking distance of a train. Therefore, the deceleration of a train needs to be partitioned across multiple block sections. At short blocks, a yellow signal is paired with a target speed. Instead of approaching the next signal with 40 km/h, a different target speed can be given such that the train can come to a standstill in time. This is called progressive speed signalling. It limits the distance that trains need to keep to each other, while still guaranteeing that two trains do not occupy a block section at the same time (Goverde et al., 2013).

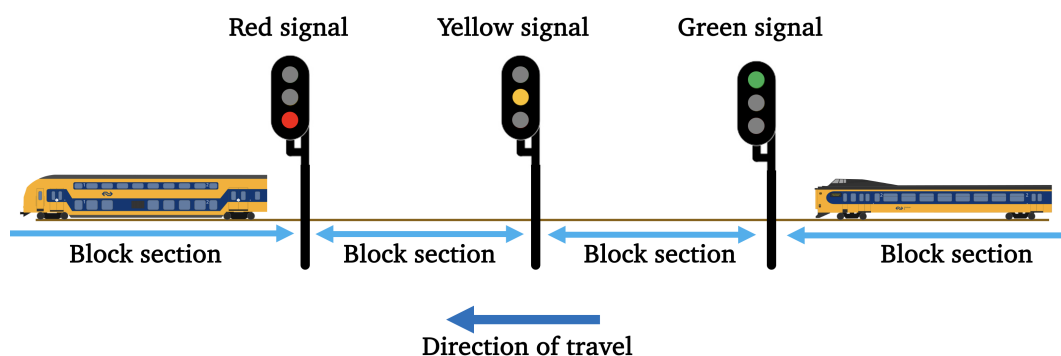


**Figure 2.1:** Illustration of a three-aspect fixed block signalling system. For short blocks, it is possible that multiple yellow signals are shown, together with target speeds. Figure was inspired by Railway Technical (2023) and uses illustrations from Wikimedia Commons (2017a, 2017b).

The majority of the Dutch railway network uses three-aspect fixed block signalling with progressive speed signalling. Besides the signalling system, NS introduced a driver advisory system in 2021. This system is called TimTim Pro and allows train drivers to check the location of other trains up to 16 kilometres ahead (Berg, [n.d.](#)). This does allow train drivers to better anticipate upcoming signals. A train driver can slow down a train to reduce the risk of encountering a yellow signal.

### 2.1.3 Macroscopic and microscopic level

Planning a timetable can be done on multiple levels of precision. Often, a distinction is made between microscopic and macroscopic level. On a microscopic level, a railway network is divided into block sections, for which a minimum running time has been determined based on the speed limit, gradient and curvature. The block sections are used to determine headways based on the signalling logic (Goverde et al., [2016](#)).

Timetables can also be designed on a macroscopic level. Then a set of macroscopic timetable points needs to be designed, which are locations where trains need to be scheduled. This often includes important stations where a lot of passengers transfer and junctions which are important for determining the order of the trains (Bešinović et al., [2016](#)). Furthermore, minimum process times and required headways need to be determined between macroscopic timetable points. These are often obtained by aggregating block sections. Based on the process and headway times, a macroscopic timetable can be created. This timetable is not always checked to be working on a microscopic level. This can result in conflicts in block usage, meaning that two trains are planned in the same block section at the same time. If times in the timetable are rounded, there can also be unattainable process times (Goverde et al., [2016](#)). So when a timetable is planned on a macroscopic level, one needs to be careful that also a feasible microscopic timetable exists.

### 2.1.4 Robustness and resilience

In this thesis, we want to create a robust timetable. We will call a timetable robust if it can recover from small delays and this leads to little hindrance for other trains. This definition is similar to the one given by Andersson et al. ([2013](#)). However, in literature, other definitions are also used to define robustness. Bešinović et al. ([2016](#)) define robustness as the ability to withstand design errors, deviating parameters and changes in operational conditions. Schöbel and Kratz ([2009](#)) also take into account whether connections are maintained if there is some delay. Dewilde et al. ([2014](#)) argue that a timetable is robust if it minimises the total (weighted) real travel time of the passengers, where also small frequently occurring disturbances are considered.

For robustness we only consider small delays. These small delays, that happen due to every-day variations, are often referred to as disturbances. Disturbances can be caused by many different reasons. Longer dwell times can be caused by variations in passenger boarding and alighting. Variations in running times can occur due to weather conditions, such as strong headwind. Additionally, the performance of a train or behaviour of a driver can influence running times (Goverde & Scheepmaker, [2023](#)).

Beside disturbances, trains can also suffer disruptions. Disruptions are larger problems which require a big adjustment to the timetable. This can be infrastructure failures or extreme weather. Often disruptions cause part of the network to be impassable, so cancellation or rerouting of trains is required. The ability to recover from disruptions can be described as the resilience of a timetable (Bešinović, [2020](#)). We do not consider the resilience of a timetable in this thesis. Disruptions often require large adjustments to a timetable, which need to be determined by real-time rescheduling. Therefore, improving the resilience of a timetable by only modifying departure and arrival times is difficult.

### 2.1.5 Dealing with disturbances

There are two important ways to limit the impact of disturbances. Firstly, running time supplement can be allocated in the timetable. This means that more time than strictly necessary is planned to complete the journey between two stations. If disturbances cause longer running times between stations, the supplement can prevent a delayed arrival at the next station or at least reduce the amount of delay. Running time supplement can also be used to make up delay if a train departs later. For trains travelling on time, running time supplement can also be used to reduce energy consumption (Goverde & Scheepmaker, 2023). To cover variations in boarding and alighting of passengers, it is also possible to plan longer dwell times at stations. We will often refer to this as dwell time supplement.

If a train is delayed due to a disturbance, this can also hinder other trains. This is called knock-on delay. If minimum headway is planned between consecutive trains, delay of the first train always leads to knock-on delay. Therefore, this is often avoided. Instead, extra buffer time is added on top of the minimum headway. More buffer time reduces the risk that delay hinders the next train (Kroon et al., 2008). To design a robust timetable, time supplements and buffer times need to be planned to limit the impact of disturbances. Additionally, knock-on delays can sometimes be reduced by adjusting the order of trains. This is especially useful if a frequently heavily delayed train is planned right in front of another train.

## 2.2 Robust timetabling

We will start this section by discussing literature which creates robust timetables, without changing the order of trains to improve robustness. Hereafter, we consider studies that do use flexibility in train order. Finally, we will summarise the discussed literature and determine a research gap.

### 2.2.1 Optimising buffer and time supplement

In Kroon et al. (2008), the robustness of an existing cyclic timetable was improved. A set of delay scenarios was generated, consisting of entrance delays, dwell time disturbances and running time disturbances. The goal was to allocate a limited amount of running time supplement, such that the weighted average delay was minimised over the generated delay scenarios. The order of the trains was kept identical to the original timetable, while the running time supplements were optimised by solving an MILP with CPLEX.

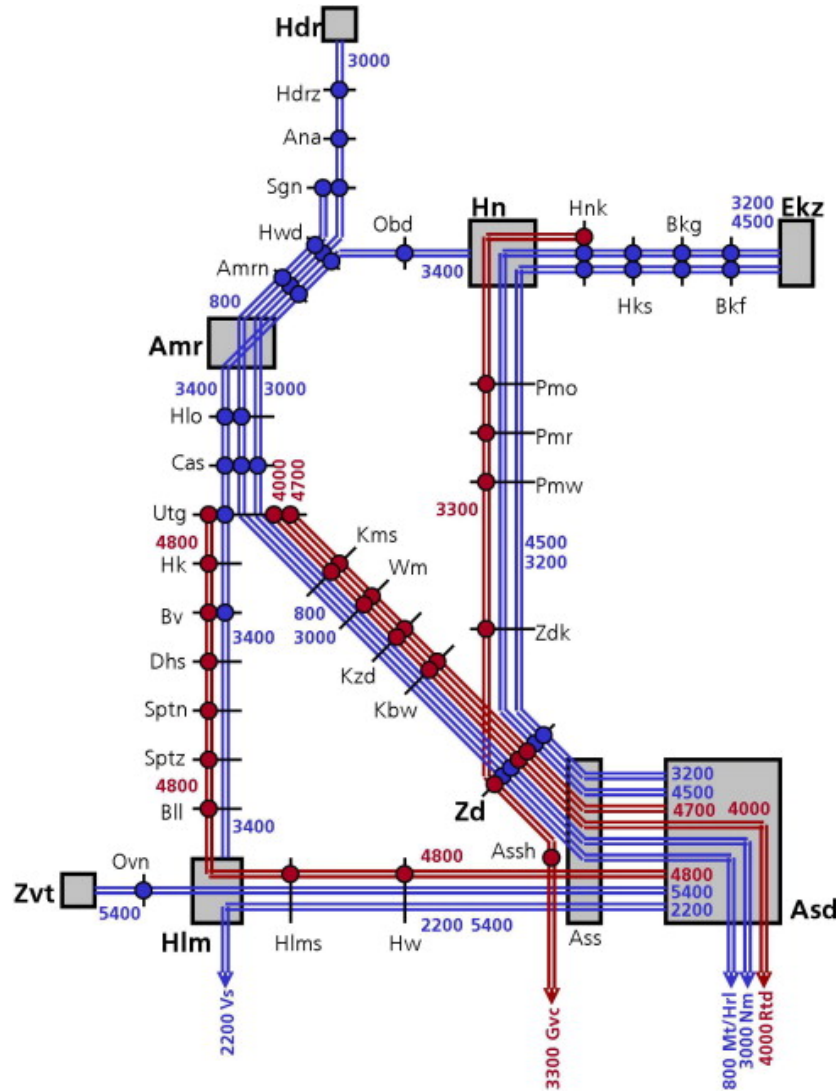


**Figure 2.2:** Railway network in the ‘Kop van Noord-Holland’. Here, Asd is Amsterdam Centraal, Hlm is Haarlem, Zd is Zaandam, Hn is Hoorn and Ekz is Enkhuizen. Figure is from Kroon et al. (2008).

Optimising based on a sample of delay scenarios is called stochastic programming with a sample average approximation approach. This approach was tested in Kroon et al. (2008) on part of the Dutch railway network, which is called the ‘Kop van Noord-Holland’ (see Figure 2.2). Twenty delay scenarios were created of the cyclic timetable being executed for 21 consecutive hours. The optimisation took 33 hours until an optimality gap of 1.5% was reached. The average delay decreased from 1.22 minutes in the original to 0.92 minutes in the optimised timetable.

Based on the obtained results, the authors convinced NS to test an adjustment to the real timetable. This was done on the ‘Zaanlijn’, which is the railway corridor between Amsterdam and Den Helder (which is part of the ‘Kop van Noord-Holland’). The total amount of running time supplement was kept equal to the original timetable, while the dwell times were not adjusted at all. The adapted timetable was tested during an eight-week period in 2006 and showed an average punctuality of 85.4% of trains being less than three minutes delayed. The original timetable on the other hand, had an average punctuality of 79.4% during the previous 13 weeks it was operated. It was therefore concluded that the adapted timetable was more robust. However, for disruptions the timetable performance did not improve, this would require adjustments to the train order.

So Kroon et al. (2008) showed promising results that reallocation of running time supplement can lead to a more robust timetable. However, optimisation took 33 hours on a relatively small part of the Dutch railway network and considered only twenty days during optimisation. Naturally, the computational power of computers have improved over the past 20 years, but it is questionable if this could lead to much faster calculations.

Verifying if a feasible cyclic timetable exists is NP-complete, when the order of trains may be adjusted. This can be shown by making a reduction from graph-colouring or the Hamiltonian circuit problem (Odijk, 1996). In Kroon et al. (2008), the order of trains has already been fixed, which does simplify the problem. However, instead of finding a feasible solution, robustness needs to be optimised. Kroon et al. (2008) uses an MILP for this. For every scheduled event (departures and arrivals), a schedule time needs to be determined. Based on this schedule and realised scenarios, all realised event times can be calculated. These realised event times are implied by the choice of the schedule times. However, to incorporate realised event times in a linear programming model, they need to be added as variables. Furthermore, the scheduled times in Kroon et al. (2008) are rounded to full minutes, while the realised event times are not. So a large scale linear programming model including integrality constraints needs to be optimised, which leads to long computation times. More details on this can be found in Appendix A.

To achieve faster timetable improvement, in Maróti (2017) a branch and bound heuristic was developed. This paper is a follow-up to Kroon et al. (2008) and mostly uses the same constraints. One extra limitation is imposed: every event can be shifted at most one minute with respect to the original timetable. Just like in Kroon et al. (2008), every event time needs to be in whole minutes. Therefore, for every event there are three options:  $-1$  minute, no change or  $+1$  minute. This does limit the search space, which can prevent finding better solutions. But the heuristic can be performed much faster than solving the MILP with CPLEX. The branch and bound heuristic finds its final solutions within 5 minutes, even when considering the entire Dutch railway network. The results do sometimes have an optimality gap of up to 30%. However, instead of a better solution existing, the optimality gap could also be caused by weak lower bounds. On the other hand, changes to event times can be at most one minute. So this does limit the search space, which can prevent finding better solutions.

In Fischetti et al. (2009), a two-stage optimisation is used to find a robust timetable. In the first stage of optimisation, a profit function is optimised that does not involve robustness. This stage determines the train order and the maximum achievable profit  $z^*$ . In the second stage of optimisation, the robustness is optimised. In this stage, only event times may be modified. The robustness is optimised among solutions with a minimum profit of  $\alpha \cdot z^*$ , for a selected parameter  $\alpha \in [0, 1]$ . Three different approaches are tested in the second stage to improve the robustness. Firstly, a stochastic model is used where the true delay at every event is measured (just like Kroon et al., 2008), which is called the fat stochastic model. Secondly, a simplified delay calculation is tested which is called a slim stochastic model. This only measures the amount of new delay that arises on each track section. This reduces the number of decision variables, but does not capture the development of the delay. It is not measured what the effect of the delay is on the arrival time at later stations. To compensate for this, delays early in the network are weighted more heavily, since they are expected to cause a delayed arrival at more stations

The third concept applied in Fischetti et al. (2009) is light robustness, which was developed by Fischetti and Monaci (2009). Here, robustness is not directly optimised. Stricter constraints are imposed instead. The minimum process time between two events of the same train or the minimum headway time is increased. However, it is usually not possible to find a solution which satisfies all stricter constraints. Therefore, slack variables are added to weaken the constraints again. An objective function based on the total amount of slack is then minimised. Because no delay propagation has to be calculated during optimisation,

the method is a lot faster than stochastic programming. But the quality of the solutions is dependent on how the constraints are made stricter. With the approach from Fischetti et al. (2009), the results of light robustness are only slightly worse than the slim stochastic model. But the approach they use to get stricter constraints uses information derived from the solution of the stochastic programming approach (Caimi et al., 2017). The slim stochastic model does come close to the results of the fat stochastic model in terms of delay within the network. However, during optimisation, delays are not tracked. This does restrict the objective function that can be used during optimisation. For example, we do not know the delay for trains at network exit. Therefore, it is hard to add extra punishment for delays at network exit to account for hindrance caused outside of the considered network.

In Liebchen et al. (2010), a two-fold objective function is optimised with the goal of minimising realised transfer times. The first part of the objective function describes the transfer times if there is no delay. The second part is an approximation of the extra wait time caused by missing transfers due to delay. It is assumed that there is a strict no-wait policy, so trains do not wait to ensure a transfer being attainable. The delay caused by missed transfers is approximated with variables similar to the slack approach used in light robustness. After a timetable has been generated, the robustness is evaluated by testing traffic management based on a set of delay scenarios. In the article, three different probability distributions are used to approximate the wait time due to missed transfers. Also three different weight combinations are tested to balance the two objective function components. By evaluating the robustness of all solutions, a trade-off could be chosen.

In Jovanović et al. (2017), buffer times are allocated to improve the robustness compared to an existing timetable. First, the existing timetable is compressed, which means that planned buffer is reduced until the time window required to complete all events is minimised. The goal is then to increase the robustness by adding buffer to the headways in the right places, while ensuring that the timetable is cyclic. The amount of buffer that is desirable to add between trains is determined. This is done based on the priority of both trains, the running time supplement that has been planned for both trains and the position in the network. But if all desirable buffer time is added, then the timetable would not be executable within the cycle time. This means that a planned train cannot be executed at the same moment during each cycle. Therefore, a decision needs to be made on which buffer times are chosen without exceeding the cycle time. Mathematically, this is known as a multidimensional knapsack problem. The downside of the approach is that longer dwelling causes trains to be in the network for longer. Therefore, the network entry or exit time needs to be modified. So this also requires a different timetable outside of the considered network.

When generating a timetable, usually one speed profile is assigned to every train. In event flexibility, event slots are used instead. These event slots specify a lower and upper bound, between which the realised event time can take place without causing problems. The goal is then to find a combination of event slots such that no conflicts occur if each train stays within the event slots. However, the concept of event slots does not scale well to a larger network (Caimi et al., 2017).

Recoverable robustness uses a set of likely delay scenarios that need to be taken into account. The goal is to find a timetable, such that for each delay scenario, the amount of traffic management stays below a set bound. The amount of traffic management is calculated based on the total delay on all events, maximum delay on an event and number of connections cancelled (Liebchen et al., 2009). Recoverable robustness assures that a high traffic management cost is unlikely, but does not minimise the average cost.

Both event flexibility and recoverable robustness do not use randomly generated delays to test the performance, so the quality of solutions is highly dependent on model choices. For recoverable robustness these are the chosen delay scenarios and for event flexibility the desired slot widths. This could be the reason that these methods are still known as less suitable for use in practice (Caimi et al., 2017).

### 2.2.2 Improving the train order

In the previous subsection, robustness was only adjusted by modifying time supplements and buffer times. In this subsection, we will discuss research, which also makes adjustments to the train order.

In Bešinović et al. (2016), an iterative micro-macro approach is used. Timetables are created on a macroscopic level. This is done based on desired transfers, required headway times, minimum and maximum running times and minimum and maximum dwell times. Timetables are created using a heuristic, based on a priority sequence of train lines. This sequence describes the order that is used to add train lines into the timetable, so not the order in which train lines will travel through the network. Sequentially, train lines are added into the timetable using a dynamic programming algorithm. This algorithm optimises an objective function based on minimising running, dwell and transfer times without altering the already scheduled trains. If there is no room left to add the train line, then a penalty term is added to the total objective function. After all trains have been considered, the robustness of the timetable is checked on some delay scenarios. The robustness is also incorporated in the objective function. For 1000 different priority sequences a timetable is created, of which the best one is selected.

The selected timetable is evaluated on a microscopic level. It is checked whether there are trains that have conflicting block usage. Furthermore, it is checked whether any capacity norms are violated. If one of these problems arises, parameters are adjusted. Thereafter, a new timetable is created on the macroscopic level based on the adjusted parameters. This iterative procedure continues until a macroscopic timetable has been created which meets all requirements on a microscopic level. So the procedure does not stop until a timetable is found, which is feasible on a microscopic level. Hereafter, it is possible to perform a post-processing step for corridor fine-tuning, as was added in Goverde et al. (2016). Departure and arrival times can be adjusted for smaller stations, which were not macroscopic timetable points. Adjustments are made to find speed profiles which improve robustness or energy-efficiency.

The micro-macro approach does have some disadvantages. The used heuristic is not likely to find an optimal solution. It is also not guaranteed to schedule all train lines. Additionally, it assumes that there is full flexibility to schedule the train lines. This is not always realistic due to trains continuing their journey beyond the considered network. So the method would need to be applied to a large network, which is not left by trains. Otherwise, alignment with other networks still needs to be added into the approach.

In the next papers, that we will discuss, the train order is determined using genetic algorithms. Genetics algorithms are an example of a random search metaheuristic. Metaheuristics are algorithms which are applicable to a large variety of optimisation problems and which try to efficiently explore a search space to find near-optimal solutions. This is different than heuristics, which have been tailored to a specific optimisation problem (Talbi, 2009). A genetic algorithm has a pool of solutions. By making combinations of these solutions and modifications to these solutions, new solutions are created. At the end, the solution with the best objective function is returned. Since randomness is used for creating solutions, genetic algorithms are random search metaheuristics (Zabinsky, 2011).

In Chow and Pavlides (2018), a genetic algorithm is used to determine the order of trains departing from their origin station. One by one, trains are then added into the schedule to minimise travel times with the help of Dijkstra's algorithm. The created timetables are evaluated based on four criteria: running time of trains, waiting time of passengers, crowdedness of trains and punctuality of service. There do exist more studies that utilise genetic algorithms to determine the train order, Dündar and Şahin (2013), Tan et al. (2020), and Tormos et al. (2008), but these studies do not focus on robustness. The disadvantage of Chow and Pavlides (2018) is that once a train order is determined, a simple heuristic is used to create the timetable. It would be interesting to see if a genetic algorithm could also be applied to update time supplements in the timetable.

There are also articles which have included variations in order in their optimisation model. This is done on a microscopic level in Burggraave and Vansteenwegen (2017), but they only consider a short corridor. In Högdahl and Bohlin (2023), a timetable is optimised for an entire railway corridor. They assume that an existing timetable is available. Trains can be shifted a maximum amount of time compared to the original timetable, several values up to 30 minutes are tested for this. The timetable is optimised based on a weighted cost function of travel times and expected delay. The expected delay is not estimated by using a sample average approximation approach within the optimisation model. Instead, delays were simulated on the original timetable. The expected delay of the adjusted timetable was then approximated by estimating how changes in supplement would impact the delay. The delay propagation is done macroscopically. In the article, it was tested how much the existing timetable could be improved both with and without fixing the order of the trains. Additionally, the impact of flexibility in the entry time was investigated. A flexible train order did give better results when the entry time was also flexible and a large shift compared to the original timetable was allowed. Otherwise, there were little possibilities to change the train order and the results were not improved.

It was tested in Högdahl and Bohlin (2023) whether the approach was scalable to a larger timetable. The original timetable that needed to be modified consisted of 87 trains that travelled during seven hours on one railway corridor. If a large planning window was used, then every event could be shifted at most 30 minutes. It was tested what would happen if the original timetable was repeated eight times and this had to be optimised. If flexibility in train order and entry time was allowed, then only in one out of five tries the optimisation succeeded within 24 hours. Smaller planning windows, no flexibility in order or no flexibility in entry time led to faster calculations. When designing timetables on a tactical level, it is not problematic if calculations take a few hours. Furthermore, using a cyclic timetable would limit the number of trains that need to be scheduled. But it is not known if the approach could be extended to a larger network and whether the delay estimation is realistic for the Dutch railway network.

### 2.2.3 Literature gap

We will now summarise what has been studied in existing literature. We will also discuss what would be interesting to investigate based on the studied research.

At the start of this section we have seen that it is possible to use an MILP combined with a sample average approximation approach to improve the robustness of a timetable, as has been done by Kroon et al. (2008). However, this does lead to long computation times, even on a network consisting of a small part of the Netherlands and using a limited number of delay scenarios. Therefore, different kinds of alternative approaches have been investigated. One approach is to identify places where adding extra running time supplement or buffer is likely to improve the robustness. This was done by Jovanović et al. (2017) with a knapsack approach, by Fischetti et al. (2009) with a light robustness approach and in Caimi et al. (2017) with event flexibility. It is also possible to design a timetable which is easily

adjustable to a few likely delay scenarios, which was the recoverable robustness approach used by Liebchen et al. (2009).

Alternatively, it is possible to use a simple approximation of the expected delay. This was done in Liebchen et al. (2010) based on the probability of missing a connection and in Högdahl and Bohlin (2023) based on the delay of the original timetable. Lastly, it is possible to avoid using a solver (like CPLEX) for optimising a mathematical programming model. Instead, a (meta)heuristic can be used to find a robust timetable. Maróti (2017) uses a branch and bound heuristic to test small changes to event times in an existing timetable to improve the robustness. Bešinović et al. (2016) create a timetable from scratch by determining a priority sequence and then using a dynamic programming algorithm to plan train lines. Chow and Pavlides (2018) use a genetic algorithm to create the order in which trains travel through the network, and then a heuristic based on Dijkstra's algorithm is used to create a timetable.

Out of the discussed approaches, we are most interested in designing a (meta)heuristic. Designing a good list of desired buffer or supplement would require extensive prior knowledge about delay propagation and we fear this is insufficiently available to create a good timetable. The same problem holds for the concept of recoverable robustness, which requires designing multiple likely delay scenarios. Also the lack of attention to the average delay is a disadvantage of recoverable robustness, since it is only tested if a maximum traffic management cost is sufficient. If the alternative approach of using a simplified delay cost function would be utilised, one needs to be careful that an accurate estimation of the delay is chosen. The delay approximation approach from Högdahl and Bohlin (2023) shows promising results, so this approach could be tested on the Dutch railway network. However, this approach does still need large computation times to create a timetable. Therefore, we prefer to design a (meta)heuristic, which is more promising to obtain timetables quickly.

The (meta)heuristics that we have discussed do have certain limitations. Some heuristics create a new timetable completely from scratch, including determining the train order. This makes the (meta)heuristic not suitable for improving an existing timetable on a smaller network. To ensure that trains can continue their journey outside of a network, further alignment would be required. This is a problem faced by e.g. Chow and Pavlides (2018) and Bešinović et al. (2016).

Additionally, it is rare that heuristics focus on allocating time supplement to improve robustness. Usually, allocation of supplement is done based on a different objective. Subsequently, the robustness of created timetables is analysed and considered when selecting a timetable. An exception is Maróti (2017), where a branch and bound heuristic is used to adjust time supplement such that the amount of delay decreases. However, this heuristic could only modify times in the timetable by one full minute, which did limit the search space. Therefore, there is a gap in the literature for more research on allocating time supplement by making use of a (meta)heuristic. Especially using random search metaheuristics could be interesting, since they have seen limited use in existing literature. Despite metaheuristics being known for using limited computation time and being suitable for optimising large problems (Talbi, 2009).

## 2.3 Research goal

Based on the literature that we have discussed, we can formulate the research goal of this thesis. We will test the performance of multiple random search metaheuristics for allocating time supplements. The time supplements of an original timetable will be reallocated to improve the robustness. No modifications will be required to the timetable outside of the considered network.

The timetable that is created should satisfy the following requirements.

- The timetable is feasible if there is no delay. Every planned departure and arrival time is attainable when minimum running and dwell times are used. The timetable is executable as a one-hour cyclic timetable and there are no conflicts between trains.
- The time of the first and last event of each train is identical to the original timetable, such that the original train schedule outside of the considered network can be maintained. This prevents problems being moved outside of the considered network.
- The timetable performs well under a set of delay scenarios, which for each train consists of an entrance delay and at each station a sampled dwell time.

The robustness of timetables created by metaheuristics needs to be evaluated. We will use directed graphs for this. Directed graphs have previously been used to evaluate delay propagation in e.g. Goverde (2007, 2010) and Kecman et al. (2013). These directed graphs are sometimes called timed event graphs.

Delay propagation can be evaluated on either a microscopic or a macroscopic level. To evaluate the effect of signals on trains, it is required to use a microscopic model. If a train encounters a yellow or red signal, this impacts the travel times of a trains. Therefore, microscopic models are more realistic. However, since the introduction of a driver advisory system at NS, train drivers can check the location of other trains up to 16 kilometres ahead (Berg, n.d.). This allows train drivers to anticipate and prevent encountering red or yellow signals. Therefore, we will work on a macroscopic level. Sufficient headway is only checked at stations and junctions, while in between it is assumed that sufficient headway can be kept with the help of a driver advisory system. Using a macroscopic level has the additional advantage that calculating the delay propagation can be done faster, since a smaller graph has to be traversed.

In a real-time rescheduling context, directed graphs are also used to determine the train order which minimises delay. These graphs are known as alternative graphs, more information on them can be found in Appendix B. However, Högdahl and Bohlin (2023) have found that changes in order are little beneficial if trains travel along the same corridor and the entrance times are fixed. Therefore, adjusting the train order will not be a priority during this thesis. But based on the delay propagation, we will check if there are any obvious adjustments to the order of trains. This can also be adjustments under certain delay scenarios only.

## Chapter 3

# Framework

In this chapter, we will introduce the framework of how timetables are created and evaluated. Firstly, in Section 3.1, we give an overview of the notation that will be used throughout the chapter. Then, in Section 3.2, we create a directed graph. This graph is used to check feasibility of a created timetable in Section 3.3 and calculate delay propagation in Section 3.4. Thereafter, the objective function is introduced in Section 3.5. In Section 3.6, an algorithm is introduced, which combines the feasibility check, delay propagation and calculating the objective function. Finally, in Section 3.7, it is discussed how the timetable is expected to perform on other areas than robustness.

### 3.1 Overview of notation

In this section, we give an overview of all notation that is used throughout the rest of this chapter. All notation will still be introduced throughout the chapter in more detail.

|                            |                                                                                              |
|----------------------------|----------------------------------------------------------------------------------------------|
| $q_j$                      | Activity                                                                                     |
| $z_{i,k}$                  | Train, which is the $k$ -th operation of train service $i$                                   |
| $Q_i$                      | Sequence of $q_j$ that need to be completed by train service $i$                             |
| $P_i$                      | Subsequence of $Q_i$ , contains only $q_j$ that need to be planned                           |
| $e_{i,j,k}$                | Both event vertex and event time for $z_{i,k}$ completing $q_j$                              |
| $x_{i,j,k}$                | Both schedule vertex and schedule time for $z_{i,k}$ completing $q_j$                        |
| $v_0$                      | Source vertex                                                                                |
| $V_e$                      | Set with all event vertices $e_{i,j,k}$                                                      |
| $V_x$                      | Set with all schedule vertices $x_{i,j,k}$                                                   |
| $V$                        | Set of all vertices, $V = V_x \cup V_e \cup \{v_0\}$                                         |
| $(u, v)$                   | Arc connecting $u \in V$ to $v \in V$                                                        |
| $t(u, v)$                  | Length of the arc $(u, v)$ , gives the constraint $v \geq u + t(u, v)$                       |
| $A$                        | Set of all arcs $(u, v)$                                                                     |
| $A_{in}(v)$                | Incoming arcs for $v \in V$ , $A_{in}(v) = \{(u, v) \in A   u \in V\}$                       |
| $G$                        | Directed weighted graph, formed by the triple $(V, A, t)$ .                                  |
| $o_{i,j}$                  | Minimum length of timetabling arc incoming to $x_{i,j,1}$                                    |
| $\varepsilon_{i,j}$        | Supplement added to $o_{i,j}$                                                                |
| $J_i$                      | Set of indices $j$ for train service $i$ , such that $o_{i,j}$ and $\varepsilon_{i,j}$ exist |
| $x_{i,0}$                  | Network entrance time for train $z_{i,1}$                                                    |
| $x_i^*$                    | Network exit time for train $z_{i,1}$                                                        |
| $s_i$                      | Total supplement available for train service $i$                                             |
| $\text{ord}(V, A)$         | Topological ordering of vertices $V$ on a graph with arcs $A$                                |
| $R$                        | Number of delay scenarios                                                                    |
| $\delta^r$                 | Delay scenario                                                                               |
| $\Delta_R$                 | Set of delay scenarios $\delta^1, \delta^2, \dots, \delta^R$                                 |
| $t^r(u, v)$                | Arc length of $(u, v)$ under delay scenario $\delta^r$                                       |
| $e_{i,j,k}^r$              | Realised event time of event $e_{i,j,k}$ under delay scenario $\delta^r$                     |
| $\text{plan}(e_{i,j,k})$   | Whether $e_{i,j,k}$ needs to be planned, so whether $x_{i,j,k}$ exists                       |
| $U$                        | Set of indices $(i, j, k)$ , such that a schedule time $x_{i,j,k}$ exists                    |
| $\mathcal{L}(x, \Delta_R)$ | Objective function for timetable $x$ and delay scenarios $\Delta_R$                          |
| $\ell(x, \delta^r)$        | Part of objective contributed by delay scenario $\delta^r$                                   |
| $h(u, v)$                  | Function for bonus term in objective function                                                |
| $b_{i,j,k}$                | Whether bonus term needs to be added for event $e_{i,j,k}$                                   |

### 3.2 Directed graph

In this section, we will build a directed graph. We will use this graph to set a timetable, consisting of departure and arrival times for each train. Subsequently, the graph is used to check if all departure and arrival times are attainable and to calculate delay propagation. Before we introduce the vertices of the graph, we will first introduce the notion of an activity.

An activity combines a location with an activity type. This can be passing a junction, entering a station or exiting a station. We will denote an activity as  $q_j$ , where the index  $j$  can be used to identify a certain activity. It can be convenient to further subdivide a location. This is especially useful for quadruple track railways, where there are multiple separated tracks for trains travelling in the same direction. By creating separate activities for each of the tracks, it becomes easier to see which trains follow behind each other. This is not required for creating the directed graph, but can make it easier to determine which vertices need to be connected. Examples of activities include: passing Lunetten junction, exiting Maarn station or entering platform 19 of Utrecht Centraal. So activities do combine a location with an activity type, but are train independent.

**Remark 3.2.1.** *Station entrance is the moment when the train reaches the platform and station exit is the moment when the train leaves the platform. If a train stops at a station, the entrance may also be called an arrival and the exit a departure. If a train does not stop at a station, we call this a through-running train. A station exit and entrance is also created for trains, that run through a station. However, a through-running train completes both activities at the same time.*

We will distinguish between three types of stations: main stations, basic stations and short stop stations (which is often shortened to short stops). Short stops are stations, where it is not possible for trains to overtake, cross or reverse direction. For the remaining stations, a distinction is made between basic and main stations. Main stations are used by a large number of passengers and often offer possibilities to change between trains travelling in different directions. We will consider all trains that saw at least 40,000 passengers on an average weekday in 2025 to be main stations. This includes the fourteen most used stations in the Netherlands (see Remark 3.2.2).

**Remark 3.2.2.** *These are all fourteen main stations in the Netherlands according to the introduced criterion: Utrecht Centraal, Amsterdam Centraal, Rotterdam Centraal, Schiphol Airport, Leiden Centraal, Den Haag Centraal, Amsterdam Zuid, Amsterdam Sloterdijk, Eindhoven Centraal, 's-Hertogenbosch, Amersfoort Centraal, Arnhem Centraal, Zwolle and Nijmegen (NS, 2025).*

We will now introduce notation for trains. We will use the term ‘train service’ to describe a train that operates once an hour in the timetable. This makes a train service different from a train line. A train line may include multiple trains an hour, which have the same route and stopping pattern. If a train line runs  $m$  times an hour, then we will create  $m$  different train services for this train line. We will denote a train service with an index  $i$ . A train service leads to a train being operated once an hour. Therefore, one train service is associated with multiple trains. We use an index  $k$  to distinguish between multiple trains which are part of the same train service. The  $k$ -th train which operates as part of train service  $i$  is denoted as  $z_{i,k}$ . We will operate 16 hours for each train service, such that we can evaluate the delay propagation across an entire day. Therefore, for each train service  $i$ , we create trains  $z_{i,1}, z_{i,2}, \dots, z_{i,16}$ .

We will now explain how we will create the vertex set  $V$  of the directed graph. For each train service  $i$ , a sequence of activities  $Q_i$  is described that need to be completed by the train service. The sequence follows the same order in which the activities need to be completed. For each train  $z_{i,k}$  and activity  $q_j \in Q_i$ , we will create an event vertex  $e_{i,j,k}$ . This event vertex will be used to determine the time that  $z_{i,k}$  completes  $q_j$ . All event vertices are added to the set  $V_e$ .

The goal of this thesis is creating a timetable in which event times are planned. However, it is not necessary to determine a scheduled time for each event. For passengers, the departure and arrival times at stations are important to map out their journey. Therefore, we will only determine schedule times for arrivals and departures. In the graph, arrivals are all vertices  $e_{i,j,k}$  where  $q_j$  is the entrance of a station where train  $z_{i,k}$  stops. Similarly, a

departure is a vertex  $e_{i,j,k}$  where  $q_j$  is the exit of a station where train  $z_{i,k}$  stops. However, at short stops, trains do not have to wait for a scheduled departure time (D. Li et al., 2014). A scheduled departure time is not communicated to passengers. Instead, an early departure time is given, based on the scheduled arrival time and the technical minimal dwell time. Therefore, we will not determine a scheduled time for departures from short stops. Thus, we only determine a scheduled time for all station arrivals and for departures from basic and main stations.

All  $q_j$  where a schedule time needs to be determined for a train service  $i$  are included in the sequence  $P_i$ . The activities in this sequence are in order of completion by train service  $i$ , therefore it is a subsequence of  $Q_i$ . For all trains  $z_{i,k}$  and  $q_j \in P_i$ , the event time  $e_{i,j,k}$  needs to be scheduled. For such events, we create an extra vertex  $x_{i,j,k}$ . We will call  $x_{i,j,k}$  a schedule vertex. A vertex  $x_{i,j,k}$  is used to determine the scheduled time for event  $e_{i,j,k}$ . All schedule vertices  $x_{i,j,k}$  are added to the set  $V_x$ . The vertex set  $V$  will consist of the event vertices  $V_e$ , schedule vertices  $V_x$  and an additional source vertex  $v_0$ . To check if an event  $e_{i,j,k}$  has a schedule time, we introduce the function:

$$\text{plan}(e_{i,j,k}) = \begin{cases} \text{True} & \text{if } x_{i,j,k} \in V_x, \\ \text{False} & \text{otherwise.} \end{cases} \quad (3.1)$$

We will now introduce the arcs that are present in the directed graph. Each arc goes from a tail vertex to a head vertex. We will denote an arc from a tail vertex  $u \in V$  to a head vertex  $v \in V$  as

$$(u, v). \quad (3.2)$$

Each arc will have a length, which we denote as  $t(u, v)$ . The letter  $t$  is used, since the length will represent a time in seconds. By slight abuse of notation, a vertex  $v \in V$  is also used for the time associated with it. So an event vertex  $e_{i,j,k}$  has event time  $e_{i,j,k}$ , and similarly a schedule vertex  $x_{i,j,k}$  has schedule time  $x_{i,j,k}$ . Now we have introduced this notation, we can explain the meaning of an arc. An arc  $(u, v)$  with length  $t(u, v)$  represents the constraint

$$v \geq u + t(u, v). \quad (3.3)$$

All arcs will be added to the set  $A$ . To calculate the time for a vertex, we need to define the set of incoming arcs. The set of incoming arcs for a vertex  $v$  is defined as:

$$A_{\text{in}}(v) := \{(u, v) \in A \mid u \in V\}. \quad (3.4)$$

We can then calculate the time of a vertex  $v$  as

$$v = \max\{u + t(u, v) \mid (u, v) \in A_{\text{in}}(v)\}. \quad (3.5)$$

A directed graph will be denoted as a triple  $G = (V, A, t)$ . We will now introduce all arcs which are part of the graph. In Figure 3.1, an example of a constructed graph with six activities and three trains can be found. During the remainder of this section, all arc types are introduced which are present in the graph. A colour-blind-friendly version of this figure is available in Figure F.1 in Appendix F.

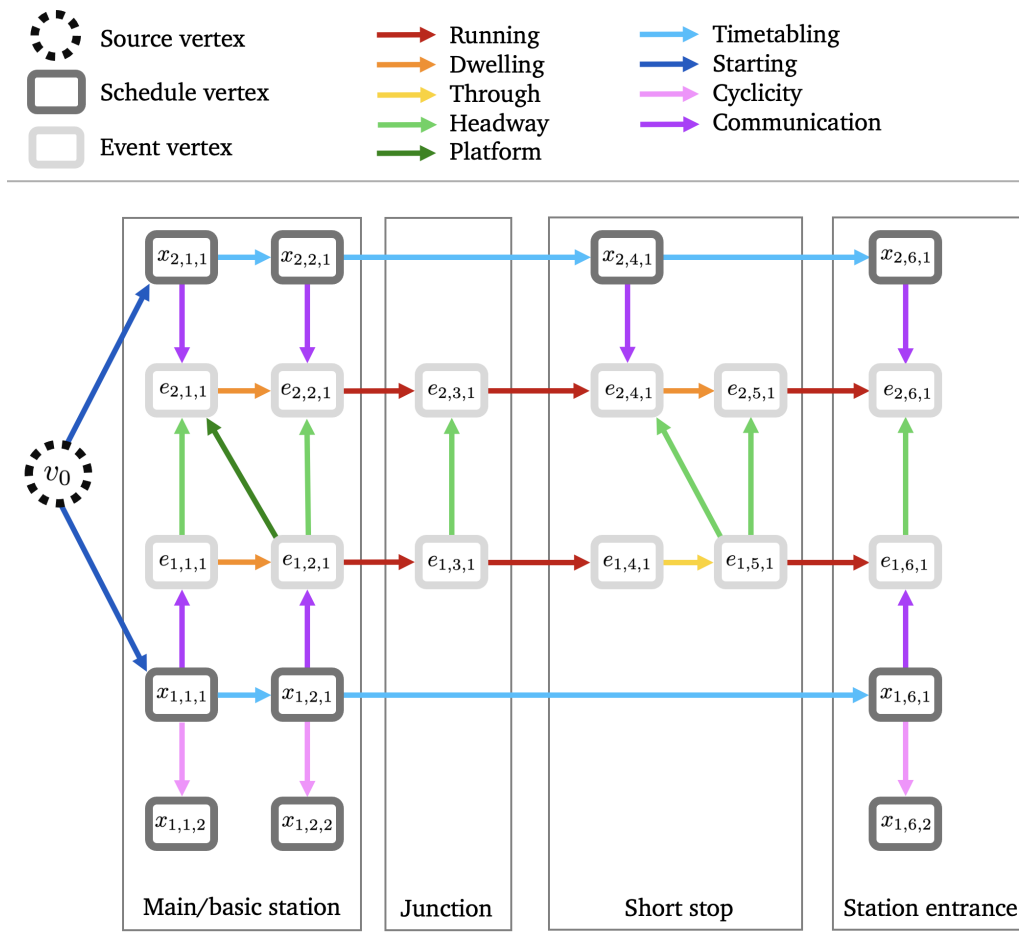


**Figure 3.1:** Example of a constructed graph. A colour-blind-friendly version of this figure is available in Figure F.1. In this graph, train  $z_{2,1}$  follows behind train  $z_{1,1}$  throughout a network. There are six activities. Trains start by entering a basic or main station. The entrance of this station is activity  $q_1$ , while the exit is activity  $q_2$ . At this station, the platform arc is only present if  $z_{1,1}$  and  $z_{2,1}$  use the same platform at this station. The trains then pass a junction, which is activity  $q_3$ . Subsequently,  $q_4$  and  $q_5$  are a short stop entrance and exit, only train  $z_{2,1}$  makes a stop here. The network ends at the station entrance  $q_6$ . To show the cyclicity arcs, also the schedule vertices for train  $z_{1,2}$  are given.

We will first introduce arcs that connect event vertices of the same train. For each train  $z_{i,k}$ , we will create such arcs based on the sequence  $Q_i$ . If an activity  $q_{j'}$  is followed in  $Q_i$  by  $q_j$ , then we create an arc  $(e_{i,j',k}, e_{i,j,k})$ . The length  $t(e_{i,j',k}, e_{i,j,k})$  describes the minimum time separation, that we want to have between the events  $e_{i,j',k}$  and  $e_{i,j,k}$ . This time separation depends on the activity  $q_j$  and the sequence  $P_i$  of planned events.

We will first consider the case that  $q_j$  is a station exit, but  $q_j, q_{j'} \notin P_i$ . Then,  $e_{i,j,k}$  represents the exit from a station where  $z_{i,k}$  runs through without stopping, while  $e_{i,j',k}$  was the entrance at this station. In such a case, we will call  $(e_{i,j',k}, e_{i,j,k})$  a through arc. For through arcs,  $t(e_{i,j',k}, e_{i,j,k}) = 0$ , since no dwell time is required between station entrance and exit.

If  $q_j$  is a station exit and  $q_{j'} \in P_i$ , then  $(e_{i,j',k}, e_{i,j,k})$  is the time spent at a station where  $z_{i,k}$  does stop. Therefore, we will call this a dwelling arc. For dwelling arcs,  $t(e_{i,j',k}, e_{i,j,k})$  is the minimum time that we want to plan for dwelling. We will call this the nominal dwell time. The nominal dwell time will depend on the type of station. We will choose a different parameter value at short stops, basic stations and main stations. These parameter values will be unveiled in Section 6.2, after a timetable has been analysed.

If  $q_j$  is a station entrance or junction passage,  $(e_{i,j',k}, e_{i,j,k})$  is called a running arc. Then  $t(e_{i,j',k}, e_{i,j,k})$  is the minimum time that is allocated for  $z_{i,k}$  to complete the journey between  $q_{j'}$  and  $q_j$ . We will have the technical minimal running time available for each running arc. On top of the technical minimal running time, margin can be included within the minimum running time.

Until now, we have described arcs which specify the time separation required between events of the same train. However, it is also important that sufficient distance is maintained between different trains. We use headway arcs for this. We will use a fixed train order. Therefore, we can determine the order in which trains complete a certain activity. Consider a train  $z_{i,k}$  and activity  $q_j \in Q_i$ . We will assume that a separate activity is created for each separated route, such that distance needs to be kept to the previous train completing the activity. This makes it easier to explain which vertices need to be connected by an arc. If  $z_{i,k}$  is the first train which visits  $q_j$ , then we will not create a headway arc with head  $e_{i,j,k}$ . Otherwise, let  $z_{i',k'}$  be the train which visits  $q_j$  before  $z_{i,k}$ .

If  $q_j$  is the passing of a junction, we will create an arc  $(e_{i',j,k'}, e_{i,j,k})$ . This means that consecutive trains need to be separated by a minimum headway time  $t(e_{i',j,k'}, e_{i,j,k})$  when passing a junction. Also if  $q_j$  is a station exit, we create a headway arc  $(e_{i',j,k'}, e_{i,j,k})$ . So minimum headway needs to be kept to the previous train exiting the station. However, if  $q_j$  is an entrance at a short stop, then we will not create an arc to the station entrance of the previous train. At short stops it is not possible to switch tracks, so the previous train at the same platform needs to have left the station before a new train can enter. Let  $q_{j'}$  be the activity for the exit from the station where  $q_j$  is the entrance. Then a headway arc  $(e_{i',j',k'}, e_{i,j,k})$  is created. At basic and main stations, it is possible that trains approach a station on the same track, but stop at a different platform. So it is not always required that the previous train which entered the station via the track has left the station. Therefore, we use a headway arc  $(e_{i',j,k'}, e_{i,j,k})$  to describe the minimum headway between consecutive station entrances.

Basic and main stations sometimes have more platforms available than there are tracks before the station. But it is still important that a train has left the platform before a new train can enter the station to stop at the same platform. We do create additional arcs for this, which are called platform arcs. Let  $e_{i,j,k}$  be the entrance of a train at a basic or main station, and  $e_{i',j',k'}$  the exit of the station for the previous train using the same platform. Then, an arc  $(e_{i',j',k'}, e_{i,j,k})$  is created (unless  $e_{i,j,k}$  was the first train to use the platform). The length of headway and platform arcs will be revealed in Section 6.2. We do not consider the possibility that a train can be diverted to another free platform. So all trains will stick to the platform that was assigned in the original timetable.

We have introduced platforms arcs, with the tail being a station exit and the head being a station entrance. We do consider networks which end at a station entrance. Therefore, we do not know when trains leave the final station and the platform becomes free for the next train. Managing the platform allocation at this final station will be part of a new network. So it is possible to omit platform arcs for the final station. However, it is also possible that a train line ends somewhere within the network. Then we only model the station entrance at its final station too. If the used platform is also used by other trains, then it is recommended to add a platform headway arc. This can be done from the station entrance of the terminated train to the station entrance of the subsequent train using the same platform. The length of this arc should contain the estimated time for the terminated train to free up the platform. Connecting station entrances of trains using the same platform with a platform arc is also an alternative approach at the final station in the network.

**Remark 3.2.3.** *It is possible to introduce more arcs between event vertices, like transfer arcs. Transfer arcs can be used to enforce a train to wait before exiting a station, until enough time has elapsed since a different train entered the station, such that crew or passengers can transfer.*

We will now discuss how we will determine all schedule times for schedule vertices  $x_{i,j,k}$  with  $k = 1$ . For each train service  $i$ , we do not modify the schedule time for the first activity. This is done to make sure that the train trajectory used before entering the considered network can be maintained. Therefore, for each train service  $i$ , we pick the activity  $q_j$  at the start of  $P_i$ . If this activity was originally scheduled at time  $x_{i,0}$  (in seconds), then we create an arc  $(v_0, x_{i,j,1})$  with length  $t(v_0, x_{i,j,1}) = x_{i,0}$ . We call this starting arcs.

We now need to set the schedule time for the remaining  $q_j \in P_i$ . If  $q_{j'}$  is followed by  $q_j$  in  $P_i$ , then we add an arc  $(x_{i,j',1}, x_{i,j,1})$ . We will call this a timetabling arc. The length of this arc will be at least  $o_{i,j}$ , which is a parameter describing the minimum time that we want to have between  $x_{i,j',1}$  and  $x_{i,j,1}$ . Here,  $o_{i,j}$  should not be lower than the minimum time required between  $e_{i,j',1}$  and  $e_{i,j,1}$  based on running, through and dwelling arcs. On top of  $o_{i,j}$ , we can add extra supplement time  $\varepsilon_{i,j}$ . So we get

$$t(x_{i,j',1}, x_{i,j,1}) = o_{i,j} + \varepsilon_{i,j}. \quad (3.6)$$

We will sometimes have to sum over all  $o_{i,j}$  or  $\varepsilon_{i,j}$  for a train service  $i$ . We know that  $o_{i,j}$  and  $\varepsilon_{i,j}$  exist for  $q_j \in P_i$ . With exception of the  $q_j$  at the start of  $P_i$ , which does not have an incoming timetabling arc (but a starting arc instead). Therefore, for each train service  $i$  we introduce a set

$$J_i := \{j | q_j \in P_i, q_j \text{ not the start of } P_i\}. \quad (3.7)$$

We do not want to modify the train timetable outside of the considered network. Therefore, for each train service  $i$  and the last  $q_j \in P_i$ ,  $x_{i,j,1}$  should be equal to the original schedule time  $x_i^*$ . This means that we cannot add an unlimited amount of supplement. Instead, for each train service it is required that

$$x_i^* = x_{i,0} + \sum_{j \in J_i} (o_{i,j} + \varepsilon_{i,j}). \quad (3.8)$$

Here,  $x_{i,0}$ ,  $x_i^*$  and all  $o_{i,j}$  are parameters, that will be determined in Chapter 6. After these parameters are fixed, we can only vary the  $\varepsilon_{i,j}$ -variables. Therefore, we have the constraint

$$s_i := x_i^* - x_{i,0} - \sum_{j \in J_i} o_{i,j} = \sum_{j \in J_i} \varepsilon_{i,j}. \quad (3.9)$$

So for each train service  $i$ ,  $s_i$  is the amount of supplement we can allocate across the route. By modifying the variables  $\varepsilon_{i,j}$ , the supplement is allocated. We will make use of random search metaheuristics to modify the  $\varepsilon_{i,j}$ -variables, as will be explained in Chapter 5.

It is inconvenient to communicate continuous departure and arrival times. Therefore, schedule times are changed to full seconds. This is more precise than the current system being used by NS, which has a six-second precision (Hoogenraad, 2021). The parameters  $o_{i,j}$  can be set by design to full seconds. The variables  $\varepsilon_{i,j}$  will be rounded down. It is possible that rounding leads to the last activity of train service  $i$  not being planned at  $x_i^*$ . If this is the case, then  $\varepsilon_{i,j}$  leading to the last vertex  $x_{i,j,1}$  for a train is adjusted upwards until it leads to  $x_{i,j,1} = x_i^*$ .

We have not yet determined  $x_{i,j,k}$  for  $k \geq 2$ . The timetable needs to be hourly cyclic. Therefore, after all  $x_{i,j,1}$  have been determined, we can set  $x_{i,j,k}$  for  $k \geq 2$ . This is done by creating an arc  $(x_{i,j,1}, x_{i,j,k})$  with  $t(x_{i,j,1}, x_{i,j,k}) = 3600(k - 1)$ . Since we work with arc lengths in seconds, this is exactly  $k - 1$  hours after the first train of the same service has completed  $q_j$ . We will call these arcs cyclicity arcs.

Lastly, to communicate the schedule time to the corresponding event vertex, we create an arc  $(x_{i,j,k}, e_{i,j,k})$  for all  $x_{i,j,k} \in V_x$ . These arcs are called communication arcs and will have length  $t(x_{i,j,k}, e_{i,j,k}) = 0$ . Communication arcs prevent an event from happening before its scheduled time.

### 3.3 Feasibility check

In the previous section, a directed graph was constructed, which can be used to set a timetable. In this section, we will explain how it can be checked that a constructed timetable is attainable. We will call this the feasibility check.

By choosing the  $\varepsilon_{i,j}$ -variables, a schedule time is determined for all events that need to be planned. We do not want to prevent that a schedule time is chosen, which can not be achieved if a train runs as expected. We consider a train run to run as expected if it enters the network without an entrance delay, dwells within the nominal dwell time and completes all running processes within the assigned margin. This will be quantified in Section 6.2.

For trains  $z_{i,k}$  with  $k = 1$ , subsequent activities are connected with arcs of length at least  $o_{i,j}$ . This  $o_{i,j}$  describes the minimum time that we want to have between two scheduled events. Therefore, for each individual train  $z_{i,1}$  (with  $k = 1$ ) events have not been planned too close together. This also holds for trains  $z_{i,k}$  with  $k \geq 2$ , since they have a timetable that has been shifted by  $k - 1$  hours compared to a train  $z_{i,1}$ . So sufficient time is planned between subsequent scheduled activities for each individual train.

The schedule vertices are not connected by headway or platform arcs, as can be seen in Figure 3.1. For determining the event times, we do have headway and platform arcs. It is possible that satisfying these constraints requires that the event time exceeds the schedule time ( $e_{i,j,k} > x_{i,j,k}$ ). But this would mean that the schedule time determined for the event is not attainable, when sufficient headway is kept. This is undesirable.

Ideally, we would create a timetable which guarantees that each schedule time is attainable when keeping the required headway. But if all interdependencies between schedule times of different trains are considered, then a lot more interdependent variables are required to modify the timetable. It would be harder to design a metaheuristic which can update these variables. Therefore, we use a simpler encoding instead, where the  $\varepsilon_{i,j}$ -variables can be modified. This also means that we do not use variables to represent the amount of buffer between two trains. We modify the time supplements, which does lead to changes in buffer times that can also impact robustness.

After a timetable is created, a feasibility check is used to make sure that all schedule times are attainable. If this is not the case, the timetable is rejected. The success of this approach will depend on how fast feasible solutions can be found. If infeasible solutions occur too often, then a repair procedure needs to be created. This repair procedure can attempt to adjust infeasible solutions by modifying particular event times that are not attainable.

So we need to check if all schedule times of a constructed timetable are attainable. This can be done by propagating through the graph and calculating all event times  $e_{i,j,k}$ . If an event time exceeds its scheduled time ( $e_{i,j,k} > x_{i,j,k}$ ), then  $x_{i,j,k}$  is unattainable. In Algorithm 1, we present the feasibility check. Calculating the (schedule or event) time of a vertex  $v$ , does require knowing the time  $u$  for all  $(u, v) \in A_{\text{in}}(v)$ . To make sure that these times have already been calculated, the vertices in the graph can be ordered in topological order. We will use  $\text{ord}(V, A)$  to denote the topological ordering of a vertex set  $V$  on a graph with arcs  $A$ . We also introduce  $\text{ord}(V_x, A)$  and  $\text{ord}(V_e, A)$ , for the vertices in  $\text{ord}(V, A)$  which are part of  $V_x$  and  $V_e$  respectively.

Algorithm 1 works as follows. First, the source vertex is set in line 1. Subsequently, all schedule times are calculated in lines 2-3. Lines 4-8 consider event vertices. An event time is calculated in line 5. Line 6 determines if the event is scheduled and line 7 checks if the schedule time is unattainable. If a schedule time is unattainable, feasibility is rejected in line 8. If all schedule times are attainable, this will be confirmed in line 9.

**Algorithm 1:** Feasibility check

**Input** : Graph  $G = (V, A, t)$   
Sequences  $\text{ord}(V_e, A)$  and  $\text{ord}(V_x, A)$   
Incoming arc set  $A_{\text{in}}(v)$  for all  $v \in V$   
Function  $\text{plan}(e_{i,j,k})$  (defined for all  $e_{i,j,k} \in V_e$ )

**Output:** Feasibility

```

1  $v_0 \leftarrow 0$ 
2 for  $x_{i,j,k} \in \text{ord}(V_x, A)$  do
3    $x_{i,j,k} \leftarrow \max\{u + t(u, x_{i,j,k}) \mid (u, x_{i,j,k}) \in A_{\text{in}}(x_{i,j,k})\}$ 
4 for  $e_{i,j,k} \in \text{ord}(V_e, A)$  do
5    $e_{i,j,k} \leftarrow \max\{u + t(u, e_{i,j,k}) \mid (u, e_{i,j,k}) \in A_{\text{in}}(e_{i,j,k})\}$ 
6   if  $\text{plan}(e_{i,j,k})$  then
7     if  $e_{i,j,k} > x_{i,j,k}$  then
8       return False (The solution is not feasible)
9 return True (The solution is feasible)

```

The time complexity of Algorithm 1 is dependent on the size of the sets  $A_{\text{in}}(v)$ . We have created a graph, where event vertices  $e_{i,j,k}$  can have at most four incoming arcs: a communication arc, a headway arc, a platform arc and a running arc. Event vertices can also have incoming dwell or through arcs, but then there is no incoming platform and running arc. Schedule vertices  $v_{i,j,k}$  always have one incoming arc, this can be a timetabling, starting or cyclicity arc. The source vertex  $v_0$  does not have any incoming arcs. So for the constructed graph  $|A_{\text{in}}(v)| \leq 4$ .

The time complexity of Algorithm 1 is equal to  $O(|V|)$  if  $|A_{\text{in}}(v)|$  is bounded by a constant. As explained, this is true for the constructed graph. If this would not be the case, the algorithm would run in  $O(|A| + |V|)$  time. The time complexity would be larger, if the vertices would still need to be sorted in topological order. However, the same vertex and arc set are used for the evaluation of all timetables. So a topological order would only need to be determined once and not every time the feasibility check is performed.

**Remark 3.3.1.** From a graph  $G$ , all communication arcs that do not lead to the first or last activity of a train can be removed to create a graph  $G'$ . By applying Algorithm 1 on  $G'$ , it can be checked whether a feasible solution exists under the given constraints. So checking if there exists a feasible solution can be done in  $O(|V|)$  time.

### 3.4 Delay propagation algorithm

In this section, we describe a delay propagation algorithm. This algorithm calculates the departure and arrival times of trains under multiple delay scenarios. This can be used to measure the robustness of the timetable. We introduce the following notation.

In the delay propagation algorithm, we consider  $R$  different delay scenarios. A delay scenario is denoted as  $\delta^r$ . A set of delay scenarios  $\delta^1, \delta^2, \dots, \delta^R$  will be denoted as  $\Delta_R$ . To calculate delay propagation, we use the same vertex set  $V$  and arc set  $A$  as in the graph  $G$  used for the feasibility check. But the arc lengths are different for each delay scenario. Therefore, we denote the arc length under delay scenario  $\delta^r$  as  $t^r(\cdot, \cdot)$ .

We will now explain how the length function  $t^r$  differ from the length function  $t$  used in the feasibility check. Firstly, for each dwelling arc, the length  $t(\cdot, \cdot)$  was a nominal dwell time. Dwelling arcs instead get a sampled dwell time as their length  $t^r(\cdot, \cdot)$ , these dwell times are all described in the delay scenario  $\delta^r$ . Furthermore,  $\delta^r$  will consist of an entrance delay for each train. Entrance delays impact the time at which a train  $z_{i,k}$  can complete its first activity. If  $q_j$  is the first activity in  $P_i$ , let the entrance delay that affects  $e_{i,j,k}$  be denoted

as  $\delta_{i,j,k}$ . The connection arc will then have length  $\delta_{i,j,k}$ . It is also possible to sample all running arcs, but we will not do this as will be explained in Section 4.2. Still, the length of running arcs will be different than in the feasibility check. In the feasibility check, running arc lengths  $t(\cdot, \cdot)$  included some margin on top of the technical minimal running time. For the delay propagation algorithm, running arcs will use the technical minimal running time as their length  $t^r(\cdot, \cdot)$ . This means that the margin incorporated in the minimum running time may be used to make up delay. All remaining arcs lengths  $t^r(\cdot, \cdot)$  are equal to  $t(\cdot, \cdot)$ .

In Algorithm 2, all realised event times are calculated. Initialisation for the source vertex happens in line 1. The schedule times are calculated in lines 2-3. The schedule times do not change based on a delay scenario, so line 3 is independent of  $r$ . Still, in lines 4-5, we need to introduce  $x_{i,j,k}^r$  values, to make sure that  $u^r$  is well defined in line 8. Calculating all realised event times happens in lines 6-8. Finally, all calculated realised event times are returned in line 9.

**Algorithm 2:** Delay propagation algorithm

**Input :** Graph  $G = (V, A, t)$   
Sequences  $\text{ord}(V_e, A)$  and  $\text{ord}(V_x, A)$   
Incoming arc set  $A_{\text{in}}(v)$  for all  $v \in V$   
Number of delay scenarios  $R$   
Arc length functions  $t^1, t^2, \dots, t^R$  based on delay scenarios  $\Delta_R$

**Output:** Realised event times  $e_{i,j,k}^r$

```

1  $v_0 \leftarrow 0$ 
2 for  $x_{i,j,k} \in \text{ord}(V_x, A)$  do
3    $x_{i,j,k} \leftarrow \max\{u + t(u, x_{i,j,k}) \mid (u, x_{i,j,k}) \in A_{\text{in}}(x_{i,j,k})\}$ 
4   for  $r \in \{1, 2, \dots, R\}$  do
5      $x_{i,j,k}^r \leftarrow x_{i,j,k}$ 
6 for  $e_{i,j,k} \in \text{ord}(V_e, A)$  do
7   for  $r \in \{1, 2, \dots, R\}$  do
8      $e_{i,j,k}^r \leftarrow \max\{u^r + t^r(u, e_{i,j,k}) \mid (u, e_{i,j,k}) \in A_{\text{in}}(e_{i,j,k})\}$ 
9 return all calculated  $e_{i,j,k}^r$ 

```

For an arbitrary graph  $G$ , Algorithm 2 runs in time complexity  $O(R \cdot (|A| + |V|))$ . For the graph constructed in Section 3.2, the time complexity reduces to  $O(R \cdot |V|)$ , since  $|A| \leq 4|V|$  (as explained below Algorithm 1). Algorithm 2 also offers possibilities for easy parallelisation. In line 8,  $e_{i,j,k}^r$  can be calculated for all  $r$  simultaneously. A possible approach is collecting all realisations of a vertex  $v$  in a vector  $[v^1, v^2, \dots, v^R]^T$ . Also, the length functions  $t^r(u, v)$  would need to be collected in a vector. Then, applying line 8 on all realisations in the vector can be done much faster than one-by-one. Using the NumPy library in Python, for  $R = 100$ , using a vector to compute lines 7-8 is around 70 times faster. For  $R = 1000$  this increases to 450 times.

The efficiency of parallel computation, does mean that the feasibility check from Algorithm 1 is relatively slow. It would be more efficient to compute the event times and schedule times for the feasibility check alongside the delay propagation. Therefore, we will combine the feasibility check and delay propagation algorithm in Section 3.6. The new algorithm will also calculate the objective function, which is introduced in the next section.

### 3.5 Objective function

In this section, we will create the objective function. In the objective function, we punish delays. Delays are scheduled events, where the realised event time exceeds the schedule time ( $e_{i,j,k}^r > x_{i,j,k}$ ). In the objective function, we will sum over all scheduled events. For readability, we introduce the notation

$$U = \{(i, j, k) | x_{i,j,k} \in V_x\}. \quad (3.10)$$

We will now introduce the first component of the objective function, which is a weighted average delay. The weight of an event  $e_{i,j,k}$  describes its importance and will be denoted as  $w_{i,j,k}$ . The weighted average delay is equal to

$$\frac{1}{R} \sum_{r=1}^R \sum_{(i,j,k) \in U} w_{i,j,k} (e_{i,j,k}^r - x_{i,j,k}). \quad (3.11)$$

Please note that by design  $e_{i,j,k}^r \geq x_{i,j,k}$ , this is ensured by communication arcs. Therefore,  $e_{i,j,k}^r - x_{i,j,k}$  is non-negative. So it is not possible that early trains cancel out the delay of late trains.

We will now determine the  $w_{i,j,k}$ -values, which describe the importance of an event. To measure the discomfort for passengers, it would be best to determine weights based on the number of impacted passengers. This is done in e.g. Dollevoet et al. (2012), but requires detailed passenger data that is not available for this thesis. The number of passengers visiting a station a day is publicly known data (NS, 2025), but this is not enough to deduce the number of passengers per train. So a different weight structure needs to be used.

In the evaluation procedure of Fischetti et al. (2009), all weights are set to 1. Maróti (2017) uses weight 1 for arrival events and weight 0 for departure events. Both these approaches have two problems. Firstly, sprinters would become disproportionally important. Since sprinters make more stops on the same route than intercity trains, a delayed sprinter will lead to a higher punishment. Secondly, it is possible that trains continue their journey outside of the considered network. If a train leaves the network delayed, passengers which are still in the train might arrive late to their destination. But this is not measured, because the delay happens outside of the considered network. Furthermore, delayed trains exiting the network can disturb other trains and can cause further knock-on delay. A third disadvantage of the method used by Maróti (2017), is that departure events will be scheduled as soon as allowed. If this is unrealistically early, train dispatchers will need to adjust the traffic in stations more frequently, which leads to a higher workload. Additionally, the departure times will be unreliable for passengers.

We will create a new weight strategy, which takes into account these disadvantages. Firstly, we will give weight 1 to departures at main and basic stations to avoid unrealistic departure times. We do not schedule departure times at short stops, so delay is not punished here. To balance the importance of sprinter and intercity trains, we use the following strategy. For each stop or station passed by a train, the weight will be added to the arrival of the next station where the train does stop. This is illustrated in Figure 3.2.

The arrival of a train at its initial station in the network gets weight 0 (as is depicted in Figure 3.2). We can not modify the arrival time at the initial station, so a train has not yet experienced a modified timetable when arriving here. Therefore, the arrival delay would be the same for each train and it is not meaningful to measure this delay in the objective function.

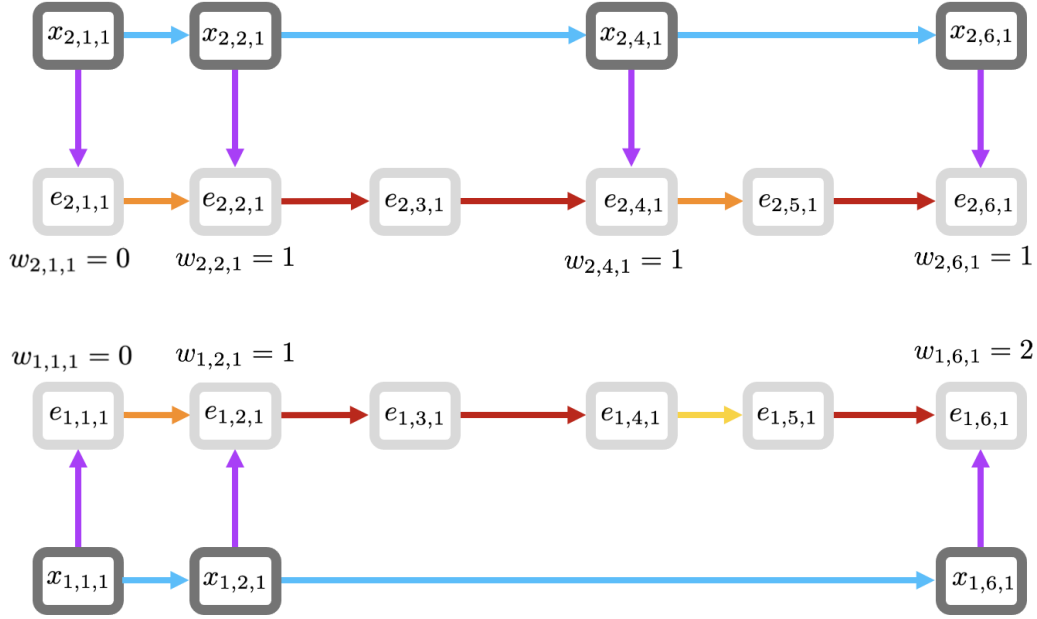


**Figure 3.2:** Weight function for the vertices of two trains in Figure 3.1. No weights are given for unscheduled events. For the bottom train,  $e_{1,4,1}$  is unscheduled. There is a different train for which the activity  $q_4$  is scheduled. Therefore, the weight is increased for  $e_{1,6,1}$ , which is the next scheduled event for the bottom train.

The introduced weight scheme does balance the importance of sprinter and intercity trains. It does not address that trains with a delayed network exit, can still cause further hindrance. We will now introduce a second component for the objective function, which takes this into account.

Exit delay will not immediately disappear, but it is possible that delay is reduced between stations. We will use the following approximation: delay can be reduced by 30 seconds between consecutive scheduled events. Furthermore, we will make the assumption that scheduled events outside of the considered network have weight 1. These assumptions will be substantiated in Section 6.2. If the exit delay is 210 seconds, then this leads to a further delay contribution of

$$180 + 150 + 120 + 90 + 60 + 30 + 0. \quad (3.12)$$

More generally, let  $e_{i,j,k}$  be the last scheduled event of train  $z_{i,k}$  and pick a delay scenario  $\delta^r$ . First, consider the case that there is  $N \in \mathbb{N}$  such that  $e_{i,j,k}^r - x_{i,j,k} = 30N$ , so the delay is a multiple of 30 seconds. This leads to a delay contribution of

$$30 \sum_{m=0}^{N-1} m = 30 \frac{N(N-1)}{2}. \quad (3.13)$$

The right hand side of this equation has been calculated by applying Gauss summation. We can substitute  $N$  into the right hand side, to obtain

$$\frac{1}{2} \left( -1 + \frac{e_{i,j,k}^r - x_{i,j,k}}{30} \right) (e_{i,j,k}^r - x_{i,j,k}). \quad (3.14)$$

We have now obtained a formula, which we can also use if  $e_{i,j,k}^r - x_{i,j,k} \neq 30N$ . However, (3.14) becomes negative for  $e_{i,j,k}^r - x_{i,j,k} \in (0, 30)$ . This is undesired, therefore we will make the term non-negative. We denote the obtained term as

$$h(e_{i,j,k}^r, x_{i,j,k}) := \max \left\{ 0, \frac{1}{2} \left( -1 + \frac{e_{i,j,k}^r - x_{i,j,k}}{30} \right) (e_{i,j,k}^r - x_{i,j,k}) \right\}. \quad (3.15)$$

We will often refer to this function as the bonus term. The bonus term needs to be applied at network exit only. A variable  $b_{i,j,k}$  will be used to denote whether an event is a network exit. This variable is defined as

$$b_{i,j,k} = \begin{cases} 1 & \text{if } q_j \text{ is the final activity in } P_i, \\ 0 & \text{otherwise.} \end{cases} \quad (3.16)$$

Now, we can introduce the objective function for a timetable  $x$  and delay scenarios  $\Delta_R$  as

$$\mathcal{L}(x, \Delta_R) = \frac{1}{R} \sum_{r=1}^R \sum_{(i,j,k) \in U} w_{i,j,k} (e_{i,j,k}^r - x_{i,j,k}) + b_{i,j,k} \cdot h(e_{i,j,k}^r, x_{i,j,k}). \quad (3.17)$$

Furthermore, we will introduce the part of the objective function for one delay scenario  $\delta^r$ , which is

$$\ell(x, \delta^r) = \sum_{(i,j,k) \in U} w_{i,j,k} (e_{i,j,k}^r - x_{i,j,k}) + b_{i,j,k} \cdot h(e_{i,j,k}^r, x_{i,j,k}). \quad (3.18)$$

So we have introduced an objective function, which consists of two components. The first component is a weighted average delay. The second component punishes delay after network exit. It is possible to extend this objective function. To measure the inconvenience for passengers, it can also be measured whether there is sufficient time to transfer between trains. For NS, it is also important that high 3-minute and 10-minute punctualities are maintained. These are performance indicators, for which a threshold value has been agreed with the Dutch government (NS, 2026). So an extra penalty can be added for all delays exceeding the 3-minute or 10-minute mark. For the 3-minute mark, this is done in Maróti (2017). But since adding an extra term to the objective function would further reduce the interpretability, we will not do this.

## 3.6 Combining algorithms

In this section, an algorithm is introduced. It combines performing a feasibility check, determining delay propagation and calculating the objective value.

As established in Section 3.4, computing 1000 delay scenarios in parallel can be done efficiently. If a solution is feasible, all vertices need to be traversed in Algorithm 1. Then, Algorithm 1 is almost as slow to compute as Algorithm 2, even when Algorithm 2 uses one thousand delay scenarios. If a timetable is feasible, it is more efficient to calculate the event times for the feasibility check alongside the realised event times. However, Algorithm 1 does have an advantage. If a vertex is considered, with an unattainable schedule time, the algorithm does immediately terminate. This approach will also be applied in the combined algorithm that we will present. Since the timetable is cyclic, an unattainable schedule time will usually lead to termination of the algorithm within the first or second cycle. Although this is not guaranteed, as is discussed in Remark 3.6.1.

**Remark 3.6.1.** *Despite using a cyclic timetable, it is not always sufficient to check the first few cycles for feasibility. Because we do not determine a schedule time for all event vertices, it is possible that event times change between consecutive hours. A shift of an event time, which is not scheduled, can cause a schedule time to become unattainable. So a timetable is only feasible when also all event times for unscheduled events do not shift any more. However, it is hard to design a good check for this. Especially for larger networks, this is difficult to check since some trains will only meet each other after a few cycles. This would also be a problem, if the methodology is extended to timetables that are not completely cyclic. Therefore, no check has been added to Algorithm 1 to conclude feasibility based on a subset of  $V$ .*

We calculate the delay propagation similar to Algorithm 2. However, besides the realised event times, we also calculate the event times  $e_{i,j,k}$  as in the feasibility check. We will refer to  $e_{i,j,k}$  as a nominal event time and they are calculated in line 9. If the nominal event time exceeds the schedule time, the algorithm is terminated in lines 13-14. Otherwise, in lines 15-16, we update the objective values based on the determined realised event times. Here the variable  $\ell^r$  will be equal to  $\ell(x, \delta^r)$  at the end of the algorithm. If the timetable is feasible, we calculate the objective value  $\mathcal{L}(x, \Delta_R)$  in line 17.

For an arbitrary graph  $G$ , Algorithm 3 runs in time complexity  $O(R \cdot (|A| + |V|))$ . For the graph constructed in Section 3.2, the time complexity reduces to  $O(R \cdot |V|)$ , since  $|A| \leq 4|V|$  (as explained below Algorithm 1). By using  $R + 1$ -dimensional vectors, easy parallelisation of lines 9-11 can be achieved.

**Algorithm 3:** Algorithm to determine feasibility, delay propagation and objective value

**Input :** Graph  $G = (V, A, t)$   
Sequences  $\text{ord}(V_e, A)$  and  $\text{ord}(V_x, A)$   
Incoming arc set  $A_{\text{in}}(v)$  for all  $v \in V$   
Number of delay scenarios  $R$   
Arc length functions  $t^1, t^2, \dots, t^R$  based on delay scenarios  $\Delta_R$   
Function  $\text{plan}(e_{i,j,k})$  (defined for all  $e_{i,j,k} \in V_e$ )  
Set of scheduled indices  $U$   
Weights  $w_{i,j,k}$  for  $(i, j, k) \in U$   
Bonus term function  $h(\cdot, \cdot)$   
Bonus term indicators  $b_{i,j,k}$  for  $(i, j, k) \in U$

**Output:** Feasibility, objective value  $\mathcal{L}$  and  $\ell^r$ -values

```

1  $v_0 \leftarrow 0$ 
2 for  $x_{i,j,k} \in \text{ord}(V_x, A)$  do
3    $x_{i,j,k} \leftarrow \max\{u + t(u, x_{i,j,k}) \mid (u, x_{i,j,k}) \in A_{\text{in}}(x_{i,j,k})\}$ 
4   for  $r \in \{1, 2, \dots, R\}$  do
5      $x_{i,j,k}^r \leftarrow x_{i,j,k}$ 
6   for  $r \in \{1, 2, \dots, R\}$  do
7      $\ell^r \leftarrow 0$ 
8   for  $e_{i,j,k} \in \text{ord}(V_e, A)$  do
9      $e_{i,j,k} \leftarrow \max\{u + t(u, e_{i,j,k}) \mid (u, e_{i,j,k}) \in A_{\text{in}}(e_{i,j,k})\}$ 
10    for  $r \in \{1, 2, \dots, R\}$  do
11       $e_{i,j,k}^r \leftarrow \max\{u^r + t^r(u, e_{i,j,k}) \mid (u, e_{i,j,k}) \in A_{\text{in}}(e_{i,j,k})\}$ 
12    if  $\text{plan}(e_{i,j,k})$  then
13      if  $e_{i,j,k} > x_{i,j,k}$  then
14        return False (the solution is not feasible)
15      for  $r \in \{1, 2, \dots, R\}$  do
16         $\ell^r \leftarrow \ell^r + w_{i,j,k}(e_{i,j,k}^r - x_{i,j,k}) + b_{i,j,k} \cdot h(e_{i,j,k}^r, x_{i,j,k})$ 
17   $\mathcal{L} \leftarrow \frac{1}{R} \sum_{r=1}^R \ell^r$ 
18 return  $\mathcal{L}$  (and all  $\ell^r$ )

```

**Remark 3.6.2.** Algorithm 3 is a more elegant redesign of the algorithm which was used to gather results for this thesis. The algorithm that was used during implementation can be found in Appendix C.2. In this appendix, we also address that further improvement of the algorithm is possible, by considering vertices in a different order.

## 3.7 Timetable performance

In Section 3.5, we have discussed how timetable performance will be evaluated. An objective function which quantifies the robustness, is used for this. In this section, we will predict how the created timetables will perform in other areas.

For each train, the arrival at both the first and last station in the network have been fixed. This also limits the adjustments that can be made to the other event times. Therefore, only small changes will be made to journey times of passengers. However, we do not ensure that transfers present in the original timetable are maintained (see Remark 3.2.3). So, an adjusted timetable can lead to longer waiting times due to a transfer being missed. On the other hand, it is also possible that new transfer possibilities are created due to adjustments to the timetable. Adjustments can also impact the regularity of a timetable, which can increase the maximum waiting time until a train stops at a station. This can be prevented by making the same adjustments to all train services which are part of the same train line. This does reduce the number of variables that need to be optimised. On the other hand, the flexibility is reduced. Based on the test case, it will be decided in Chapter 6 if the timetable should be regular.

We have made sure that trains cannot arrive at a station early. An energy-efficient speed profile can then be selected to enter the station on time. However, in the delay propagation algorithm from Section 3.4, we also calculate event times for unscheduled events. These event times are set as early as possible based on all constraints. So trains travel at the maximum possible speed (while maintaining sufficient headway) past all intermediate stations. If an intermediate station is passed, shortly before a train stops at a station, this would not be energy efficient. The train would travel at maximum speed for a great portion of its route and then break hard to enter the station on time. It is more energy efficient to slow down earlier and coast towards the station. For more information on coasting, see e.g. Scheepmaker and Goverde (2015).

It is not problematic that unscheduled event times at intermediate stations are set as early as possible. Often there is still room to pick a different trajectory, which is more energy efficient. This can be determined in a post-processing step. It is important that this does not hinder the planned trajectory for the next train. In Figure 3.3, an example is presented of the freedom to adjust a train trajectory. Naturally, if there are fluctuations in running time, a more energy-efficient trajectory could be less robust. In that case, a trade-off needs to be made between robustness and energy efficiency. But during this thesis, we will only optimise timetable robustness.

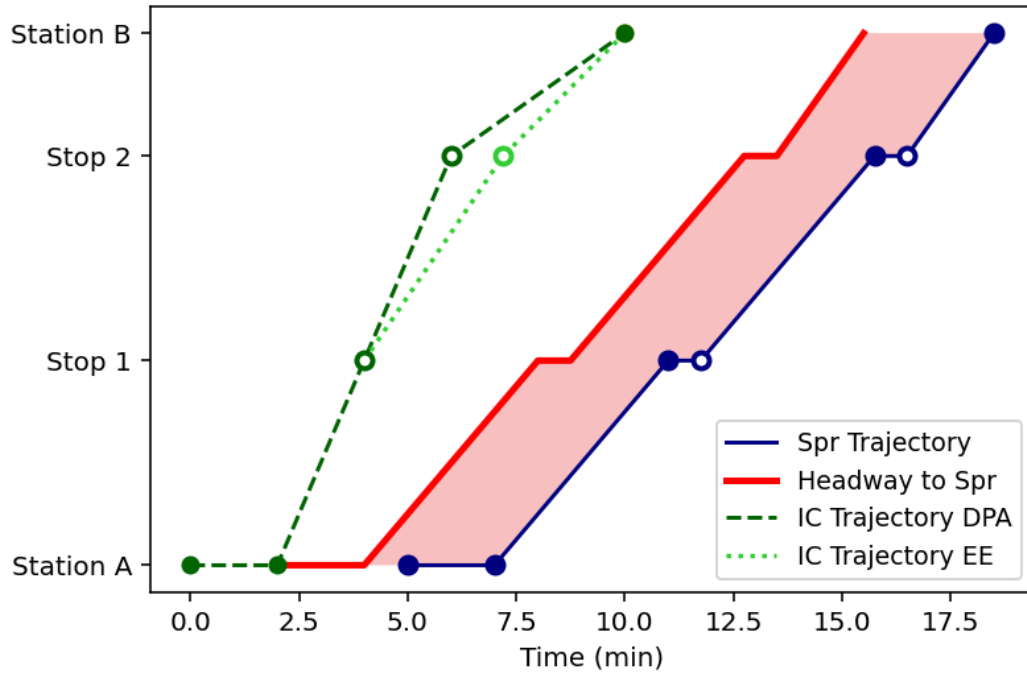


**Figure 3.3:** Time-distance diagram for the trajectory of an intercity (IC) and sprinter (Spr) train. All trajectories in between stations and stops are depicted with straight lines for simplicity. All scheduled events are depicted using filled circles, while unscheduled events are depicted using open circles. The sprinter train stops at stop 1 and stop 2, so the scheduled events largely determine the trajectory. The intercity train does not stop at stop 1 and stop 2, so a trajectory in between station A and B may be determined. The IC trajectory DPA describes the trajectory as determined in Algorithms 2 and 3, where intermediate stops are passed as soon as possible. The IC trajectory EE suggests a more energy-efficient trajectory, where coasting is used. All trajectories, which do not pass through the shaded pink area will not disturb the sprinter train.

## Chapter 4

# Delay scenarios

In this chapter, delay scenarios will be designed. Unfortunately, little research has been done on finding appropriate probability distributions for running times (Larsen et al., 2014). Therefore, we have chosen not to include variations in running time into the delay scenarios. Instead, we only vary dwell times (Section 4.3) and entrance delays (Section 4.4). In Section 4.2, we will discuss the number of delay scenarios required. But first we will explain why sampling delay scenarios leads to a good approximation of the expected delay.

### 4.1 Validity of sampling

In Section 3.5, we have introduced an objective function which quantifies the robustness based on sampled delay scenarios  $\delta^1, \delta^2, \dots, \delta^R \in \Delta_R$ . When this objective function is used, a timetable  $x$  can be found which optimises the objective function. The optimal timetable has the best allocation of supplement, so the best choice for  $\varepsilon_{i,j}$ -variables. In this section, we will use results from Kleywegt et al. (2002) to prove that the optimum found under  $\Delta_R$  converges to the solution which optimises the expected value of the objective function. Also the speed of convergence will be discussed.

We will now explain the form of the optimisation problems used in Kleywegt et al. (2002). The goal is to find a solution  $x \in S$  that minimises the expected value of  $\ell(x, \delta)$ . Here,  $\delta \sim P$  denotes a random vector, which has probability distribution  $P$ . We can denote this as

$$\min_{x \in S} \{g(x) := \mathbb{E}_P [\ell(x, \delta)]\}.$$

We will work with the following setup. The variable  $x$  represents the timetable. Let the set  $S$  consist of all timetables, which have valid  $\varepsilon_{i,j}$ -variables (non-negative and total supplement per train service equal to  $s_i$ ). The random vector  $\delta$  consists of all delays that need to be sampled. Which probability distributions are used to sample  $\delta$ , will be decided later this chapter. The delay scenarios  $\delta^1, \dots, \delta^R$  are independent identically distributed (i.i.d.) realisations of this vector  $\delta$ . In (3.18), we introduced the function  $\ell(x, \delta^r)$ . This function determines the objective value for a sampled delay scenario  $\delta^r$  and timetable  $x$ . We will now work with a general vector  $\delta$  rather than sampled vectors  $\delta^r$ . We will denote the arc length under  $\delta$  as  $t^\delta$  rather than  $t^r$ . Furthermore, we denote  $e_{i,j,k}^\delta$  rather than  $e_{i,j,k}^r$ . By slight abuse of notation, we will also use

$$\ell(x, \delta) = \sum_{(i,j,k) \in U} w_{i,j,k} (e_{i,j,k}^\delta - x_{i,j,k}) + b_{i,j,k} \cdot h(e_{i,j,k}^\delta, x_{i,j,k}). \quad (4.1)$$

We will now state the results presented in Kleywegt et al. (2002) about the convergence of solutions. Thereafter, we will show that these results are applicable under the introduced setup.

In Kleywegt et al. (2002), the following conditions need to be satisfied.

**Condition 4.1.1.** *The optimisation problem*

$$\min_{x \in S} \{g(x) := \mathbb{E}_P [\ell(x, \delta)]\}$$

is considered under the following conditions:

- (i)  $\delta$  is a random vector with probability distribution  $P$
- (ii)  $S$  is a finite set
- (iii)  $\ell(x, \delta)$  is a real-valued function of two vectors  $x$  and  $\delta$  with expected value  $\mathbb{E}_P[\ell(x, \delta)] = \int \ell(x, z)P(dz)$
- (iv)  $\ell(x, \cdot)$  is measurable for all  $x \in S$
- (v)  $\mathbb{E}_P [|\ell(x, \delta)|] < \infty$  for each  $x \in S$

We will now introduce some notation. Let  $\delta^1, \dots, \delta^R$  be independent realisations of  $\delta$ . Then the sample average function is defined as

$$\hat{g}_R(x) := \frac{1}{R} \sum_{r=1}^R \ell(x, \delta^r).$$

We will call  $\hat{g}_R(x)$  the sample average approximation (SAA) of  $g(x)$ . We will denote

$$v^* := \min_{x \in S} g(x) \text{ and } \hat{v}_R := \min_{x \in S} \hat{g}_R(x).$$

Additionally, let  $S^*$  consist of all  $x \in S$  such that  $g(x) = v^*$  and  $\hat{S}_R$  consist of all  $x \in S$  such that  $\hat{g}_R(x) = \hat{v}_R$ . Now we can introduce the concept of  $\xi$ -optimal sets.

**Definition 4.1.2** ( $\xi$ -optimal set). *The set of  $\xi$ -optimal solutions  $S_\xi$  consists of all  $x \in S$  such that  $g(x) \leq v^* + \xi$ .*

Similarly,  $\hat{S}_{R,\xi}$  are all  $x \in S$  such that  $\hat{g}_R(x) \leq \hat{v}_R + \xi$ . This allows us to formulate the following result. Please note that convergence with probability 1 is also commonly referred to as almost sure convergence.

**Proposition 4.1.3** (Kleywegt et al., 2002). *If the conditions from Condition 4.1.1 are satisfied, then the following properties hold:*

- (i)  $\hat{v}_R \rightarrow v^*$  with probability 1 as  $R \rightarrow \infty$
- (ii) for arbitrary  $\xi \geq 0$ ,  $\hat{S}_{R,\xi} \subseteq S_\xi$  happens with probability 1 as  $R \rightarrow \infty$

A proof of this result can be found after proposition 2.1 of Kleywegt et al. (2002). We will now continue with a result about the convergence rate.

**Proposition 4.1.4** (Kleywegt et al., 2002). *If Condition 4.1.1 holds and  $\delta$  is chosen such that  $\ell(x, \delta)$  is bounded for all  $x \in S$ , then for  $0 \leq \zeta \leq \xi$  it holds that there is a function  $\psi(\zeta, \xi) > 0$  such that*

$$\mathbb{P} [\hat{S}_{R,\zeta} \not\subseteq S_\xi] \leq |S \setminus S_\xi| e^{-R\psi(\zeta, \xi)}.$$

In proposition 2.2 of Kleywegt et al. (2002), this is proven under more general assumptions and for a specific  $\psi(\zeta, \xi)$ . These assumptions are satisfied if  $\ell(x, \delta)$  is bounded for all  $x \in S$  and  $\delta$  (Kroon et al., 2008). So the convergence happens exponentially fast.

We will now check if the results are applicable when using the setup introduced at the start of this section. Firstly, Condition 4.1.1(iv) is checked.

**Proposition 4.1.5.** *Consider a directed acyclic graph  $G(V, A, t^\delta)$  as introduced in Chapter 3. Then,  $\ell(x, \delta)$  as presented in (4.1) is measurable.*

*Proof.* Take a general graph  $G$ . We will use strong induction to show that all event times  $e_{i,j,k}^\delta$  and schedule times  $x_{i,j,k}$  are measurable. By slight abuse of notation, we will denote the time for a vertex  $v \in V$  as  $v^\delta$ . Despite  $x_{i,j,k}$  and  $v_0$  not changing based on  $\delta$ . We will use  $L(v)$  to denote the length of the longest path in  $G$ , leading to  $v \in V$ .

If  $L(v) = 0$ ,  $v$  must be the source vertex  $v_0$ . Therefore,  $v_0^\delta = 0$ . Since constants are measurable (Carothers, 2000),  $v_0^\delta$  is measurable.

Now take arbitrary  $m \in \mathbb{N}$  and assume that for  $L(v) \leq m$  it holds that  $v^\delta$  is measurable. Take an arbitrary vertex such that  $L(v) = m + 1$ . We know that  $v^\delta$  can be calculated as

$$v^\delta = \max\{u^\delta + t^\delta(u, v) \mid (u, v) \in A_{\text{in}}(v)\}. \quad (4.2)$$

We know that an arc length  $t^\delta(u, v)$  can either be a parameter or a random variable. Parameters are constants, and therefore measurable (Carothers, 2000). Also random variables are measurable (Gut, 2006). So all arc lengths are measurable.

Based on the induction hypothesis, we can show that all  $u^\delta$  in (4.2) are measurable. Fix a vertex  $u$ , such that  $(u, v) \in A_{\text{in}}(v)$ . Assume that  $L(u) \geq m + 1$ . If  $v$  is in the longest path to  $u$ , then there is a path from  $v$  to  $u$ . By adding the arc  $(u, v)$  we would create a cycle. Since  $G$  is acyclic,  $v$  can not be in the longest path to  $u$ . Therefore, we could create a path of length  $L(u) + 1 \geq m + 2$  to  $v$ , by adding  $v$  to the longest path to  $u$ . This would contradict  $L(v) = m + 1$ . Thus  $L(u) \leq m$  and by the induction hypothesis  $u^\delta$  is measurable.

Finite sums and maxima are both continuous. So (4.2) is a continuous function of  $u^\delta$  and  $t^\delta(u, v)$ . Measurability is maintained under continuous functions (Carothers, 2000), so  $v^\delta$  is measurable for all  $v \in V$ . So all  $e_{i,j,k}^\delta$  and  $x_{i,j,k}$  are measurable. We will now show that  $\ell(x, \delta)$  is measurable. Both  $h(e_{i,j,k}^\delta, x_{i,j,k})$  and  $(e_{i,j,k}^\delta - x_{i,j,k})$  are continuous functions of  $e_{i,j,k}^\delta$  and  $x_{i,j,k}$  and therefore measurable. Also,  $b_{i,j,k}$  and  $w_{i,j,k}$  are measurable due to being parameters. Lastly, measurability is maintained under finite sums and finite products (Carothers, 2000). Thus we conclude that  $\ell(x, \delta)$  is indeed measurable.  $\square$

So Condition 4.1.1(iv) holds as long as  $G$  is acyclic. A cycle in the directed graph would require an event to precede itself, so no feasible timetable would exist and constraints would need to be modified. Therefore, we can assume that there is no cycle. The timetable  $x$  is created based on the parameters  $o_{i,j}$  and  $x_{i,0}$ , and the variables  $\varepsilon_{i,j}$ . The parameters are not adapted for different timetables and  $\varepsilon_{i,j}$  is rounded to full seconds. Therefore, we have a discrete search space  $S$ . We also know that each  $\varepsilon_{i,j}$  is lower bounded by zero and upper bounded by  $s_i$ . Therefore,  $S$  is also finite and Condition 4.1.1(ii) holds. Condition 4.1.1(i) also holds, since  $\delta$  is a random vector. The function  $\ell(x, \delta)$  is real-valued, so we can define the expected value to satisfy Condition 4.1.1(iii). Lastly, if  $\delta$  consists of bounded random variables, then  $\ell(x, \delta)$  is bounded too for  $x \in S$ . But then  $\mathbb{E}_P[|\ell(x, \delta)|] < \infty$  as well. So we have also satisfied Condition 4.1.1(v).

We have now satisfied all conditions in Propositions 4.1.3 and 4.1.4. Please note that for using  $\ell(x, \delta)$  as described in (4.1),  $\hat{g}_R$  becomes the objective function  $\mathcal{L}(x, \Delta_R)$ . So we have established that asymptotically, the solution  $x \in S$  which minimises  $\mathcal{L}(x, \Delta_R)$  would also minimise  $\mathbb{E}_P[\ell(x, \delta)]$ . Therefore, we consider using a sample average approximation approach valid.

## 4.2 Number of scenarios

We will use a total of  $R$  delay scenarios, where the same delay scenarios are used to evaluate all timetables. It is important that a sufficient number of delay scenarios is used to give a good estimation of the delay. Since knock-on delays can propagate beyond one hour, each delay scenario will consist of an entire day. A similar approach is used in Kroon et al. (2008), where days of 21 hours are used. Kroon et al. (2008) used 20 days during optimisation. To evaluate the created timetables, up to 1000 days were used. We will work with a similar approach. But since fewer trains run in the late evening and at night, we will only use 16-hour days. This is used to mimic the timetable from 6:00 to 22:00.

We do not know if the twenty delay scenarios used during optimisation in Kroon et al. (2008) is sufficient to give meaningful results. In other literature, the number of delay scenarios varies greatly. Corman et al. (2011) generated 40 instances of entrance delays and 15 instances of dwell time delays, which were combined to form 600 delay scenarios. Corman et al. (2012) use 25 delay scenarios, Corman and Quaglietta (2015) use 30 scenarios and Kecman et al. (2013) use 200 scenarios.

The number of delay scenarios  $R$  will be important to determine whether a timetable  $x$  is significantly better than a different timetable  $x'$ . The hypothesis  $\mathbb{E}[\ell(x, \delta)] < \mathbb{E}[\ell(x', \delta)]$  can be tested based on the differences  $\ell(x, \delta^r) - \ell(x', \delta^r)$  for  $r \in \{1, 2, \dots, R\}$ . The student's t-test is a strong statistical test, which could be applied. However, this test requires the differences being normally distributed (Bijma et al., 2017). This could be checked by using QQ-plots, but this is inconvenient if a lot of timetables need to be compared. So we will use a different statistical test.

The Wilcoxon signed-rank test is a non-parametric test. It only requires differences to be symmetric (Rohatgi & Ehsanes Saleh, 2015). For testing whether the timetable  $x$  is better than the timetable  $x'$ , a one sided hypothesis can be used:

$H_0$ : The median of  $\ell(x, \delta) - \ell(x', \delta)$  is equal to 0

$H_1$ : The median of  $\ell(x, \delta) - \ell(x', \delta)$  is below 0

We do need to have symmetric differences to apply the Wilcoxon signed-rank test. We will check sample skewness to see if the differences are approximately symmetric. We will consider a sample to be approximately symmetric if the sample skewness lies in the interval  $[-0.5, 0.5]$ . This is often used as a rule of thumb, see e.g. Bulmer (2012), Hatem et al. (2022), and Piovesana and Senior (2018).

**Remark 4.2.1.** *The Wilcoxon signed-rank test can also be associated with a different null hypothesis. Instead of assuming that the distribution is symmetric and testing if the median is equal to zero, it can also be tested whether the distribution is symmetric around zero (H. Li & Johnson, 2014). This version also rejects the null hypothesis if the median is equal to zero, but the distribution is asymmetric. Therefore, this is less suitable to compare the performance of two timetables.*

**Remark 4.2.2.** *The Wilcoxon signed-rank test should not be confused with the Wilcoxon rank sum test, which is also known as the Mann-Whitney U test. The Mann-Whitney U test does assume that data is not paired, but instead all observations are independent (McElduff et al., 2010). Therefore, this is a weaker test than the Wilcoxon signed-rank test.*

If the differences are not symmetric, the Wilcoxon signed-rank test can not be applied. However, in that case, it is still possible to apply the sign test. The sign test can test the same null and alternative hypothesis, but is generally weaker. The sign test only looks at the signs of  $\ell(x, \delta^r) - \ell(x', \delta^r)$ , so whether differences are positive or negative. A binomial test is applied to check whether the number of negative differences is large enough to accept  $H_1$  (Bijma et al., 2017). In Chapter 7, we will discover that  $R = 1000$  is sufficient to give meaningful results on the test case that we will use.

### 4.3 Dwell times

In this section, we will determine probability distributions for the dwell times. This is important since delays can easily arise in stations. The number of passengers that want to enter or exit a train can vary and sometimes cause the scheduled dwell time to be exceeded. The development of delay in station areas was investigated by analysing data from Eindhoven station from 1997 by Goverde (2005). It was found that the departure delay of trains at stations was often higher than the arrival delay. For intercity or international trains, about 85% of trains arrived punctually at a station (at most 3 minutes late). At departure, the average punctuality had dropped below 80%. Longer dwell times were not only caused by longer passenger boarding and alighting. Waiting for transfers or clearance of outbound routes, also played an important role.

When determining delay propagation, we already have headway arcs to take into account outbound route clearance. So only the dwell time caused by passenger boarding and alighting needs to be incorporated in to the model. We would like to use the concept of free dwell time, which is defined as the required time for passengers to board, assuming no hindrance from other trains. In Yuan (2006), seven different probability distributions are tested to mimic the free dwell time at The Hague Hollands Spoor station, based on 10,000 trains which were recorded during September 1999. These distributions are uniform, normal, log-normal, gamma, beta, exponential and Weibull.

A statistical fitting procedure is used on a set of trains that are unlikely to have been hindered by other trains. For early trains, this were trains that had a clear outbound route at the scheduled departure time. For late trains, a clear outbound route was required 30 seconds after the arrival. Due to planned transfers being rare at the station, the impact of transfers was not considered. In total, 33 cases were examined, which were train lines divided per direction and into early and late arrivals. Parameters of distributions are determined by using maximum likelihood estimation, followed by removing some outliers that improve performance on a test set and additional fine-tuning. The fit of every distribution is assessed using the Kolmogorov-Smirnov test. The Weibull distribution is found to be most suitable to fit free dwell times. It perform best in 15 out of 33 cases and second-best in 15 more cases (Yuan, 2006).

We will now discuss the parameters found in Yuan (2006). For early trains, the minimum free dwell time is equal to the scheduled dwell time. This equates to a shift parameter of 60 for most trains, although some train lines also have 120 seconds. In Figure 4.1, the cumulative distribution functions are plotted for the domestic trains lines which have a planned dwell time of sixty seconds. The average scale parameter of these cases is 26, while the average shape parameter is 1.28. The distribution for the average parameters is also added to Figure 4.1. The median dwell time is equal to 80 seconds. For late arriving trains, trains can dwell shorter than the scheduled time to compensate a delayed arrival. Therefore, the shift parameter is fitted for late trains. Fourteen domestic train lines have been investigated, which are depicted in Figure 4.2. The shift parameter varies from 20 to 37, with an average of 28. The average scale parameter is 49, while the average shape parameter is 2.0. This results in a median dwell time of 69 seconds. The distribution for the average parameters is added to Figure 4.2 too.

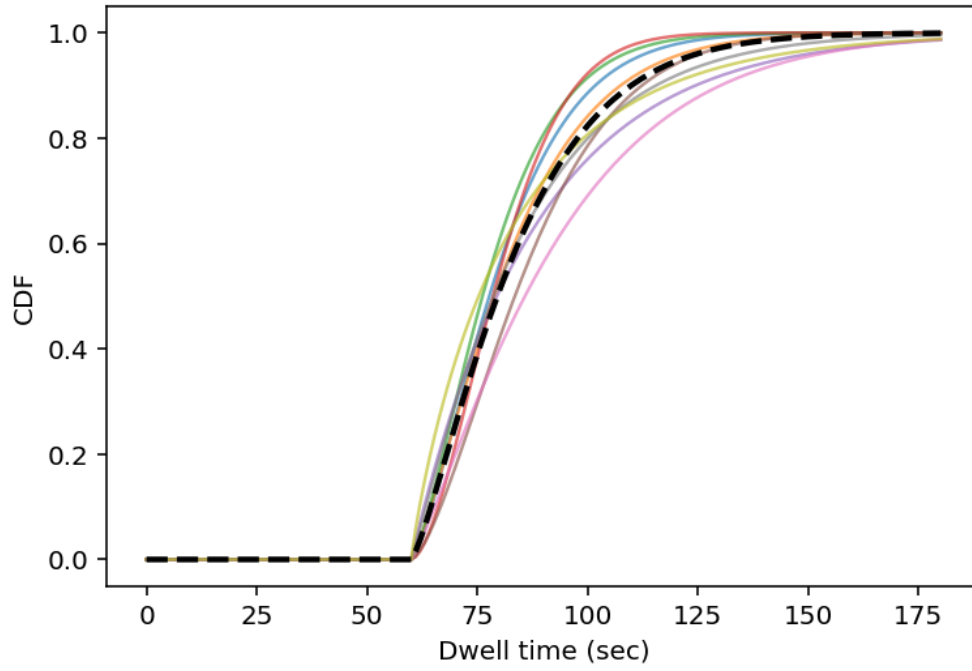


**Figure 4.1:** Cumulative distribution functions of Weibull distributions for the free dwell times of early arriving trains at The Hague Hollands Spoor station. Each of the nine continuous lines display a distribution fitted for one train line in Yuan (2006). The black dashed line displays the distribution found when using the average parameters of the nine train lines.

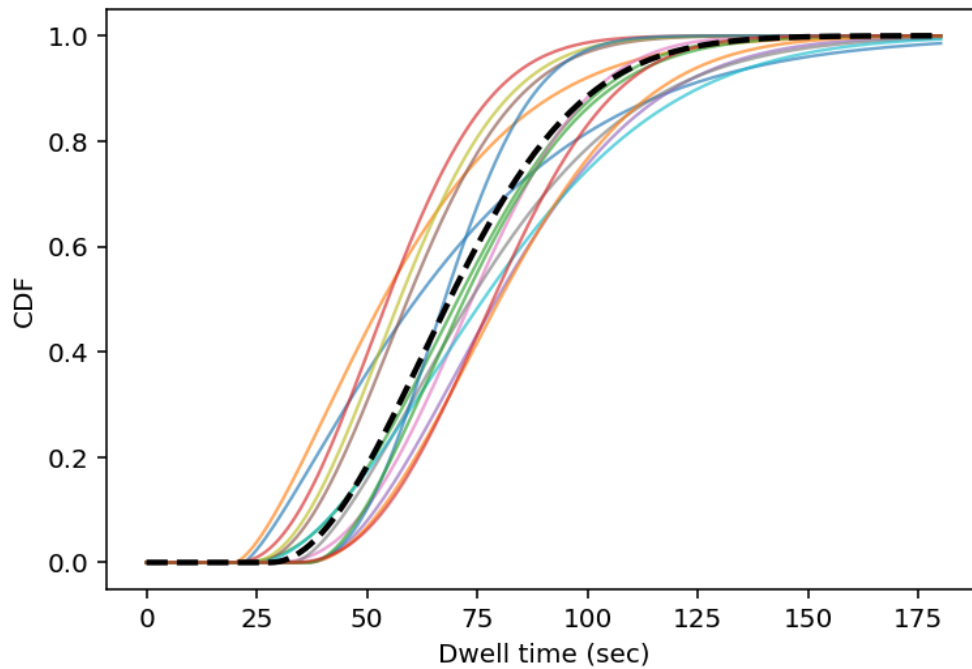


**Figure 4.2:** Cumulative distribution functions of Weibull distributions for the free dwell times of late arriving trains. Each of the fourteen continuous lines display a distribution fitted for one train line in Yuan (2006). The black dashed line displays the distribution found when using the average parameters of the fourteen train lines.

To sample free dwell times, we will use the average parameters for late arriving trains. Based on Figure 4.2 this is a reasonable approximation based on all train lines. We will sample the dwell times from the distribution for late arriving trains. Naturally, a train still must wait for its scheduled dwell time if the sampled free dwell time would allow for an early departure. The downside of this approach is that it overestimates the number of trains which depart exactly on time. But we do not want to sample a different dwell time for early and late arrivals, since this would not give a fair comparison between generated timetables. The downside of using the parameters as fitted in Yuan (2006), is that it is based on data gathered in 1999. More recently, dwell times have been determined based on data gathered at Utrecht Centraal in 2008. However, full dwell times were used instead of free dwell times (see Remark 4.3.1). Therefore, these distributions were less suitable for this thesis.

**Remark 4.3.1.** In Corman et al. (2011), dwell times are determined based on a dataset of 33000 events recorded at Utrecht Centraal in April 2008. A Weibull distribution is fitted with scale 252, shape 2.1 and shift 4 for the dwell time in seconds. This means, that the median dwell time is 215 seconds and 5.7% of dwelling events take more than 7 minutes. The authors do not mention only considering trains with a conflict free outbound path, so likely the total dwell time is measured. Furthermore, at Utrecht Centraal, some trains switch direction. This requires extra time for the train driver to be swapped or walk to the other end. This also causes the dwell times to be higher than at most stations.

The research of Yuan (2006) concluded that the Weibull distribution was most suitable for sampling dwell times based on the station The Hague Hollands Spoor. This is a basic station, so it is not guaranteed that the Weibull distribution performs best for short stops as well.

D. Li et al. (2014) investigated dwell times on three consecutive short stops on the Utrecht-Den Bosch railway corridor: Houten, Houten Castellum and Culemborg. Over 17000 trains are analysed that travelled from September to November 2012. The dwell times approximately obey a log-normal distribution. Different parameters are fitted for every short stop, they are given in Table 4.1.

**Table 4.1:** Mean and standard deviation (sd) of the dwell times of three different short stops on the Utrecht-Den Bosch corridor. The columns  $\mu$ -estimate and  $\sigma$ -estimate give the fitted parameters of the log-normal distribution (D. Li et al., 2014).

| Short stop       | Mean  | sd    | $\mu$ -estimate | $\sigma$ -estimate |
|------------------|-------|-------|-----------------|--------------------|
| Houten           | 42.05 | 9.79  | 3.71            | 0.23               |
| Houten Castellum | 34.51 | 8.10  | 3.51            | 0.23               |
| Culemborg        | 40.18 | 12.84 | 3.64            | 0.33               |

In D. Li et al. (2014), it is also investigated what influences the dwell times. It is found that there is only a weak relationship between the dwell time of a train at two consecutive short stops. The same holds for two consecutive trains stopping at a short stop and consecutive trains of the same line stopping at a short stop. This justifies sampling dwell times independently. However, dwell times are influenced by train lengths, weekday and time of day (D. Li et al., 2014). Since train lengths are not always known in advance, we cannot incorporate this into the model. A box-plot of dwell times per hour is given. This is differentiated for Monday until Friday (shown in Figure 4.3), Saturday and Sunday. We could set a different mean for every hour of the day. For basic and main stations, we do not use different distributions throughout the day. We could adopt the same approach for short stops, but this would neglect a lot of differences. Additionally, using different mean values across the day allows us to evaluate the effect of more diverse delay scenarios.

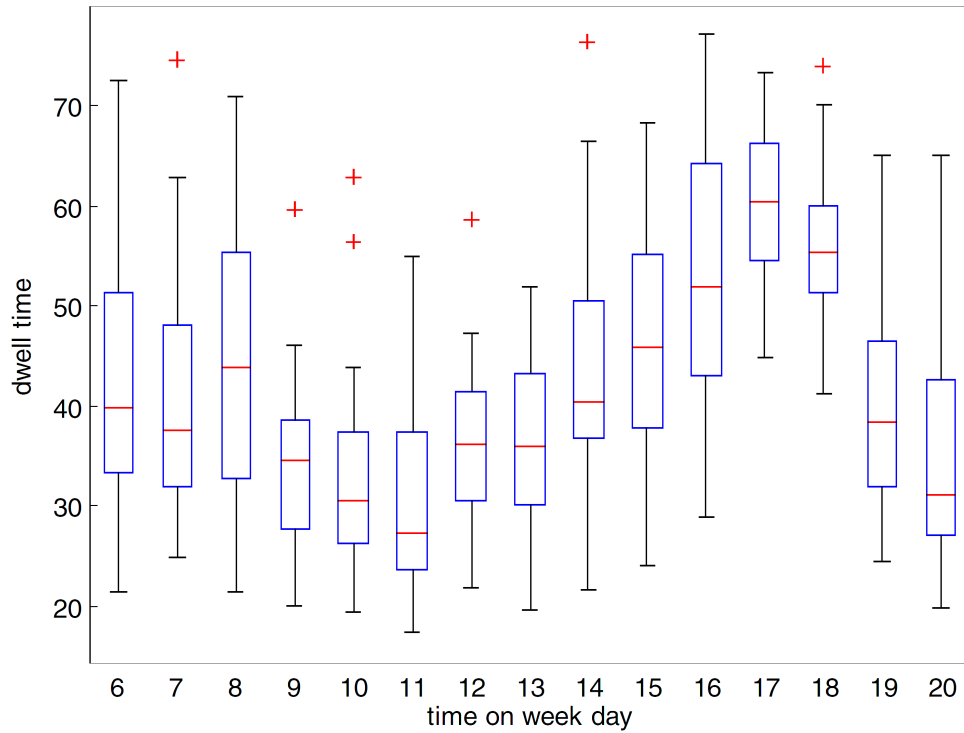


**Figure 4.3:** Box-plot of dwell times across the day on a weekday (Monday-Friday), figure is adapted from D. Li et al. (2014).

**Table 4.2:** Median dwell times across the day on a weekday (Monday-Friday) (D. Li et al., 2014).

| Hour | Median (s) |
|------|------------|
| 6    | 40         |
| 7    | 38         |
| 8    | 44         |
| 9    | 34         |
| 10   | 30         |
| 11   | 27         |
| 12   | 36         |
| 13   | 36         |
| 14   | 40         |
| 15   | 46         |
| 16   | 52         |
| 17   | 60         |
| 18   | 55         |
| 19   | 38         |
| 20   | 31         |

The different median-values that we will use can be found in Table 4.2, these values were obtained from Figure 4.3. We use the probability distributions for weekdays, since we will also work with a timetable for weekdays. The value for hour 6 will be used between 6:00 and 7:00 and so on. After 21:00, we will also use the value for hour 20. For determining which trains falls under which time of day, we will make use of the original timetable. This is done since we do not want to resample a delay if a schedule time is adjusted to a different hour.

The median  $m_i$  of a lognormal  $(\mu_i, \sigma_i)$ -distribution is equal to  $e^{\mu_i}$ . Therefore, we will set  $\mu_i = \ln(m_i)$ . This is not equal to the maximum likelihood estimator, which for a sample  $x_1, \dots, x_N$  is equal to  $\hat{\mu}_i = \frac{1}{N} \sum_k \ln(x_k)$  (Pavlovic, 2022). But since we do not have the full sample available, using  $\mu_i = \ln(m_i)$  is our best option. The maximum likelihood estimator for  $\sigma_i$  (Pavlovic, 2022) is equal to

$$\hat{\sigma}_i = \frac{1}{N} \sum_{k=1}^N (\ln(x_k) - \hat{\mu}_i)^2.$$

We cannot calculate this, because we do not have the dataset available. We do know which value was obtained for the three investigated stations (see Figure 4.1). The parameter  $\sigma_i = 0.23$  was found for two out of the three investigated stations. We will use this parameter for the entire day. This is a rough estimation of the variance. But we consider this as a more realistic approach than neglecting all information from D. Li et al. (2014) and using a single Weibull distribution. It should be noted that the different results for three stations suggest that the dwell time distribution likely differs per short stop. Further research on distinguishing station types and finding better parameters based on real data is therefore recommended.

Since we do not modify the order of trains, we do not want to have extreme delay scenarios. Therefore, we will resample free dwell times at main and basic stations if they are higher than 240 seconds. This should generally be enough time for passengers to enter or exit the train. The used probability distribution already guarantees that the free dwell time is at least 28 seconds. For sprinter rolling stock, minimum dwell times can vary between 18 and 30 seconds. Therefore, we find a minimum dwell time of 28 seconds acceptable. However, intercity rolling stock usually has a minimum dwell time of 42 seconds (Goverde, 2005). Therefore, for intercity trains we will also resample dwell times below 42 seconds. The discussed probability distribution was determined at The Hague Hollands Spoor station, which is a basic station. To take into account the extra passengers, at main stations we will resample all dwell times below 88 seconds.

For sprinter trains at short stops, we will impose a maximum dwell time of 80 seconds based on Figure 4.3. Furthermore, we will set a minimum dwell time of 24 seconds. This value was used in Quaglietta et al. (2013) and is in line with the minimum dwell time of 18-30 seconds. For intercity trains, which sometimes stop at short stops too, we adjust the minimum dwell time to 42 seconds. Also the maximum dwell time is adjusted to 98 seconds. So intercity trains use a probability distribution that has been shifted by 18 seconds compared to sprinter trains.

In conclusion, we will differentiate based on station type. For main and basic stations, we will use the Weibull distributions from Yuan (2006). For short stops we will use log-normal distributions based on D. Li et al. (2014). For the short stops we will use the same variance across the entire day, but differentiate the mean for every hour of the day. There is also a minimum and maximum possible dwell time.

## 4.4 Entrance delays

It is possible that trains travel part of their route before entering the considered network. This can lead to trains already being delayed when they arrive at the first station in the network. We need to find distributions to sample these entrance delays.

Yuan (2006) has tested different probability distributions to fit the arrival delay of trains, once again these distributions are uniform, normal, log-normal, gamma, beta, exponential and Weibull. Both the arrival time at the signal before The Hague Hollands Spoor station and the arrival time at the actual platform was investigated. In a majority of cases, the lognormal distribution performed best. If only non-negative arrivals at the platform track are considered, then the Weibull distribution performs best in most cases.

Corman et al. (2011) determined probability distributions for entrance delays based on data gathered at Utrecht Centraal during April 2008. They created different Weibull distributions for intercities, sprinters, international trains and freight trains. This led to the parameters from Table 4.3. The parameters have been fitted with a statistical procedure from Yuan (2006). The entrance delays can also be negative, so perhaps it would have been better to use a log-normal distribution instead.

**Table 4.3:** Parameters and properties of the Weibull distributions fitted in Corman et al. (2011) for the entrance delay in seconds of several train types.

| Train type    | Scale | Shape | Shift | Median | $\leq 0$ s | $> 360$ s  |
|---------------|-------|-------|-------|--------|------------|------------|
| Sprinter      | 235   | 3.0   | -186  | 22     | 39.1%      | $< 0.01\%$ |
| Intercity     | 395   | 2.2   | -315  | 19     | 45.5%      | 3.9%       |
| International | 560   | 1.4   | -205  | 226    | 21.7%      | 36.3%      |
| Freight       | 1099  | 2.6   | -885  | 70     | 43.4%      | 25.1%      |

**Remark 4.4.1.** In Corman et al. (2011), also the train type ‘stop train’ has been investigated. Stop trains have a distribution similar to the sprinter. Since NS meanwhile classifies all frequently stopping trains as sprinters, we have omitted stop train from this table.

We will use Weibull distributions with the parameters provided in Table 4.3 to create entrance delays. However, if the entrance delay exceeds 360 seconds, we will resample it. Larger delays are definitely possible, especially for international and freight trains, but then it would be common to re-order a train. Sometimes a train would also be cancelled in practice, if the train is almost caught up by a train going to the same destination. But we will not cancel any trains, just limit the delay instead. We have chosen a threshold of 360 seconds. If two trains are planned at minimum headway, this allows giving priority to the second train without hindering the delayed train. If a negative entrance delay is obtained, then we will adjust it to 0 seconds. So we assume that a train cannot enter the network before its scheduled time, but instead try to find a more energy-efficient speed profile. There are also trains that start their journey from a station within the network, which are less likely to be delayed. The entrance delay is always added to the arrival time, so by setting the arrival time at the initial station a bit earlier we can reduce the entrance delay for such trains.

**Remark 4.4.2.** Since we do not modify the order of trains, we also do not want to create heavier delay scenarios. Which is the reason, we do not use the heavier delay scenarios determined in Corman and Quaglietta (2015) and Corman et al. (2012).

We will sample the entrance delays independently. In practice, it can happen that multiple trains are hindered by the same disruption or disturbance. This could e.g. be a speed limit on a certain track section due to an animal being spotted next to the track. For real-time delay prediction, the dependence between trains using the same infrastructure can be incorporated into models. Corman and Kecman (2018) used this information in a Bayesian network for predicting delay.

However, we are not working in real-time. So instead of looking at the current disturbances, we would need to take into account all disturbances that might happen. Although it would be interesting to look at dependencies in future research, we will stick to sampling independently. The entrance delays are also sampled from the dwell time delays. D. Li et al. (2014) found that dwell time delays at short stops are not influenced by the delay at the previous station. For stations, the dwell time is influenced by the delay. This is partially caused by trains not being allowed to depart from a station before its scheduled departure time. Therefore, a train arriving on time often waits longer at a station until the scheduled departure time. Meanwhile, delayed trains only need to wait until all passengers are on board and the outbound route is clear (Hansen et al., 2010).

## Chapter 5

# Random search metaheuristics

In this chapter, we are going to describe several random search metaheuristics. These random search metaheuristics will be used to optimise supplement-variables ( $\varepsilon_{i,j}$ ), such that a more robust timetable is found. The chapter starts with an overview of metaheuristics in Section 5.1. Then, in Section 5.2, it is discussed how an initial solution set can be generated. Subsequently, Sections 5.3 to 5.7 each describe a metaheuristic. Finally, in Section 5.8, we will summarise the advantages and disadvantages of the considered metaheuristics.

### 5.1 Overview of metaheuristics

Metaheuristics are algorithms which are applicable to a large variety of optimisation problems and which try to efficiently explore a search space to find near-optimal solutions. This is different than heuristics, which have been tailored to a specific optimisation problem. Metaheuristics are often used for fast problem solving, solving large problems or creating robust solutions. Metaheuristics can be divided into single solution-based algorithms and population-based algorithms (Talbi, 2009). The metaheuristics discussed in this thesis all utilise randomness and are therefore random search metaheuristics (Zabinsky, 2011).

A single solution-based algorithm starts with one initial solution. In each step, a new solution is selected based on the current solution until the algorithm is terminated. Well-known single solution-based algorithms are simulated annealing and tabu search (Pant et al., 2020). In Section 5.3, we will discuss simulated annealing. Here, we will also discuss the applicability of tabu search for timetable optimisation.

Population-based algorithms work with a set of solutions, which is called the population. In each generation, the population is adjusted. There are two kinds of population-based algorithms: swarm intelligence and evolutionary algorithms. Swarm intelligence algorithms are inspired by social behaviour observed in animal populations. A popular swarm intelligence algorithm is particle swarm optimisation, which will be discussed in Section 5.6. Evolutionary algorithms are inspired by the evolution of species. This includes genetic algorithms and differential evolution (Pant et al., 2020), which will be discussed in Sections 5.4 and 5.5 respectively.

For all metaheuristics, a stopping criterion needs to be selected. It is possible to stop a metaheuristic after a fixed number of steps or generations has been completed. It is also possible to terminate the algorithm if the best solution found has not changed for a certain number of steps or generations (or the objective value of the best solution has not changed by much). Lastly, it is possible to stop after a certain computation time has passed.

## 5.2 Initial solution generation

Using a metaheuristic requires defining an initial population or an initial solution in the case of single solution methods. We will have a dataset available, consisting of the timetable which was used in the Netherlands during 2025 (more information on this will follow in Chapter 6). We will refer to this as the original timetable, which we will denote as  $x^{\text{orig}}$ . We can determine the time supplements  $\varepsilon_{i,j}^{\text{orig}}$ , that have been used in the original timetable. In this chapter, we will refer to solutions mainly using the letters  $x$  and  $y$ . The variables of a solution  $x$  that can be adapted are the  $\varepsilon_{i,j}$ -variables, which determine all  $x_{i,j,k}$ -variables which form a timetable.

The original solution can be used as the initial solution for single-solution methods, as long as it is feasible. For population-based methods, additional solutions can be generated based on the original timetable. We will use the procedure from Algorithm 4 to create a new solution  $x'$  based on  $x^{\text{orig}}$ . During this procedure we can get negative  $\varepsilon'_{i,j}$ -variables. This would violate the minimum time required between two scheduled events. It is also possible that the total amount of supplement planned for a train service exceeds  $s_i$ . Therefore, in Algorithm 4 we use a repair procedure, which is given in Algorithm 5. This repair procedure adjusts negative  $\varepsilon_{i,j}$  back to 0 and also makes sure that the total amount of supplement used for each train service  $i$  is equal to  $s_i$ . Applying the repair procedure does not guarantee that a feasible solution is obtained. But if the created solution  $x'$  is feasible, we can add it to the initial solution set  $X_0$ . The procedure from Algorithm 4 can be applied on  $x^{\text{orig}}$ , until the desired number of solutions is collected to fill  $X_0$ .

For Algorithm 4, a multiplicative constant  $c^{\text{mult}} \in [0, 1]$  and additive constant  $c^{\text{add}} \geq 0$  need to be set. These need to be chosen such that a diverse population is created. For this purpose, large modifications to the original timetable are preferred. However, large modifications are also expected to lead to fewer feasible solutions. So more solutions would need to be generated until the initial population  $X_0$  is complete. It will need to be tested how to set the parameters  $c^{\text{mult}}$  and  $c^{\text{add}}$ , such that solutions are created that are both diverse and frequently feasible.

**Algorithm 4:** Solution generation. Naturally, it is possible to adjust this procedure such that  $Z^{\text{add}}$  and  $Z^{\text{mult}}$  come from a different probability distribution.

**Input** : Timetable  $x$  consisting of variables  $\varepsilon_{i,j}$  (for each train service  $i$  and  $j \in J_i$ )  
 Supplement  $s_i$  for each train service  $i$   
 Multiplicative constant  $c^{\text{mult}}$   
 Additive constant  $c^{\text{add}}$   
**Output:** Candidate timetable  $x'$

- 1 **for each**  $\varepsilon_{i,j}$  **do**
- 2     Sample  $Z^{\text{mult}} \sim \text{Unif}(1 - c^{\text{mult}}, 1 + c^{\text{mult}})$
- 3     Sample  $Z^{\text{add}} \sim \text{Unif}(-c^{\text{add}}, c^{\text{add}})$
- 4      $\varepsilon'_{i,j} \leftarrow Z^{\text{mult}} \cdot \varepsilon_{i,j} + Z^{\text{add}}$
- 5 Apply repair procedure from Algorithm 5 on  $x'$
- 6 **return**  $x'$

**Algorithm 5:** Repair procedure. Lines 5-8 increase supplement for a train line if it is almost zero, to prevent zero division in line 10.

**Input :** Timetable  $x$  consisting of variables  $\varepsilon_{i,j}$  (for each train service  $i$  and  $j \in J_i$ )  
 Supplement  $s_i$  for each train service  $i$   
**Output:** Repaired timetable  $x$

```

1 for each  $\varepsilon_{i,j}$  do
2    $\varepsilon_{i,j} \leftarrow \max\{\varepsilon_{i,j}, 0\}$ 
3 for each train service  $i$  do
4    $s'_i \leftarrow \sum_{j \in J_i} \varepsilon_{i,j}$ 
5   if  $s'_i < 10^{-6}$  then
6     for  $j \in J_i$  do
7        $\varepsilon_{i,j} \leftarrow \varepsilon_{i,j} + 1$ 
8      $s'_i \leftarrow \sum_{j \in J_i} \varepsilon_{i,j}$ 
9   for  $j \in J_i$  do
10     $\varepsilon_{i,j} \leftarrow \frac{s_i}{s'_i} \varepsilon_{i,j}$ 
11 return  $x$ 

```

If it turns out that obtaining feasible solutions is hard, then a different solution generation method is required. We could instead check what is the first time that each event  $e_{i,j,k}$  can take place, without violating a constraint. And set  $x_{i,j,k} = e_{i,j,k}$  for this scenario. This would be a timetable where each event is scheduled as early as possible. Then random supplement can be added until it is not possible to add supplement without obtaining an infeasible solution. A similar procedure starting from a compressed timetable has been used in Jovanović et al. (2017). Alternatively, the amount of margin incorporated in running arcs on top of the technical running time may be reduced. Although this margin is planned with a reason, it might not be problematic if less margin is used on a small number of running arcs. It is also possible to use a single solution metaheuristic, such that no initial solution set needs to be created. In the next section, the single solution metaheuristic simulated annealing will be discussed.

### 5.3 Simulated Annealing

The simulated annealing algorithm (SA) can be described as follows, assuming that the objective function  $\mathcal{L}(x)$  needs to be minimised (Locatelli, 2000). To simplify notation, during this chapter we denote the objective value for a timetable  $x$  as  $\mathcal{L}(x)$  instead of  $\mathcal{L}(x, \Delta_R)$ .

**Step 0** Start with an initial solution  $x_0$ . Also set the best solution  $\hat{x} \leftarrow x_0$  and the step count  $m \leftarrow 0$ .

**Step 1** Sample a candidate point  $y_{m+1}$  from a distribution  $D(\cdot; x_m)$ .

**Step 2** Determine the probability  $p_m$  that  $x_{m+1}$  will be set to  $y_{m+1}$ .

**Step 3** Sample a random variable  $Z \sim \text{Unif}(0,1)$ . If  $Z < p_m$ , set  $x_{m+1} \leftarrow y_{m+1}$ . Otherwise, set  $x_{m+1} \leftarrow x_m$ .

**Step 4** If  $\mathcal{L}(x_{m+1}) < \mathcal{L}(\hat{x})$ , set  $\hat{x} \leftarrow x_{m+1}$ .

**Step 5** If the stopping criterion has been satisfied, return  $\hat{x}$ . Otherwise, set  $m \leftarrow m + 1$  and go back to step 1.

The simulated annealing algorithm requires choosing a distribution  $D(\cdot; x_m)$ , an acceptance probability  $p_m$  and a stopping criterion. Commonly  $p_m$  is set as the Metropolis function, meaning that for some ‘temperature’  $T_m > 0$  (Locatelli, 2000),

$$p_m = \min \left\{ 1, \exp \left( \frac{\mathcal{L}(x_m) - \mathcal{L}(y_{m+1})}{T_m} \right) \right\}.$$

This acceptance probability always selects the candidate point  $y_{m+1}$  if it has a better objective value than  $x_m$ . If  $y_{m+1}$  has a worse objective value than  $x_m$ , then the acceptance probability increases if  $\mathcal{L}(y_{m+1})$  improves. The acceptance probability also increases if the variable  $T_m$  (known as the temperature) is higher. The temperature  $T_m$  is an additional variable that needs to be determined. Usually,  $T_m$  is decreased during the process such that the acceptance probability of worse solutions (ascent steps) decreases. The reason that simulated annealing has ascent steps, is that they can help the algorithm to escape from local minima. Besides the stopping criteria from Section 5.1, simulated annealing can also be stopped after  $T_m$  drops below a selected threshold.

In Locatelli (2000), it is shown that  $D(\cdot; x_m)$  and  $T_m$  can be set, such that the created sequence  $x_m$  converges in probability to the optimal solution. We do not satisfy the assumptions required for this, since a continuous search space with a finite number of global minima is required. We want to have scheduled times which are rounded to full seconds, otherwise Assumption 4.1.1(ii) does not hold. We can work with continuous time supplements  $\varepsilon_{i,j}$ , which are rounded when evaluating the objective value. But then there are infinitely many solutions which get rounded to the same global minimum.

Despite the convergence results from Locatelli (2000) not being applicable, the underlying idea remains interesting. First, consider the case that  $\mathcal{L}(x_m) - \mathcal{L}(\hat{x}) \leq \eta$  for a selected threshold  $\eta$ , so the objective value of  $x_m$  is close to the best found so far. Then a new candidate solution  $y_{m+1}$  is found by uniformly selecting a solution in the feasible part of the ball  $B(x_m, r_m)$  with centre  $x_m$  and radius  $r_m$ . The radius  $r_m$  is selected from a sequence which converges to 0. Also the temperature  $T_m$  is selected from a sequence which converges to 0, meaning that an ascent step is less likely to be selected as  $m$  increases. However, if  $\mathcal{L}(x_m) - \mathcal{L}(\hat{x}) > \eta$  instead,  $r_m$  and  $T_m$  are not taken from sequences converging to 0. Instead,  $T_m \geq \tilde{T} > 0$  and  $r_m \geq \tilde{r} > 0$ . Here  $\tilde{T}$  and  $\tilde{r}$  are constants which are bounded away from zero. So bigger steps are possible and an ascent step is more likely to be accepted.

Inspired by Locatelli (2000), we introduce the following approach. Let  $\mathbb{1}$  denote an indicator function. The temperature is set as

$$T_m = (\alpha_T)^m \cdot T_0 + T_{\text{bonus}} \cdot \mathbb{1}_{\{\mathcal{L}(x_m) - \mathcal{L}(\hat{x}) > \eta\}}. \quad (5.1)$$

Furthermore, we sample a new point  $y_{m+1}$  by applying Algorithm 4 on  $x_m$ . This requires setting constants  $c^{\text{add}}$  and  $c^{\text{mult}}$ , which during step  $m$  will be denoted as  $c_m^{\text{add}}$  and  $c_m^{\text{mult}}$ . These will be of the form

$$c_m^{\text{add}} = (\alpha_{\text{step}})^m \cdot c_0^{\text{add}} + c_{\text{bonus}}^{\text{add}} \cdot \mathbb{1}_{\{\mathcal{L}(x_m) - \mathcal{L}(\hat{x}) > \eta\}}, \quad (5.2)$$

and

$$c_m^{\text{mult}} = (\alpha_{\text{step}})^m \cdot c_0^{\text{mult}} + c_{\text{bonus}}^{\text{mult}} \cdot \mathbb{1}_{\{\mathcal{L}(x_m) - \mathcal{L}(\hat{x}) > \eta\}}. \quad (5.3)$$

The updating procedure for temperature is based on the popular geometric cooling schedule (Talbi, 2009). But a bonus term is added if the current solution performs significantly worse than the best one found so far, based on Locatelli (2000). In Locatelli (2000), solutions were generated in a ball around  $x_m$ . Based on this procedure, we will set  $c_m^{\text{mult}} = 0$ . So we sample in a hypercube with radius  $c_m^{\text{add}}$ . We use a hypercube instead of a ball, since this makes the radius more easily interpretable. Based on Locatelli (2000), the radius  $c_m^{\text{add}}$  is larger if the current objective value is significantly worse than the best one seen so far. After a solution is created, the feasibility of the solution is checked. If the solution is infeasible, we resample until a feasible solution is found (without increasing the step count  $m$ ).

Even with  $c_m^{\text{mult}} = 0$ , SA requires tuning seven parameters:  $\alpha_T$ ,  $T_0$ ,  $T_{\text{bonus}}$ ,  $\alpha_{\text{step}}$ ,  $c_0^{\text{mult}}$ ,  $c_{\text{bonus}}^{\text{mult}}$  and  $\eta$ . We also want to test a simpler random search procedure, which does require less parameters to be tuned. Therefore, we introduce the local search procedure (LS). The local search procedure does not use the Metropolis acceptance probability. Instead, only solutions with a better objective value are accepted, so

$$p_m = \mathbb{1}_{\{\mathcal{L}(y_{m+1}) < \mathcal{L}(x_m)\}}. \quad (5.4)$$

This does not require setting a temperature. Furthermore, since the previous solution is always the best seen solution, we do not need the threshold  $\eta$ . Therefore, we only need to determine  $\alpha_{\text{step}}$ ,  $c_0^{\text{add}}$  and optionally  $c_0^{\text{mult}}$ . So these are less parameters to tune, but the disadvantage of local search is that it can get stuck in local minima.

In Liebchen et al. (2008), simulated annealing is tested to modify schedule times. This is not done to improve the robustness of a timetable, instead the required number of rolling stock and passenger waiting times are used in the objective function. Simulated annealing leads to results which are described as ‘rather bad’, also when compared against genetic algorithms (which will be covered in the next section). This was based on a geometric cooling schedule, but it is not explained how the distribution  $D(\cdot; x_m)$  has been set. This explanation was excluded because the results were poor. It is possible that we can find a suitable distribution  $D(\cdot; x_m)$ , which does give good results. Therefore, we will test the performance of simulated annealing.

Simulated annealing and local search will be the only single-solution methods that are tested to improve the robustness of a timetable. We will not make use of tabu search. Tabu search requires defining a neighbourhood of solutions, which can be reached from the current solution  $x_m$ . Solutions in the neighbourhood are marked as tabu, if they have recently been visited by the algorithm. From the neighbourhood, out of the solutions which are not tabu, the solution with the best objective value is picked as  $x_{m+1}$ . This is used to avoid ending up in a cycle (Talbi, 2009). We could define the neighbourhood of a timetable in many different ways. If we choose a large neighbourhood, then it would be computationally expensive to compute the objective value of each solution in the neighbourhood. However, if a small neighbourhood is considered, then only small modifications to the timetable can be tested. Because of these drawbacks, tabu search is not studied further in this thesis.

## 5.4 Genetic Algorithms

Genetic algorithms (GA) are an example of evolutionary algorithms. Assuming that an objective function needs to be minimised, genetic algorithms work as follows.

**Initialisation** We start with an initial pool of solutions  $X_0$ , of size  $n$ . Set  $m \leftarrow 0$  and the desired population size after crossover as  $n_c$ .

**Crossover** Repeat until  $|X_m| = n_c$ :  
 Select  $x$  and  $x'$  randomly in  $X_m$ , apply a crossover procedure to obtain solutions  $y$  and  $y'$ . If  $|X_m| = n_c - 1$ , add  $y$  to  $X_m$ . Otherwise, add both  $y$  and  $y'$  to  $X_m$ .

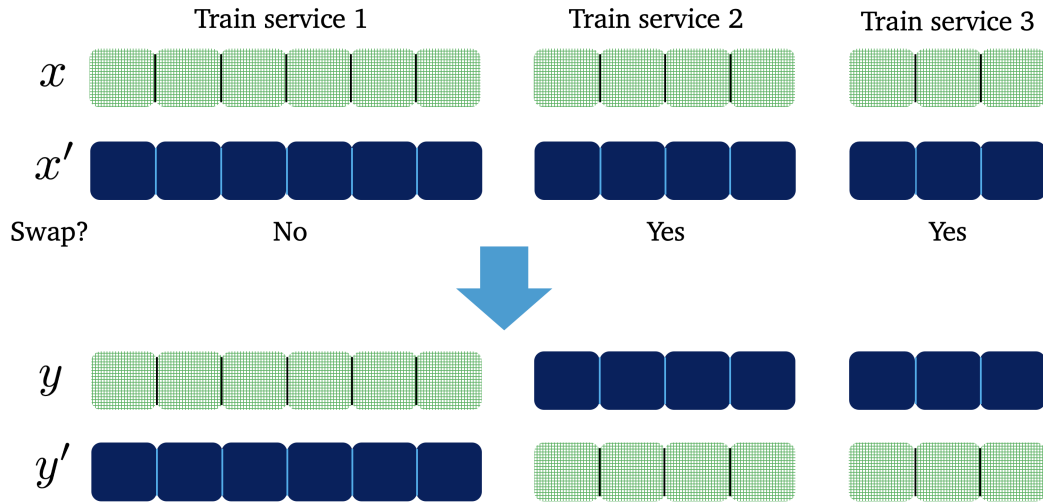
**Mutation** For each solution  $x \in X_m$ :  
 Apply a mutation procedure to create a mutated copy  $y$ , which replaces  $x$  in  $X_m$ .

**Selection** Set  $X_{m+1} \leftarrow \emptyset$ . Apply a selection procedure, which selects solutions from  $X_m$  to add to  $X_{m+1}$  until  $|X_{m+1}| = n$ .

**Evaluation** Check the stopping criterion. If this has been satisfied, return the solution in  $X_{m+1}$  with the best objective value. Otherwise, set  $m \leftarrow m + 1$  and go back to Crossover.

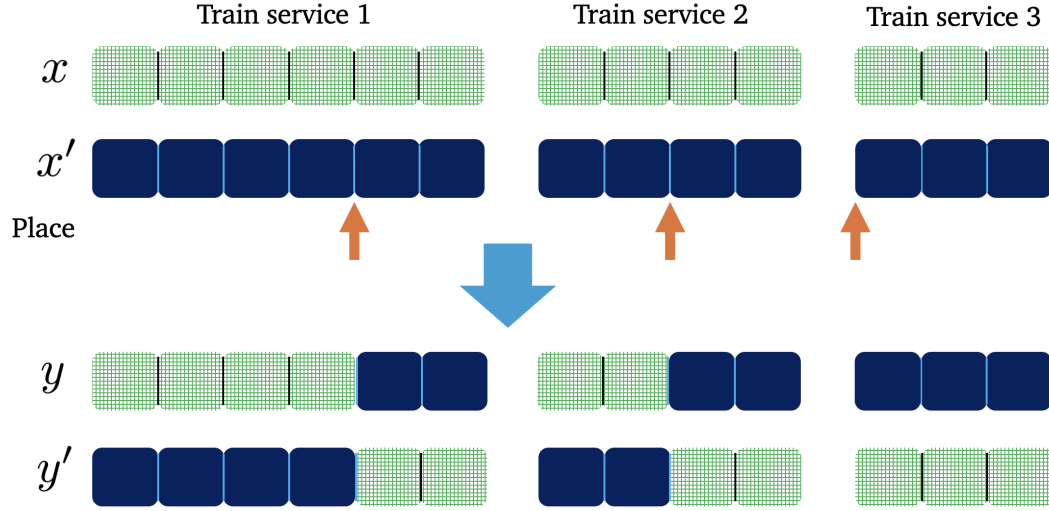
To use a genetic algorithm; a crossover, mutation and selection procedure need to be defined. Also the initial population size  $n$  and population size after crossover  $n_c$  need to be set. The design choices of the genetic algorithm also depend on the problem encoding. Most commonly, a problem is encoded with binary variables. Since we want to adjust the time supplements, we want to update supplement-values. This is known as value encoding. Value encoding does require tailored crossover and mutation procedures (Katoch et al., 2021).

Commonly used crossover techniques include uniform crossover and  $k$ -point crossover. For uniform crossover, it is decided for every variable whether the genes of the selected chromosomes need to be swapped. For time supplement optimisation, we do not expect this to work well. But we can apply uniform crossover per train (as showcased in Figure 5.1). Then, for every train it is decided whether the supplements planned for that train should be swapped between the two genes. This would require a mutation operator, which also makes sufficiently large adjustments to the allocation of supplement for each train. In  $k$ -point crossover,  $k$  points are selected to divide a solution in  $k + 1$  blocks. Alternately, blocks are selected from the first and second solution (Katoch et al., 2021). We could select a random crossover point per train service, so part of the variables are swapped for each train service. This is showcased in Figure 5.2. For both crossover procedures, the repair procedure from Algorithm 5 needs to be applied on  $y$  and  $y'$ . Infeasible solutions which are created, will not be added to  $X_m$ .



### Uniform crossover per train service

**Figure 5.1:** New solutions  $y, y'$  are created based on the solutions  $x, x' \in X_m$ . Each cell represents one variable  $\varepsilon_{i,j}$ . For each train service, there are  $|J_i|$  cells. These  $\varepsilon_{i,j}$  are placed in a block in the same order as  $q_j$  occurs in  $P_i$ . In this example, the variables are swapped for both train services 2 and 3.



### 1-point crossover for each train service

**Figure 5.2:** New solutions  $y, y'$  are created based on the solutions  $x, x' \in X_m$ . Each cell represents one variable  $\varepsilon_{i,j}$ . For each train service, there are  $|J_i|$  cells. These  $\varepsilon_{i,j}$  are placed in a block in the same order as  $q_j$  occurs in  $P_i$ . The orange arrow shows the crossover point which has been determined for each train service.

There are two mutation strategies that we want to test. Firstly, mutation can be done with the procedure in Algorithm 4. However, altering all  $\varepsilon_{i,j}$  might result in mutations which are too heavy. Instead, for each variable  $\varepsilon_{i,j}$  a mutation (multiplying with  $Z^{\text{mult}}$  and adding  $Z^{\text{add}}$ ) is only applied with a certain probability  $p_{\text{mut}}$ . We will call this mutation per variable. Secondly, we propose a procedure, which structurally increases or decreases the amount of supplement before a station for each train. We will call this switchover mutation. It works as follows and is also illustrated in Figure 5.3.

**Step 1** For each train service  $i$ , draw  $Z_{i,1}^{\text{mult}}, Z_{i,2}^{\text{mult}} \sim \text{Unif}(1 - c^{\text{mult}}, 1 + c^{\text{mult}})$ .

**Step 2** For each train service  $i$ , randomly select an index  $j' \in J_i$ .

**Step 3** For each train service, draw  $Z_i \sim \text{Unif}(0, 1)$ . If  $Z_i < p_{\text{mut}}$ :

Multiply  $\varepsilon_{i,j}$  with  $Z_{i,1}^{\text{mult}}$  for all  $j \in J_i$  such that  $q_j$  precedes  $q_{j'}$  in  $P_i$ . Multiply the remaining  $\varepsilon_{i,j}$  with  $Z_{i,2}^{\text{mult}}$ .

**Step 4** Apply the repair procedure in Algorithm 5.

For both mutation procedures, a mutated copy enters the population if it is feasible. It does not need to have a better objective value than the individual it replaces. If the created solution is bad, it will likely be discarded during the selection procedure. But if it survives, it will lead to an increased diversity. However, we do not want to ruin the best solution in a population. Therefore, we never mutate the best individual in a population. This is known as elitism (Katoch et al., 2021). The mutation probability for other solutions will need to be tested.

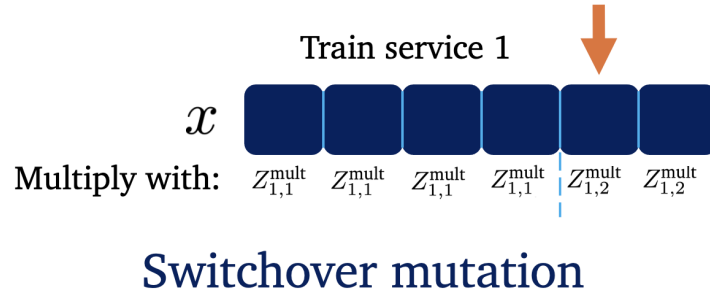


**Figure 5.3:** Switchover mutation for a train service that is mutated (which happens with probability  $p_{\text{mut}}$ ). Each cell represents one variable  $\varepsilon_{i,j}$ . These  $\varepsilon_{i,j}$  are placed in a block in the same order as  $q_j$  occurs in  $P_i$ . Here the arrow points at the variable  $\varepsilon_{i,j'}$  for the index  $j'$  selected during step 2 of the switchover mutation procedure.

For the selection procedure, we will use tournament selection. This is the most popular selection procedure for genetic algorithms. In a tournament selection, multiple solutions are randomly selected from the solution pool. The solution with the best objective value is added to  $X_{m+1}$ . This process is repeated until  $X_{m+1}$  has reached the desired population size  $n$  (Shukla et al., 2015). A population size  $n$  between 50 and 100 is commonly used (Patil & Pawar, 2015). We will always take  $n_c$  to be a multiple of  $n$ , so  $n_c = \kappa \cdot n$  for  $\kappa \in \mathbb{N}$ . We will test both  $\kappa = 2$  and  $\kappa = 5$ . After mutation, the solution pool  $X_m$  is partitioned into groups of size  $\kappa$ . Out of each group, the best solution is added to  $X_{m+1}$ . This guarantees that each solution in  $X_m$  takes part in exactly one tournament, so the best solution will always be part of  $X_{m+1}$ .

There do exist alternatives to tournament selection, see e.g. Jebari and Madiafi (2013) and Katoch et al. (2021). But most other methods require sorting the solution based on their objective value, which leads to a higher time complexity. Also the convergence rate of tournament selection outperforms alternative selection procedures (Shukla et al., 2015). This is why we will work with tournament selection.

Genetic algorithms are suitable to adjust permutations (Katoch et al., 2021). Therefore, they are commonly used in railway scheduling to determine a train order. This was the case in e.g. Chow and Pavlides (2018), Dündar and Şahin (2013), Tan et al. (2020), and Tormos et al. (2008). We want to use the genetic algorithm to adjust time supplements instead. In Liebchen et al. (2008), a genetic algorithm has been used to adjust the departure and arrival times of trains. This gives good quality solutions reasonably fast. The crossover and mutation procedures are not described. But it is mentioned, that an additional local improvement procedure is used where every schedule time is altered sequentially to minimise the objective function. The obtained solutions are of good quality and the runtime of the algorithm is also good. Furthermore, the results are not sensitive to parameter choices. Therefore, Liebchen et al. (2008) recommended using genetic algorithms to Deutsche Bahn, instead of using simulated annealing or an MILP solver.

It is possible that the crossover and mutation techniques that we have designed do not work well. In crossover, the variable-values are coming from two different solutions. It is possible that this kind of combination will not work well and yield many infeasible solutions. But since Liebchen et al. (2008) got promising results using GA, we will test the performance of this metaheuristic too.

## 5.5 Differential Evolution

We will now discuss a second evolutionary algorithm. Differential evolution (DE) uses linear combinations of solutions in the population. This is done in the following way, assuming that the objective function needs to be minimised (Zhong et al., 2012).

**Initialisation** We start with an initial pool of solutions  $X$  of size  $n$ . Let  $\hat{x}$  be the solution with the best objective function. Set  $m \leftarrow 0$ .

**Mutation** For each  $x_l \in X$ , use a mutation procedure to create a solution  $y_l$ .

**Crossover** For each  $x_l \in X$ , use a crossover procedure to create a solution  $x'_l$  which inherits part of its variables from  $y_l$  and the rest from  $x_l$ .

**Selection** For each  $x_l \in X$ :

If  $\mathcal{L}(x'_l) < \mathcal{L}(x_l)$ ,  $x'_l$  replaces  $x_l$  in  $X$ . If  $\mathcal{L}(x'_l) < \mathcal{L}(\hat{x})$  too, set  $\hat{x} \leftarrow x'_l$ .

**Evaluation** Check the stopping criterion. If this has been satisfied, return  $\hat{x}$ . Otherwise, set  $m \leftarrow m + 1$  and go back to Mutation.

Differential evolution requires choosing a mutation and crossover procedure. Also a population size  $n$  needs to be determined. Often,  $n$  is selected proportionally to the dimension  $D$ , like  $n = 5D$  (Pant et al., 2020). The mutation strategies in differential evolution, always take a linear combination of solutions in  $X$ . Some common mutation schemes can be found in Table 5.1. The rand and current-to-rand solutions perform global searches, while the best and current-to-best solutions perform a local search. The current-to-pbest method has been introduced to have a balance between local and global search (Zhong et al., 2012). The mutation operator also depends on the multiplication factor  $w$ . Often  $w$  is chosen in the interval  $[0.5, 1]$  (Pant et al., 2020). We will test  $w = 0.9$ , which is suggested in Ronkkonen et al. (2005) as a good value for avoiding premature convergence, while also limiting the computation time until convergence.

**Table 5.1:** Possible mutation strategies for differential evolution (Zhong et al., 2012). Here  $x_{r1}, \dots, x_{r5}$  are solutions, which are randomly selected from  $X$ . The best solution in  $X$  is denoted as  $\hat{x}$ , while  $x_{\text{pbest}}$  is a solution which is randomly selected out of the best  $p \cdot n$  solutions in  $X$  (for some  $p < 1$ ). The parameter  $w > 0$  is a weight.

| Name               | Formula                                                      |
|--------------------|--------------------------------------------------------------|
| Best/1             | $y_l = \hat{x} + w(x_{r1} - x_{r2})$                         |
| Rand/1             | $y_l = x_{r1} + w(x_{r2} - x_{r3})$                          |
| Rand/2             | $y_l = x_{r1} + w(x_{r2} - x_{r3}) + w(x_{r4} - x_{r5})$     |
| Current-to-rand/1  | $y_l = x_l + w(x_{r1} - x_l) + w(x_{r2} - x_{r3})$           |
| Current-to-best/1  | $y_l = x_l + w(\hat{x} - x_l) + w(x_{r1} - x_{r2})$          |
| Current-to-pbest/1 | $y_l = x_l + w(x_{\text{pbest}} - x_l) + w(x_{r1} - x_{r2})$ |

After generating a mutated copy  $y_l$ , we do always apply the repair procedure from Algorithm 5, before continuing to the crossover procedure. There are two popular crossover procedures for DE: binomial and exponential. We work with a solution  $x$  consisting of  $\varepsilon_{i,j}$ -variables. For the purpose of explaining exponential crossover, we will denote  $x$  as a  $D$ -dimensional vector consisting of elements  $x_d$  (where each  $x_d$  is one  $\varepsilon_{i,j}$ ). For exponential crossover, an integer  $Z_{l,1} \in [0, D - 1]$  is randomly generated as the starting point for crossover. Also a variable  $Z_{l,2} \in [1, D]$  is generated (independently of  $Z_{l,1}$ ), which determines the number of variables that will be taken from  $y_l$ . This crossover requires setting a crossover rate CR. Then, we can draw  $Z_{l,2} \sim \text{Geom}(1 - \text{CR})$ . If this results in  $Z_{l,2} > D$ , we set  $Z_{l,2} \leftarrow D$ .

We will denote  $a$  modulo  $D$  as  $\langle a \rangle_D$ . Exponential crossover selects  $x'_l$  as follows (Storn, 1996), as is also illustrated in Figure 5.4,

$$(x'_l)_d = \begin{cases} (y_l)_d & \text{for } d \in \{\langle Z_{l,1} \rangle_D, \langle Z_{l,1} + 1 \rangle_D, \dots, \langle Z_{l,1} + Z_{l,2} - 1 \rangle_D\}, \\ (x_l)_d & \text{for other } d \in [0, D - 1]. \end{cases}$$

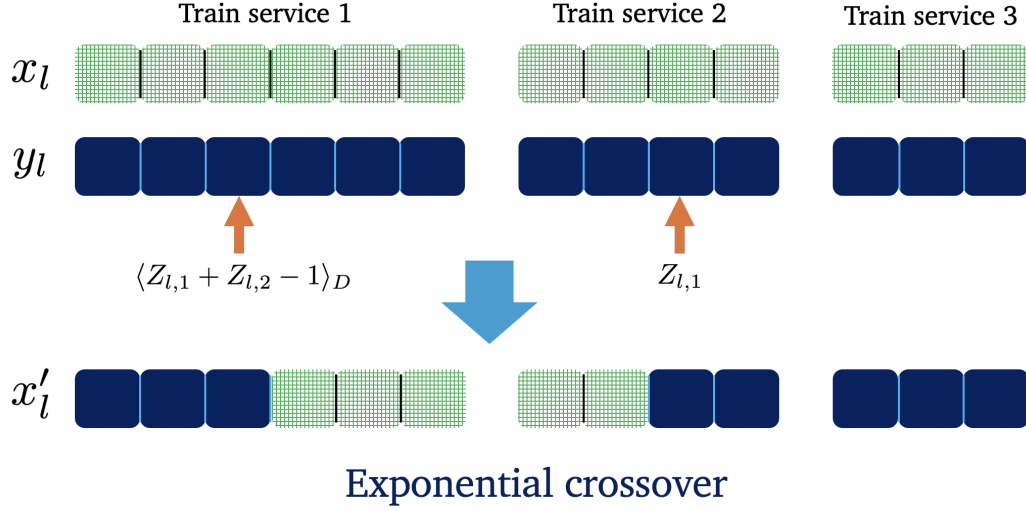


**Figure 5.4:** A new solution  $x'_l$  is created based on  $x_l$  and  $y_l$ . Each cell represents one variable  $\varepsilon_{i,j}$ . These  $\varepsilon_{i,j}$  are placed in a block in the same order as  $q_j$  occurs in  $P_i$ . For this mutation procedure, it does make a difference in which order the train lines are placed. In the depicted scenario,  $Z_{l,1} = 8$ , so we start taking variables from  $y_l$  starting at the variable with index 8 (where indexing starts at 0). We also have  $Z_{l,2} = 8$ , so 8 variables are selected from  $y_l$ .

The second crossover method that we will test is binomial crossover. We will perform binomial crossover per train service. One train service is randomly selected, for which  $x'_l$  gets the  $\varepsilon_{i,j}$  from  $y_l$ . This assures that at least one change is made. For the other train services, there is a probability of CR that all  $\varepsilon_{i,j}$  are taken from  $y_l$ . Otherwise, they are taken from  $x_l$  (Storn & Price, 1997). This is illustrated in Figure 5.5.

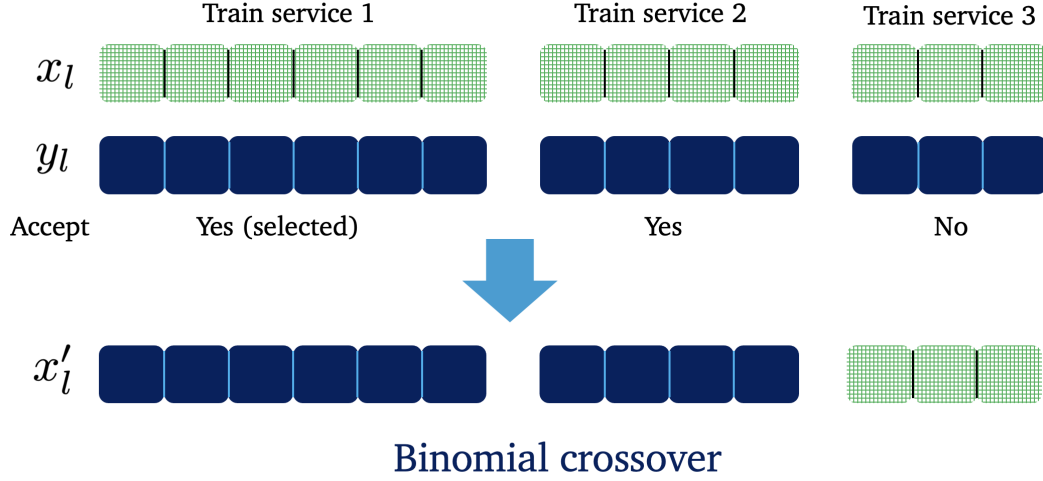


**Figure 5.5:** A new solution  $x'_l$  is created based on  $x_l$  and  $y_l$ . Each cell represents one variable  $\varepsilon_{i,j}$ . These  $\varepsilon_{i,j}$  are placed in a block in the same order as  $q_j$  occurs in  $P_i$ . In the depicted scenario, train service 1 has been randomly selected as the train service where variables are taken from  $y_l$ . For train services 2 and 3, there was a probability of CR that variables were taken from  $y_l$ .

Binomial crossover is more popular under researchers, but for some high-dimensional optimisation problems exponential crossover works better (Pant et al., 2020). Both mutation procedures require choosing a crossover rate CR. The variables that we are optimising are dependent. For this scenario,  $CR \in [0.9, 1]$  is suggested for binomial crossover by Ronkkonen et al. (2005). For exponential crossover, the crossover rate works differently and is highly dependent on the dimension  $D$ , so we will determine a value for this at a later point. After crossover, we will apply the repair procedure from Algorithm 5 on each  $x'_l$ . If a solution  $x'_l$  is feasible and has a better objective value than  $x_l$ , then  $x'_l$  will replace  $x_l$  in the population.

Differential evolution is applied in Zhong et al. (2012) to minimise the total waiting time of passengers in the Guangzhou metro network. The running time of vehicles through the network is fixed, but the start time from the first station and the frequency of each line during different times of the day can be determined. They use a dual population approach, where small modifications are made to the best known solution in one population. In the other population, larger modifications are done, to see if a better solution can be found. In total, eight evolutionary algorithms are tested, including a regular differential evolution strategy and a genetic algorithm. The double population differential evolution approach does give promising results, also when compared against the other algorithms.

We will not use the double population approach from Zhong et al. (2012). We will use regular differential evolution instead, since we rather use one solution pool to limit computation times. This has not commonly been applied for creating timetables, possibly because the population size is often large. Still, it is possible that the linear combinations used for differential evolution work well for adjusting supplements. Therefore, we will test the performance of the algorithm.

## 5.6 Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) is a swarm intelligence algorithm. There is a pool of solutions, which is called the swarm. In each generation, solutions are updated using a velocity vector. Each solution in the swarm has a separate velocity vector, which is updated based on the best version the solution has been and the best solution seen in the entire population. This is described in more detail in Appendix D.

The downside of PSO is its ability to work on constrained optimisation problems. The velocity vector can push a solution outside of the feasible region. We can accept this, but then solutions can spend a lot of time outside of the feasible region. We can also reject an update step if this leads to an infeasible solution. This would mean that a solution is stuck in its place until the velocity vector has changed enough such that it leads to a feasible solution. So we can not simply discard infeasible solutions, as was possible for the previously discussed metaheuristics. Research has been done on applying PSO on constrained optimisation problems, this is described in Appendix D. However, we do see PSO as a less promising method due to it being challenging to maintain feasible solutions. We will not test out the performance of PSO or a different swarm intelligence method. Instead, we give priority to the previously discussed metaheuristics.

## 5.7 Convex Combinations

We have discussed multiple well-known random search metaheuristics in the previous sections. These metaheuristics could all run into the problem that finding feasible solutions is hard, which can slow down the algorithm. It would be great to have a metaheuristic which mostly preserves feasibility. Therefore, we introduce an evolutionary algorithm that uses convex combinations. We will abbreviate this algorithm as CC.

We will use the mutation formulas from differential evolution as inspiration. Using these formulas can easily lead to  $\varepsilon_{i,j}$  becoming negative or the total amount of supplement for a train to be exceeded. Naturally, the repair procedure in Algorithm 5 can be used to correct this. But this does mean the original linear combination is distorted. This would not be the case if we would use convex combinations of solutions in the population. Then,  $\varepsilon_{i,j}$  can never become negative. If we do not apply a crossover procedure, then also the amount of supplement used per train will be as intended. It would be interesting to see if this gives more feasible solutions.

The combination algorithm that we have described is similar to the mutation strategy used in Nitisiri et al. (2019). Their chromosome consists of the buffer times and running time supplements between stations (which is used for every train since they only consider a single line). Here the best solution  $\hat{y}$  is used as a teacher. Then every solution  $y_i$  is used to create  $y'_i = \hat{y} + \lambda(\hat{y} - y_i)$  for  $\lambda \in [0, 1]$ . This would have been a convex combination if  $\lambda \in [-1, 0]$ . This is likely what they intended, otherwise infeasible solution could occur. Apart from this mutation operator, they also use a two-point crossover (which was explained in Section 5.4). But we do not want to add a two-point crossover operation, because it would require the repair procedure to be used.

There are limited examples of convex metaheuristics present in literature, these have not been applied to timetabling to the authors knowledge. El-Shorbagy and Abd Elazeem (2025) introduce a convex combination search. In each generation, a new population is created from the set  $S$  consisting of the 10% best solutions of the previous generation. The solutions in  $S$  are directly transferred to form 10% of the new generation. Then several different convex combination techniques on individuals in  $S$  are used to create 60% of the solutions. New solutions are generated between updated lower and upper bounds to create the remaining

30% of the new population. The algorithm outperforms genetic algorithms and particle swarm approaches on most test instances which were used in the article. Yan et al. (2011) introduce a convex differential evolution algorithm. They combine the differential evolution mutation scheme (rand/1) with convex mutations. This does outperform the standard differential algorithm on most of the tested problems.

The big downside of taking convex combinations, is that it has a limited search space. We only search in the convex polytope of the current solutions. Therefore, the method is not able to explore the entire search space. To improve the diversity of solutions in the search space, new solutions can be added in each generation (like in El-Shorbagy and Abd Elazeem, 2025). It is also possible to combine convex combining with a different metaheuristic, like Yan et al. (2011) did by combining it with differential evolution.

We will also use convex combinations on solution sets, which were found by applying another metaheuristic. It will be interesting to see if convex combinations can improve on the best solution found by the other metaheuristic. We will use the following procedure.

**Initialisation** Start with a population  $X$ , where the solution with the best objective value is denoted as  $\hat{x}$ .

**Crossover** For  $x_l \in X$ , generate  $\lambda_l \sim \text{Unif}(0, 1)$ . Set  $x'_l = \lambda_l x_l + (1 - \lambda_l) \hat{x}$ . If  $x'_l$  is feasible and  $\mathcal{L}(x'_l) < \mathcal{L}(x_l)$ , then  $x'_l$  replaces  $x_l$  in  $X$ . If also  $\mathcal{L}(x'_l) < \mathcal{L}(\hat{x})$ , set  $\hat{x} \leftarrow x'_l$ .

**Evaluation** If the stopping criterion has been satisfied, return  $\hat{x}$ . Otherwise, go back to Crossover.

## 5.8 Conclusion

During this chapter we have discussed multiple metaheuristics. Firstly, we have discussed the single-solution method simulated annealing. SA has the advantage that only one solution needs to be updated, instead of an entire population. But it does require many parameters to be tuned. Therefore, we also introduced a local search procedure, which required tuning less parameters. But since LS only accepts better solutions, it could get stuck in local minima. A different single solution method, tabu search, was rejected due to the difficulties of defining a suitable neighbourhood.

We have also discussed two evolutionary algorithms: genetic algorithms and differential evolution. GA has shown promising results in literature, but tailored crossover and mutation procedures had to be designed. It will therefore be interesting to see whether created solutions are frequently infeasible. DE does allow standard crossover and mutation procedures to be used. But limited literature is available on the performance of DE to improve timetables. We will not consider swarm intelligence algorithms, due to difficulties with respecting constraints. But we have introduced a convex combinations approach. This algorithm is expected to deliver mostly feasible solutions. This method is not suitable for exploring the entire search space, so it is best used in combination with another metaheuristic.

In Chapter 7, we will test the performance of LS, SA, GA and DE. Firstly, the model choices are tested for all three solutions. Once the best model choices have been selected for each metaheuristic, we will compare their performance. To achieve a fair comparison, each metaheuristic will have the same computation time to find a solution. Since every run of a random search metaheuristic is different, the performance will be evaluated based on multiple runs. Additionally, we will see whether applying CC on a population obtained by GA or DE can further improve the best solution found.

# Chapter 6

## Test case

In this chapter, we describe the test case that we will use. First, in Section 6.1, we will describe the network that we consider. Then, in Section 6.2, parameter choices are discussed. Lastly, the original timetable is introduced in Section 6.3.

### 6.1 Network

We will use the railway corridor from Utrecht to Arnhem as a test case, including the branch line Utrecht-Rhenen which travels partly on the Utrecht-Arnhem corridor. We will use this network as a test case, since it is heavily used. But since a large part of the corridor has only one track per direction, trains do travel close behind each other. So this can result in delay propagation.

We will first describe the train lines that run in the network, which are also depicted in Figure 6.1. In the network, there are three intercity lines from Utrecht to Arnhem. We will number these train lines as 30, 31 and 32; based on the train line numbering used by NS. Each intercity line runs twice per hour and enters Utrecht from the direction of Amsterdam. Each intercity train starts their journey in the network at Utrecht Centraal and ends it at Arnhem Centraal, which are both main stations. All intercity lines makes an intermediate stop at the basic station Ede-Wageningen, for train line 31 this is the only intermediate stop. Train line 30 also stops at Driebergen-Zeist, which is a basic station. Train line 32 stops at the short stop Veenendaal-De Klomp instead. After reaching Arnhem Centraal, lines 30 and 31 travel on to Nijmegen. But the corridor Arnhem-Nijmegen is not part of the network that we examine. Train line 32 ends its journey at Arnhem Centraal.

There are also two sprinter lines which enter the network at Utrecht Centraal: 73 and 74. Both lines reach Utrecht from the direction of Amsterdam. Lines 73 and 74 both stop at the short stops Utrecht Vaartsche Rijn and Bunnik before reaching Driebergen-Zeist. This station is the terminus for line 74. Line 73, on the other hand, continues to stop at the short stop Maarn before reaching De Haar junction. At this junction the Utrecht-Arnhem corridor is left by line 73. Subsequently, stops are made at Veenendaal West (short stop) and Veenendaal Centrum (basic station). Train line 73 ends at its terminus Rhenen. Both train lines 73 and 74 run twice an hour, but line 74 only operates during peak hours. The third sprinter line in the network is line 75, which operates between Ede-Wageningen and Arnhem. It also stops at the short stops Oosterbeek and Wolfheze. This sprinter train operates twice an hour.

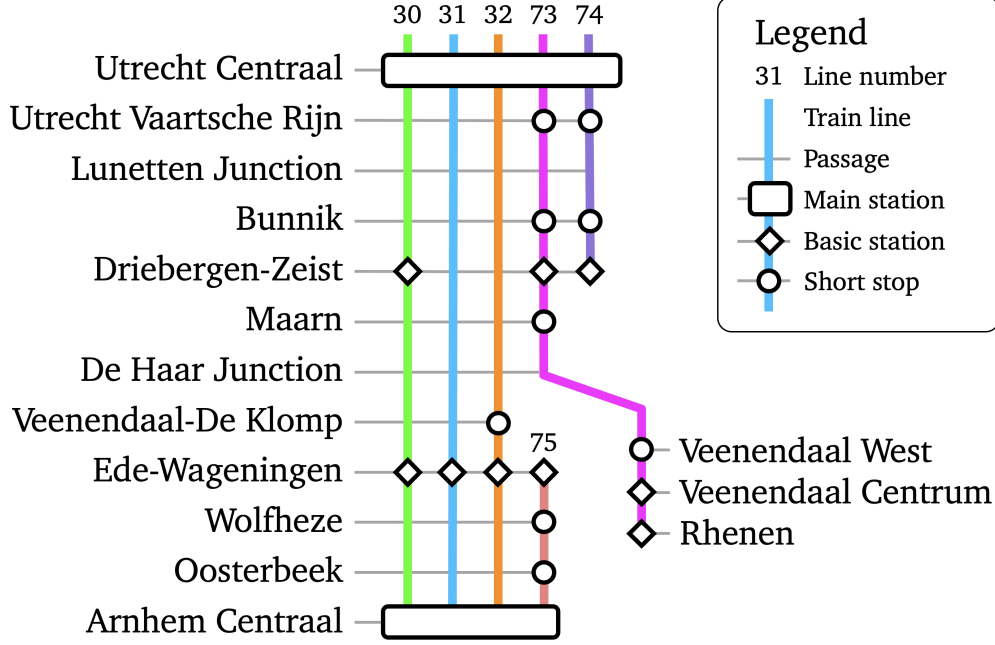


**Figure 6.1:** Line plan on the Utrecht-Arnhem network. If a train passes through a station, without stopping there, this is shown as a passage. Train lines 30, 31 and 32 are intercity lines, while 73, 74 and 75 are sprinters. Each train line is operated twice an hour.

The Utrecht-Arnhem corridor is also used by an international train to Germany ones every two hours. There is little freight traffic along the corridor. In 2025, only 150 freight trains travelled between Utrecht and Arnhem in either direction (ProRail, 2026). We would like to test the random search metaheuristics on a perfectly cyclic timetable. The international trains to Germany only run once every two hours. Furthermore, if an international train runs, the timetable of several other trains deviates from their regular schedule. Therefore, we have decided to exclude the international trains from the test case. We do also not include freight trains. But to achieve cyclicity, we have decided to let train line 74 run throughout the entire day instead of during peak hours only.

Excluding international trains does have another advantage: we can create a timetable which is cyclic every 30 minutes. The existing timetable plans two train services of the same train line exactly 30 minutes apart, except when international trains require deviations from this schedule. We will maintain this regularity in the timetable and create a timetable which is cyclic every thirty minutes. So if train services  $i$  and  $i'$  are part of the same train line, then we will use  $\varepsilon_{i,j} = \varepsilon_{i',j}$ . This reduces the number of  $\varepsilon_{i,j}$ -variables that need to be optimised to 33.

We will use repetitions of the same hourly timetable, starting at Utrecht Centraal at 6:00 (so including all trains leaving the station until 22:00). At Ede-Wageningen, all trains coming from Utrecht pass after 6:30. We have decided to also add all sprinter trains at Ede-Wageningen starting from 6:30 up until 22:30. In practice, different train lines start and end their journey at different times. But the current approach leads to the same train pattern being repeated during each hour, which is convenient for testing a cyclic timetable.

We will continue by describing the infrastructure on which these train lines operate. The infrastructure is depicted in Figure 6.2. Before entering the network, trains travel along the corridor Amsterdam-Utrecht. This corridor is a quadruple-track railway, so there are two tracks running in each direction. The sprinter lines that run from Amsterdam to Utrecht (73 and 74) use one of these tracks. This track is shared with freight trains running on this corridor. At Utrecht Centraal, sprinters use track 15, which is exclusively used by train lines

73 and 74. The second track from Amsterdam to Utrecht is reserved for intercity trains. The intercity trains travelling on to Arnhem (lines 30, 31 and 32) have to share this track with intercity trains going in the direction of Den Bosch. At Utrecht Centraal, an exclusive platform is used for the trains to Arnhem, namely platform 19.

**Remark 6.1.1.** When an international train to Germany stops at Utrecht Centraal, the platform allocation is altered for multiple other trains. But we will not take this into account, since we have excluded international trains from the test case.

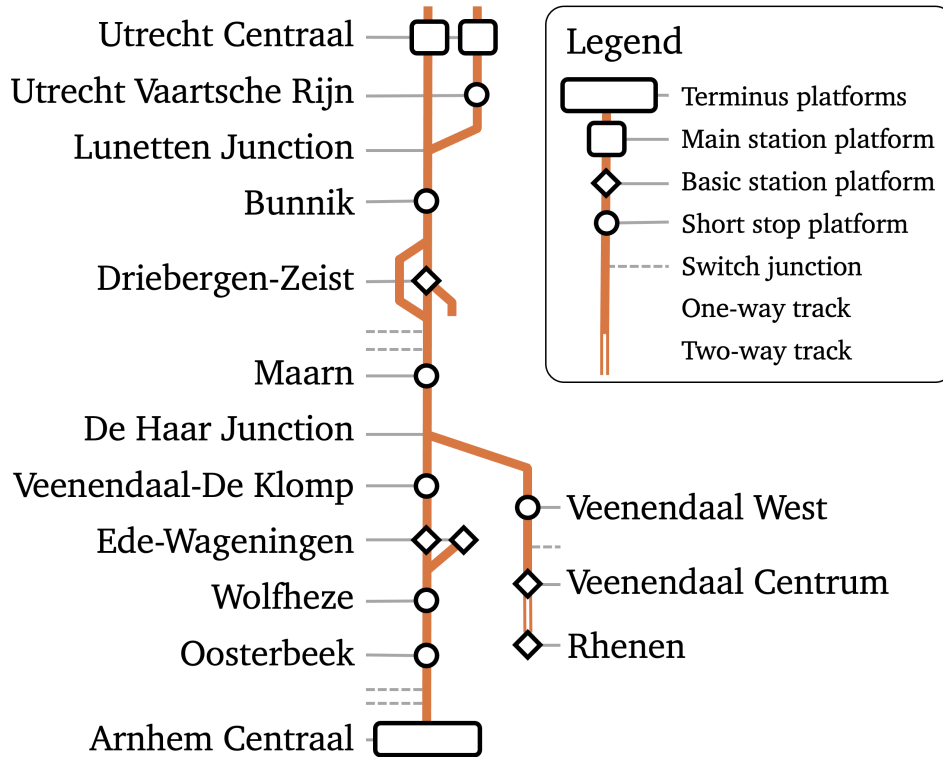


**Figure 6.2:** Infrastructure on the Utrecht-Arnhem network. One-way track is only used by trains running in one direction, while two-way track is used by trains running in both directions.

We do not have to model different train lines, which could interfere with the train travelling on the Utrecht-Arnhem corridor. When exiting Utrecht Centraal, intercity lines use a track which is not crossed by trains travelling in a different direction. So the outbound route is exclusively used by train lines 30, 31 and 32. The sprinter trains have an outbound route which has to be crossed to reach a shunting yard. Since only four sprinters an hour use the outbound route, trains can cross it most of the time without disturbing a sprinter train. Therefore, we will make the assumption that trains travelling from or to this shunting yard can be operated without disturbing the sprinter lines.

Naturally, the other trains travelling from Amsterdam to Utrecht can cause delay. But for hindered trains, a higher entrance delay at Utrecht Centraal will have been observed. So the possibility that a train is hindered up to the initial station by a train travelling towards another direction, is already reflected in the probability distribution for the entrance delays. Although it is possible that the amount of delay propagation is underestimated by the entrance delay, since the number of trains has increased after the entrance delays were calibrated in 2008.

We will now describe the track layout in the Utrecht-Arnhem network. After Utrecht Centraal, there is still a separate track for intercity lines and one for sprinter lines. At Utrecht Vaartsche Rijn the sprinter track has a platform. The trains will then reach Lunetten junction, where the two tracks will merge. After Lunetten junction, there is only one track per direction on the remaining corridor to Arnhem. However, at Driebergen-Zeist there is an extra track available to pass by a train that is stopping at the platform. From the platform, also a tail track can be reached, which is used by train line 74 before continuing service back towards Utrecht. At Ede-Wageningen station, all intercity lines use the same platform. But a different platform is used by sprinter line 75, which has an inbound route that is separated from the intercity lines.

Since the network ends at Arnhem Centraal, we do not model the platform allocation within Arnhem Centraal (therefore we denote this as terminus platforms in Figure 6.2). Since we do not model the departure times from Arnhem, we do not check when a platform becomes available for a new train. However, train lines 32 and 75 both have a dedicated platform. Only lines 30 and 31 share a platform. At least thirteen minutes is planned between arrivals of consecutive trains of these train lines. So the risk that a platform is unavailable is small. Although it is good to note, that lines 30 and 31 continue in opposite direction from Arnhem Centraal. Consequently, train drivers have to walk to the other end of the train, which does lead to longer dwell times than usual.

The branch line to Rhenen, which is exclusively used by line 73, starts at De Haar junction. One track per direction is available until Veenendaal Centrum. After Veenendaal Centrum, the railway becomes single track. So trains cannot continue to Rhenen before the train from the other direction has reached Veenendaal Centrum. We do not model this. This decision has been made, since we already punish delayed entrance at the final station. If a train from Rhenen holds up another train at Veenendaal Centrum, it will often have arrived late at Rhenen too. And the hindrance caused by a train continuing on delayed from Rhenen (back to Veenendaal Centrum) is already considered in the bonus term in the objective function.

There are seven locations in the network which are junctions. This includes the junctions Lunetten and De Haar. The other five junctions are places where there is a switch to another track, which is not inside a station area. These ‘switch junctions’ have been marked without a name in Figure 6.2. This includes switches to a track which is normally used for travel in the opposite direction. These switches are mainly used for re-routing trains in case of a broken track. There are also switches to (freight) shunting yards. Since these switch junctions are part of the timetable used by ProRail, we have included passing these junctions as activities.

The test case leads to a directed graph consisting of 4833 vertices and 9909 arcs. There are 3584 event vertices, 1248 schedule vertices and 1 source vertex. The number of supplement-variables that need to be optimised is equal to 33. There are six variables for line 30, four for line 31, five for line 32, ten for line 73, four for line 74 and four for line 75.

## 6.2 Parameter choices

We constructed a data set for the 2025 timetable on the railway corridor Utrecht-Arnhem. Based on this dataset, the lengths of arcs in the directed graph are chosen.

In the timetable, the technical minimal running time is described between consecutive activities for a train line. These are the fastest running times that a train is expected to achieve between two activities. In the timetable, the planned running time is generally set to 107% of the technical minimal running time. This is recommended by the UIC (International Union of Railways, Liebchen et al. (2010)). But we want to give the random search metaheuristic

more freedom to reallocate planned margin. Therefore, we will use a minimum running time of 104% of the technical minimal running time. The reason that margin is added on top of the technical minimal running time, is that it is based on average performance. Bad weather conditions, suboptimal driving or subpar train performance can lead to longer running times (Goverde & Scheepmaker, 2023). So if the technical minimal running time is planned, small delays will frequently occur. Therefore, we plan 4% margin on top of the technical minimal running time, such that small fluctuations can be absorbed. Running time supplement can be added on top of this.

For running time arcs, the arc length  $t(\cdot, \cdot)$  is set to 104% of the technical running time. However,  $t^r(\cdot, \cdot)$  is set to the technical minimal running time for running arcs. This allows a delayed train to use the 4% planned margin to catch up part of its delay. In practice, the running time can still vary due to weather, driving behaviour or rolling stock performance. But as previously explained, we do not vary running times due to lack of research on suitable probability distributions.

**Remark 6.2.1.** *In a few cases, an activity was missing for a train line in the timetable. The minimal running times were then estimated based on the schedule for other train lines. The timetable does also not give a separate arrival time at most basic stations and short stops. However, the time planned for dwelling at such stations could be estimated by looking how much time was added in the timetable on top of 107% of the technical minimum running time. For line 75, we set the arrival time at Ede-Wageningen three minutes before the planned departure time. This does influence the amount of entrance delay that can be absorbed at this station.*

We will now describe the length  $t(\cdot, \cdot)$  of dwelling arcs, which we call the nominal dwell time. At stations, one or two minutes is usually planned for dwelling (Wiggenraad, 2001). We will set the nominal dwell time at main stations to 120 seconds. This is also the shortest dwell time planned at Utrecht Centraal in the current timetable. Intercities will have a nominal dwell time of 60 seconds at both short stops and regular stations, this was approximately the dwell time observed at Driebergen-Zeist, Veenendaal-De Klomp and Ede-Wageningen. For sprinter trains we have used 30 seconds at short stops. In the timetable, dwell times usually vary between 36 and 45 seconds. We have rounded this down to 30 seconds. In Table 4.3, it can be seen that during most hours of the day, the median dwell time is higher than 30 seconds. Therefore, it is expected that supplement will be added on top of the nominal dwell time.

For sprinters, we did first set the nominal dwell time at basic stations to 60 seconds. However, a shorter dwell time was planned at Driebergen-Zeist in the original timetable. Using a nominal dwell time of 60 seconds made this timetable infeasible, therefore the nominal dwell time at this station was reduced to 42 seconds. This allows us to start the optimisation process from a feasible timetable. The short dwell time at Driebergen-Zeist in the original timetable does indicate that a further differentiation in station type for the dwell time distributions would be a good topic for future research.

For a timetabling arc  $(x_{i,j',1}, x_{i,j,1})$ ,  $t(x_{i,j',1}, x_{i,j,1}) = o_{i,j} + \varepsilon_{i,j}$ . Here,  $o_{i,j}$  is calculated by aggregating the lengths  $t(\cdot, \cdot)$  of running, through and dwelling arcs connecting  $e_{i,j',1}$  to  $e_{i,j,1}$ . In the delay propagation graph, the arc length  $t^r(\cdot, \cdot)$  for timetabling arcs are still equal to  $o_{i,j} + \varepsilon_{i,j}$ . Otherwise, we would adjust the timetable that we have set.

We also need to determine the length of headway arcs. Since 2022, ProRail has not used standard headway times to determine the timetable. Instead, the headway time is calculated separately for each segment in the network (ProRail, 2020). We have looked in the 2025 timetable, what the minimum headway times in the network are. For junctions, we will use a minimum headway time of 144 seconds (which was observed at Lunetten junction). A headway of 144 seconds is also used between station entrances or exits with a conflicting

route (as observed at the entrance of Arnhem Centraal). However, if a train runs through a station, while the previous train does stop at the station, then only 120 seconds of headway will be required between these trains. This was observed at Driebergen-Zeist. Also if a train exits a station after it stopped, while the previous train ran through the station, it can exit 120 seconds after the previous train.

At short stops, 174 seconds headway needs to be kept between consecutive station exits on the same track (as was observed at Oosterbeek). Furthermore, a train can not enter the station until 144 seconds has passed since the previous train exited the station. This does allow a train to enter the stop, do nominal dwelling and then continue their journey. Lastly, a train can not enter a station until its platform is free. We require a train to wait 144 seconds after the previous train using the same platform has exited the station, before the train can enter the station. However, at Driebergen-Zeist train line 74 ends, so we do not model the station exit of trains of this line. We have assumed, that this train leaves the station 60 seconds after its entrance. Please note that at Arnhem station, we do not check platform allocation, so here we do not assume that the previous train using the platform needs to have exited the station.

To create the bonus term in (3.17), we assumed that delays could be reduced by 30 seconds between consecutive events. Also, all delays got weight 1, which is usually true for sprinter trains. Sprinter trains travel about five minutes between consecutive stations, leading to 21 seconds of running time supplement when a 7% margin is used. By adding some additional dwell supplement, we obtained the approximation of 30 seconds reduction. We have calculated this based on the arrival events, but assume the same reduction can happen before a departure event. Intercity trains have longer running times between events, but also higher weights. We have decided not to incorporate this into the bonus term. So the same approximation is used for both intercity and sprinter trains.

## 6.3 Original timetable

We will now discuss the constructed timetable, which we will refer to as the original timetable. A time distance diagram of this timetable is given in Figure 6.3. There are three cases where trains follow at minimum headway. Firstly, train line 73 follows at minimum headway behind line 31 at Lunetten Junction. Please note that these train lines use a separated route until Lunetten junction, otherwise they would violate the minimum headway. Secondly, train line 32 runs through Driebergen-Zeist at minimum headway behind line 74. Lastly, train line 32 arrives at Arnhem Centraal at minimum headway behind line 75. Additionally, delay propagation is likely between train lines 73 and 30 at De Haar junction. Here, 96 seconds of buffer time is available on top of the minimum headway.

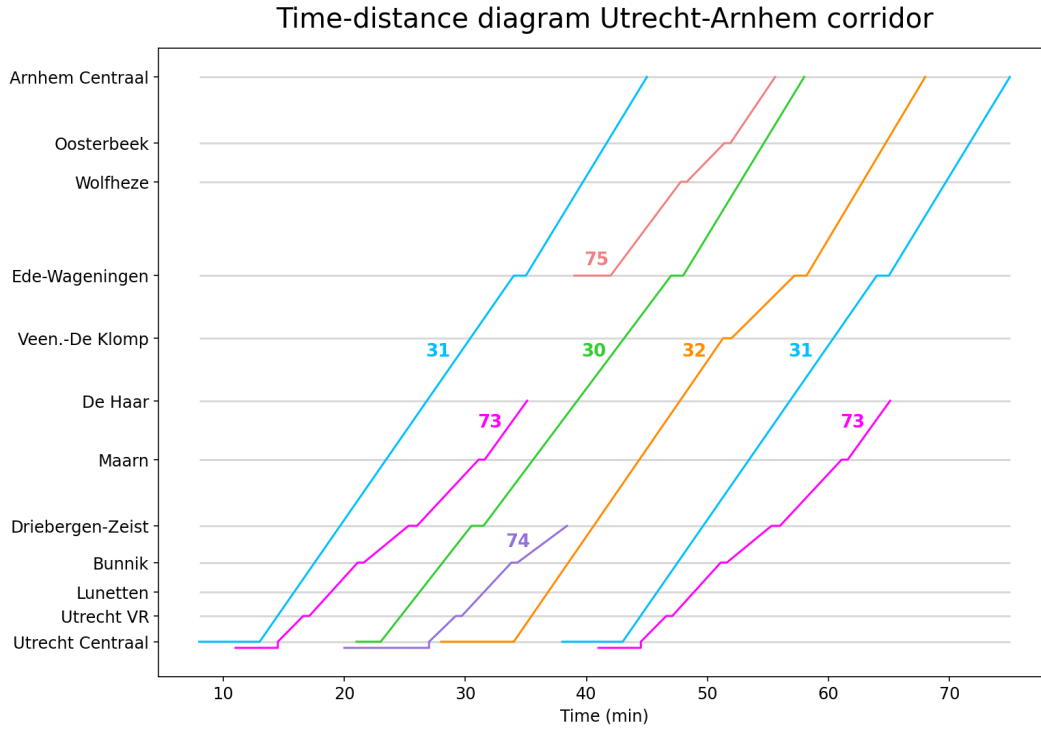


**Figure 6.3:** This is the time-distance diagram for the 2025 timetable. The scale of the vertical axis is proportional to the travel times for the international trains, which travel from Utrecht to Arnhem without an intermediate stop. Only the Utrecht-Arnhem corridor is shown, so for train line 73, the journey from De Haar junction on to Rhenen is missing. For simplicity, the departure and arrival times at stations have been connected with straight lines. To show the dwell times at Utrecht Centraal, sprinter trains are depicted slightly lower on the vertical axis at this station. Train lines 30 and 73 are depicted twice, such that the difference between the last train of a cycle and the first train of the next cycle can be observed. Line 73 only is the first train of the second cycle travelling on a track, until Lunetten junction.

The time supplements used in the original timetable are summarised in Table 6.1. Only train line 30 uses the nominal dwell time (120 seconds) at Utrecht Centraal, other trains stop here for up to five minutes longer. At basic stations, there is usually no extra supplement planned. Only at Veenendaal Centrum, does train line 73 stop for 120 instead of 60 seconds. This is likely to create some extra margin for the train in the opposite direction to come into the station. After short stops, extra time is added in the timetable, which can vary from 6 to 15 seconds. Lastly, the planned running time between stations includes approximately 7% margin on top of the technical minimal running time. Since we work with a minimum of 4% margin, the remaining 3% is part of the supplement  $s_i$ .

**Table 6.1:** Supplement available in the original timetable for each train line. The total amount of supplement available is given in the column  $s_i$ .

| Line | $s_i$ (s) | Supplement at initial station (s) |
|------|-----------|-----------------------------------|
| 30   | 88        | 0 (Utrecht Centraal)              |
| 31   | 269       | 180 (Utrecht Centraal)            |
| 32   | 285       | 240 (Utrecht Centraal)            |
| 73   | 220       | 90 (Utrecht Centraal)             |
| 74   | 350       | 300 (Utrecht Centraal)            |
| 75   | 192       | 120 (Ede-Wageningen)              |

The original timetable gives an objective value of 51669, using the objective function in (3.17). This would have been 45702 if we did not include the bonus term for network exit delays. As can be seen in Table 6.2, train line 30 is responsible for over half of the delay. This is mainly caused by train line 30 having no supplement at Utrecht Centraal, which makes it hard to recover from an entrance delay. Furthermore, line 30 passes De Haar junction with limited buffer to line 73 and arrives at Arnhem Centraal closely behind line 75. Out of the sprinter lines, line 73 has the largest contribution to the objective value. This line also has the least supplement available at the initial station and follows closely behind line 31. Additionally, line 73 has more stops than lines 74 and 75.

**Table 6.2:** Breakdown of contribution of each train line to the objective value.

| Line  | Contribution to $\mathcal{L}$ |
|-------|-------------------------------|
| 30    | 30182                         |
| 31    | 3251                          |
| 32    | 6798                          |
| 73    | 9216                          |
| 74    | 1675                          |
| 75    | 548                           |
| Total | 51669                         |

# Chapter 7

## Experiments

In this chapter, we will test the performance of several random search metaheuristics, on the test case introduced in the previous chapter. First, in Section 7.1, the setup of the experiments is explained. In the subsequent sections, design options are tested for several metaheuristics. Finally, in Section 7.7, the metaheuristics are compared.

### 7.1 Setup

For each considered metaheuristic, we will perform a grid search. For each parameter combination in the grid search, we will do three experiments. During each experiment ten minutes of computation time is used to get a solution. This stopping criterion is always tested after a generation (or step for single solution methods) has been completed. So the algorithm can still finish the generation it is working on after ten minutes. For each parameter combination, we measure the best objective value found in each of the three experiments. By slight abuse of notation, ‘Mean  $\mathcal{L}$ ’ will be used to denote the average of these three objective values. The best objective value among them will be denoted as ‘Best  $\mathcal{L}$ ’. All results are obtained using a HP ProBook 450 G10 with an Intel Core i5 1.9Ghz, 16GB Ram and an Intel Iris Xe Graphics GPU with Python 3.13.5.

For population-based methods, we will create new solutions based on the original timetable using the procedure described in Algorithm 4. We have chosen to use  $c^{\text{add}} = 20$  and  $c^{\text{mult}} = 0.25$ . Using these settings, over 30% of generated timetables are feasible. But these changes are expected to be large enough to create a diverse population. For single solution methods, we will start from the original timetable. The number of delay scenarios that we will use is 1000. This was found to give favourable computation times, while also giving meaningful results.

In Appendix C, we explain how the ideas presented in Chapter 3 were implemented. In Section C.3, we also explain a mistake that was made. This could have led to infeasible timetables being approved. However, the impact of this mistake turned out to be negligible (as explained in Section C.3). Therefore, it was decided that the gathered results for the grid searches in this chapter were still representative. The performance evaluation at the end of each section was done with a corrected implementation to rule out impact on the final results.

**Remark 7.1.1.** *During this chapter, we use a slightly different objective function than presented in (3.17). Firstly, the constant 30 in the bonus term is replaced by 60. This is an optimistic reduction of delay. Secondly, no maximum was taken over the bonus term. The bonus term could be slightly negative due to this. Optimisation in this chapter does start from an original objective value of 47537 for the original timetable. In Chapter 8, we will switch to the improved objective function, as it was already presented in (3.17).*

## 7.2 Local Search

The first meta-heuristic that we will test is the local search algorithm. There are three parameters that we will vary during the first grid search, for which an overview is given in Table 7.1. For the multiplication factor of the step size ( $\alpha_{\text{step}}$ ), we test two options. Firstly,  $\alpha_{\text{step}} = 1$  is tested, which does not decrease the step size during optimisation. Secondly, we test  $\alpha_{\text{step}} = 0.9999$ . Since the local search algorithm can complete around 25000 steps within ten minutes, this means that  $c_m^{\text{mult}}$  and  $c_m^{\text{add}}$  are reduced to 8.2% of their starting value at the end of the algorithm. We will also test five different  $c_0^{\text{mult}}$ -values and three different  $c_0^{\text{add}}$ -values.

**Table 7.1:** Design options for local search algorithm, which were tested in the first grid search.

| $\alpha_{\text{step}}$ | $c_0^{\text{mult}}$ | $c_0^{\text{add}}$ |
|------------------------|---------------------|--------------------|
| 0.9999                 | 0.00                | 5                  |
| 1                      | 0.05                | 10                 |
|                        | 0.10                | 20                 |
|                        | 0.15                |                    |
|                        | 0.25                |                    |

We will now describe the results obtained during the first grid search, which are presented in Table 7.2. The results clearly indicate that the step size should be decreased during optimisation. When the same  $c_0^{\text{mult}}$  &  $c_0^{\text{add}}$  are used, we can compare the average objective value over three experiments. With multiplication factor 1, the average objective value was always at least 140 worse than for factor 0.9999. Therefore, we have excluded the results with factor 1 from Table 7.2 and only discuss the results for factor 0.9999.

The parameter combination with the best results was  $c_0^{\text{mult}} = 0.25$  and  $c_0^{\text{add}} = 5$ . This led to both the best average objective value over three experiments and the experiment with the best objective value. It stands out that for each  $c_0^{\text{mult}}$ -value,  $c_0^{\text{add}} = 5$  always outperforms  $c_0^{\text{add}} = 10$ . Additionally,  $c_0^{\text{add}} = 10$  also outperforms  $c_0^{\text{add}} = 20$  for all  $c_0^{\text{mult}}$ -values. Furthermore, if you fix  $c_0^{\text{add}} = 5$  or  $c_0^{\text{add}} = 10$ , then the results improve when  $c_0^{\text{mult}}$  increases. For  $c_0^{\text{add}} = 20$  this is not the case. But the results suggest that the local search works best, when a large  $c_0^{\text{mult}}$  and small  $c_0^{\text{add}}$  is used.

The number of steps performed by the local search algorithm varied from 24000 to 27000. Which part of solutions was feasible, could vary greatly per run. It was possible that this varied by 20 percentage points between different experiments for the same parameter choice. That the local search algorithm only uses one solution at a time greatly influences the feasibility. However, the portion of feasible solutions was always between 22% and 61%.

**Table 7.2:** The results obtained during the first grid search of the local search algorithm. In this table, the results are given for  $\alpha_{\text{step}} = 0.9999$ . Various combinations of  $c_0^{\text{mult}}$  and  $c_0^{\text{add}}$  are tested. For each combination, in the column ‘Mean  $\mathcal{L}$ ’ the mean is taken of the obtained objective values at the end of the three experiments. The column ‘Best  $\mathcal{L}$ ’ contains the best objective value obtained among the three experiments. The best results in a column have been written in bold.

| $c_0^{\text{mult}}$ | $c_0^{\text{add}}$ | Mean $\mathcal{L}$ | Best $\mathcal{L}$ |
|---------------------|--------------------|--------------------|--------------------|
| 0.00                | 5                  | 37163              | 37101              |
| 0.00                | 10                 | 37193              | 37100              |
| 0.00                | 20                 | 37197              | 37182              |
| 0.05                | 5                  | 37149              | 37114              |
| 0.05                | 10                 | 37184              | 37155              |
| 0.05                | 20                 | 37254              | 37169              |
| 0.10                | 5                  | 37122              | 37092              |
| 0.10                | 10                 | 37150              | 37090              |
| 0.10                | 20                 | 37213              | 37138              |
| 0.15                | 5                  | 37117              | 37087              |
| 0.15                | 10                 | 37142              | 37090              |
| 0.15                | 20                 | 37204              | 37152              |
| 0.25                | 5                  | <b>37089</b>       | <b>37086</b>       |
| 0.25                | 10                 | 37107              | 37101              |
| 0.25                | 20                 | 37218              | 37156              |

Based on the results of the first grid search, we will test the parameters in Table 7.3 during a second grid search. We have added two more multiplication factors. A multiplication factor of 0.99985 will result in  $c_m^{\text{mult}}$  and  $c_m^{\text{add}}$  decreasing towards 2.4% of their starting value (for 25000 steps). For factor 0.99995, this is 28.6%. Based on  $c_0^{\text{mult}} = 0.25$  giving the best results, we will also test the options  $c_0^{\text{mult}} = 0.20$  and  $c_0^{\text{mult}} = 0.30$ . Furthermore, we will test smaller  $c_0^{\text{add}}$ -values. We do not want to test  $c_0^{\text{add}} = 0$ , since it should always be possible to change zero supplement to a higher value

**Table 7.3:** Design options for local search, which were tested in a second grid search.

| $\alpha_{\text{step}}$ | $c_0^{\text{mult}}$ | $c_0^{\text{add}}$ |
|------------------------|---------------------|--------------------|
| 0.99985                | 0.20                | 1                  |
| 0.99990                | 0.25                | 2                  |
| 0.99995                | 0.30                | 3                  |
|                        |                     | 4                  |
|                        |                     | 5                  |

An overview of all results for the second grid search can be found in Table E.1. For  $c_0^{\text{add}}$  it is noticeable that  $c_0^{\text{add}} = 1$  does not work well when it is combined with a multiplication factor of 0.99985. In this case,  $c_0^{\text{add}}$  does likely decrease too fast to obtain good exploration of the search space. No other patterns become clear of certain  $c_0^{\text{add}}$  performing better across certain parameter combinations. To get a better overview of the performance of  $\alpha_{\text{step}}$  and  $c_0^{\text{mult}}$ -combinations, we have taken the average across multiple  $c_0^{\text{add}}$ -values. For  $\alpha_{\text{step}}$ -values of 0.99990 and 0.99995, the average was taken across the five tested  $c_0^{\text{add}}$ -values. For  $\alpha_{\text{step}} = 0.99985$ , we excluded  $c_0^{\text{add}} = 1$  from the average due to its worse performance. The obtained averages can be found in the column ‘Aggregate mean’ in Table 7.4.

The aggregate means showcase that  $\alpha_{\text{step}} = 0.99995$  gives the worst result. So for this multiplication factor, the step size does not decrease quickly enough. Other combinations of  $\alpha_{\text{step}}$  and  $c_0^{\text{mult}}$  performed similar. Therefore, we will not perform a third grid search to test more options.

**Table 7.4:** Aggregated results obtained during the second grid search for local search. For a combination of  $\alpha_{\text{step}}$  and  $c_0^{\text{mult}}$ , the average is taken of the mean  $\mathcal{L}$  achieved for multiple  $c_0^{\text{add}}$ -values. For  $\alpha_{\text{step}}$  equal to 0.99990 or 0.99995,  $c_0^{\text{add}} \in \{1, 2, 3, 4, 5\}$  is used. For  $\alpha_{\text{step}} = 0.99985$ ,  $c_0^{\text{add}} = 1$  has been excluded from the aggregate mean.

| $\alpha_{\text{step}}$ | $c_0^{\text{mult}}$ | Aggregate mean |
|------------------------|---------------------|----------------|
| 0.99985                | 0.20                | 37113          |
| 0.99985                | 0.25                | 37109          |
| 0.99985                | 0.30                | 37105          |
| 0.99990                | 0.20                | 37104          |
| 0.99990                | 0.25                | 37108          |
| 0.99990                | 0.30                | 37107          |
| 0.99995                | 0.20                | 37122          |
| 0.99995                | 0.25                | 37127          |
| 0.99995                | 0.30                | 37133          |

The parameter combination that we select for local search is  $\alpha_{\text{step}} = 0.9999$ ,  $c_0^{\text{mult}} = 0.25$  and  $c_0^{\text{add}} = 3$ . This parameter combination gave the best results in terms of mean  $\mathcal{L}$  (37087), while also having the joint best objective value during a single experiment (37084). Compared to the results of the first grid search, the same  $c_0^{\text{mult}}$  and  $\alpha_{\text{step}}$  is selected. Only,  $c_0^{\text{add}} = 5$  was changed to  $c_0^{\text{add}} = 3$ . For the selected parameters, around 25200 steps can be completed during a ten-minute experiment.

The results of random search metaheuristics are inherently random. Since we have tested many parameter combinations, it is possible that the best parameter combination obtained a better mean  $\mathcal{L}$  during the experiments than is usually achieved. To get a better idea of the performance of the selected parameters, we will perform 20 new experiments for this parameter combination. The best objective value which was achieved by the local search algorithm was 37083, this value was achieved for three out of twenty experiments. The three worst objective values achieved are 37106, 37117 and 37122. The mean objective value is equal to 37096. An overview of all twenty experiments will be available in Table 7.17 in Section 7.7.

We have also kept track of the convergence during the twenty experiments, by storing the objective value every ten seconds. In Figure 7.1, we showcase the minimum, mean and maximum objective value at each time point. The objective value of an experiment is equal to the best objective value it has seen before a time point. Please note that optimisation starts from an objective value of 47537. But during all experiments the objective value is already below 37600 within the first minute. Therefore, the vertical axis has been limited to 37600 to better showcase the convergence during the remaining nine minutes. It does become clear from Figure 7.1 that the speed of convergence can differ considerably per run. The largest difference is present at achieving an objective value below 37150, this can vary between 2 and 8 minutes.

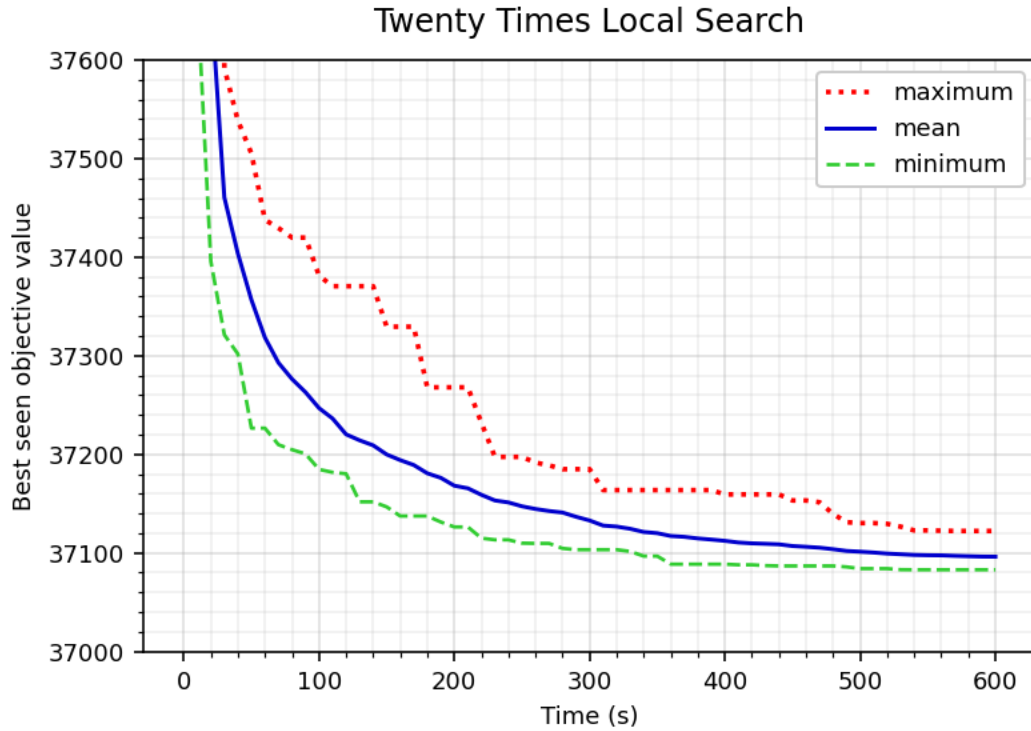


**Figure 7.1:** Convergence for 20 instances of the local search algorithm, using the parameters  $\alpha_{\text{step}} = 0.9999$ ,  $c_0^{\text{mult}} = 0.25$  and  $c_0^{\text{add}} = 3$ . Optimisation starts from an objective value of 47537, but the vertical axis is limited to 37600 to better showcase the convergence towards the end.

### 7.3 Simulated Annealing

We are now going to test the simulated annealing algorithm. Simulated annealing does work similar to the local search algorithm, which was tested in the previous section. The big difference is that SA sometimes accepts worse solutions, which can help to escape from a local minimum. That the selected step search for the local search algorithm was quite large, reduces the risk of getting stuck in local minima. Therefore, it will be interesting to see if the simulated annealing algorithm manages to outperform local search. It may well be that accepting worse solutions during optimisation leads to worse results due to unnecessary divergence.

We are now going to test design options for the simulated annealing algorithm. We will test four starting temperatures ( $T_0$ -values). The impact of the starting temperatures on acceptance probabilities is illustrated in Table 7.5. We will also test three multiplication factors for the temperature (which is denoted as  $\alpha_T$ ). For  $\alpha_T = 0.9999$  we end at approximately 8.2% of  $T_0$  (for 25000 steps). When  $\alpha_T = 0.9998$ , this is 0.7% of  $T_0$  instead. For  $\alpha_T = 0.9995$  we even end at  $4 \cdot 10^{-4}\%$  of  $T_0$ .

**Table 7.5:** The first column selects a temperature. The remaining columns describe the acceptance probabilities for that temperature, if the objective value is worse by a certain value. This value varies from 10 to 1000. If a cell is empty, the acceptance probability is below 0.1%.

| $T$ | 10    | 20    | 50    | 100   | 200   | 500  | 1000 |
|-----|-------|-------|-------|-------|-------|------|------|
| 25  | 67.0% | 44.9% | 13.5% | 1.8%  |       |      |      |
| 50  | 81.9% | 67.0% | 36.8% | 13.5% | 1.8%  |      |      |
| 100 | 90.5% | 81.9% | 60.7% | 36.8% | 13.5% | 0.7% |      |
| 200 | 95.1% | 90.5% | 77.9% | 60.7% | 36.8% | 8.2% | 0.7% |

Simulated annealing was only meant to have additive variables to the supplements (so  $c_m^{\text{mult}} = 0$ ), we will test this with  $c_0^{\text{add}} = 10$  and  $c_0^{\text{add}} = 20$ . Based on the results of the local search algorithm, we are also going to test  $c_0^{\text{add}} = 3$  combined with  $c_0^{\text{mult}} = 0.25$ . Based on the results of the local search algorithm, we also fix  $\alpha_{\text{step}}$  to 0.9999. This is used as a multiplication factor for both  $c_0^{\text{mult}}$  and  $c_0^{\text{add}}$ . During the first grid search, we will not increase temperature and step size if a threshold  $\eta$  is exceeded. We will only test this in the second grid search, to reduce the computation time required for the first grid search. An overview of the design options tested during the first grid search for SA can be found in Table 7.6.

**Table 7.6:** Design options for simulated annealing, which were tested during the first grid search.

| $T_0$ | $\alpha_T$ | $c_0^{\text{mult}}$ & $c_0^{\text{add}}$ | $\alpha_{\text{step}}$ |
|-------|------------|------------------------------------------|------------------------|
| 25    | 0.9995     | 0 & 10                                   | 0.9999                 |
| 50    | 0.9998     | 0 & 20                                   |                        |
| 100   | 0.9999     | 0.25 & 3                                 |                        |
| 200   |            |                                          |                        |

The results of the first grid search showcase that the combination  $c_0^{\text{mult}} = 0.25$  and  $c_0^{\text{add}} = 3$  performs best. Only for the parameter combination  $T_0 = 200$  with  $\alpha_T = 0.9999$  does it not give the best mean  $\mathcal{L}$  and usually there is a clear margin to the other options. Therefore, in Table 7.7 only the results for  $c_0^{\text{mult}} = 0.25$  and  $c_0^{\text{add}} = 3$  are given. Table E.2 does contain all results. There it can be seen that for  $c_0^{\text{mult}} = 0$ ,  $c_0^{\text{add}} = 10$  does outperform  $c_0^{\text{add}} = 20$ .

When we fix  $c_0^{\text{mult}} = 0.25$  and  $c_0^{\text{add}} = 3$ , for the same  $T_0$  the worst results are always obtained by  $\alpha_T = 0.9999$ . Especially when  $T_0 = 200$ , the mean  $\mathcal{L}$  is higher, which indicates that the temperature is not cooled enough. Good results are obtained for both  $\alpha_T = 0.9995$  and  $\alpha_T = 0.9998$ . There is also not an initial temperature  $T_0$  which clearly performs best. The best mean objective value is obtained using  $T_0 = 50$  and  $\alpha_T = 0.9998$ , while the single best experiment used  $T_0 = 200$  and  $\alpha_T = 0.9995$ .

**Table 7.7:** Results obtained during the first grid search for simulated annealing, when  $c_0^{\text{mult}} = 0.25$  and  $c_0^{\text{add}} = 3$ .

| $T_0$ | $\alpha_T$ | Mean $\mathcal{L}$ | Best $\mathcal{L}$ |
|-------|------------|--------------------|--------------------|
| 25    | 0.9995     | 37089              | 37084              |
| 25    | 0.9998     | 37094              | 37087              |
| 25    | 0.9999     | 37094              | 37089              |
| 50    | 0.9995     | 37096              | 37091              |
| 50    | 0.9998     | <b>37086</b>       | 37084              |
| 50    | 0.9999     | 37121              | 37112              |
| 100   | 0.9995     | 37119              | 37112              |
| 100   | 0.9998     | 37091              | 37090              |
| 100   | 0.9999     | 37149              | 37106              |
| 200   | 0.9995     | 37088              | <b>37083</b>       |
| 200   | 0.9998     | 37108              | 37105              |
| 200   | 0.9999     | 37304              | 37221              |

Based on the results of the first grid search, a second grid search is conducted with the design options in Table 7.8. We have fixed  $c_0^{\text{mult}} = 0.25$  and  $c_0^{\text{add}} = 3$ , while we still use  $\alpha_{\text{step}} = 0.9999$ . Since it performed worst in the first grid search, we have removed the option  $\alpha_T = 0.9999$ . To limit computation times, we have also removed  $T_0 = 100$ . All other starting temperatures achieved a slightly better mean  $\mathcal{L}$  for their best parameter combination. During the second grid search, we will also add the threshold  $\eta$ . We test both  $\eta = 100$  and  $\eta = 500$ . If the threshold is exceeded, the temperature is increased by either 25 or 50 (this is  $T_{\text{bonus}}$ ). We set  $c_{\text{bonus}}^{\text{add}} = 3$ , while  $c_{\text{bonus}}^{\text{mult}}$  can be either 0.05 or 0.10.

**Table 7.8:** Design options for simulated annealing, which were tested in a second grid search.

| $T_0$ | $\alpha_T$ | $\eta$ | $T_{\text{bonus}}$ | $c_0^{\text{mult}}$ & $c_0^{\text{add}}$ | $c_{\text{bonus}}^{\text{mult}}$ & $c_{\text{bonus}}^{\text{add}}$ | $\alpha_{\text{step}}$ |
|-------|------------|--------|--------------------|------------------------------------------|--------------------------------------------------------------------|------------------------|
| 25    | 0.9995     | 100    | 25                 | 0.25 & 3                                 | 0.05 & 3                                                           | 0.9999                 |
| 50    | 0.9998     | 500    | 50                 |                                          | 0.10 & 3                                                           |                        |
| 200   |            |        |                    |                                          |                                                                    |                        |

The results of the second grid search are available in Table E.3. A lot of parameter combinations perform similar. In 43 out of 48 parameter combinations, the achieved mean  $\mathcal{L}$  is below 37110. All five outliers use  $\eta = 100$ , so the bonus-term if a threshold is exceeded might reduce the probability of convergence. The best mean  $\mathcal{L}$  is obtained using  $\eta = 500$ ,  $T_0 = 25$ ,  $\alpha_T = 0.9998$ ,  $T_{\text{bonus}} = 25$  and  $c^{\text{mult}} = 0.10$ . All other parameter combinations with  $\eta = 500$  and  $T_0 = 25$  also obtain a good mean  $\mathcal{L}$  (between 37092 and 37098). This is exactly the combination where the lowest number of worse solutions is expected to be accepted, since the lowest temperature is used. This could indicate that simulated annealing does not have an added benefit compared to local search. Therefore, we will not perform a third grid search for simulated annealing.

The parameter combination which we select is  $T_0 = 25$ ,  $\alpha_T = 0.9998$ ,  $\eta = 500$ ,  $T_{\text{bonus}} = 25$  and a  $c_{\text{bonus}}^{\text{mult}}$  of 10. Around 24900 steps could be completed using these parameters during a ten-minute experiment. The parameter combination obtained a mean  $\mathcal{L}$  of 37087, which was the best mean  $\mathcal{L}$  found for the local search algorithm. The best  $\mathcal{L}$  found during a single experiment using this parameter combination was equal to 37085. There were seven parameter combinations, which achieved a lower objective value of 37083 during their best experiment. However, these combinations all had a higher mean  $\mathcal{L}$  than the selected parameter combination.

For the selected parameter combination, we have done twenty experiments. The best objective value achieved was 37083, which was seen during three out of twenty experiments. The three worst objective values are 37099, 37102 and 37147. The mean obtained objective value was equal to 37092. The convergence is showcased in Figure 7.2, where the vertical axis is again limited to 37600.

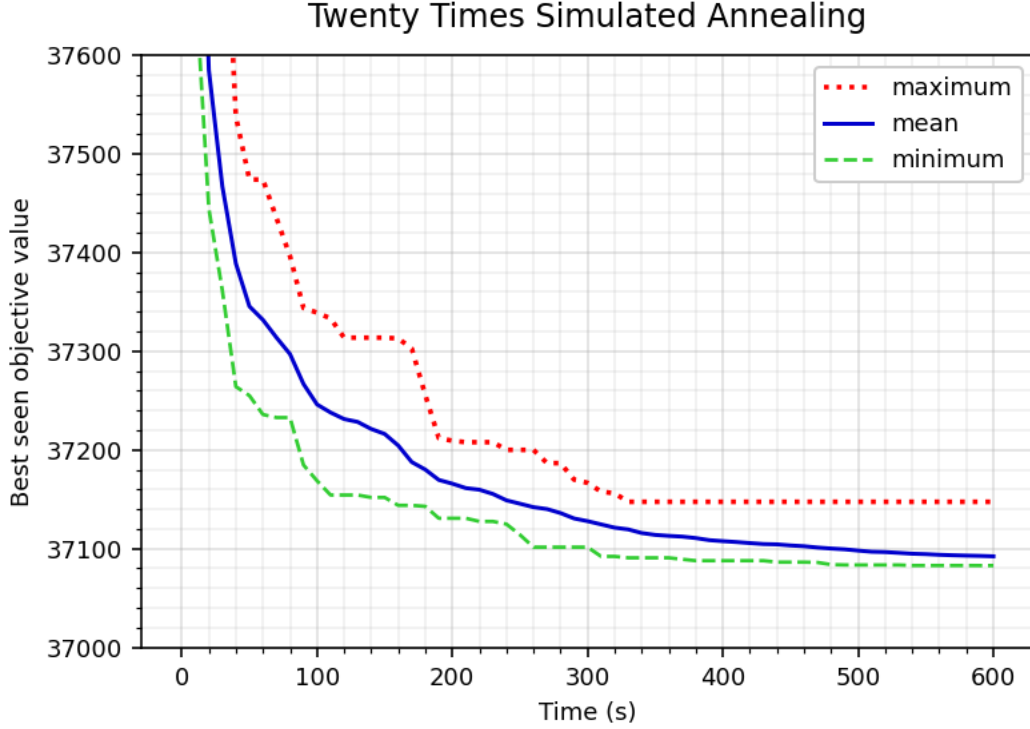


**Figure 7.2:** Convergence for 20 instances of the simulated annealing algorithm, with parameter choices  $T_0 = 25$ ,  $\alpha_T = 0.9998$ ,  $\eta = 500$ ,  $T_{\text{bonus}} = 25$ ,  $\alpha_{\text{step}} = 0.9999$ ,  $c_0^{\text{mult}} = 0.25$ ,  $c_0^{\text{add}} = 3$ ,  $c_{\text{bonus}}^{\text{add}} = 3$  and  $c_{\text{bonus}}^{\text{mult}} = 10$ . Optimisation starts from an objective value of 47537, but the vertical axis is limited to 37600 to better showcase the convergence towards the end.

Since simulated annealing accepts worse solutions, it is possible that the objective value of a selected solution is higher than the best seen objective value. These differences are showcased in Figure 7.3. Please note that maximum and minimum in this figure refer to the minimum and maximum deviation, this is not necessarily the deviation for the minimum and maximum from Figure 7.2. In fact, the experiment which achieved the worst objective value was very close to its best objective value from 240 seconds on and does not manage to improve upon this solution. It is also good to note, that the largest deviation is approximately 150. This means that the threshold of 500 is never exceeded, so  $T_{\text{bonus}}$ ,  $c_{\text{bonus}}^{\text{mult}}$  and  $c_{\text{bonus}}^{\text{add}}$  are never added.

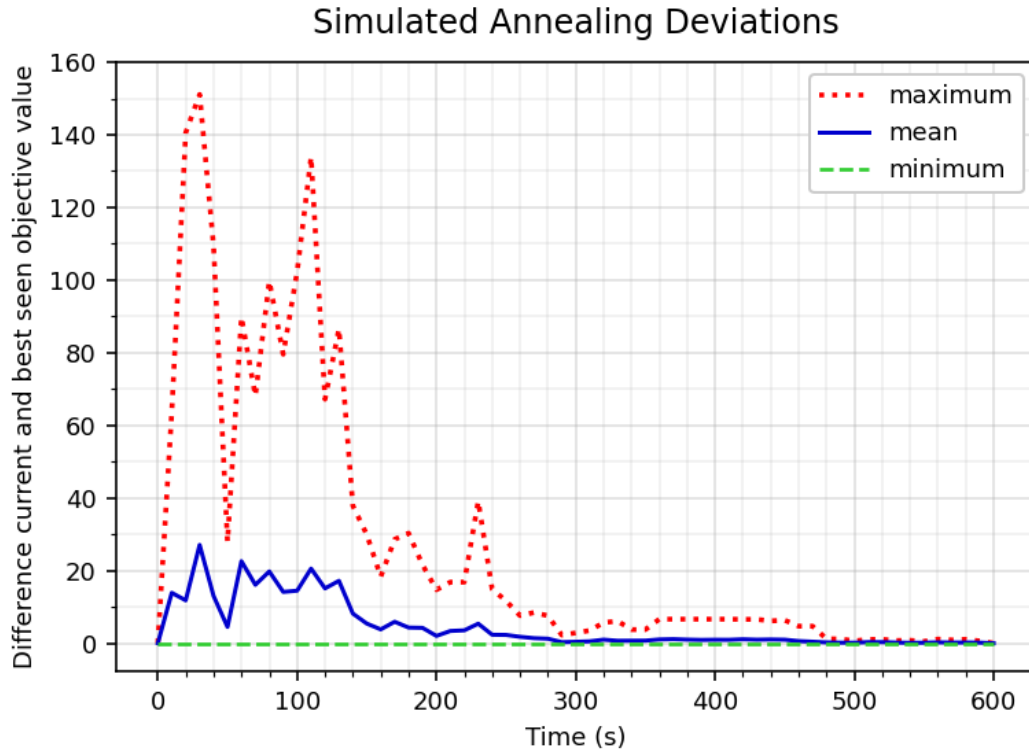


**Figure 7.3:** Deviations from best seen objective value, for the 20 instances of simulated annealing which are shown in Figure 7.2. The maximum and minimum show the maximum and minimum deviation seen across 20 experiments, so this is not necessarily the deviations for the maximum and minimum in Figure 7.2.

## 7.4 Genetic Algorithm

We will now test design options for a genetic algorithm. During the first grid search, we will test the design options in Table 7.9. The population size at the start of each generation is equal to 50. After crossover, this is either increased to twice this size or five times this size. We test both uniform and 1-point crossover. We also test mutation per variable and switchover mutation, both with  $p_{\text{mut}} = 0.1$ . We test two combinations for  $c^{\text{mult}}$  and  $c^{\text{add}}$ , one smaller and one larger step size.

**Table 7.9:** Design options for genetic algorithms, which were tested in a grid search.

| $n$ | $\kappa$ | Crossover | Mutation, with prob. | $c^{\text{mult}}$ & $c^{\text{add}}$ |
|-----|----------|-----------|----------------------|--------------------------------------|
| 50  | 2        | Uniform   | Per variable, 0.1    | 0.05 & 5                             |
|     | 5        | 1-point   | Switchover, 0.1      | 0.10 & 10                            |

The results of the first grid search can be found in Table 7.10. It becomes clear that the switchover mutation procedure does not give good results. With mutation per variable, it is possible to obtain results similar to the ones found for local search and simulated annealing. In two cases, the mean  $\mathcal{L}$  is equal to 37087. Both cases use  $\kappa = 2$ ,  $c^{\text{mult}} = 0.10$  and  $c^{\text{add}} = 10$ , while the used crossover procedure is different. Good results were also obtained if  $\kappa$  was changed to 5. Also  $c^{\text{mult}} = 0.05$  &  $c^{\text{add}} = 5$  can give good results, if  $\kappa = 2$  and mutation per variable is used. However, using  $c^{\text{mult}} = 0.10$  &  $c^{\text{add}} = 10$  instead of  $c^{\text{mult}} = 0.05$  &  $c^{\text{add}} = 5$  always gives a better mean  $\mathcal{L}$ , when other parameters are taken equally.

After concluding that switchover mutation gives bad results, we also tested the population size  $n = 100$  for all other parameter combinations from Table 7.9. The best results were obtained with the same parameter combinations as worked best for  $n = 50$ , this time by a clear margin. However, the best mean  $\mathcal{L}$  was equal to 37109, so this is worse than for  $n = 50$ . The interested reader can find these results in Table E.4.

**Table 7.10:** Results for the first grid search for genetic algorithms. All experiments use  $n = 50$  and  $p_{\text{mut}} = 0.1$ .

| $\kappa$ | Mutation     | Crossover | $c^{\text{mult}}$ & $c^{\text{add}}$ | Mean $\mathcal{L}$ | Best $\mathcal{L}$ |
|----------|--------------|-----------|--------------------------------------|--------------------|--------------------|
| 2        | Per variable | Uniform   | 0.05 & 5                             | 37092              | 37089              |
| 2        | Per variable | Uniform   | 0.10 & 10                            | <b>37087</b>       | 37085              |
| 2        | Per variable | 1-point   | 0.05 & 5                             | 37092              | 37089              |
| 2        | Per variable | 1-point   | 0.10 & 10                            | <b>37087</b>       | <b>37084</b>       |
| 2        | Switchover   | Uniform   | 0.05 & 5                             | 40268              | 39803              |
| 2        | Switchover   | Uniform   | 0.10 & 10                            | 39703              | 38910              |
| 2        | Switchover   | 1-point   | 0.05 & 5                             | 39220              | 39117              |
| 2        | Switchover   | 1-point   | 0.10 & 10                            | 38652              | 38548              |
| 5        | Per variable | Uniform   | 0.05 & 5                             | 37190              | 37133              |
| 5        | Per variable | Uniform   | 0.10 & 10                            | 37091              | 37089              |
| 5        | Per variable | 1-point   | 0.05 & 5                             | 37154              | 37140              |
| 5        | Per variable | 1-point   | 0.10 & 10                            | 37089              | 37086              |
| 5        | Switchover   | Uniform   | 0.05 & 5                             | 40846              | 40693              |
| 5        | Switchover   | Uniform   | 0.10 & 10                            | 39831              | 39451              |
| 5        | Switchover   | 1-point   | 0.05 & 5                             | 38748              | 38554              |
| 5        | Switchover   | 1-point   | 0.10 & 10                            | 38442              | 38348              |

For the second grid search, we keep using  $n = 50$ . We only use mutation per variable, but test three different mutation probabilities: 0.05, 0.10 and 0.25. For crossover, we still use both uniform and 1-point crossover. We still use the combination  $c^{\text{mult}} = 0.10$  and  $c^{\text{add}} = 10$ , which worked best in the previous grid search. Inspired by the results of local search, we now do also test lower  $c^{\text{add}}$  and higher  $c^{\text{mult}}$ -values. And we also test the heavy mutation combination  $c^{\text{mult}} = 0.25$  and  $c^{\text{add}} = 20$ , which was used to generate the initial solution set. Both  $\kappa = 2$  and  $\kappa = 5$  are still tested, since a larger population after crossover might work well combined with a higher mutation probability. Alternatively, we could have increased  $n$ , but this is less promising based on the first grid search.

**Table 7.11:** Design options for genetic algorithms, which were tested in a second grid search.

| $n$ | $\kappa$ | Crossover | Mutation, $p_{\text{mut}}$ | $c^{\text{mult}}$ & $c^{\text{add}}$ |
|-----|----------|-----------|----------------------------|--------------------------------------|
| 50  | 2        | Uniform   | Per variable, 0.05         | 0.10 & 3                             |
|     | 5        | 1-point   | Per variable, 0.10         | 0.10 & 5                             |
|     |          |           | Per variable, 0.25         | 0.10 & 10                            |
|     |          |           |                            | 0.25 & 3                             |
|     |          |           |                            | 0.25 & 5                             |
|     |          |           |                            | 0.25 & 20                            |
|     |          |           |                            |                                      |

The results of the second grid search for  $\kappa = 2$  are given in Table E.5. Using  $\kappa = 5$ , only worked better in terms of mean  $\mathcal{L}$ , for two parameter combinations that did not have an objective value close to the best result. Therefore, we have omitted the results for  $\kappa = 5$ . In the results for  $\kappa = 2$ , a lot of parameter combinations perform similar. The parameter combinations that do not work well, combine a small mutation probability with a small step size. This apparently does not lead to good exploration of the search space.

There were five experiments with mean  $\mathcal{L}$  equal to 37087, those are given in Table 7.12. In three cases,  $c^{\text{mult}} = 0.10$  and  $c^{\text{add}} = 0.05$ . The remaining two cases also included  $c^{\text{mult}} = 0.10$  (combined with  $c^{\text{add}} = 3$ ) or  $c^{\text{add}} = 5$  (combined with  $c^{\text{mult}} = 0.25$ ). Therefore,  $c^{\text{mult}} = 0.10$  and  $c^{\text{add}} = 0.05$  seems to be the best working combination. Out of the best performing parameter combinations, the one which converged fastest uses uniform crossover and  $p_{\text{mut}} = 0.25$ . Therefore, we select this as the best parameter combination. Although it should be noted, that the convergence is not clearly faster, the lower value was caused by one experiment reaching the threshold of 37100 within 384 seconds. But since there is no clear reason why a different parameter combination is preferred, we will still select it. Since the results were close to other mutation probabilities,  $c^{\text{mult}}$  and  $c^{\text{add}}$ ; we do not perform a third grid search.

**Table 7.12:** The parameter combinations that achieved mean  $\mathcal{L} = 37087$  during the second grid search for genetic algorithms. The column  $< 37100$  gives the average number of seconds that were required to reach an objective value below 37100.

| $\kappa$ | Crossover | $p_{\text{mut}}$ | $c^{\text{mult}}$ | $c^{\text{add}}$ | Mean $\mathcal{L}$ | Best $\mathcal{L}$ | $< 37100$ |
|----------|-----------|------------------|-------------------|------------------|--------------------|--------------------|-----------|
| 2        | 1-point   | 0.05             | 0.25              | 5                | 37087              | 37086              | 445s      |
| 2        | Uniform   | 0.10             | 0.10              | 5                | 37087              | 37086              | 484s      |
| 2        | 1-point   | 0.10             | 0.10              | 5                | 37087              | 37087              | 478s      |
| 2        | Uniform   | 0.25             | 0.10              | 5                | 37087              | 37086              | 427s      |
| 2        | 1-point   | 0.25             | 0.10              | 3                | 37087              | 37087              | 465s      |

For the chosen parameters, GA can complete 287 generations within 10 minutes. Around 98% of solutions which are generated during crossover are feasible, while 42% of mutation results in a feasible solution. To better analyse the performance of GA, we have done 20 experiments using the selected parameters. The best experiment obtained an objective value of 37084, while the worst objective value was equal to 37098. The mean  $\mathcal{L}$  obtained was 37089. The convergence of solutions is shown in Figure 7.4. Contrary to the previous sections, the vertical axis is limited to 42000. This is the value achieved by all 20 experiments during the first sixty seconds.

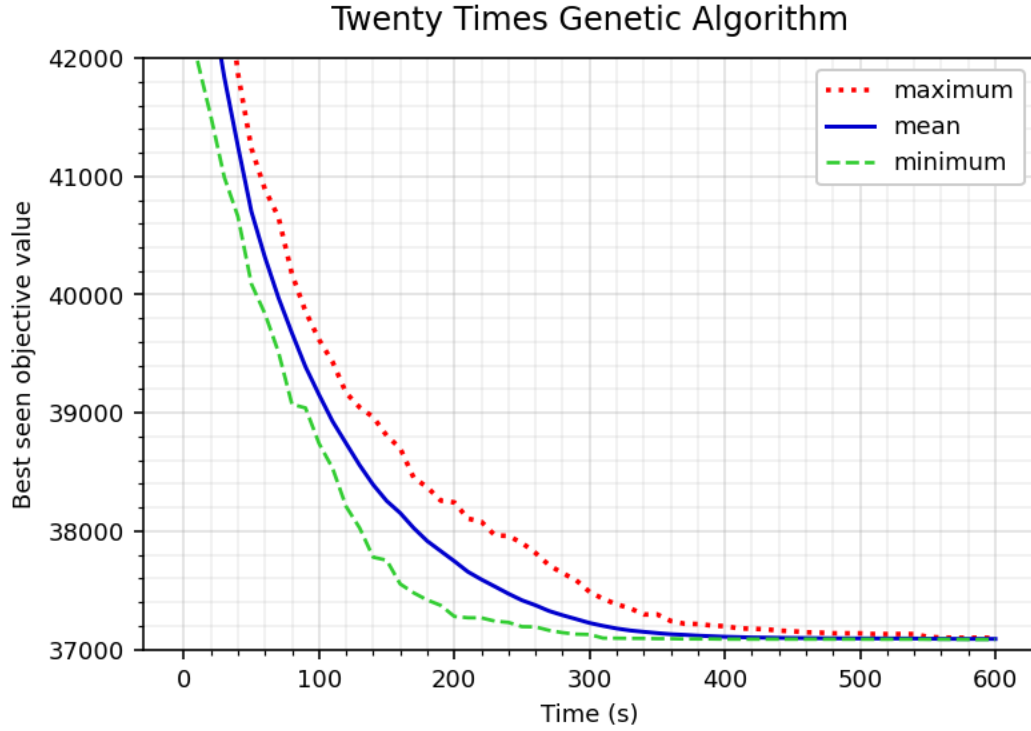


**Figure 7.4:** Convergence results for 20 instances of GA. Solutions were created using mutation per variable, uniform crossover with  $p_{\text{mut}} = 0.25$ ,  $c^{\text{mult}} = 0.10$ ,  $c^{\text{add}} = 5$ ,  $\kappa = 2$  and  $n = 50$ . Optimisation starts from an objective value of 47537, but the vertical axis is limited to 42000 to better showcase the convergence towards the end.

## 7.5 Differential Evolution

In this section, we will test design options for differential evolution. In the first grid search, we will vary the population size, crossover procedure and mutation procedure. An overview of the tested design options can be found in Table 7.13. We test seven different mutation procedures, which were the ones introduced in Table 5.1. We will test the mutation procedure current-to-pbest for both  $p = 0.10$  and  $p = 0.25$ . We denote this as current-to-0.10-best/1 and current-to-0.25-best/1 respectively. The parameter  $w$ , that is used in each mutation formula, is fixed to 0.9 as was suggested by Ronkkonen et al. (2005). We use two different population sizes: 100 and 200. Since we work with a 33-dimensional ( $D = 33$ ) solution, this is approximately  $3D$  and  $6D$  respectively.

For crossover, we will test three different procedures. Firstly, full crossover has crossover rate 1. So the created solution  $x'_i$  will always be equal to  $y_i$ . Secondly, we test binomial crossover with crossover rate 0.9 as suggested by Ronkkonen et al. (2005). On average,  $x'_i$  will consist for 90.3% of variables from  $y_i$  (see Remark 7.5.1). Lastly, we test exponential crossover. As can be seen in Figure 5.4, the order in which the train lines are placed makes a difference. We have used the order 30-31-32-73-74-75. Exponential crossover will be used with crossover rate 0.99, which means that 85.5% of  $x'_i$  is coming from  $y_i$ . We avoid using 90.3%, which was used for binomial crossover. For exponential crossover, this would require a crossover rate of 0.9935. This would mean that in 80.7% of cases, we would have  $y_i = x'_i$ . Since this would be similar to full crossover in many cases, we use a lower crossover rate for exponential crossover.

**Table 7.13:** Design options for differential evolution, which were tested during the first grid search.

| $n$ | $w$ | Crossover & CR     | Mutation               |
|-----|-----|--------------------|------------------------|
| 100 | 0.9 | Full & 1           | Best/1                 |
| 200 |     | Binomial & 0.9     | Rand/1                 |
|     |     | Exponential & 0.99 | Rand/2                 |
|     |     |                    | Current-to-rand/1      |
|     |     |                    | Current-to-best/1      |
|     |     |                    | Current-to-0.10-best/1 |
|     |     |                    | Current-to-0.25-best/1 |

**Remark 7.5.1.** Please note that the mentioned percentages for crossover are only true for  $D = 33$ . Especially for exponential crossover, the crossover rate is highly dependent on the dimension  $D$ . The expected part of  $x'_i$  coming from  $y_i$  for binomial crossover is equal to

$$\frac{CR(D-1)+1}{D},$$

while for exponential crossover this is (Pan et al., 2026)

$$\frac{1 - CR^D}{D(1 - CR)}.$$

The results of the grid search are given in Table 7.14. Population size 200 only managed to outperform population size 100 on mean  $\mathcal{L}$  in one case: when mutation procedure current-to-best/1 and full crossover was used. The other results for  $n = 200$  have been omitted from Table 7.14 and we will now discuss the results for  $n = 100$ . It stands out that for each of the seven mutation procedures, the best results are obtained with exponential crossover and the worst results with full crossover. This holds both for the mean  $\mathcal{L}$  and the best  $\mathcal{L}$ . That full crossover does perform worse, indicates that  $x'_i$  should consist of a mix of  $y_i$  and  $x_i$ .

The two mutation procedures that perform best are best/1 and current-to-best/1. For exponential crossover, they both manage to achieve an average objective value of 37085 and a best objective value of 37083. Also binomial crossover manages to get near the optimal solution for these mutation procedures. Both best/1 and current-to-best/1 perform local searches involving the best optimal solution so far. This outperforms the current-to-pbest mutation procedures, which were designed to balance local and global search. The global search procedures rand/1, rand/2 and current-to-rand/1 give the worst results. Especially the performance of rand/2 stays behind. This might be due to rand/2 only creating feasible  $x'_i$  in 26% of cases, while this varies from 45% to 60% for the other mutation procedures.

**Table 7.14:** Results obtained for the first grid search of differential evolution. Several population sizes ( $n$ ), mutation procedures and crossover procedures were used. The parameter  $w$  is always set to 0.9.

| $n$ | Mutation               | Crossover   | Mean $\mathcal{L}$ | Best $\mathcal{L}$ |
|-----|------------------------|-------------|--------------------|--------------------|
| 100 | Best/1                 | Full        | 37429              | 37342              |
| 100 | Best/1                 | Binomial    | 37111              | 37087              |
| 100 | Best/1                 | Exponential | <b>37085</b>       | <b>37083</b>       |
| 100 | Rand/1                 | Full        | 38297              | 38176              |
| 100 | Rand/1                 | Binomial    | 37646              | 37628              |
| 100 | Rand/1                 | Exponential | 37326              | 37303              |
| 100 | Rand/2                 | Full        | 39461              | 39031              |
| 100 | Rand/2                 | Binomial    | 38787              | 38630              |
| 100 | Rand/2                 | Exponential | 37747              | 37747              |
| 100 | Current-to-rand/1      | Full        | 38248              | 38115              |
| 100 | Current-to-rand/1      | Binomial    | 37601              | 37572              |
| 100 | Current-to-rand/1      | Exponential | 37329              | 37270              |
| 200 | Current-to-best/1      | Full        | 37150              | 37123              |
| 100 | Current-to-best/1      | Full        | 37219              | 37135              |
| 100 | Current-to-best/1      | Binomial    | 37092              | 37087              |
| 100 | Current-to-best/1      | Exponential | <b>37085</b>       | <b>37083</b>       |
| 100 | Current-to-0.10-best/1 | Full        | 38294              | 38002              |
| 100 | Current-to-0.10-best/1 | Binomial    | 37338              | 37306              |
| 100 | Current-to-0.10-best/1 | Exponential | 37219              | 37190              |
| 100 | Current-to-0.25-best/1 | Full        | 37814              | 37786              |
| 100 | Current-to-0.25-best/1 | Binomial    | 37321              | 37299              |
| 100 | Current-to-0.25-best/1 | Exponential | 37228              | 37218              |

We will now introduce the parameters for a second grid search, which are shown in Table 7.15. Since  $n = 100$  worked better than  $n = 200$ , we will test a smaller population size of 50 beside  $n = 100$ . We will only use exponential crossover, but besides a crossover rate of 0.99 also  $CR = 0.98$  and  $CR = 0.95$  are tested. A crossover rate of 0.98 results in the variables in  $x'_i$  coming on average 73.7% from  $y_i$ , for  $CR = 0.95$  this is 49.5%. We will only use the mutation procedures best/1 and current-to-best/1, but the parameter  $w$  can now also be 0.85 and 0.95.

**Table 7.15:** Design options for differential evolution, which were tested in a second grid search.

| $n$ | $w$  | Crossover & CR     | Mutation          |
|-----|------|--------------------|-------------------|
| 50  | 0.85 | Exponential & 0.95 | Best/1            |
| 100 | 0.90 | Exponential & 0.98 | Current-to-best/1 |
|     | 0.95 | Exponential & 0.99 |                   |

The results of the second grid search for differential evolution can be found in Table E.6. Remarkably, for 26 out of the 36 parameter combinations a best value for 37083 is found. For six more parameter combinations, the best value is equal to 37084. There are even fourteen parameter combinations where the average result over three experiments is also 37083. This makes it reasonable to assume that 37083 is the optimal solution that can be obtained.

We will first analyse the six parameter combinations, where the average objective value exceeded 37090. This means that at least one out of three experiments done did not converge towards the optimal solution. In all six cases, the population size is 50. In five out of those cases, the crossover rate was equal to 0.99. In the remaining case, the crossover rate was equal to 0.98. This suggests that using a population size of 50 combined with a high

crossover rate does not work well. The mutation procedures always creates  $y_l$  using the best solution found so far. That variables in  $x'_l$  are mainly coming from  $y_l$ , likely leads to a fast decline in diversity for a small population size. Therefore, we select a crossover rate of 0.95.

So using a smaller population size might lead to preliminary convergence. But using a smaller population size also has advantages. For  $n = 50$ , around 870 generations are completed within ten minutes, while this is only 520 for  $n = 100$ . An objective value below 37100 is usually achieved after 3 minutes when  $n = 50$ , while this takes about 5.5 minutes with  $n = 100$ . So using a smaller population size can lead to faster convergence. Using  $w = 0.85$  generally showed faster convergence than  $w = 0.90$  or  $w = 0.95$ . Therefore, we will perform a third grid search where we test some lower values for  $w$ : 0.20, 0.40, 0.60, 0.70, 0.75, 0.80 and 0.85. We also use both the mutation procedures best/1 and current-to-best/1.

When  $w = 0.20$  or  $w = 0.40$  is used, this leads to a mean  $\mathcal{L}$  above 39000. For the other  $w$ -values the results can be found in Table 7.16. It stands out that the mutation procedure best/1 reaches a higher mean  $\mathcal{L}$  for  $w = 0.60$  and  $w = 0.85$ . This could indicate that best/1 mutation struggles more with a decline in diversity than current-to-best/1. This could be explained, since best/1 replaces a solution based on a linear combination of a randomly selected solution and the best solution (if it has a better objective value). If the same solution is randomly selected multiple times, this can reduce the diversity in the population quickly. The current-to-best/1 mutation procedure does not run into this problem, so we will pick this mutation procedure.

For the selected mutation procedure, convergence seems slower for both  $w = 0.60$  and  $w = 0.85$ . Where for  $w = 0.60$  also not all experiments reach the optimal solution. The other  $w$ -values differ little in terms of speed of convergence. Therefore, we decide to pick the middle one:  $w = 0.75$ .

**Table 7.16:** Third grid search for differential evolution.

| Mutation          | $w$  | Mean $\mathcal{L}$ | Best $\mathcal{L}$ | < 37100 (s) |
|-------------------|------|--------------------|--------------------|-------------|
| Best/1            | 0.60 | 37128              | 37083              | -           |
| Best/1            | 0.70 | 37083              | 37083              | 101         |
| Best/1            | 0.75 | 37083              | 37083              | 116         |
| Best/1            | 0.80 | 37083              | 37083              | 130         |
| Best/1            | 0.85 | 37158              | 37083              | -           |
| Current-to-best/1 | 0.60 | 37086              | 37083              | 162         |
| Current-to-best/1 | 0.70 | 37083              | 37083              | 124         |
| Current-to-best/1 | 0.75 | 37083              | 37083              | 126         |
| Current-to-best/1 | 0.80 | 37083              | 37083              | 127         |
| Current-to-best/1 | 0.85 | 37083              | 37083              | 155         |

For the selected parameters, DE can complete around 850 generations within ten minutes. Out of the generated  $x'_l$ , 61.3% is feasible. To better analyse the performance, we did twenty experiments for the selected parameters. In all twenty cases, an objective value of 37083 was achieved. The convergence is shown in Figure 7.5. Please note that the vertical axis has been limited to 42000. Fast convergence is achieved in all cases, with an objective value below 37200 always achieved within 100 seconds. The final objective value of 37083 is always achieved between 180 and 270 seconds.

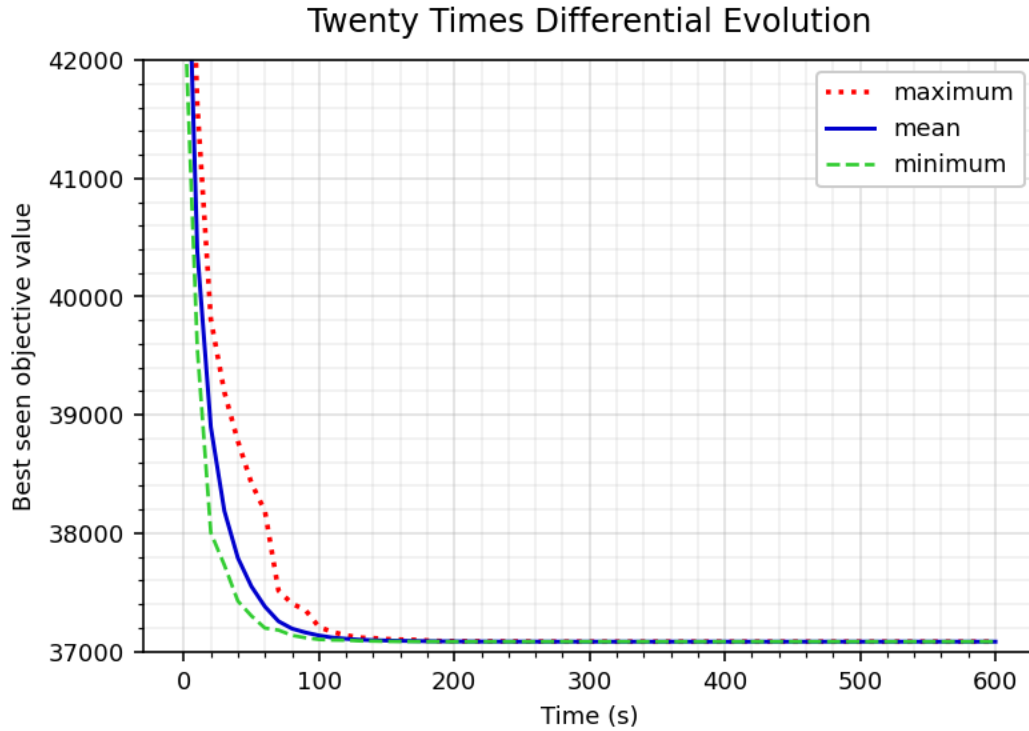


**Figure 7.5:** Convergence results for 20 instances of differential evolution. Experiments were done using  $n = 50$ ,  $w = 0.75$ , current-to-best/1 mutation and exponential crossover with  $CR = 0.95$ . Optimisation starts from an objective value of 47537, but the vertical axis is limited to 42000.

## 7.6 Convex Combinations

In this section, we will test the performance of the convex combinations algorithm which was introduced in Section 5.7. Since the differential evolution algorithm already converges quickly to the presumed optimal solution, it is not necessary to combine it with convex combinations. But we will test whether CC can help GA.

We will create twenty different instances for GA, using the parameters found in Section 7.4. These are not the same instances that have been used in Figure 7.4. For each of the instances we complete a ten minute run. But after eight minutes we save the solution set. CC will have two minutes available to work with the solutions in this set. We can then compare whether CC or GA achieved better improvements during the last two minutes.

An overview of the achieved objective values can be found in Table E.7. In four out of twenty instances, neither CC nor GA can improve the solution during the last two minutes. In ten instances, only GA manages to improve upon the solution found after eight minutes. In the remaining six instances, both GA and CC achieve an improvement. In only one case, CC reaches a better improvement. There an objective value of 37092 is improved to 37086 instead of an objective value of 37088. But in a different instance, GA reaches an objective value which is lower by twelve.

So despite convex combinations managing to improve the best solution found sometimes, it does not outperform GA in the same computation times in a majority of instances. For differential evolution, we already saw that good results were not obtained when full crossover was used. CC does not use a crossover procedure, this could explain why it performs

underwhelming. It is possible to use a procedure which does feature a crossover procedure, like the exponential crossover from DE. But since DE already converges fast, it is not needed to combine this with GA. However, if preliminary convergence becomes a problem for DE, when working with a different test case, it might become interesting to combine GA and DE.

## 7.7 Comparing results

In this section, we will compare the results obtained by LS, SA, GA and DE. We start by comparing the results on twenty experiments for each metaheuristic, which were already mentioned at the end of previous sections. These results are available side-by-side in Table 7.17.

The metaheuristic which obtains the worst mean objective value is LS with 37096. SA obtains a slightly better mean  $\mathcal{L}$  of 37092. It is noticeable that SA achieves more objective values below 37090 than LS and less objective values above 37100. But SA does also have the experiment with the largest objective value. GA has an even better mean objective value of 37089. Also all experiments reach an objective value below 37100. But in none of the cases the presumed optimal objective value of 37083 is obtained. DE does clearly perform best, with the presumed optimal objective value achieved during all twenty experiments.

We will now compare the convergence of the four metaheuristics. In two figures, we will compare the mean objective value during each time step for each metaheuristic. This is a combination of the means which were plotted in Figures 7.1, 7.2, 7.4 and 7.5. The results are shown in Figure 7.6 and 7.7. The convergence of LS and SA is very similar. They start with the best convergence, already achieving an objective value of 37600 within 20 seconds. DE starts a bit slower, but performs better than LS and SA after about 60 seconds. After 130 seconds, DE achieves an objective value below 37100 and the final objective value of 37083 is reached after 270 seconds. GA converges very slow at the start, but does manage to overtake LS and SA after about 400 seconds.

Both single solution methods, SA and LS, follow a very similar trajectory. But between roughly 60 and 180 seconds, LS performs slightly better. In Figure 7.3, we saw that this was when larger differences were present between the best seen objective value and the current objective value. When the differences go to zero, is also when SA starts performing slightly better than LS. So it is possible that the deviations from the best seen solution did help SA to find better solutions. But the difference is quite small, so this can not be concluded with certainty.

**Table 7.17:** The results for 20 instances of 4 metaheuristics, for the parameters selected at the end of each section. The first row is the mean of all experiments, the other results are sorted from best to worst.

| Statistic | LS    | SA    | GA    | DE    |
|-----------|-------|-------|-------|-------|
| Mean      | 37096 | 37092 | 37089 | 37083 |
| (1)       | 37083 | 37083 | 37084 | 37083 |
| (2)       | 37083 | 37083 | 37085 | 37083 |
| (3)       | 37083 | 37083 | 37085 | 37083 |
| (4)       | 37084 | 37084 | 37086 | 37083 |
| (5)       | 37085 | 37085 | 37086 | 37083 |
| (6)       | 37088 | 37085 | 37087 | 37083 |
| (7)       | 37091 | 37086 | 37087 | 37083 |
| (8)       | 37091 | 37087 | 37087 | 37083 |
| (9)       | 37095 | 37088 | 37087 | 37083 |
| (10)      | 37096 | 37088 | 37088 | 37083 |
| (11)      | 37096 | 37088 | 37088 | 37083 |
| (12)      | 37097 | 37090 | 37089 | 37083 |
| (13)      | 37099 | 37091 | 37091 | 37083 |
| (14)      | 37100 | 37092 | 37091 | 37083 |
| (15)      | 37101 | 37093 | 37092 | 37083 |
| (16)      | 37103 | 37094 | 37092 | 37083 |
| (17)      | 37104 | 37096 | 37092 | 37083 |
| (18)      | 37106 | 37099 | 37094 | 37083 |
| (19)      | 37117 | 37102 | 37095 | 37083 |
| (20)      | 37122 | 37147 | 37098 | 37083 |

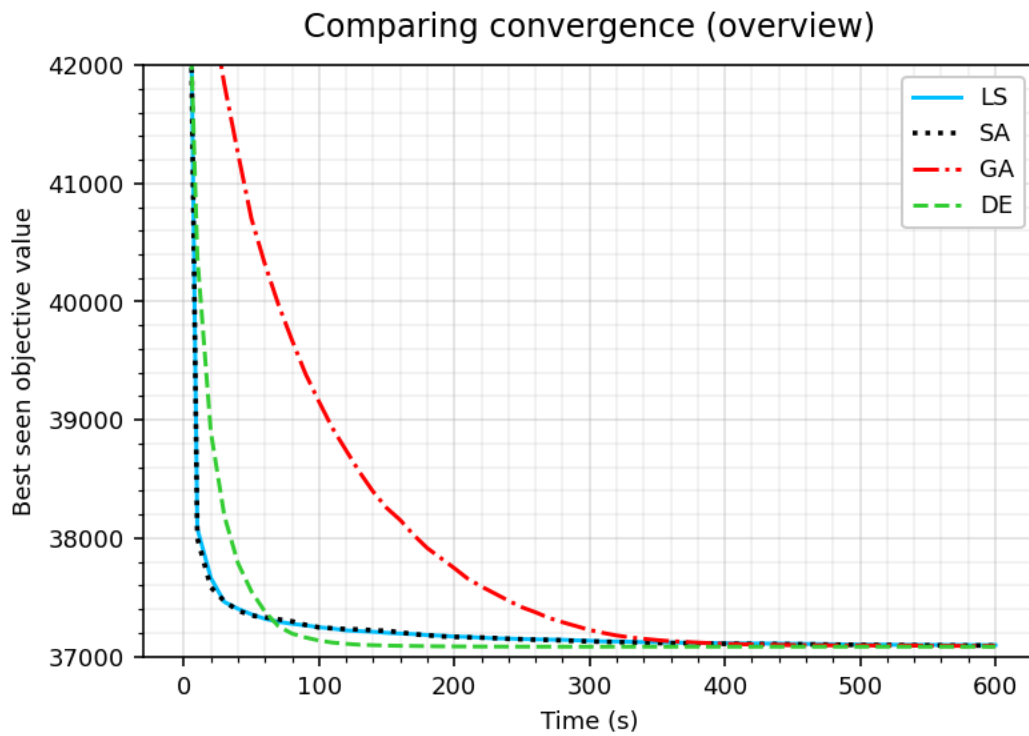


**Figure 7.6:** Comparing convergence of several metaheuristics. For each metaheuristic, the mean objective value over twenty experiments is shown. The vertical axis has been limited to 42000, while optimisation starts from 47537.

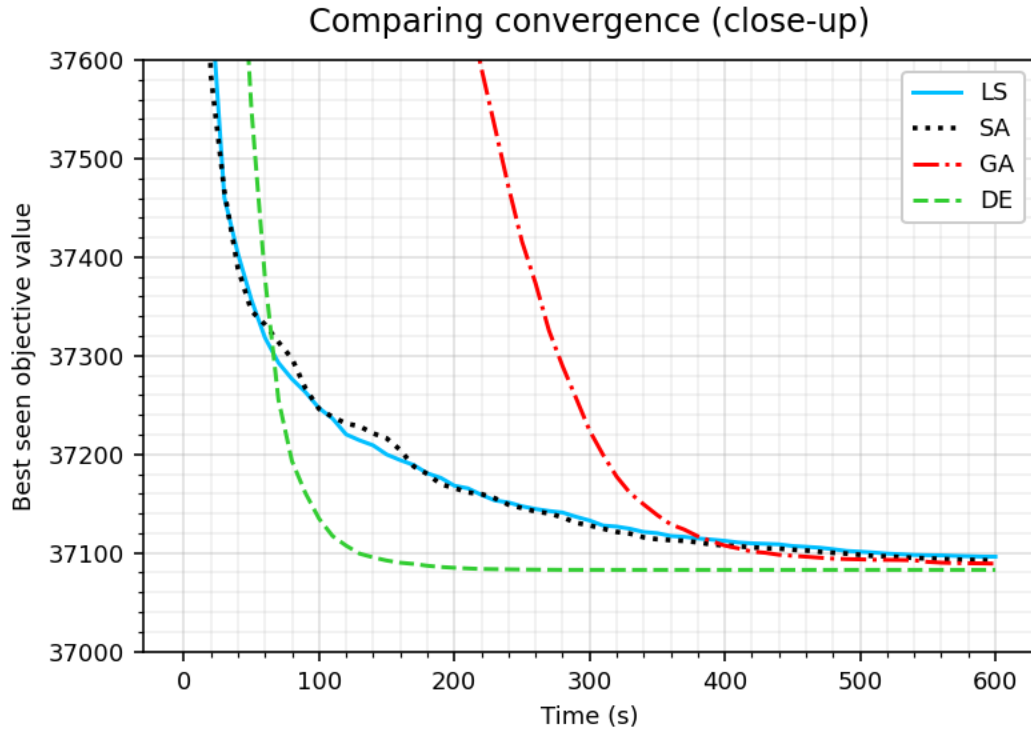


**Figure 7.7:** Comparing convergence of several metaheuristics. For each metaheuristic, the mean objective value over twenty experiments is shown. The vertical axis has been limited to 37600, while optimisation starts from 47537.

During optimisation of all metaheuristics, we kept track of the improvements. Each time that a new best objective value was found during an experiment, the delay was compared to the previous best on each of the delay scenarios in  $\Delta_{1000}$ . If the differences had a skewness between  $-0.5$  and  $0.5$ , the Wilcoxon signed rank test was applied. Sometimes a higher skewness was measured. This happened for 10% of improvements for GA and 4-5% of improvements for the other metaheuristics. For these cases, the sign test was applied instead. We will consider an improvement to be significant, if the  $p$ -value of the applied test is below 0.05.

A metaheuristic is not necessarily better if it produces a higher number of significant improvements. We always compare the performance of a solution against the previous best. So if a lot of small improvements are made, this likely leads to less significant improvements. It is not bad if a metaheuristic makes more intermediate steps to reach the same solution within similar time. Therefore, we are not interested in the total percentage of significant improvements. Instead, we want to know how large an improvement to the objective value needs to be for an improvement to be significant. The results in Table 7.18 include all improvements made during the twenty test runs for the selected parameters for LS, SA, GA and DE. Between the different metaheuristics, there were no notable differences in significance.

It can be seen that all improvements above 30 are significant. A large majority of improvements between 10 and 30 is also significant. Even a majority of improvements between 2 and 10 is still significant. Only improvements up to 2 are mostly insignificant. So based on a thousand delay scenarios, even relatively small improvements are often significant. So this is a good number of delay scenarios to consider during optimisation.

**Table 7.18:** Significance of improvements separated by the size of the improvement. The column total gives the number of improvements in each category. The data that was used for this table are the 20 experiments done for the metaheuristics LS, SA, GA and DE.

| Improvement | Total | Significant |
|-------------|-------|-------------|
| All         | 9184  | 72.9%       |
| > 30        | 3741  | 100.0%      |
| (20, 30]    | 572   | 98.3%       |
| (15, 20]    | 440   | 95.0%       |
| (10, 15]    | 527   | 90.5%       |
| (5, 10]     | 776   | 70.6%       |
| (2, 5]      | 887   | 55.1%       |
| $\leq 2$    | 2219  | 20.7%       |

Based on the convergence shown in Figure 7.7, we select DE as the metaheuristic with the most promising results. It achieves fast convergence in all twenty cases. We will use the parameters as selected in Section 7.5. So the mutation procedure is equal to current-to-best/1 with  $w = 0.75$ , the population size is 50 and exponential crossover is used with crossover rate of 0.95.

The parameters that we have found are different to the ones used for DE in Zhong et al. (2012). They used binomial crossover with crossover rate 0.9. The mutation procedure was set to Rand/1 with  $w = 0.5$ . The population size was equal to 100. Although it should be noted, that Zhong et al. (2012) also tested multiple variants on DE, including a dual population approach. This did result in improved performance in Zhong et al. (2012), so this could be worth exploring. However, we are satisfied with performance of regular DE on the test case. Therefore, we will use regular DE with the selected parameters during the rest of this thesis.

## Chapter 8

# Analysing obtained timetable

In the previous chapter, we have judged the quality of a timetable solely by looking at the objective value. In this chapter, we will evaluate how an optimal timetable looks like, compared to the original timetable. Firstly, in Section 8.1, we will switch to the objective value as introduced in (3.17). Subsequently, in Section 8.2, we will see how the allocation of supplement in the obtained timetable has changed compared to the original timetable. Then, in Section 8.3, we will investigate how this has impacted the delay. Lastly, in Section 8.4, we will debate if the obtained timetable is actually an improvement over the original timetable.

### 8.1 Adjust objective function

During the previous chapter we used an adjusted objective function, as explained in Remark 7.1.1. We will now investigate the impact of using the objective function as introduced in (3.17). If we take the optimal solution found in Chapter 7, the objective value for (3.17) becomes 41354. For the original timetable, we get an objective value of 51669. If we optimise over (3.17), then the optimal solution only changes marginally. Three experiments were done with differential evolution, which all gave an objective value of 41334. The allocation of supplement changes little compared to the optimal timetable found in Chapter 7. A total of twelve seconds is moved from the initial station, further down the lines. During this chapter, we will analyse the timetable found when optimising based on (3.17).

### 8.2 Obtained timetable

The difference between the original and optimal timetable is visualised in Figure 8.1. Here, the optimal timetable is depicted using dotted lines, while the original timetable is depicted as previously in Figure 6.3.

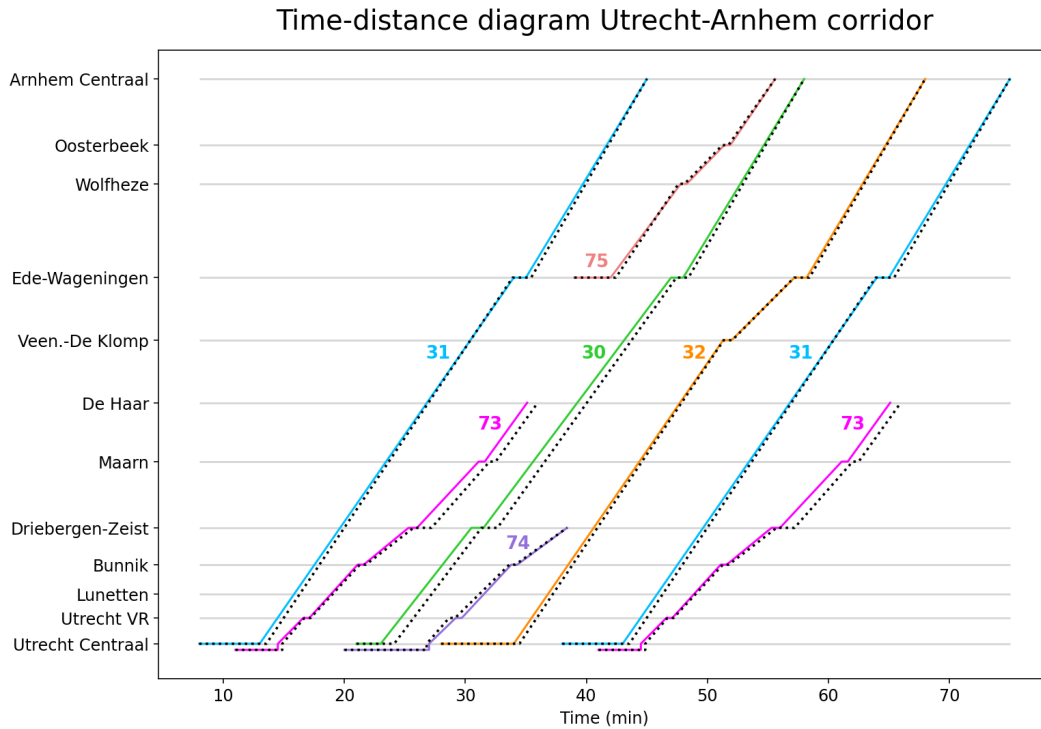


**Figure 8.1:** In this time distance-diagram, the original timetable is depicted identical to Figure 6.3. On top of this, the optimal timetable as found by DE when optimising over (3.17) is added. This is depicted using black dotted lines. A larger version of this figure is available in Figure F.2.

We will first discuss the adjusted allocation of supplement for train line 30. This train line previously had zero supplement at Utrecht Centraal, in the optimal timetable this has been increased to 67 seconds. Furthermore, 19 seconds of dwelling supplement is allocated at Driebergen-Zeist, which was previously 0. The optimised timetable does not add supplement between stations or at Ede-Wageningen. In the original timetable, there was respectively 17, 43 and 26 seconds running time supplement planned between stations. That not much supplement is added towards the end of the line, could be a result of this train line coming into Arnhem Centraal right behind line 75.

Also for intercity lines 31 and 32, supplement is shifted to stations. Train line 31 gets 33 seconds extra dwelling supplement at Utrecht Centraal, while the running time supplement between Utrecht and Ede-Wageningen has decreased from 62 to 23 seconds. It is likely that this would have been decreased to zero, if this would not have hindered line 73. Also 32 seconds of dwelling supplement is added at Ede-Wageningen, while the running time supplement between Ede-Wageningen and Arnhem Centraal drops from 35 to 0 seconds. Train line 32 gets 16 seconds dwelling supplement at Ede-Wageningen, while the running time supplement between Ede-Wageningen and Arnhem Centraal is reduced from 14 to 0 seconds. The remainder of the running time supplement is moved to Utrecht Centraal, to create 28 seconds of extra dwell time supplement.

For train line 73, 22 seconds dwell time supplement is added at Utrecht Centraal. At Driebergen-Zeist, even 47 seconds dwell time supplement is added. After each short stop, there is running time supplement planned to take into account variations in dwell time. These supplements are 19, 14, 13 and 12 seconds, so the supplement decreases further down the line. The one minute dwelling supplement present at Veenendaal Centrum in the original timetable has been removed.

Line 74 had 300 seconds dwelling supplement at Utrecht Centraal in the original timetable, this is the only train line where the supplement at the initial station has decreased (although only by twelve seconds). Also the running time supplement to Utrecht Vaartsche Rijn was removed. The supplement between Vaartsche Rijn and Bunnik increased to 43 seconds (previously 20 seconds). This is probably to avoid delay by being hindered by train line 30, which merges in front of train line 74 between these stations. Also after the short stop Bunnik, the supplement increased from 8 to 18 seconds. Lastly, we consider line 75. This line adds 26 seconds extra supplement at Ede-Wageningen, while the 37 seconds of running time supplement was removed between Ede-Wageningen and the next stop Wolfheze. Also a few seconds extra supplement was added after Oosterbeek, which resulted in 21 seconds of supplement being present after Wolfheze and 24 seconds after Oosterbeek.

So for most train lines, extra supplement is added at the initial station to reduce the entrance delay. Additionally, dwell time supplement is allocated at stations. After short stops, there is running time supplement to take care of variations in dwell time. But no running time supplement is usually added after basic or main stations, the model seems to prefer planning a later departure time. There is one exception for train line 31, where later departure would hinder the next train. But it is clearly noticeable that supplement is allocated exactly where we have added delay. In Section 8.4, we will discuss whether this leads to a better timetable. But first we see how the optimal timetable has improved the measured delay.

### 8.3 Delay breakdown

We will analyse where in the network delay occurs and how this has shifted compared to the original timetable. We will most often express the average delay per train for a certain activity and train line. Since every train line is operated 32 times, the contribution to the objective value can be calculated by multiplying the average delay by 32 and the weight of the activity for this train line.

As can be seen in Table 8.1, the largest improvements have been made for lines 30 and 73. For line 30, adding dwell time supplement at Utrecht Centraal has had a large influence on the objective function. The average departure delay per train dropped from 84 to 46 seconds. Also the departure and arrival delay at Driebergen-Zeist dropped from 73 and 88 to 46 and 49 second respectively (where the arrival delay is multiplied with weight 3 in the objective function). Also the arrival and departure delays at Ede-Wageningen decrease for line 30. Only at the final station, Arnhem Centraal, does the average delay increase from 58 to 65 seconds per train. The contribution to the objective value (including the bonus term) increases from 9683 to 10412 at this station.

For train line 73 the departure delay from Utrecht Centraal has also decreased, from 12 to 8 seconds per train. However, most impactful is the dwell time supplement added at Driebergen-Zeist. This decreased departure delays from 59 to 14 seconds per train. This also leads to a lower arrival delay at the next stop (Maarn). In the original timetable, the extra delay which arises at Driebergen-Zeist is 30 seconds on average for trains on line 73. It is questionable whether this is realistic. So perhaps Driebergen-Zeist actually has dwell times for sprinters which resemble a short stop. That less supplement is planned towards the end of the line for line 73, does result in a total of 446 weighted delay more at the last two stations. This equals 3 and 5 seconds extra delay per train at these stations.

In total, the extra dwell time supplement planned at stations results in a decrease in departure delays from 13522 to 8086. This is more than half of the improvement in the objective value. Especially the departure delay at Utrecht Centraal is notable, which decreases from 3322 to 1858. A big part of this improvement is realised for line 30, as mentioned earlier this section.

**Table 8.1:** Breakdown of contribution of each train line to the objective value.

| Line  | Original | Optimal | Reduction |
|-------|----------|---------|-----------|
| 30    | 30182    | 24632   | 18.4%     |
| 31    | 3251     | 2631    | 19.1%     |
| 32    | 6798     | 6250    | 8.1%      |
| 73    | 9216     | 5801    | 37.1%     |
| 74    | 1675     | 1589    | 5.1%      |
| 75    | 548      | 431     | 21.4%     |
| Total | 51669    | 41334   | 20.0%     |

We have described the results based on the 1000 delay scenarios that were used to optimise the timetable. In Figure 8.2, we compare the objective value of the original timetable with the objective value of the improved timetable for delay scenarios which were not seen during optimisation. We generated  $\Delta_{1000}$ , a thousand different times. The objective value for the optimal solution has an almost linear relationship with the objective value on the original timetable. The objective values for the timetable used during optimisation are in line with this. So the timetable is not a result of overfitting. This fits in with the results from Section 7.7. There, it was discovered that relatively small improvements on the objective value, were significant improvements for the underlying probability distributions.

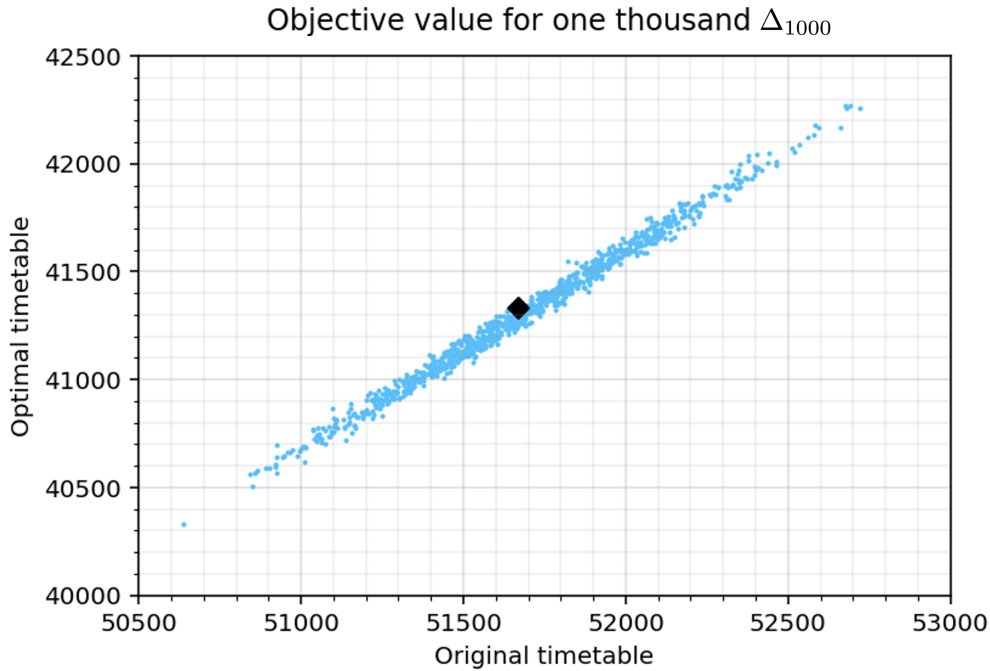


**Figure 8.2:** One thousand times, a set  $\Delta_{1000}$  consisting of 1000 delay scenarios was generated. For each set of delay scenarios, the objective value of the optimal solution (as found by DE) was compared with the objective value of the original solution. The delay scenario which was used during optimisation is also added as a black diamond.

For the optimisation of the timetable, all entrance delays above 360 seconds were resampled. Since the optimised timetable has more supplement at the initial station for most trains, this could help reduce delays for larger delay scenarios. To test this, all entrance delays which were resampled due to being too high were placed back in the delay scenarios. On the original timetable, the objective value increased to 80092. For the optimal timetable, the objective value became 69059. This is an increase of 28423 and 27725 respectively. So the optimal timetable is also more robust against higher entrance delays, but only 2.5% better.

Even if we use unbounded entrance delays, the timetable offers little possibilities to swap the order of consecutive trains. Based on Figure 8.1, we note the following. In most cases, sprinter trains and intercity trains are alternated. This makes it harder to swap consecutive trains. Consider the following example. If line 75 arrives delayed at Arnhem Centraal, line 30 will also arrive delayed. If already at Ede-Wageningen line 75 would have a delayed departure, we could let it wait at Ede-Wageningen until line 30 has passed by. But this would require line 75 to depart eight minutes behind its schedule, assuming that line 30 does not suffer delay. Similarly, line 74 would need to depart about 7.5 minutes behind schedule from Utrecht Centraal to end up behind line 32. Line 31 can not swap order with line 73 without also hindering line 30. And swapping line 30 with either line 73 or 74 is also difficult, because all these train lines use the same platform at Driebergen-Zeist.

So trains need to depart much later to be able to swap order. Since most trains also have supplement available at Utrecht Centraal, this would only be favourable for entrance delays that rarely occur for intercity and sprinter trains. If we would have included international trains in the timetable, this could lead to a different situation. As seen in Table 4.3, international trains suffer larger entrance delays. But for the timetable that we consider, there are no adjustment measures for the train order that could frequently be applied. It is also not possible to create a timetable with an adjusted train order, without earlier entrance or later exit from the network. Therefore, we do not further investigate changing the order of trains.

## 8.4 Impact of adjustments

In the previous two sections, we have discussed the obtained timetable. It became clear that the optimal timetable has longer dwell times at stations, especially at the initial station. This leads to less departure delay. Also the arrival delay in the beginning of the network reduces. But due to a lack of running time supplement, there can be more delay towards the end of the network. We will now discuss whether these are desirable adjustments to the timetable.

The largest concern with the adjusted timetable, is that the planned amount of running time supplement could be too low. The downside of waiting longer at the initial station, is that less supplement is planned on the corridor between Utrecht and Arnhem. We have not considered fluctuations in running times, due to a lack of reliable probability distributions available to mimic these fluctuations. Since we do not know the scale of these fluctuations, it is hard to assess if running times frequently exceed the 4% margin that is scheduled in the timetable. The margin suggested by the international union of railways is 7%, which was used to create the original timetable. The created timetable does not comply with the 7% running time margin norm between Utrecht and Arnhem. But it is hard to assess the impact of using a lower margin without knowing the scale of fluctuations. This is the largest doubt about the robustness of the created timetable. Also if the lack of running time supplement does not form a problem for robustness, it is still less energy-efficient.

We have now discussed the most important disadvantages of the created timetable, but there are also advantages. That more supplement is planned at the start of the network, makes the departure and arrival times at the start of the network more reliable. Especially line 30 was susceptible to delay, since this line had no dwell time supplement at Utrecht Centraal. Passengers will now see a more reliable departure time, which is one minute later. The other modified departure times at Utrecht Centraal are changed by less than a minute, such that the rounded down departure time communicated to passengers does not change.

That the trains depart later than their communicated departure time from Utrecht Centraal, does make it more likely for people to make their train. Utrecht Centraal is an important transfer hub. Out of the 242 thousand passengers using the station each day, 62 thousand passengers transfer to a different train (NS, 2025). Slightly later departure times can help people to complete their transfer. Although it should be noted, that if the first station in the network is not important for transfers, it is still likely that a lot of supplement is planned at the initial station to reduce the entrance delays. In that case, planning extra supplement at the first station is less useful. Later departure times do also lead to slightly longer travel times for passengers which entered the train in a station before the network (so before Utrecht Centraal for the used test case). However, in vehicle time is generally perceived by passengers as less annoying than waiting time (Fan et al., 2016; Fosgerau et al., 2007; Ingvardson et al., 2018). Research by Shelat et al. (2021) found that passengers accept more than seven minutes extra in-vehicle time, if this led to more certainty in waiting time. So having more reliable departure and arrival times, is important for the satisfaction of passengers.

## Chapter 9

# Model adjustments

In this chapter, we will test multiple adjustments to the used setup. Firstly, in Section 9.1, an adjustment to the objective function is tested. Subsequently, in Section 9.2, the dwell times at Driebergen-Zeist are adjusted for line 73. Then, in Section 9.3, it is investigated what happens if the departure times from the initial station are fixed. Lastly, in Section 9.4, the running times of trains are varied. All experiments in this chapter are done using differential evolution with the parameters as selected in Chapter 7. At least two experiments were done for each adjustment, if this gave an identical timetable multiple times, we consider this the presumable optimal timetable. Otherwise, we did more experiments until the same timetable with best objective value was found twice.

### 9.1 Different objective function

In Section 3.5, it was discussed that sometimes departure delays are not punished in the objective function. We will test if the allocation of supplement changes when the weight of all departure events is set to zero. Naturally, if the departure delays are not punished, it does not matter if dwell time supplement is planned at a station or running time supplement is planned directly after a station. But apart from this, only a couple of seconds of supplement differs if we optimise without punishing departure delays. This change to the objective value will not be used throughout the rest of the chapter, so departure delays will still be punished in the upcoming sections.

### 9.2 Change dwell times at Driebergen-Zeist

As previously mentioned in Section 8.3, the dwell times sampled for sprinters at Driebergen-Zeist could be too pessimistic for sprinter line 73. We use the probability distribution from a basic station, while the scheduled dwell time in the original timetable is in line with a short stop. The currently used distribution, causes an average delay of about 30 seconds in the original timetable. To reduce this, a lot of dwell time supplement was added at the station in the optimal timetable. We will now investigate how the timetable changes if the dwell times at Driebergen-Zeist for line 73 are sampled from the short stop distribution.

Changing the dwell times, causes the objective function of the original timetable to drop from 51669 to 47957. For the previous optimal timetable, the objective value decreases from 41335 to 39447. When we optimise over the adjusted delay scenarios, an objective value of 38743 is achieved. The amount of dwell time supplement planned at Driebergen-Zeist for line 73 drops from 47 to 11 seconds. Of this supplement, 20 seconds is added as dwell time supplement at Utrecht Centraal. Also, 8 seconds of dwell time supplement is added at Veenendaal Centrum. The remaining seconds are allocated as running time supplement after short stops. The later departure of line 73 also allows line 31 to depart later from

Utrecht Centraal, the departure time of this line is therefore delayed by 22 seconds. This supplement is taken away from the running time supplement between Utrecht Centraal and Ede-Wageningen, which drops to just one second.

So using short stop dwell times for line 73 at Driebergen-Zeist, does lead to an altered timetable. For the rest of this chapter, we will keep using short stop dwell times for line 73 at Driebergen-Zeist.

### 9.3 Fixed departure time from initial station

In the adjusted timetable, a lot of dwell time supplement is allocated at the initial station for each train line. This leaves a limited amount of supplement between Utrecht and Arnhem. This is bad for the energy-efficiency and if the running times vary heavily, also bad for the robustness of the timetable. Therefore, we will test how the timetable is adjusted if the departure time from the initial station may not be modified. As was to be expected, the majority of supplement planned at the initial station is now moved to directly after the station. Although a limited amount of supplement is also allocated further along the line. This involves 10 seconds for line 73, 6 seconds for line 75 and 5 seconds for line 30. Most of this supplement is already allocated as dwell time supplement at the first intermediate station. The resulting objective value is 40552.

### 9.4 Varying train running times

In this section, we will see what happens if the running times of trains vary. We use the uniform probability distributions from Larsen et al. (2014). For the running times between two activities, the technical minimal running time is multiplied with a factor which is uniformly sampled within an interval. The intervals  $[0.975, 1.05]$ ,  $[0.95, 1.10]$ ,  $[0.925, 1.15]$  and  $[0.90, 1.20]$  from Larsen et al. (2014) will be tested, so there are four fluctuation levels. It should be noted, that Larsen et al. (2014) selected these probability distributions without examining train recorder data. So it is possible that using uniform distributions is not completely realistic, but it does still give an interesting insight in the effect of adding running time fluctuations. We do sample all multiplication factors independently from each other, as was also done in Larsen et al. (2014). We will also remove the limitation from the previous section, such that departure times from the initial station may be modified again. This gives insight in the balance between entrance delays, varying dwell times and fluctuations in running times.

Four timetables were obtained, by optimising on each of the four fluctuation levels. Line 30 had no dwell time supplement at the initial station Utrecht Centraal in the original timetable. When we optimised without running time fluctuations, 66 seconds of supplement was added here. There was no supplement directly after Utrecht Centraal, by which we mean that there is no running time supplement for the running time between Utrecht Centraal and the next station where the train stops. For all fluctuation levels, the dwell time supplement at Utrecht Centraal and the running time supplement directly after this station adds up to a number between 53 and 59 seconds. For the lowest fluctuation level, 81% of this is dwell time supplement, while for the highest fluctuation level only 30% is dwell time supplement. So the balance between dwell time and running time supplement changes as the level of fluctuations changes. In all cases, the supplement added at and directly after Utrecht Centraal has increased a lot compared to the 17 seconds present in the original timetable.

Line 30 also has between 27 and 33 seconds of supplement at and directly after Driebergen-Zeist station. Where the balance between dwell time supplement at the station and running time supplement directly after the station varies based on the level of fluctuations. No supplement is allocated at and directly after Ede-Wageningen station. So less supplement is

allocated towards the end of the line. In the original timetable, there was still 43 seconds running time supplement directly after Driebergen-Zeist and 26 seconds directly after Ede-Wageningen.

Line 31 had 242 seconds of supplement at and directly after Utrecht Centraal in the original timetable, of which 74% is dwell time supplement. For the timetables optimised on a fluctuation level, this varies between 234 and 238 seconds of which 48% to 74% is dwell time supplement. For higher levels of fluctuations, more time supplement is added directly after Ede-Wageningen instead of at this station.

Line 32 has the longest running time between Utrecht Centraal and Veenendaal-De Klomp, here 25 seconds of running time supplement was available in the original timetable. For increasing levels of fluctuations this becomes 29, 53, 72 and 85 seconds. In contrast, the dwell time supplement at Utrecht Centraal decreases to 199 seconds for the highest fluctuations, compared to the original 240 seconds. Also the supplement on the segment Veenendaal de Klomp-Arnhem Centraal decreases, for the largest two levels of fluctuations even to zero.

Line 73 had 106 seconds of supplement at or directly after the initial station Utrecht Centraal in the original timetable. For the lowest level of fluctuations, this increases by 13. For the highest level of fluctuations, 28 seconds supplement is taken away and reallocated across the rest of the route. Line 74 had 300 seconds dwell time supplement at its initial station Utrecht Centraal in the original timetable. When fluctuations in running time are added, this varies from 270 (lowest fluctuations) to 213 seconds (highest fluctuations). The freed up supplement is allocated across the rest of the route. Line 75 had 120 seconds of dwell time supplement at its initial station Ede-Wageningen in the original timetable. With fluctuations this varies from 132 (lowest fluctuations) to 90 seconds (highest fluctuations). So the running time supplement increases for higher levels of fluctuations.

To summarise, some adjustments compared to the original timetable are made for all fluctuation levels. Line 74 always departs earlier, such that more running time supplement is available for the rest of the route. Line 30 always gets more time supplement at or directly after Utrecht Centraal, to reduce the amount of entrance delay. There are also changes which do depend on the level of fluctuations. Like all sprinter lines having less dwell time supplement at the initial station for higher fluctuation levels, such that more supplement is available for the rest of the route. So the level of fluctuations does influence the balance between running and dwell time supplement.

The results for optimising over several fluctuation levels can be found in Table 9.1. The timetable which is optimised with no fluctuations in running time, performs worse than the original timetable for the three largest fluctuation levels. So the previous strategy of placing a lot of supplement at the initial station, does not work well if the running times fluctuate.

Improving upon the original timetable is still possible when fluctuations in running time are added. We have optimised timetables based on four different levels of fluctuations. We can also evaluate timetables based on these four levels of fluctuations or with no fluctuations in running time. All four timetables optimised using fluctuations in running time outperform the original timetable on all these five evaluation options. The best improvement in objective value compared to the original timetable is 19.3% if no fluctuations in running time are added. The best improvement is 15.2% for  $[0.975, 1.05]$ , 14.4% for  $[0.95, 1.10]$ , 15.5% for  $[0.925, 1.15]$  and 17.2% for  $[0.90, 1.20]$ . So a considerable improvement of the objective value is possible, no matter the level of fluctuations.

The timetables obtained for the four different fluctuation levels are different, as previously discussed. However, in Table 9.1 it can be seen that timetables optimised on a certain fluctuation level can also perform well on other fluctuation levels. This is especially true for the

timetable optimised using fluctuation factors in  $[0.95, 1.10]$ . This timetable has a reduction in objective value compared to the original timetable, which varies between 12.8% and 14.4% for the five considered evaluation options. So a similar improvement would be achieved, no matter the fluctuation level. This shows that a considerable improvement to the original timetable can be made, without knowing the true level of fluctuations.

**Table 9.1:** In the column timetable, six different timetables are given. Firstly, there is the original timetable. The other timetables have been optimised based on a thousand delay scenarios, using factors on the technical minimal running times sampled uniformly within the given intervals. Each column describes the objective values, when evaluated for a certain factor interval on the running times. Evaluation is done using 100,000 delay scenarios, which were not used during optimisation.

| Timetable       | 1            | $[0.975, 1.05]$ | $[0.95, 1.10]$ | $[0.925, 1.15]$ | $[0.90, 1.20]$ |
|-----------------|--------------|-----------------|----------------|-----------------|----------------|
| Original        | 47951        | 58208           | 74042          | 95924           | 123736         |
| 1               | <b>38694</b> | 57061           | 80878          | 109160          | 141658         |
| $[0.975, 1.05]$ | 39845        | <b>49363</b>    | 66168          | 90317           | 119776         |
| $[0.950, 1.10]$ | 41304        | 50392           | <b>63391</b>   | 82491           | 107931         |
| $[0.925, 1.15]$ | 42777        | 51770           | 64108          | <b>81036</b>    | 103342         |
| $[0.900, 1.20]$ | 43967        | 52975           | 65183          | 81591           | <b>102434</b>  |

Up until now, all running times were sampled independently. In practice, not all running times are independent. Fluctuations in running times can be caused by train driver behaviour or the characteristics of a specific train. This will influence the performance of a train across its entire route, unless the train driver is swapped at a station in the network. Fluctuations in running times can also be caused by unfavourable weather conditions, like strong headwinds (Goverde & Scheepmaker, 2023). Although it is a popular saying that nothing is as changeable as the weather, often multiple trains are influenced by the same weather conditions. And the weather conditions can be similar throughout the network.

Sampling independently was also the approach from Larsen et al. (2014), because there is little research available on the dependencies between running times. So more research is needed, to get a reliable estimate of the dependencies between several running times. But sampling the length of each running arc independently, can lead to more moderate fluctuations. Since there can be multiple running arcs between two planned events, the fluctuations can partially cancel each other out. Therefore, we will test incorporating some dependencies into the model.

We allow the multiplication factor on the technical minimal running time to vary between 0.95 and 1.10, which is one of the intervals used before. But the multiplication is now decided based on four variables. There is still a multiplication factor which is independent for each running arc. This is sampled uniformly in the interval  $[0.995, 1.01]$ . To this, three factors are added which are all sampled uniformly in  $[-0.015, 0.03]$ . Firstly, there is a factor which is the same per train. This factor is used to represent fluctuations in train performance and train driver behaviour. Secondly, there is a factor which represents global conditions. This is the same for all trains and locations in the network. The factor is resampled every time that two hourly cycles have passed, so once every two hours. Thirdly, we add a factor for local conditions. This factor is the same for all running arcs at a specific location, and is also resampled every time that two hourly cycles have passed. For quadruple-track railways, we have used the same factor on both tracks heading in the same direction. The independent factor which is sampled for each running process is used to represent further uncertainty. It is uncertain whether the used factors have the correct proportions, but it is still interesting to see the effect of these dependencies.

We will compare the timetable for the dependent multiplication factors to the timetable obtained when sampling independently in the interval  $[0.95, 1.10]$ . Using dependent sampling has limited effect on the obtained timetable. For train line 30, the supplement is allocated exactly the same. For lines 31 and 32 about five seconds of dwelling supplement at Utrecht Centraal is moved to the subsequent running time. For the sprinter lines, 5-10 seconds extra supplement is created at the initial station. As can be seen in Table 9.2, the independent delay factors lead to slightly more delay. We expected that independent delays would partially cancel each other out, which would lead to less delay. But it seems like independent factors make it more likely that some delay occurs. We further see that the evaluated objective value is approximately 100 better if the same approach was used for optimisation as evaluation.

**Table 9.2:** The first column describes whether optimisation of a timetable was done with independent or dependent factors, in the interval  $[0.95, 1.10]$ . The remaining columns evaluate using independent and dependent factors respectively. Evaluation is done using 100,000 delay scenarios, which were not used during optimisation.

| Timetable                | Independent evaluation | Dependent evaluation |
|--------------------------|------------------------|----------------------|
| Independent optimisation | <b>63391</b>           | 63157                |
| Dependent optimisation   | 63488                  | <b>63064</b>         |

It is possible that differential evolution would struggle with optimising a timetable, when there are fluctuations in running times. To assess this, we did twenty optimisations (of ten minutes each). In sixteen cases, an objective value of 62998 was found (on the delay scenarios used for optimisation). In three more cases, the objective value was 62999. One time, the objective value was 63031. For all other cases, an objective value of 63000 was obtained within 300 seconds. For the worst performing, the objective value only improved by 1 past 240 seconds. So adding fluctuations on the running times, does not lead to slower convergence. However, we have learned that preliminary converge is still possible for DE with the design options that we selected. Therefore, it is recommended to pick the best solution after performing multiple experiments. Although the worst performing run out of the twenty experiments, does still reach an optimality gap of 0.05% within 240 seconds.

To showcase how the obtained timetable looks like after all modifications made in this chapter, a time-distance diagram is available in Figure 9.1. The black dotted lines show the timetable which was obtained for the dependent factors, while the continuous lines are the original timetable.

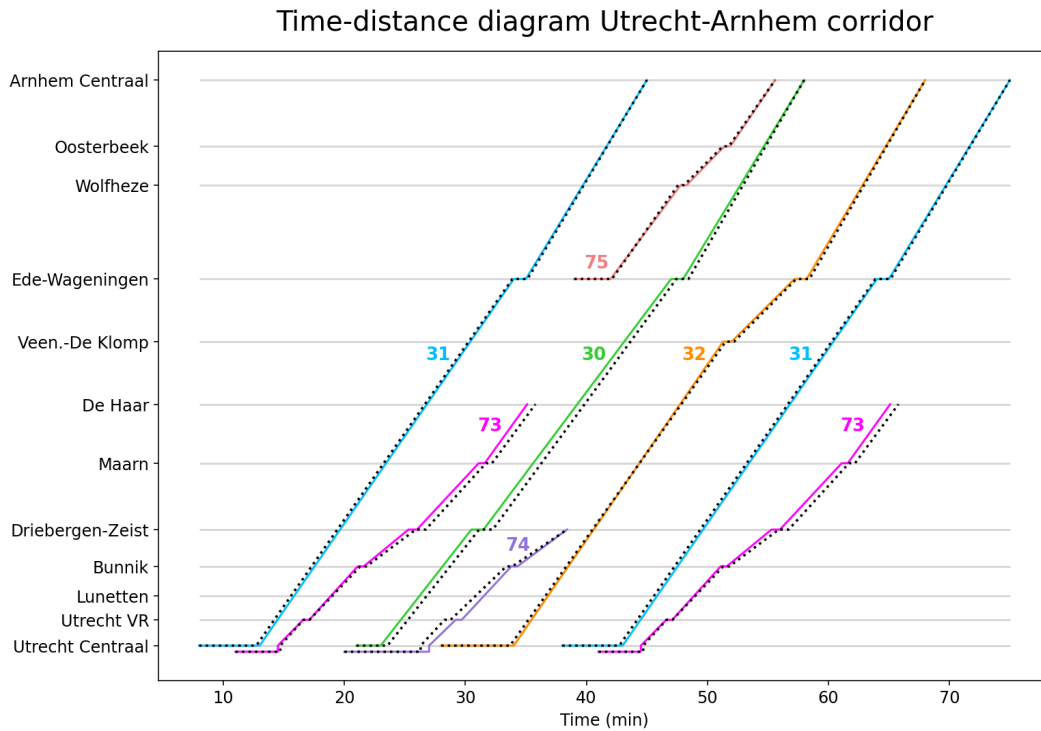


**Figure 9.1:** The black dotted lines represent the timetable obtained by optimising over the dependent delay factors. These are overlaid on the original timetable, as depicted in Figure 6.3. A larger version of this figure is available in Figure F.3.

## Chapter 10

# Conclusion & Discussion

During this thesis, we utilised random search metaheuristics to reallocate time supplements in a railway timetable. This was done to improve the robustness of the timetable. Since delays lead to a decrease in passenger satisfaction, creating a robust timetable is important for the attractiveness of railways compared to other modes of transport.

Firstly, in Chapter 2, it was discovered that optimising time supplements by solving an ILP is computationally expensive. However, also limited research is available where (meta)heuristics are used to allocate time supplements. Therefore, we decided to use random search metaheuristics to optimise timetables. In Chapter 3, it was decided how the timetable would be modified by a random search metaheuristic and how the robustness is evaluated. We created a framework where timetables could be modified on a network, without having to adjust the timetable outside of the network, contrary to many (meta)heuristics found in literature. To evaluate the robustness, one thousand delay scenarios were designed in Chapter 4 consisting of sampled entrance delays and dwell times. These are more delay scenarios than used in previous literature, which allows for a more accurate estimation of the delay.

In Chapter 5, we introduced several metaheuristics, where local search, simulated annealing, genetic algorithms and differential evolution were selected. For these metaheuristics, design options were tested in Chapter 7. This was done on the railway line between Utrecht and Arnhem, including the branch line to Rhenen (as introduced in Chapter 6). Out of the investigated metaheuristics, the best results were obtained using differential evolution. When tested twenty times, DE always managed to find the presumed optimal solution within a computation time of 270 seconds. The optimal timetable was analysed in Chapter 8. A lot of supplement was allocated at the initial station for each train, such that entrance delays could be recovered. On the other hand, little supplement was planned between stations. This would make the timetable vulnerable to variations in running time, which were not accounted for when evaluating the delay propagation. Therefore, it was decided to include simple fluctuations in running times in Chapter 9, among with some other adjustments to the model setup.

Several different levels of fluctuations were tested for the running times. Adding these fluctuations resulted in a more balanced allocation of supplement. Using the previous setup, mostly dwell time supplement was planned. The biggest part of which was added at the initial station. After adding fluctuations in running time, also running time supplement was planned. The balance between running and dwell time supplement does depend on the level on fluctuations. However, for each level of fluctuations a considerable improvement of the robustness was achieved compared to the original timetable. The achieved improvement in objective value varied between 14.4% and 17.2%. Furthermore, timetables optimised on one level of fluctuations, could also perform well when evaluated on other fluctuation levels. Thus, knowing the exact level of fluctuations is not required to improve the robustness.

The main limitation of this thesis are the probability distributions, which have been used to create delay scenarios. Firstly, only simple distributions have been used to create fluctuations in running times. More research is needed on this topic. Especially, the scale and distribution of fluctuations needs to be investigated. To further improve accuracy, also dependencies between fluctuations can be investigated. The free dwell times used in this thesis were based on data from 2014 for short stops and 1999 for basic and main stations. Therefore, these distributions are due for an update. Here, it would also be good to further differentiate based on station and train type. Also the entrance delays are based on data from 2008 and could use an update. Special attention could be paid to the entrance delays for trains at the beginning of a line.

A second possibility for improvement is the objective function. Currently a simplified weight system is used. For each station that is passed by a train, the weight is added at the next station where a train stops. This does make some stations more important than they truly are. To better estimate the discomfort of passengers, the objective function should be based on the number of passengers. Arrival delays could be weighted with the number of passengers exiting a train at the station, while departure delays can be weighted with the number of passengers entering the train. It is possible to further extend the quantification of passenger discomfort, by estimating extra travel time due to missed transfers. Furthermore, the bonus term in the objective function was an estimation of the amount of delay outside of the network. This can be further refined. It can be considered that trains often have waiting time available at the end of the line before starting a new journey. This can reduce the amount of hinder caused by delay at network entrance. So further refinement of the delay reduction outside of the network is recommended. The objective value could also punish delays above three and ten minutes more heavily, such that the punctuality which needs to be reported to the Dutch government improves. Finally, robustness could also be balanced with other objectives, like energy-efficiency.

An advantage of the methodology presented in this thesis, is that the arrival time at the first and last station in the network are fixed. This ensures that the methodology can be applied on smaller networks too, without having to adjust the timetable outside of the network. During this thesis, metaheuristics were tested on the Utrecht-Arnhem corridor. It would be interesting to see how the metaheuristics perform on a larger network. Using the methodology, it is possible to partition a larger network into corridors that can be optimised separately. However, more flexibility is obtained if optimisation is performed on a larger network. This does require more supplement variables to be adjusted. This could require modifications to the metaheuristics. If all time supplements are adjusted at once, it is possible that mostly infeasible timetables are generated. Then, it might be better to update only a smaller part of the timetable, during at least some generations.

For the scalability of the presented methodology, it is promising that evaluation of a timetable scales linearly in terms of the number of vertices in the graph. However, larger networks could also require a larger population size for metaheuristics. Especially differential evolution, which performed best on the Utrecht-Arnhem corridor, usually has a population size which scales with the number of variables that need to be adjusted. It is possible to modify all solutions for differential evolution in parallel, this could be used to apply differential evolution on larger networks too. In Appendix C, more possibilities are discussed to improve the implementation and reduce computation times.

Further modifications to DE could be needed to apply the metaheuristic on larger networks. The exponential crossover procedure for DE has a crossover rate that is dependent on the dimension (as explained in Remark 7.5.1). Therefore, the crossover rate needs to be adjusted on other networks. Furthermore, exponential crossover only selects one block of variables from both the solutions  $x_l$  and  $y_l$ . For larger networks, it might be better to combine smaller blocks from both solutions. Furthermore, we have seen that DE could suffer preliminary

convergence, especially if the population size was too small. To avoid preliminary convergence, the ‘current-to-best/1’-mutation procedure could be combined with a different mutation procedure. It is also possible to test further variations on DE, like using a double population approach which worked well in Zhong et al. (2012).

There are more possibilities for future research. This thesis can easily be extended to timetables which are not perfectly cyclic, like the multi-period timetables found in Van der Knaap et al. (2025). The only downside is that more train services need to be optimised in such timetables. Also creating adjustments to train order can be considered, this can also be adjustments which should only be performed if specific delay scenarios happen. This would require extending the metaheuristics to adjust both time supplements and train orders.

To summarise, the following contributions were done in this thesis. The methodology from this thesis can be used to reallocate time supplements with metaheuristics. This can also be done without modifying the timetable outside of a considered network, contrary to many other heuristics found in literature. Promising results have been obtained for applying metaheuristics to adjust time supplements, which is a topic which has received limited attention in existing literature. Within limited computation time, the robustness could be improved considerably on a railway network between Utrecht and Arnhem. Especially the metaheuristic differential evolution showed fast convergence. Even though evaluation of robustness was done based on one thousand delay scenarios, which is more than used in most literature. The results in this thesis suggest that metaheuristics are a promising tool to achieve fast improvement of robustness of a railway timetable, which deserve more attention in future research.

# Bibliography

- Andersson, E. V., Peterson, A., & Krasemann, J. T. (2013). Quantifying railway timetable robustness in critical points. *Journal of Rail Transport Planning & Management*, 3(3), 95–110.
- Berg, P. (n.d.). *Ns-machinisten rijden nauwkeuriger en energiezuiniger dankzij timtim pro*. Info Support. <https://www.infosupport.com/cases/ns-machinisten-rijden-nauwkeuriger-en-energiezuiniger-dankzij-timtim-pro/>
- Bešinović, N. (2020). Resilience in railway transport systems: A literature review and research agenda. *Transport Reviews*, 40(4), 457–478.
- Bešinović, N., Goverde, R. M. P., Quaglietta, E., & Roberti, R. (2016). An integrated micro–macro approach to robust railway timetabling. *Transportation Research Part B: Methodological*, 87, 14–32.
- Bijma, F., Jonker, M., & Van der Vaart, A. (2017). *An introduction to mathematical statistics*. Amsterdam University Press.
- Bonyadi, M. R., & Michalewicz, Z. (2017). Particle swarm optimization for single objective continuous space problems: A review. *Evolutionary computation*, 25(1), 1–54.
- Bulmer, M. G. (2012). *Principles of statistics*. Courier Corporation.
- Burggraeve, S., & Vansteenwegen, P. (2017). Optimization of supplements and buffer times in passenger robust timetabling. *Journal of rail transport planning & management*, 7(3), 171–186.
- Cacchiani, V., Huisman, D., Kidd, M., Kroon, L. G., Toth, P., Veelenturf, L., & Wagenaar, J. (2014). An overview of recovery models and algorithms for real-time railway rescheduling. *Transportation Research Part B: Methodological*, 63, 15–37.
- Caimi, G., Kroon, L. G., & Liebchen, C. (2017). Models for railway timetable optimization: Applicability and applications in practice. *Journal of Rail Transport Planning & Management*, 6(4), 285–312.
- Carothers, N. L. (2000). *Real analysis*. Cambridge University Press.
- CBS. (2026). *Passagiersvervoer per spoor; reizigers, reizigerskilometers*. <https://www.cbs.nl/nl-nl/cijfers/detail/86172NED>
- Chow, A. H. F., & Pavlides, A. (2018). Cost functions and multi-objective timetabling of mixed train services. *Transportation Research Part A: Policy and Practice*, 113, 335–356.
- Corman, F., D'Ariano, A., Pacciarelli, D., & Pranzo, M. (2009). Evaluation of green wave policy in real-time railway traffic management. *Transportation Research Part C: Emerging Technologies*, 17(6), 607–616.
- Corman, F., D'Ariano, A., Pacciarelli, D., & Pranzo, M. (2012). Bi-objective conflict detection and resolution in railway traffic management. *Transportation Research Part C: Emerging Technologies*, 20(1), 79–94.
- Corman, F., D'Ariano, A., Pranzo, M., & Hansen, I. A. (2011). Effectiveness of dynamic reordering and rerouting of trains in a complicated and densely occupied station area. *Transportation Planning and Technology*, 34(4), 341–362.
- Corman, F., & Kecman, P. (2018). Stochastic prediction of train delays in real-time using bayesian networks. *Transportation Research Part C: Emerging Technologies*, 95, 599–615.

- Corman, F., & Quaglietta, E. (2015). Closing the loop in real-time railway control: Framework design and impacts on operations. *Transportation Research Part C: Emerging Technologies*, 54, 15–39.
- D'Ariano, A., Corman, F., Pacciarelli, D., & Pranzo, M. (2008). Reordering and local rerouting strategies to manage train traffic in real time. *Transportation science*, 42(4), 405–419.
- D'Ariano, A., Pacciarelli, D., & Pranzo, M. (2007). A branch and bound algorithm for scheduling trains in a railway network. *European journal of operational research*, 183(2), 643–657.
- D'Ariano, A., Pranzo, M., & Hansen, I. A. (2007). Conflict resolution and train speed coordination for solving real-time timetable perturbations. *IEEE Transactions on intelligent transportation systems*, 8(2), 208–222.
- De Jong, A. (2022). *The effectiveness of alternative graph-based rail traffic rescheduling models versus infrastructure layouts and traffic patterns* [Master's thesis, Delft University of Technology].
- Dewilde, T., Sels, P., Cattrysse, D., & Vansteenwegen, P. (2014). Improving the robustness in railway station areas. *European Journal of Operational Research*, 235(1), 276–286.
- Dollevoet, T., Huisman, D., Schmidt, M., & Schöbel, A. (2012). Delay management with rerouting of passengers. *Transportation science*, 46(1), 74–89.
- Duarte, P. J. C., Dauzère-Pérès, S., Huisman, D., Mannino, C., Sartor, G., Weik, N., & Widmann, P. (2025). 50 years of or in railway timetabling and rolling stock planning. *EURO Journal on Transportation and Logistics*, 14, 100155.
- Dündar, S., & Şahin, İ. (2013). Train re-scheduling with genetic algorithms and artificial neural networks for single-track railways. *Transportation Research Part C: Emerging Technologies*, 27, 1–15.
- El-Shorbagy, M. A., & Abd Elazeem, A. M. (2025). Convex combination search algorithm: A novel metaheuristic optimization algorithm for solving global optimization and engineering design problems. *Journal of Engineering Research*, 13, 2320–2339.
- Fan, Y., Guthrie, A., & Levinson, D. (2016). Waiting time perceptions at transit stops and stations: Effects of basic amenities, gender, and security. *Transportation Research Part A: Policy and Practice*, 88, 251–264.
- Fischetti, M., & Monaci, M. (2009). Light robustness. In R. K. Ahuja, R. H. Möhring, & C. D. Zaroliagis (Eds.), *Robust and online large-scale optimization: Models and techniques for transportation systems* (pp. 61–84). Springer.
- Fischetti, M., Salvagnin, D., & Zanette, A. (2009). Fast approaches to improve the robustness of a railway timetable. *Transportation Science*, 43(3), 321–335.
- Flamini, M., & Pacciarelli, D. (2008). Real time management of a metro rail terminus. *European Journal of Operational Research*, 189(3), 746–761.
- Fosgerau, M., Hjorth, K., & Vincent Lyk-Jensen, S. (2007). The danish value of time study: Data description. *The Danish Transport Research Institute*.
- Goverde, R. M. P. (1999). Improving punctuality and transfer reliability by railway timetable optimization. *Proceedings of the 5th TRAIL Annual Congress, TRAIL Research School, Delft*.
- Goverde, R. M. P. (2005). *Punctuality of railway operations and timetable stability analysis* [Doctoral dissertation, Delft University of Technology]. <https://resolver.tudelft.nl/uuid:a40ae4f1-1732-4bf3-bbf5-fdb8dfd635e7>
- Goverde, R. M. P. (2007). Railway timetable stability analysis using max-plus system theory. *Transportation Research Part B: Methodological*, 41(2), 179–201.
- Goverde, R. M. P. (2010). A delay propagation algorithm for large-scale railway traffic networks. *Transportation Research Part C: Emerging Technologies*, 18(3), 269–287.
- Goverde, R. M. P., Bešinović, N., Binder, A., Cacchiani, V., Quaglietta, E., Roberti, R., & Toth, P. (2016). A three-level framework for performance-based railway timetabling. *Transportation Research Part C: Emerging Technologies*, 67, 62–83.

- Goverde, R. M. P., Corman, F., & D'Ariano, A. (2013). Railway line capacity consumption of different railway signalling systems under scheduled and disturbed conditions. *Journal of rail transport planning & management*, 3(3), 78–94.
- Goverde, R. M. P., & Scheepmaker, G. M. (2023). Energy-efficient train timetabling. In S. Su, Z. Tian, & R. M. P. Goverde (Eds.), *Energy-efficient train operation: A system approach for railway networks* (pp. 69–101). Springer.
- Government of the Netherlands. (n.d.). *Mitigating climate change*. <https://www.government.nl/topics/climate-change/mitigating-climate-change>
- Gut, A. (2006). *Probability: A graduate course*. Springer.
- Hansen, I. A., Goverde, R. M. P., & Van der Meer, D. J. (2010). Online train delay recognition and running time prediction. *13th International IEEE Conference on Intelligent Transportation Systems*, 1783–1788.
- Hatem, G., Zeidan, J., Goossens, M., & Moreira, C. (2022). Normality testing methods and the importance of skewness and kurtosis in statistical analysis. *BAU Journal-Science and Technology*, 3(2), 7.
- Hensher, D. A., Ho, C. Q., Mulley, C., Nelson, J. D., Smith, G., & Wong, Y. Z. (2020). *Understanding mobility as a service (maas): Past, present and future*. Elsevier.
- Högdahl, J., & Bohlin, M. (2023). A combined simulation-optimization approach for robust timetabling on main railway lines. *Transportation Science*, 57(1), 52–81.
- Hoogenraad, J. H. (2021). Design and evaluation of a safe and effective das. *IRSE NEWS*, 278, 2–5. <https://www.spoorglaren.nl/site/sites/default/files/RolTijdIRSE2021.pdf>
- Ingvardson, J. B., Nielsen, O. A., Raveau, S., & Nielsen, B. F. (2018). Passenger arrival and waiting time distributions dependent on train service frequency and station characteristics: A smart card data analysis. *Transportation Research Part C: Emerging Technologies*, 90, 292–306.
- Jamili, A., Shafia, M. A., Sadjadi, S. J., & Tavakkoli-Moghaddam, R. (2012). Solving a periodic single-track train timetabling problem by an efficient hybrid algorithm. *Engineering Applications of Artificial Intelligence*, 25(4), 793–800.
- Jebari, K., & Madiafi, M. (2013). Selection methods for genetic algorithms. *International Journal of Emerging Sciences*, 3(4), 333–344.
- Jovanović, P., Kecman, P., Bojović, N., & Mandić, D. (2017). Optimal allocation of buffer times to increase train schedule robustness. *European Journal of Operational Research*, 256(1), 44–54.
- Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, 302–311.
- Katoch, S., Chauhan, S. S., & Kumar, V. (2021). A review on genetic algorithm: Past, present, and future. *Multimedia tools and applications*, 80(5), 8091–8126.
- Kecman, P., Corman, F., D'Ariano, A., & Goverde, R. M. P. (2013). Rescheduling models for railway traffic management in large-scale networks. *Public Transport*, 5(1), 95–123.
- Kleywegt, A. J., Shapiro, A., & Homem-de-Mello, T. (2002). The sample average approximation method for stochastic discrete optimization. *SIAM Journal on optimization*, 12(2), 479–502.
- Kompeer, J. (2023, December 21). *Spoorafspraken definitief: Tot 2033 alleen ns-treinen op belangrijkste lijnen*. NOS. <https://nos.nl/artikel/2502434-spoorafspraken-definitief-tot-2033-alleen-ns-treinen-op-belangrijkste-lijnen>
- Kroon, L. G., Dekker, R., & Vromans, M. J. C. M. (2007). Cyclic railway timetabling: A stochastic optimization approach. *Algorithmic Methods for Railway Optimization: International Dagstuhl Workshop, Dagstuhl Castle, Germany, June 20-25, 2004, 4th International Workshop, ATMOS 2004, Bergen, Norway, September 16-17, 2004, Revised Selected Papers*, 41–66.
- Kroon, L. G., Maróti, G., Retel Helmrich, M. J., Vromans, M. J. C. M., & Dekker, R. (2008). Stochastic improvement of cyclic railway timetables. *Transportation Research Part B: Methodological*, 42(6), 553–570.

- Larsen, R., Pranzo, M., D'Ariano, A., Corman, F., & Pacciarelli, D. (2014). Susceptibility of optimal train schedules to stochastic disturbances of process times. *Flexible Services and Manufacturing Journal*, 26(4), 466–489.
- Li, D., Goverde, R. M. P., Daamen, W., & He, H. (2014). Train dwell time distributions at short stop stations. *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2410–2415.
- Li, H., & Johnson, T. (2014). Wilcoxon's signed-rank statistic: What null hypothesis and why it matters. *Pharmaceutical statistics*, 13(5), 281–285.
- Liebchen, C., Lübbecke, M., Möhring, R., & Stiller, S. (2009). The concept of recoverable robustness, linear programming recovery, and railway applications. In R. K. Ahuja, R. H. Möhring, & C. D. Zaroliagis (Eds.), *Robust and online large-scale optimization: Models and techniques for transportation systems* (pp. 1–27). Springer.
- Liebchen, C., Proksch, M., & Wagner, F. H. (2008). Performance of algorithms for periodic timetable optimization. In M. Hickman, P. Mirchandani, & S. Voß (Eds.), *Computer-aided systems in public transport* (pp. 151–180). Springer.
- Liebchen, C., Schachtebeck, M., Schöbel, A., Stiller, S., & Prigge, A. (2010). Computing delay resistant railway timetables. *Computers & Operations Research*, 37(5), 857–868.
- Locatelli, M. (2000). Simulated annealing algorithms for continuous global optimization: Convergence conditions. *Journal of Optimization Theory and applications*, 104(1), 121–133.
- Luijt, R. S., Van den Berge, M. P. F., Willeboordse, H. Y., & Hoogenraad, J. H. (2017). 5 years of dutch eco-driving: Managing behavioural change. *Transportation Research Part A: Policy and Practice*, 98, 46–63.
- Maróti, G. (2017). A branch-and-bound approach for robust railway timetabling. *Public Transport*, 9(1), 73–94.
- Mascis, A., & Pacciarelli, D. (2002). Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, 143(3), 498–517.
- Mazzarello, M., & Ottaviani, E. (2007). A traffic management system for real-time traffic optimisation in railways. *Transportation Research Part B: Methodological*, 41(2), 246–274.
- McElduff, F., Cortina-Borja, M., Chan, S.-K., & Wade, A. (2010). When t-tests or wilcoxon-mann-whitney tests won't do. *Advances in physiology education*, 34(3), 128–133.
- Monsuur, F., Enoch, M., Quddus, M., & Meek, S. (2021). Modelling the impact of rail delays on passenger satisfaction. *Transportation Research Part A: Policy and Practice*, 152, 19–35.
- Nitisiri, K., Gen, M., & Ohwada, H. (2019). A parallel multi-objective genetic algorithm with learning based mutation for railway scheduling. *Computers & Industrial Engineering*, 130, 381–394.
- NS. (2025). *Reizigersgedrag 2025*. <https://dashboards.nsjaarverslag.nl/reizigersgedrag>
- NS. (2026). *Performance agreements*. <https://www.nsannualreport.nl/annual-report-2025/performance-what-we-have-achieved/our-operations/performance-agreements>
- Odijk, M. A. (1996). A constraint generation algorithm for the construction of periodic railway timetables. *Transportation Research Part B: Methodological*, 30(6), 455–464.
- Pan, J.-S., Chen, J., Li, J., & Meng, Z. (2026). Differential evolution with exponential crossover: A survey. *Computer Science Review*, 62, 100990.
- Pant, M., Zaheer, H., Garcia-Hernandez, L., Abraham, A., & Bilal. (2020). Differential evolution: A review of more than two decades of research. *Engineering Applications of Artificial Intelligence*, 90, 103479.
- Papadimitriou, C. H. (1981). On the complexity of integer programming. *Journal of the ACM (JACM)*, 28(4), 765–768.
- Patil, V. P., & Pawar, D. D. (2015). The optimal crossover or mutation rates in genetic algorithm: A review. *International Journal of Applied Engineering and Technology*, 5(3), 38–41.

- Pavlovic, M. (2022, February 16). *Log-normal distribution - a simple explanation*. Medium. <https://medium.com/data-science/log-normal-distribution-a-simple-explanation-7605864fb67c>
- Pinedo, M. L. (2022). *Scheduling: Theory, algorithms, and systems* (6th ed.). Springer.
- Piovesana, A., & Senior, G. (2018). How small is big: Sample size and skewness. *Assessment*, 25(6), 793–800.
- ProRail. (2020, June 29). *Strakker plannen vanaf dienstregeling 2022*. <https://www.prorail.nl/nieuws/strakker-plannen-vanaf-dienstregeling-2022>
- ProRail. (2026, February 6). *Ontwikkeling spoorgoederen in nederland*. <https://www.prorail.nl/siteassets/homepage/over-ons/documenten/pr-jaarrapport-spoorgoederenverkeer-2025.pdf>
- Quaglietta, E., Corman, F., & Goverde, R. M. P. (2013). Stability analysis of railway dispatching plans in a stochastic and dynamic environment. *Journal of Rail Transport Planning & Management*, 3(4), 137–149.
- Railway Technical. (2023). *Signalling*. <http://www.railway-technical.com/signalling/>
- Rijksoverheid. (2023). *Concessie voor het hoofdrailnet: 2025-2033*. Ministerie van Infrastructuur en Waterstaat. <https://open.overheid.nl/documenten/dpc-365c3ea388f96bee ff9723733902db8c526d3a81/pdf>
- Rijksoverheid. (2026, January 13). *Beperkte mogelijkheden om te investeren in infrastructuur*. <https://www.rijksoverheid.nl/actueel/nieuws/2026/01/13/beperkte-mogelijkheid-en-om-te-investeren-in-infrastructuur>
- Rijkswaterstaat. (2025). *Mobiliteitsvraag per vervoerwijze*. <https://www.rijkswaterstaat.nl/zakelijk/open-data/modellen-en-applicaties/verkeers-en-vervoermodellen/jaarlijks-e-prognoses/belangrijkste-prognoses/mobiliteitsvraag-per-vervoerwijze>
- Rohatgi, V. K., & Ehsanes Saleh, A. K. M. (2015). *An introduction to probability and statistics*. John Wiley & Sons.
- Ronkkonen, J., Kukkonen, S., & Price, K. V. (2005). Real-parameter optimization with differential evolution. *2005 IEEE congress on evolutionary computation*, 1, 506–513.
- Scheepmaker, G. M., & Goverde, R. M. P. (2015). The interplay between energy-efficient train control and scheduled running time supplements. *Journal of Rail Transport Planning & Management*, 5(4), 225–239.
- Schöbel, A., & Kratz, A. (2009). A bicriteria approach for robust timetabling. In R. K. Ahuja, R. H. Möhring, & C. D. Zaroliagis (Eds.), *Robust and online large-scale optimization: Models and techniques for transportation systems* (pp. 119–144). Springer.
- Shelat, S., Cats, O., & Van Lint, J. W. C. (2021). Quantifying travellers' evaluation of waiting time uncertainty in public transport networks. *Travel Behaviour and Society*, 25, 209–222.
- Shukla, A., Pandey, H. M., & Mehrotra, D. (2015). Comparative review of selection techniques in genetic algorithm. *2015 international conference on futuristic trends on computational analysis and knowledge management (ABLAZE)*, 515–519.
- Storn, R. (1996). On the usage of differential evolution for function optimization. *Proceedings of North American fuzzy information processing*, 519–523.
- Storn, R., & Price, K. (1997). Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4), 341–359.
- Talbi, E.-G. (2009). *Metaheuristics: From design to implementation*. John Wiley & Sons.
- Tan, Y.-Y., Li, Y.-X., & Wang, R.-X. (2020). Scheduling extra train paths into cyclic timetable based on the genetic algorithm. *IEEE Access*, 8, 102199–102211.
- Tormos, P., Lova, A., Barber, F., Ingolotti, L., Abril, M., & Salido, M. A. (2008). A genetic algorithm for railway scheduling problems. In F. Khafa & A. Abraham (Eds.), *Metaheuristics for scheduling in industrial and manufacturing applications* (pp. 255–276). Springer.
- Van der Knaap, R. J. H., Van Oort, N., & Goverde, R. M. P. (2025). Multi-period railway timetabling to serve time-dependent demand. *Journal of Rail Transport Planning & Management*, 35, 100536.

- Van Seters, D., & Van den Berg, R. (2026). *Stream personenvervoer: Emissiekentallen modaliteiten 2025*. CE Delft. [https://ce.nl/wp-content/uploads/2026/03/CE\\_Delft\\_210506\\_STREAM\\_Personenvervoer\\_2025\\_Def.pdf](https://ce.nl/wp-content/uploads/2026/03/CE_Delft_210506_STREAM_Personenvervoer_2025_Def.pdf)
- Wang, M., Wang, L., Xu, X., Qin, Y., & Qin, L. (2019). Genetic algorithm-based particle swarm optimization approach to reschedule high-speed railway timetables: A case study in china. *Journal of Advanced Transportation*, 2019(1), 6090742.
- Wiggenraad, P. B. L. (2001). Alighting and boarding times of passengers at dutch railway stations. *TRAIL Research School, Delft*, 1–21.
- Wikimedia Commons. (2017a, November 5). *File:ns ddz treinstel.svg*. [https://commons.wikimedia.org/wiki/File:NS\\_DDZ\\_Treinstel.svg](https://commons.wikimedia.org/wiki/File:NS_DDZ_Treinstel.svg)
- Wikimedia Commons. (2017b, November 5). *File:ns koploper treinstel.svg*. [https://commons.wikimedia.org/wiki/File:NS\\_DDZ\\_Treinstel.svg](https://commons.wikimedia.org/wiki/File:NS_DDZ_Treinstel.svg)
- Wu, Q., Cole, C., & McSweeney, T. (2016). Applications of particle swarm optimization in the railway domain. *International Journal of Rail Transportation*, 4(3), 167–190.
- Xu, P., Corman, F., Peng, Q., & Luan, X. (2017). A train rescheduling model integrating speed management during disruptions of high-speed traffic under a quasi-moving block system. *Transportation Research Part B: Methodological*, 104, 638–666.
- Yan, J., Guo, C., & Gong, W. (2011). Hybrid differential evolution with convex mutation. *Journal of software*, 6(11), 2321–2328.
- Yuan, J. (2006). *Stochastic modelling of train delays and delay propagation in stations* [Doctoral dissertation, Delft University of Technology].
- Zabinsky, Z. B. (2011). Random search algorithms. *Wiley Encyclopedia of Operations Research and Management Science*.
- Zhong, J.-H., Shen, M., Zhang, J., Chung, H. S.-H., Shi, Y.-H., & Li, Y. (2012). A differential evolution algorithm with dual populations for solving periodic railway timetable scheduling problem. *IEEE Transactions on Evolutionary Computation*, 17(4), 512–527.

# Appendix A

## Complexity class

In Kroon et al. (2008), the problem has been formulated as an MILP. For readability, we will use the term event times to refer to both departure and arrival times. For each event time that needs to be planned, a variable is created. Constraints are also formulated on these variables. To check the quality of a timetable, delay scenarios are used. In the case of Kroon et al. (2008), 20 days consisting of 21 hours are used, which leads to 420 realised event times for each scheduled event. The delay is then measured by comparing all realised event times to the scheduled times. If the variables for all scheduled times have been determined, the realised event times can easily be calculated. The following constraints need to be satisfied, a realised event time needs to be bigger than

- The schedule time
- Realised event time of previous event of the same train + process time
- Realised event time of previous train with conflicting route + minimum headway time

To minimise delay, realised event times should be as low as possible. The realised event time can therefore be calculated as the maximum of these three times. So once scheduled times have been determined, the realised event times are implied by this. However, in a linear programming setting, we cannot use a maximum directly. Instead, an extra variable has to be created for each realised event time. Also three constraints are added per realised event time, one for each element in the maximum. So the number of variables and constraints in a linear programming setting grow asymptotically linear with the number of delay scenarios. Even though the schedule times are the only variables that actually need to be determined, while the other variables are implied by this.

So we do get a large-scale linear programming model. It is known that a linear programming model can be solved in polynomial time in terms of the number of variables (Karmarkar, 1984). But this only holds if there are no integrality constraints on the variables. Solving general integer linear programming problems is NP-complete (Papadimitriou, 1981). Since event times are rounded to full minutes in Kroon et al. (2008), this is disadvantageous. Some timetabling problems can be formulated with constraints  $Ax \leq b$  for  $A$  totally unimodular and  $b$  integer (Goverde, 1999). Then, if the integrality constraints are removed, an integral solution is still obtained. But since the realised event times are not rounded in Kroon et al. (2007), an MILP needs to be optimised. This does not satisfy the given conditions to remove integrality constraints. So a large scale linear programming model including integrality constraints needs to be optimised, which leads to long computation times.

## Appendix B

# Alternative graphs

In this appendix, we will discuss alternative graph models. They are a popular tool to determine an adjusted train order during real-time rescheduling. The directed graph introduced in Chapter 3 was inspired by alternative graphs. However, since we do not vary the train order, alternative graphs are not used in this thesis. Therefore, reading the information in this appendix is not required to understand the rest of the thesis.

Alternative graphs have been developed to solve job shop scheduling problems. We will first discuss the concept of job shop scheduling and how this can be applied to railway timetabling in Section B.1. Thereafter, we will introduce alternative graphs in Section B.2 and discuss their use for real-time rescheduling in Section B.3.

### B.1 Job shop scheduling

Job shop scheduling is a popular mathematical framework used for scheduling. It considers the scenario where  $n$  jobs need to be scheduled in a factory. In the factory, there are  $m$  different machines available. For every job there is a predetermined route of machines planned that need to be visited, and there also is a time that needs to be spent on each of the machines. It is possible that jobs are not available from the start of the planning period, then jobs can have a release date. There can also be deadlines for when a job needs to be finished. The goal of the job shop scheduling problem is finding a schedule for jobs on machines that optimises a certain objective function. This can be minimising the completion time of the last job, minimising the (weighted) average completion time or minimising the total time that jobs are late to their deadline. Although many more choices for the objective function are possible (Pinedo, 2022). There also exist a lot of variations on the job shop scheduling problem. We will now explain how a railway scheduling problem can be formulated as a job scheduling problem.

To model railway scheduling as a job shop scheduling problem, it needs to be decided what are the jobs and machines. A popular choice is to see the trains as jobs and the block sections in the network as machines. Every train will have a predetermined route of block sections it needs to complete inside a network. The time that needs to be spent in a block section is the time required for a train to complete the block section. This can also include time for dwelling if the block section is inside a station. A train cannot enter a block section until the train ahead has completed that block section. So the next block must be free, before a train can leave the current block. This is called a blocking constraint in a job shop scheduling context (Pinedo, 2022). This name is not originating from block sections, but from a machine being blocked until the next one is available. An additional assumption that can be made are no-wait constraints (also known as no store constraints). This means that a train must spend exactly the required time inside a block section. It is required that the next

block section must always be available after the current block section is completed. Often this assumption is not present at stations, where longer dwelling is allowed to accomplish satisfying the no-wait constraints on open track.

## B.2 Alternative graph model

In the previous section, we have discussed the job shop scheduling problem. We will now discuss how a job shop scheduling problem with blocking constraints was transformed into a graph-based formulation by Mascis and Pacciarelli (2002). In this graph, vertices denote the time a job starts its operation on a machine. There are two additional dummy vertices. The first dummy vertex  $a$  is the source vertex required for incorporating release dates of jobs in the model. The other dummy vertex  $z$  is the sink vertex used to measure the exit time of the last completed job.

In the graph, there are two kind of arcs present at the start: fixed and alternative. The weight of an arc describes the minimum time that needs to be spent between the operation in the head and the one in the tail of the arc. There are fixed arcs present from the source vertex  $a$  to the start of the first operation for every job, this is used to make sure that jobs cannot start their first operation before their release date. Fixed arcs are also present between consecutive machines of the same job. A job cannot enter the next machine until it has spend the required time on the current machine. Therefore, an arc is added from the processing start on the current machine to the processing start on the next machine, where the weight describes the time that needs to be spend on the current machine. At the last operation that needs to be completed on a job, the arc leads to the dummy vertex  $z$  which measures the completion time.

The second type of arcs are alternative arcs, they are used to determine the order in which jobs are processed. We have previously established that under blocking constraints, a job needs to move to a new machine before the current machine can be used. Arcs need to be added to the graph, which make sure that two jobs can not use the same machine simultaneously. Assume that there are jobs A and B that both need to visit machine 1, followed by machine 2. If job A gets priority, then an arc needs to be added from A2 to B1. This means that job B cannot enter machine 1 before job A has entered machine 2. If job B gets priority instead, then an arc from B2 to A1 needs to be added. These two possible arcs are called a pair of alternative arcs, an example of this is shown in Figure B.1. A pair of alternative arcs is created for every pair of jobs that need to be processed on the same machine. One of the arcs in a pair needs to be selected. A selection consists of all alternative arcs that have been selected. Once an arc has been selected from every pair, the selection is complete. A larger network with three jobs and four machines is shown in Figure B.2.

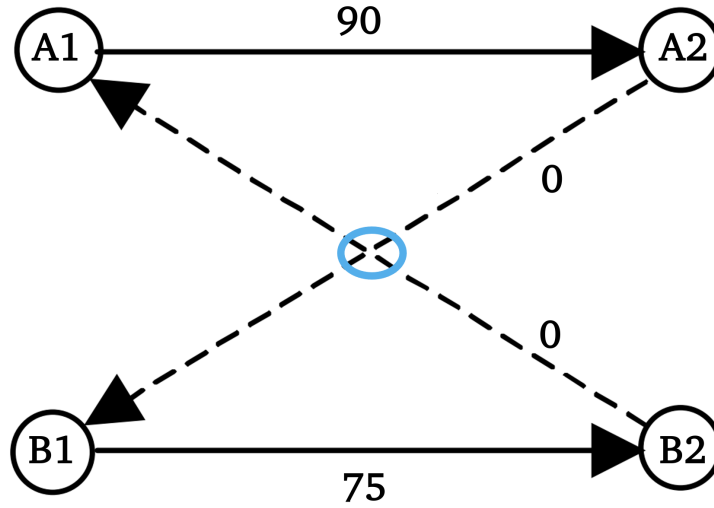


**Figure B.1:** Example of a pair of alternative arcs. Jobs A and B need to visit machines 1 and 2. A fixed arc (uninterrupted arrow) for job A is given with weight 90, to describe that 90 seconds need to be spent on machine 1 before the job can continue to machine 2. Similarly, for job B there is a fixed arc that describes that 75 seconds need to be spent on machine 1 before it can continue to machine 2. The pair of alternative arcs is denoted with dashed arrows. If the arrow from bottom right to top left is selected, then job A cannot start on machine 1 until job B has entered machine 2. If the arrow from top right to bottom left is selected instead, then job B cannot enter machine 1 until job A has entered machine 2. Since the weight of the alternative arcs is 0, there is no additional waiting time required. This figure is adapted from Figure 3 in Mascis and Pacciarelli (2002).

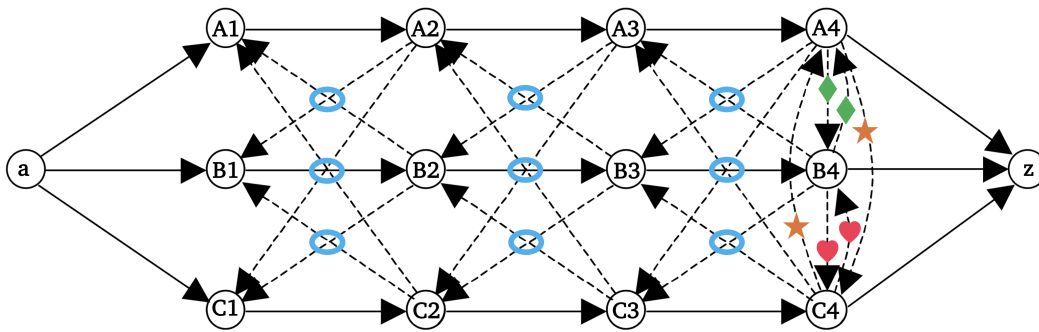


**Figure B.2:** Example of an alternative graph for three jobs (A, B and C) and four machines (1, 2, 3 and 4). The weights of the arcs have been omitted for clarity. Two alternative arcs that form a pair have been marked with a blue ellipse if the arcs cross. For the arcs connecting A4, B4 and C4 the pairs have been marked with the same shape instead (orange star, green diamond or pink heart). This figure is adapted from Figure 5 in Mascis and Pacciarelli (2002).

The elegance of the alternative graph is that it is easy to check if a solution is feasible. A selection is feasible if there is no positive cycle in the graph. If a positive cycle would exist, then this would mean that the execution of a job on a machine should precede itself. A selection which does not cause a positive cycle is called consistent. The goal is to find a complete consistent selection that optimises an objective function. By using a branch and bound approach, an optimal complete consistent solution can be found as long as a complete consistent solution exists. There do exist implication rules to help avoid cycles. For example,

if two jobs have the same route, then the order will be the same across the entire route. So if one alternative arc has been selected for these jobs, the other alternative arcs between these jobs are implied by this (Mascis & Pacciarelli, 2002).

### B.3 Real-time rescheduling

In this section, the concept of alternative graphs will be used for real-time rescheduling. For real-time rescheduling, a timetable has already been made. So time supplements and buffer times have already been fixed. Adjustments to the timetable need to be made to minimise the impact of disturbances and disruptions. This often includes swapping the order of trains. To quantify the minimisation of hindrance, often delay is subdivided into primary and secondary delay. Primary delay includes all delay which a train would suffer without being hindered by other trains, these delays are caused by disturbances or disruptions. Secondary delay includes all extra delay that arises due to being hindered by other trains. Sometimes, in real time rescheduling only the amount of secondary delay is minimised.

In D'Ariano, Pacciarelli, and Pranzo (2007), the alternative graph from Mascis and Pacciarelli (2002) was used to create a branch and bound algorithm for real-time rescheduling of trains. Most vertices in the graph represented entrance times of trains into a block section. Every train also had an additional vertex for network exit. At stations, additional vertices were created for platform arrival and departure. Trains should not depart from a station before their scheduled time, to avoid passengers missing their train. To prevent this from happening, arcs were added from the source vertex to each departure event. Furthermore, arcs were added from each arrival to a sink vertex. This was used to measure the secondary delay. The objective function was minimising the maximum secondary delay that was measured in the network.

As explained in Section 2.1.2, trains need a green signal to travel at full speed. A green signal is only shown if the two upcoming block sections are available. To make sure that intercity trains only encounter green signals, arcs are added in D'Ariano, Pacciarelli, and Pranzo (2007). These arcs connect the block entrance of a train, with the next train entering two block sections before it. They do not use the same assumption for sprinter trains, since they are considered slow. But this is not realistic, since sprinter trains travel at much higher speeds than the 40 km/h usually allowed under a yellow signal. The alternative arcs are shown in Figure B.3.

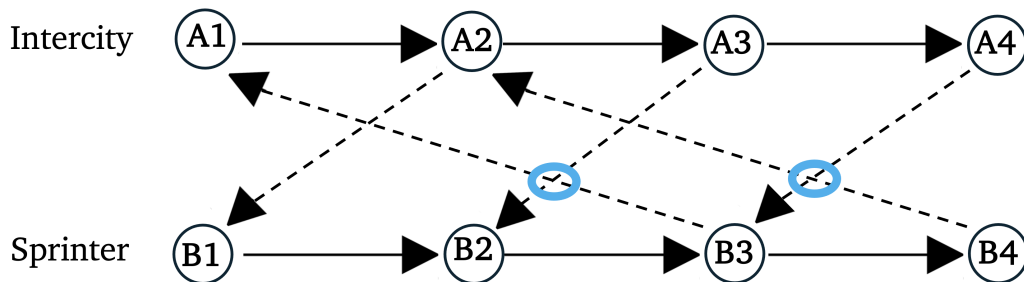


Figure B.3: Blocking constraint used in D'Ariano, Pacciarelli, and Pranzo (2007).

There are two more flaws in the assumptions of D'Ariano, Pacciarelli, and Pranzo (2007). Firstly, they assume that trains can wait in a block section until a green signal is encountered. However, this requires a driver advisory system which allows drivers to monitor the train ahead. Such a system was not in use at NS at the time (Luijt et al., 2017). However, even if green signals can be avoided by driving slower, this will impact the attainable travel time over the next block section. Therefore, in a follow-up paper (D'Ariano, Pranzo, & Hansen, 2007) an iterative procedure was used where travel times were updated if a yellow or red signal is encountered. On the same delay scenarios, this meant that 30% of trains were delayed more than two minutes, while this was previously 7.7% (D'Ariano, Pranzo, & Hansen, 2007). The created algorithm was later extended to also include rerouting possibilities (D'Ariano et al., 2008). This led to the creation of the ROMA solver (Railway traffic Optimisation by Means of Alternative graphs). ROMA has been used for rescheduling in at least ten more papers (De Jong, 2022).

There do exist more studies that create a real-time rescheduling model based on alternative graphs: Flamini and Pacciarelli (2008), Mazzarello and Ottaviani (2007), and Xu et al. (2017). Corman et al. (2009) investigated the effect of using a green-wave policy for real-time rescheduling. This means that trains cannot encounter yellow or red signals, contrary to D'Ariano, Pranzo, and Hansen (2007). Instead, trains must stay at a station, until they have a path to the next station where they do not need to slow down due to hindrance from other trains. This is equal to using no-wait constraints in a job shop scheduling context.

# Appendix C

## Implementation

In this appendix, the implementation used to gather results will be explained. Firstly, Section C.1 contains the general explanation of how the ideas from Chapter 3 were implemented. Subsequently, Section C.2 explains how Algorithm 3 worked slightly different in the code. Section C.3 explains a mistake that was made in this algorithm, which did have negligible impact on the results. Finally, Section C.4 features a link to a GitHub-repository, which contains the implementation.

### C.1 General implementation

In Chapter 3, a graph  $G = (V, A, t)$  with additional weight functions  $t^1, \dots, t^r$  is used to evaluate the timetable. When looking at the implementation, the graph is not as clearly noticeable. We will now explain how the implementation works instead.

In Chapter 3, the vertex set consists of a source vertex, schedule vertices and event vertices. In the implementation, schedule vertices are not formally defined. The values for  $x_{i,j,k}$  are calculated based on  $o_{i,j}$  and  $\varepsilon_{i,j}$ . Subsequently, this value is coupled to the vertex  $e_{i,j,k}$  with the help of object oriented programming. So no separate schedule vertices are created, contrary to the the approach from Chapter 3.

In the implementation, also no arc set  $A$  or length function  $t$  is defined. Instead, for a vertex  $e_{i,j,k}$ , we attach incoming arcs and their corresponding length. We do always create an incoming headway arc and an incoming process arc. The process arc can be a running, dwelling or through arc. Process arcs are not required for the first activity of a train, while headway arcs are not required for the first train travelling on a track. If this was the case, we did instead create a process or headway arc from the source vertex. This was done to simplify implementation, but did result in 235 extra arcs created for the test case. For each vertex, we did create extra variables to check if an incoming platform arc existed. Therefore, an event time was calculated as follows, where  $\text{plan}(e_{i,j,k})$  denotes whether an event is scheduled (as introduced in Chapter 3).

**Step 1** Calculate  $e_{i,j,k}$  based on incoming process and headway arc.

**Step 2** If a platform arc exists, update  $e_{i,j,k}$  based on the platform arc.

**Step 3** If  $\text{plan}(e_{i,j,k})$ , then  $e_{i,j,k} \leftarrow \max\{e_{i,j,k}, v_{i,j,k}\}$ .

There is a downside to using this implementation instead of taking the maximum using the set  $A_{\text{in}}(v)$ . An unnecessary if-statement needs to be checked in step 2.

So incoming arcs are used to calculate event times, it does need to be determined which vertices are connected. This is especially hard for the headway arcs, where it needs to be

decided which trains run consecutively. To accomplish this, the network has been split into segments where trains travel in the same order. For each train, we determined where in the order it was on each segment. As explained in Chapter 3, we created both a station entrance and exit for trains running through a station. This ensured that trains travelling on the same segment, also had to complete the same activities. This was convenient for implementation. However, it is not the most efficient. For through-running trains, it is also possible to combine station entrance and exit into one through running event. For the test case, this would reduce the number of vertices by 14.5% and the number of arcs by 7%. So this is a possibility for further improvement.

Rounding solutions such that all schedule times are in full seconds is done differently than explained in Section 3.2. The  $o_{i,j}$ -parameters are calculated based on 104% of the technical minimal running time. The total supplement  $s_i$  for each train line was calculated based on these  $o_{i,j}$ -parameters, without rounding all  $o_{i,j}$ -parameters to full seconds. Subsequently, after the  $\varepsilon_{i,j}$ -variables were determined, rounding schedule times to full seconds was done as follows. The  $o_{i,j}$ -parameters were rounded up to full seconds and the  $\varepsilon_{i,j}$ -variables were rounded down to full seconds. If this led to the last schedule time not being equal to the schedule time from the original timetable, the last  $\varepsilon_{i,j}$ -variables was adjusted to correct this. However, with unlucky rounding it is possible that the last  $\varepsilon_{i,j}$ -variable for a train became negative due to this. If this happened, the created timetable was rejected due to being infeasible.

If all  $o_{i,j}$ -parameters would have been rounded before we determined the  $s_i$ -variables, we would generate less infeasible solutions. In that case, the trajectory of each individual train would be attainable. Only if a train is hindered by another train, the timetable could become unattainable. Therefore, in Section 3.2 we did include rounding  $o_{i,j}$ -parameters before calculating  $s_i$ . The rounding system that was used during implementation, did still lead to about 60% of solutions being feasible for the best parameters selected for differential evolution. We will also see in the next section that discarding infeasible solutions is relatively cheap. Therefore, it is likely that getting extra infeasible solutions due to the different rounding procedure, had little impact on the performance on the considered test case. However, for a larger test case with more train lines there is a higher risk of getting at least one train line with unlucky rounding, which makes a solution infeasible. Therefore, it is recommended to adjust the rounding procedure from the implementation before examining a larger test case.

## C.2 Adjusted algorithm 3

As announced in Remark 3.6.2, we have implemented an alternative version to Algorithm 3. In Algorithm 3, we calculate the nominal event times (as calculated in the feasibility check) alongside the delay propagation. At every considered event vertex that is scheduled, we check if the nominal event time  $e_{i,j,k}$  exceeds the schedule time  $x_{i,j,k}$ .

A different approach is used in the implemented version, which is given in Algorithm 6. We will not check if the nominal event time exceeds the schedule time at each single vertex. In the first phase of the algorithm, lines 6-10, we do compute only the nominal event times for a smaller set of vertices  $V_e^f$ . These vertices will be traversed in topological order, which we denote as  $\text{ord}(V_e^f, A)$ . We used a set  $V_e^f$ , which contained all vertices for the eight trains depicted in Figure C.1. On these vertices, we do immediately check if the nominal event time exceeds the schedule time (lines 9-10). If there is no vertex in  $V_e^f$  with an unattainable schedule time, then we presume that the solution is feasible.

In the second phase of the algorithm (lines 11-21), we calculate the delay propagation. The delays  $\ell(x, \delta^1), \dots, \ell(x, \delta^R)$  are calculated as the variables  $\ell^1, \dots, \ell^R$ . We also calculate a

variable  $\ell$ , which is the delay under the nominal event times. This can be calculated at little additional cost, by parallelising lines 19-21. After calculating all delay propagation, in line 22 the presumption that the timetable is feasible is checked.

The advantage of Algorithm 6 compared to Algorithm 3 is that we do not have to check that  $e_{i,j,k} > x_{i,j,k}$  at every vertex. This does have the downside that the algorithm is not terminated intermediately if this happens at a vertex. So it is important that the set  $V_e^f$  can filter out a large majority of infeasible timetables. The advantage of Algorithm 6 is that no set  $V_e^f$  has to be created and that we do not traverse a part of the graph twice. We have decided to include Algorithm 3 in the main text because it is easier to understand and is slightly faster. However, since it is only about 2.5% faster on the test case, it was decided not to redo the experiments performed with Algorithm 6.

**Remark C.2.1.** As explained in the previous section, calculating  $e_{i,j,k}$  and  $e_{i,j,k}^r$  is not done using a set  $A_{in}(v)$ . But for readability, we have not incorporated this in Algorithm 6.

**Algorithm 6:** Algorithm to determine feasibility, delay propagation and objective value

**Input :** Graph  $G = (V, A, t)$   
Sequences  $\text{ord}(V_e, A)$ ,  $\text{ord}(V_x, A)$  and  $\text{ord}(V_e^f, A)$   
Incoming arc set  $A_{in}(v)$  for all  $v \in V$   
Number of delay scenarios  $R$   
Arc length functions  $t^1, t^2, \dots, t^R$  based on delay scenarios  $\Delta_R$   
Function  $\text{plan}(e_{i,j,k})$  (defined for all  $e_{i,j,k} \in V_e$ )  
Set of scheduled indices  $U$   
Weights  $w_{i,j,k}$  for  $(i, j, k) \in U$   
Bonus term function  $h(\cdot, \cdot)$   
Bonus term indicators  $b_{i,j,k}$  for  $(i, j, k) \in U$

**Output:** Feasibility, objective value  $\mathcal{L}$  and  $\ell^r$ -values

```

1  $v_0 \leftarrow 0$ 
2 for  $x_{i,j,k} \in \text{ord}(V_x, A)$  do
3    $x_{i,j,k} \leftarrow \max\{u + t(u, x_{i,j,k}) \mid (u, x_{i,j,k}) \in A_{in}(x_{i,j,k})\}$ 
4   for  $r \in \{1, 2, \dots, R\}$  do
5      $x_{i,j,k}^r \leftarrow x_{i,j,k}$ 
6 for  $e_{i,j,k} \in \text{ord}(V_e^f, A)$  do
7    $e_{i,j,k} \leftarrow \max\{u + t(u, e_{i,j,k}) \mid (u, e_{i,j,k}) \in A_{in}(e_{i,j,k})\}$ 
8   if  $\text{plan}(e_{i,j,k})$  then
9     if  $e_{i,j,k} > x_{i,j,k}$  then
10      return False (The solution is not feasible)
11  $\ell \leftarrow 0$ 
12 for  $r \in \{1, 2, \dots, R\}$  do
13    $\ell^r \leftarrow 0$ 
14 for  $e_{i,j,k} \in \text{ord}(V_e, A)$  do
15    $e_{i,j,k} \leftarrow \max\{u + t(u, e_{i,j,k}) \mid (u, e_{i,j,k}) \in A_{in}(e_{i,j,k})\}$ 
16   for  $r \in \{1, 2, \dots, R\}$  do
17      $e_{i,j,k}^r \leftarrow \max\{u^r + t^r(u, e_{i,j,k}) \mid (u, e_{i,j,k}) \in A_{in}(e_{i,j,k})\}$ 
18   if  $\text{plan}(e_{i,j,k})$  then
19      $\ell \leftarrow \ell + w_{i,j,k}(e_{i,j,k} - x_{i,j,k}) + b_{i,j,k} \cdot h(e_{i,j,k}, x_{i,j,k})$ 
20     for  $r \in \{1, 2, \dots, R\}$  do
21        $\ell^r \leftarrow \ell^r + w_{i,j,k}(e_{i,j,k}^r - x_{i,j,k}) + b_{i,j,k} \cdot h(e_{i,j,k}^r, x_{i,j,k})$ 
22 if  $\ell > 0$  then
23   return False (the solution is not feasible)
24  $\mathcal{L} \leftarrow \frac{1}{R} \sum_{r=1}^R \ell^r$ 
25 return  $\mathcal{L}$  (and all  $\ell^r$ )

```

Algorithm 6 does make good use of the cyclicity of the timetable. For cyclic timetables, an unattainable schedule time will usually occur already during the first cycle. So it is advisable to first check all vertices corresponding to the first few trains, to have a higher chance to filter out infeasibility early on. This is also a point of attention for Algorithm 3, it does make a difference in which order the vertices in  $V_e$  are considered.

What both Algorithms 3 and 6 do not do efficiently, is calculating the schedule times for infeasible solutions. We always calculate all schedule times before working on event times. This does not matter for feasible solutions, where we need to traverse the entire graph anyway. However, if infeasibility can be concluded based on the first cycle, it is a waste to set the schedule times for all cycles. We could create a topological order, where only the strictly necessary schedule vertices precede an event vertex. Then, determining infeasibility with the check from lines 1-10 of Algorithm 6 would be about 30% faster.

The inefficiency of filtering out infeasible solutions has had a limited effect on the performance measured in Chapter 7. Calculating lines 1-10 in Algorithm 6 is only about 6% of the computation time of the entire algorithm (if there is no intermediate termination, which is true for feasible solutions). All parameter combinations tested in Chapter 7 achieved a feasibility of at least 20%. For this level of feasibility, the computation time is reduced by at most 7%. This makes it unlikely that some parameter combinations suddenly start performing much better. For the selected parameters for differential evolution, about 60% of created solutions is feasible. Then, the computation time would only be reduced by at most 1.5%. However, a more efficient check for infeasibility can help for improving computation times on larger networks, when it is possible that much more infeasible solutions are created.

### C.3 Insufficient feasibility check

For performing the grid searches in Chapter 7, Algorithm 6 did not work as intended. During lines 11-24, we did not calculate  $e_{i,j,k}$  and update  $\ell$ . Lines 11, 15, 19, 22 and 23 were missing. Therefore, it was only checked whether a solution was feasible on the set  $V_e^f$ . Luckily, this was sufficient to filter out infeasible solutions.

Algorithm 4 with  $c^{\text{add}} = 20$  and  $c^{\text{mult}} = 0.25$  was used to test whether infeasible timetables were often approved. Firstly, this algorithm created 1 000 000 solutions based on the original timetable, that were feasible according to the incorrect feasibility check. In zero cases, the timetable was actually infeasible. Subsequently, also timetables were generated with Algorithm 4 starting from the optimal solution found by the Differential Evolution algorithm in Chapter 7. Out of 600 000 timetables that were feasible according to the incorrect feasibility check, none turned out to be infeasible. So it was really rare that an infeasible timetable was generated. Therefore, it is unlikely that infeasible solutions have affected the grid searches. Thus it was decided that results of the grid searches were still representative and did not need to be redone.

That no infeasible timetables were generated can be explained. In Remark 3.6.1, it was already mentioned that infeasibility in later cycles only occurs if event times still shift. An event time can shift, if a different train hinders it. For the  $V_f$  that we considered, this could have happened for line 31. This can best be explained on the basis of Figure C.1, which contains all trains in  $V_f$ . The first operation of line 31 is not hindered by a previous train. Therefore, a second operation of line 31 is considered, which is hindered by line 32. It is possible that the second operation of line 31 has to use different event times to keep headway to line 32. If this leads to a schedule time of line 31 to be unattainable, it is determined that the timetable is infeasible. However, it is not checked whether the shift in event times could hinder line 75 and cause a schedule time for this line to be unattainable.

In the original timetable, at least four minutes of buffer was available between lines 32 and 31 on top of the minimum headway. Therefore, a large adjustment to the timetable would have been required for line 32 to hinder line 31. Therefore, the two operations of line 31 could follow the same trajectory. Since all trajectories of other lines also led to attainable schedule times, this timetable could be replicated for an entire day. So no more infeasibilities could occur in later cycles, unless the timetable was heavily modified.

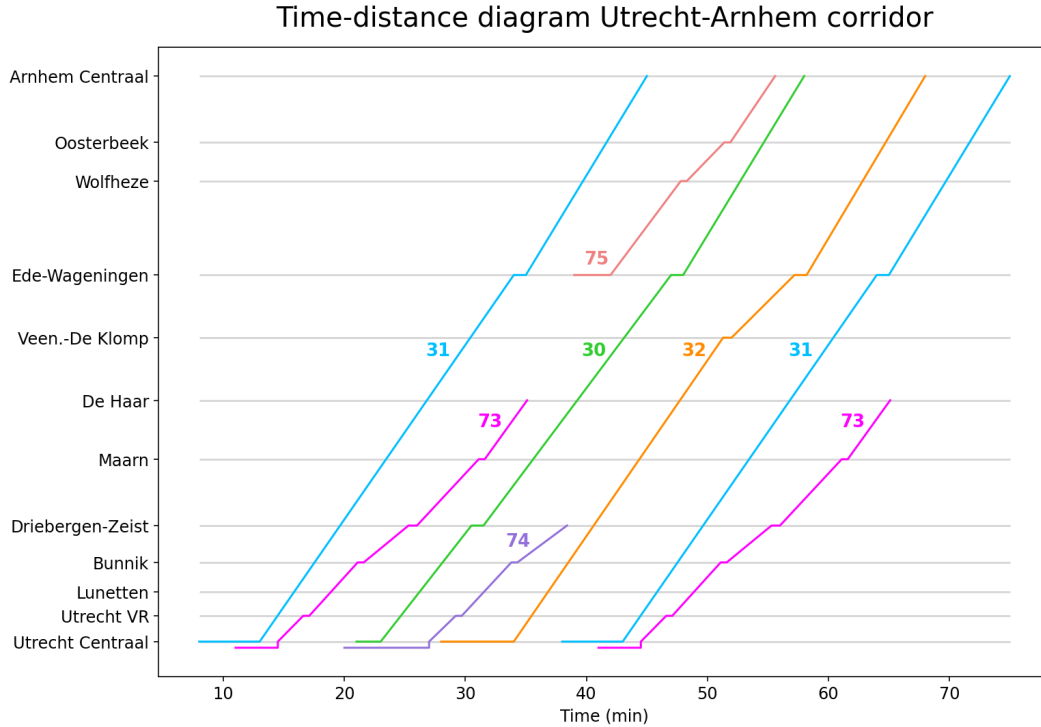


**Figure C.1:** This time-distance diagram of the 2025 timetable was earlier shown in Figure 6.3. The scale of the vertical axis is proportional to the travel times for the international trains, which travel from Utrecht to Arnhem without an intermediate stop. Only the Utrecht-Arnhem corridor is shown, so for train line 73, the journey from De Haar junction on to Rhenen is missing. For simplicity, the departure and arrival times at stations have been connected with straight lines. To show the dwell times at Utrecht Centraal, sprinter trains are depicted slightly lower on the vertical axis at this station.

The mistake also had a negligible impact on the computation speed of Algorithm 6. Calculating line 15 could easily be combined with lines 16-17 if  $R + 1$ -dimensional vector is used. Also line 19 can easily be combined with lines 20-21. Using a  $R + 1$ -dimensional vector instead of a  $R$ -dimensional vector has a small impact, especially because of parallelisation. Calculating lines 11, 22 and 23 also had a negligible impact on computation times.

## C.4 Used code

The implementation is available in the following GitHub repository:

<https://github.com/JesseJaspers/Optimising-railway-timetable-robustness-using-random-search-metaheuristics>.

## Appendix D

# Particle Swarm Optimisation

In this appendix, the swarm intelligence algorithm Particle Swarm Optimisation (PSO) is discussed. Since the algorithm is population-based, there is a pool of solutions which are modified. Each solution is called a particle, while the solution pool is called the swarm. The variables of a particle are updated with a velocity vector. Let  $x_l$  be a particle, then we denote its  $d$ -th variable as  $(x_l)_d$ . In step  $m$ , all  $(x_l)_d$  are updated with a velocity vector  $v_{l,d}(m)$ , which we will soon introduce (Wu et al., 2016):

$$(x_l)_d = (x_l)_d + v_{l,d}(m).$$

The velocity vector  $v_{l,d}(m)$  uses the best solution that has been seen in the swarm, which is denoted as  $\hat{x}$ . Also the best version seen for particle  $x_l$  is used, which is denoted as  $\hat{x}_l$ . Also variables  $Z_{1,d}, Z_{2,d}$  are randomly generated in  $(0, 1)$  and weights  $c_1, c_2, c_3$  are set. The velocity vector is updated in step  $m$  as follows (Wu et al., 2016),

$$v_{l,d}(m) = c_1 \cdot v_{l,d}(m-1) + c_2 \cdot Z_{1,d} [(\hat{x})_d - (x_l)_d] + c_3 \cdot Z_{2,d} [(\hat{x}_l)_d - (x_l)_d].$$

An overview of particle swarm optimisation being applied for research on railways can be found in Wu et al. (2016). For train order scheduling, Jamili et al. (2012) used PSO combined with SA to update priority keys. Trains with a higher key value, get priority on track over trains with a lower key value. Wang et al. (2019) combined PSO with a genetic algorithm to update the departure and arrival times of trains for a Chinese high-speed rail corridor.

Contrary to Wang et al. (2019), we do not update departure and arrival times directly. Instead, we adapt  $\varepsilon_{i,j}$ -variables, which determine the allocation of time supplements. Adding the velocity vector can lead to solutions with negative supplement or which use too much supplement. We can use the repair procedure from Algorithm 5 to fix this. But then a solution can still be infeasible. For the previously discussed metaheuristics, we could discard infeasible solutions. For PSO, not accepting feasible solutions would mean a particle is stuck in its place until the velocity vector has changed enough such that it leads to a feasible solution. Alternatively, infeasible solutions can be accepted. But then it is possible that a particle spends a lot of time outside of the feasible region.

Research has been done on applying PSO to constrained optimisation problems. This can be done by applying a penalty term to the objective function for solutions which are not feasible. Then, infeasible particles are encouraged to return back towards the best solution obtained by the particle (which was feasible). Alternatively, a more extensive repair procedure can be designed that recovers the feasibility of solutions. This can become computationally expensive. There also exist approaches, which alternate satisfying constraints and optimising the objective function (Bonyadi & Michalewicz, 2017). So applying PSO to constrained optimisation problems is possible, but maintaining feasible solutions is challenging. Therefore, we see PSO as a less promising method than the other metaheuristics covered in Chapter 5.

# Appendix E

## Grid search tables

This appendix contains results for some grid searches performed in Chapter 7. Section [E.1](#) contains the results for LS, Section [E.2](#) is about SA, [E.3](#) is about GA, Section [E.4](#) about DE and Section [E.5](#) about CC.

## E.1 Local Search

**Table E.1:** Results for the second grid search of the local search algorithm.

| $\alpha_{\text{step}}$ | $c_0^{\text{mult}}$ | $c_0^{\text{add}}$ | Mean $\mathcal{L}$ | Best $\mathcal{L}$ |
|------------------------|---------------------|--------------------|--------------------|--------------------|
| 0.99985                | 0.20                | 1                  | 37181              | 37111              |
| 0.99985                | 0.20                | 2                  | 37119              | 37103              |
| 0.99985                | 0.20                | 3                  | 37107              | 37090              |
| 0.99985                | 0.20                | 4                  | 37119              | 37087              |
| 0.99985                | 0.20                | 5                  | 37105              | 37093              |
| 0.99985                | 0.25                | 1                  | 37296              | 37140              |
| 0.99985                | 0.25                | 2                  | 37088              | 37087              |
| 0.99985                | 0.25                | 3                  | 37116              | 37094              |
| 0.99985                | 0.25                | 4                  | 37123              | 37089              |
| 0.99985                | 0.25                | 5                  | 37109              | <b>37084</b>       |
| 0.99985                | 0.30                | 1                  | 37145              | 37138              |
| 0.99985                | 0.30                | 2                  | 37098              | <b>37084</b>       |
| 0.99985                | 0.30                | 3                  | 37117              | 37111              |
| 0.99985                | 0.30                | 4                  | 37093              | 37086              |
| 0.99985                | 0.30                | 5                  | 37113              | 37093              |
| 0.99990                | 0.20                | 1                  | 37098              | 37089              |
| 0.99990                | 0.20                | 2                  | 37100              | 37092              |
| 0.99990                | 0.20                | 3                  | 37112              | 37091              |
| 0.99990                | 0.20                | 4                  | 37106              | 37094              |
| 0.99990                | 0.20                | 5                  | 37103              | 37090              |
| 0.99990                | 0.25                | 1                  | 37140              | 37110              |
| 0.99990                | 0.25                | 2                  | 37113              | 37110              |
| 0.99990                | 0.25                | 3                  | <b>37087</b>       | <b>37084</b>       |
| 0.99990                | 0.25                | 4                  | 37102              | 37087              |
| 0.99990                | 0.25                | 5                  | 37096              | 37089              |
| 0.99990                | 0.30                | 1                  | 37107              | 37094              |
| 0.99990                | 0.30                | 2                  | 37118              | 37098              |
| 0.99990                | 0.30                | 3                  | 37107              | <b>37084</b>       |
| 0.99990                | 0.30                | 4                  | 37104              | 37095              |
| 0.99990                | 0.30                | 5                  | 37100              | 37097              |
| 0.99995                | 0.20                | 1                  | 37124              | 37106              |
| 0.99995                | 0.20                | 2                  | 37107              | 37092              |
| 0.99995                | 0.20                | 3                  | 37112              | 37108              |
| 0.99995                | 0.20                | 4                  | 37128              | 37099              |
| 0.99995                | 0.20                | 5                  | 37137              | 37119              |
| 0.99995                | 0.25                | 1                  | 37107              | 37102              |
| 0.99995                | 0.25                | 2                  | 37111              | 37101              |
| 0.99995                | 0.25                | 3                  | 37140              | 37133              |
| 0.99995                | 0.25                | 4                  | 37130              | 37128              |
| 0.99995                | 0.25                | 5                  | 37145              | 37133              |
| 0.99995                | 0.30                | 1                  | 37108              | 37106              |
| 0.99995                | 0.30                | 2                  | 37131              | 37125              |
| 0.99995                | 0.30                | 3                  | 37120              | 37107              |
| 0.99995                | 0.30                | 4                  | 37180              | 37129              |
| 0.99995                | 0.30                | 5                  | 37126              | 37116              |

## E.2 Simulated Annealing

**Table E.2:** Results obtained during the first grid search for simulated annealing.

| $T_0$ | $\alpha_T$ | $c_0^{\text{mult}}$ | $c_0^{\text{add}}$ | mean $\mathcal{L}$ | best $\mathcal{L}$ |
|-------|------------|---------------------|--------------------|--------------------|--------------------|
| 25    | 0.9995     | 0.00                | 10                 | 37165              | 37116              |
| 25    | 0.9995     | 0.00                | 20                 | 37359              | 37206              |
| 25    | 0.9995     | 0.25                | 3                  | 37089              | 37084              |
| 25    | 0.9998     | 0.00                | 10                 | 37142              | 37117              |
| 25    | 0.9998     | 0.00                | 20                 | 37236              | 37200              |
| 25    | 0.9998     | 0.25                | 3                  | 37094              | 37087              |
| 25    | 0.9999     | 0.00                | 10                 | 37125              | 37103              |
| 25    | 0.9999     | 0.00                | 20                 | 37243              | 37141              |
| 25    | 0.9999     | 0.25                | 3                  | 37094              | 37089              |
| 50    | 0.9995     | 0.00                | 10                 | 37142              | 37094              |
| 50    | 0.9995     | 0.00                | 20                 | 37316              | 37205              |
| 50    | 0.9995     | 0.25                | 3                  | 37096              | 37091              |
| 50    | 0.9998     | 0.00                | 10                 | 37154              | 37124              |
| 50    | 0.9998     | 0.00                | 20                 | 37203              | 37169              |
| 50    | 0.9998     | 0.25                | 3                  | <b>37086</b>       | 37084              |
| 50    | 0.9999     | 0.00                | 10                 | 37132              | 37107              |
| 50    | 0.9999     | 0.00                | 20                 | 37175              | 37149              |
| 50    | 0.9999     | 0.25                | 3                  | 37121              | 37112              |
| 100   | 0.9995     | 0.00                | 10                 | 37243              | 37182              |
| 100   | 0.9995     | 0.00                | 20                 | 37251              | 37180              |
| 100   | 0.9995     | 0.25                | 3                  | 37119              | 37112              |
| 100   | 0.9998     | 0.00                | 10                 | 37124              | 37100              |
| 100   | 0.9998     | 0.00                | 20                 | 37222              | 37144              |
| 100   | 0.9998     | 0.25                | 3                  | 37091              | 37090              |
| 100   | 0.9999     | 0.00                | 10                 | 37149              | 37126              |
| 100   | 0.9999     | 0.00                | 20                 | 37208              | 37151              |
| 100   | 0.9999     | 0.25                | 3                  | 37149              | 37106              |
| 200   | 0.9995     | 0.00                | 10                 | 37183              | 37165              |
| 200   | 0.9995     | 0.00                | 20                 | 37239              | 37155              |
| 200   | 0.9995     | 0.25                | 3                  | 37088              | <b>37083</b>       |
| 200   | 0.9998     | 0.00                | 10                 | 37137              | 37114              |
| 200   | 0.9998     | 0.00                | 20                 | 37208              | 37116              |
| 200   | 0.9998     | 0.25                | 3                  | 37108              | 37105              |
| 200   | 0.9999     | 0.00                | 10                 | 37244              | 37172              |
| 200   | 0.9999     | 0.00                | 20                 | 37285              | 37240              |
| 200   | 0.9999     | 0.25                | 3                  | 37304              | 37221              |

**Table E.3:** Results for the second grid search of simulated annealing. The five results with a mean  $\mathcal{L}$  above 37110 have been marked in red.

| $T_0$ | $\alpha_T$ | $\eta$ | $T_{\text{bonus}}$ | $c_{\text{bonus}}^{\text{mult}}$ | Mean $\mathcal{L}$ | Best $\mathcal{L}$ |
|-------|------------|--------|--------------------|----------------------------------|--------------------|--------------------|
| 25    | 0.9995     | 100    | 25                 | 0.05                             | 37096              | 37085              |
| 25    | 0.9995     | 100    | 25                 | 0.10                             | 37094              | 37087              |
| 25    | 0.9995     | 100    | 50                 | 0.05                             | 37098              | <b>37083</b>       |
| 25    | 0.9995     | 100    | 50                 | 0.10                             | 37095              | 37086              |
| 25    | 0.9998     | 100    | 25                 | 0.05                             | 37096              | 37085              |
| 25    | 0.9998     | 100    | 25                 | 0.10                             | 37089              | 37087              |
| 25    | 0.9998     | 100    | 50                 | 0.05                             | 37151              | 37098              |
| 25    | 0.9998     | 100    | 50                 | 0.10                             | 37103              | <b>37083</b>       |
| 50    | 0.9995     | 100    | 25                 | 0.05                             | 37088              | 37084              |
| 50    | 0.9995     | 100    | 25                 | 0.10                             | 37089              | <b>37083</b>       |
| 50    | 0.9995     | 100    | 50                 | 0.05                             | 37097              | 37091              |
| 50    | 0.9995     | 100    | 50                 | 0.10                             | 37120              | 37110              |
| 50    | 0.9998     | 100    | 25                 | 0.05                             | 37090              | 37086              |
| 50    | 0.9998     | 100    | 25                 | 0.10                             | 37094              | 37085              |
| 50    | 0.9998     | 100    | 50                 | 0.05                             | 37100              | 37094              |
| 50    | 0.9998     | 100    | 50                 | 0.10                             | 37101              | <b>37083</b>       |
| 200   | 0.9995     | 100    | 25                 | 0.05                             | 37095              | 37089              |
| 200   | 0.9995     | 100    | 25                 | 0.10                             | 37096              | 37091              |
| 200   | 0.9995     | 100    | 50                 | 0.05                             | 37109              | 37088              |
| 200   | 0.9995     | 100    | 50                 | 0.10                             | 37098              | <b>37083</b>       |
| 200   | 0.9998     | 100    | 25                 | 0.05                             | 37130              | 37111              |
| 200   | 0.9998     | 100    | 25                 | 0.10                             | 37153              | 37099              |
| 200   | 0.9998     | 100    | 50                 | 0.05                             | 37101              | 37084              |
| 200   | 0.9998     | 100    | 50                 | 0.10                             | 37150              | 37102              |
| 25    | 0.9995     | 500    | 25                 | 0.05                             | 37098              | 37091              |
| 25    | 0.9995     | 500    | 25                 | 0.10                             | 37092              | 37085              |
| 25    | 0.9995     | 500    | 50                 | 0.05                             | 37093              | 37085              |
| 25    | 0.9995     | 500    | 50                 | 0.10                             | 37092              | 37085              |
| 25    | 0.9998     | 500    | 25                 | 0.05                             | 37093              | <b>37083</b>       |
| 25    | 0.9998     | 500    | 25                 | 0.10                             | <b>37087</b>       | 37085              |
| 25    | 0.9998     | 500    | 50                 | 0.05                             | 37093              | 37090              |
| 25    | 0.9998     | 500    | 50                 | 0.10                             | 37092              | 37086              |
| 50    | 0.9995     | 500    | 25                 | 0.05                             | 37109              | 37101              |
| 50    | 0.9995     | 500    | 25                 | 0.10                             | 37091              | 37084              |
| 50    | 0.9995     | 500    | 50                 | 0.05                             | 37102              | 37094              |
| 50    | 0.9995     | 500    | 50                 | 0.10                             | 37104              | 37085              |
| 50    | 0.9998     | 500    | 25                 | 0.05                             | 37093              | 37090              |
| 50    | 0.9998     | 500    | 25                 | 0.10                             | 37088              | <b>37083</b>       |
| 50    | 0.9998     | 500    | 50                 | 0.05                             | 37091              | 37084              |
| 50    | 0.9998     | 500    | 50                 | 0.10                             | 37089              | 37086              |
| 200   | 0.9995     | 500    | 25                 | 0.05                             | 37106              | 37084              |
| 200   | 0.9995     | 500    | 25                 | 0.10                             | 37096              | 37094              |
| 200   | 0.9995     | 500    | 50                 | 0.05                             | 37093              | 37090              |
| 200   | 0.9995     | 500    | 50                 | 0.10                             | 37091              | 37084              |
| 200   | 0.9998     | 500    | 25                 | 0.05                             | 37105              | 37089              |
| 200   | 0.9998     | 500    | 25                 | 0.10                             | 37098              | 37094              |
| 200   | 0.9998     | 500    | 50                 | 0.05                             | 37103              | 37084              |
| 200   | 0.9998     | 500    | 50                 | 0.10                             | 37098              | 37090              |

## E.3 Genetic Algorithm

**Table E.4:** Population size  $n = 100$  for the genetic algorithm. Results are obtained with a mutation probability of 0.1.

| $\kappa$ | Mutation     | Crossover | $c^{\text{mult}}$ & $c^{\text{add}}$ | Mean $\mathcal{L}$ | Best $\mathcal{L}$ |
|----------|--------------|-----------|--------------------------------------|--------------------|--------------------|
| 2        | Per variable | Uniform   | 0.05 & 5                             | 37698              | 37598              |
| 2        | Per variable | Uniform   | 0.10 & 10                            | 37116              | <b>37098</b>       |
| 2        | Per variable | 1-point   | 0.05 & 5                             | 37278              | 37199              |
| 2        | Per variable | 1-point   | 0.10 & 10                            | <b>37109</b>       | 37105              |
| 5        | Per variable | Uniform   | 0.05 & 5                             | 38450              | 38400              |
| 5        | Per variable | Uniform   | 0.10 & 10                            | 37328              | 37214              |
| 5        | Per variable | 1-point   | 0.05 & 5                             | 37854              | 37771              |
| 5        | Per variable | 1-point   | 0.10 & 10                            | 37212              | 37176              |

**Table E.5:** The results of the second grid search for genetic algorithms.

| $\kappa$ | Crossover | $p_{\text{mut}}$ | $c^{\text{mult}}$ | $c^{\text{add}}$ | Mean $\mathcal{L}$ | Best $\mathcal{L}$ |
|----------|-----------|------------------|-------------------|------------------|--------------------|--------------------|
| 2        | Uniform   | 0.05             | 0.10              | 3                | 37629              | 37512              |
| 2        | Uniform   | 0.05             | 0.10              | 5                | 37122              | 37089              |
| 2        | Uniform   | 0.05             | 0.10              | 10               | 37090              | 37086              |
| 2        | Uniform   | 0.05             | 0.25              | 3                | 37142              | 37103              |
| 2        | Uniform   | 0.05             | 0.25              | 5                | 37090              | 37086              |
| 2        | Uniform   | 0.05             | 0.25              | 20               | 37092              | 37090              |
| 2        | 1-point   | 0.05             | 0.10              | 3                | 37367              | 37296              |
| 2        | 1-point   | 0.05             | 0.10              | 5                | 37093              | 37086              |
| 2        | 1-point   | 0.05             | 0.10              | 10               | 37088              | 37088              |
| 2        | 1-point   | 0.05             | 0.25              | 3                | 37095              | <b>37085</b>       |
| 2        | 1-point   | 0.05             | 0.25              | 5                | <b>37087</b>       | 37086              |
| 2        | 1-point   | 0.05             | 0.25              | 20               | 37094              | 37091              |
| 2        | Uniform   | 0.10             | 0.10              | 3                | 37182              | 37093              |
| 2        | Uniform   | 0.10             | 0.10              | 5                | <b>37087</b>       | 37086              |
| 2        | Uniform   | 0.10             | 0.10              | 10               | 37089              | 37088              |
| 2        | Uniform   | 0.10             | 0.25              | 3                | 37109              | <b>37085</b>       |
| 2        | Uniform   | 0.10             | 0.25              | 5                | 37090              | <b>37085</b>       |
| 2        | Uniform   | 0.10             | 0.25              | 20               | 37101              | 37097              |
| 2        | 1-point   | 0.10             | 0.10              | 3                | 37160              | 37101              |
| 2        | 1-point   | 0.10             | 0.10              | 5                | <b>37087</b>       | 37087              |
| 2        | 1-point   | 0.10             | 0.10              | 10               | 37090              | 37089              |
| 2        | 1-point   | 0.10             | 0.25              | 3                | 37090              | 37090              |
| 2        | 1-point   | 0.10             | 0.25              | 5                | 37089              | 37086              |
| 2        | 1-point   | 0.10             | 0.25              | 20               | 37091              | 37087              |
| 2        | Uniform   | 0.25             | 0.10              | 3                | 37100              | 37087              |
| 2        | Uniform   | 0.25             | 0.10              | 5                | <b>37087</b>       | 37086              |
| 2        | Uniform   | 0.25             | 0.10              | 10               | 37098              | 37096              |
| 2        | Uniform   | 0.25             | 0.25              | 3                | 37088              | 37087              |
| 2        | Uniform   | 0.25             | 0.25              | 5                | 37089              | 37086              |
| 2        | Uniform   | 0.25             | 0.25              | 20               | 37104              | 37093              |
| 2        | 1-point   | 0.25             | 0.10              | 3                | <b>37087</b>       | 37087              |
| 2        | 1-point   | 0.25             | 0.10              | 5                | 37089              | 37086              |
| 2        | 1-point   | 0.25             | 0.10              | 10               | 37092              | 37088              |
| 2        | 1-point   | 0.25             | 0.25              | 3                | 37088              | 37087              |
| 2        | 1-point   | 0.25             | 0.25              | 5                | 37089              | 37086              |
| 2        | 1-point   | 0.25             | 0.25              | 20               | 37096              | 37089              |

## E.4 Differential Evolution

**Table E.6:** Second grid search for differential evolution, with exponential crossover. The last column describes the average number of seconds required to reach an objective value below 37100. If not all experiments reached an objective value below 37100, this column contains an -. The convergence speed can vary largely between experiments for the same parameter combination. So comparing two parameter combinations directly is not always justifiable, but the data can still be used to spot general trends.

| $n$ | $w$  | Mutation          | CR   | Mean $\mathcal{L}$ | Best $\mathcal{L}$ | < 37100 (s) |
|-----|------|-------------------|------|--------------------|--------------------|-------------|
| 50  | 0.85 | Best/1            | 0.95 | <b>37083</b>       | <b>37083</b>       | 164         |
| 50  | 0.90 | Best/1            | 0.95 | <b>37083</b>       | <b>37083</b>       | 173         |
| 50  | 0.95 | Best/1            | 0.95 | <b>37083</b>       | <b>37083</b>       | 194         |
| 50  | 0.85 | Best/1            | 0.98 | 37084              | <b>37083</b>       | 135         |
| 50  | 0.90 | Best/1            | 0.98 | <b>37083</b>       | <b>37083</b>       | 207         |
| 50  | 0.95 | Best/1            | 0.98 | 37084              | <b>37083</b>       | 172         |
| 50  | 0.85 | Best/1            | 0.99 | 37278              | 37091              | -           |
| 50  | 0.90 | Best/1            | 0.99 | 37591              | 37493              | -           |
| 50  | 0.95 | Best/1            | 0.99 | 37166              | 37092              | -           |
| 50  | 0.85 | Current-to-best/1 | 0.95 | <b>37083</b>       | <b>37083</b>       | 181         |
| 50  | 0.90 | Current-to-best/1 | 0.95 | <b>37083</b>       | <b>37083</b>       | 178         |
| 50  | 0.95 | Current-to-best/1 | 0.95 | <b>37083</b>       | <b>37083</b>       | 201         |
| 50  | 0.85 | Current-to-best/1 | 0.98 | <b>37083</b>       | <b>37083</b>       | 166         |
| 50  | 0.90 | Current-to-best/1 | 0.98 | 37159              | <b>37083</b>       | -           |
| 50  | 0.95 | Current-to-best/1 | 0.98 | 37084              | <b>37083</b>       | 185         |
| 50  | 0.85 | Current-to-best/1 | 0.99 | <b>37083</b>       | <b>37083</b>       | 172         |
| 50  | 0.90 | Current-to-best/1 | 0.99 | 37394              | <b>37083</b>       | -           |
| 50  | 0.95 | Current-to-best/1 | 0.99 | 37160              | 37084              | -           |
| 100 | 0.85 | Best/1            | 0.95 | <b>37083</b>       | <b>37083</b>       | 299         |
| 100 | 0.90 | Best/1            | 0.95 | 37084              | <b>37083</b>       | 356         |
| 100 | 0.95 | Best/1            | 0.95 | 37085              | 37084              | 352         |
| 100 | 0.85 | Best/1            | 0.98 | <b>37083</b>       | <b>37083</b>       | 310         |
| 100 | 0.90 | Best/1            | 0.98 | 37085              | 37084              | 326         |
| 100 | 0.95 | Best/1            | 0.98 | 37085              | <b>37083</b>       | 394         |
| 100 | 0.85 | Best/1            | 0.99 | 37084              | <b>37083</b>       | 274         |
| 100 | 0.90 | Best/1            | 0.99 | 37084              | <b>37083</b>       | 322         |
| 100 | 0.95 | Best/1            | 0.99 | 37085              | <b>37083</b>       | 343         |
| 100 | 0.85 | Current-to-best/1 | 0.95 | 37084              | <b>37083</b>       | 332         |
| 100 | 0.90 | Current-to-best/1 | 0.95 | 37084              | 37084              | 343         |
| 100 | 0.95 | Current-to-best/1 | 0.95 | 37084              | 37084              | 408         |
| 100 | 0.85 | Current-to-best/1 | 0.98 | 37086              | <b>37083</b>       | 295         |
| 100 | 0.90 | Current-to-best/1 | 0.98 | 37085              | <b>37083</b>       | 386         |
| 100 | 0.95 | Current-to-best/1 | 0.98 | 37087              | 37086              | 437         |
| 100 | 0.85 | Current-to-best/1 | 0.99 | <b>37083</b>       | <b>37083</b>       | 310         |
| 100 | 0.90 | Current-to-best/1 | 0.99 | <b>37083</b>       | <b>37083</b>       | 361         |
| 100 | 0.95 | Current-to-best/1 | 0.99 | 37084              | <b>37083</b>       | 432         |

## E.5 Convex Combinations

**Table E.7:** Testing if CC can improve upon GA for 20 instances. The column GA 8 min, gives the best objective value found by GA after 8 minutes. GA 10 min gives the best objective achieved by GA after 2 more minutes. The column CC gives the objective value after CC spends two minutes on the solutions obtained by GA after 8 minutes.

| Experiment | GA 8 min | GA 10 min | CC    |
|------------|----------|-----------|-------|
| 1          | 37089    | 37089     | 37089 |
| 2          | 37094    | 37089     | 37094 |
| 3          | 37089    | 37089     | 37089 |
| 4          | 37100    | 37091     | 37100 |
| 5          | 37091    | 37086     | 37088 |
| 6          | 37097    | 37090     | 37095 |
| 7          | 37091    | 37088     | 37091 |
| 8          | 37108    | 37095     | 37101 |
| 9          | 37088    | 37086     | 37088 |
| 10         | 37088    | 37085     | 37088 |
| 11         | 37086    | 37084     | 37085 |
| 12         | 37090    | 37090     | 37090 |
| 13         | 37093    | 37087     | 37093 |
| 14         | 37088    | 37088     | 37088 |
| 15         | 37101    | 37086     | 37098 |
| 16         | 37097    | 37092     | 37097 |
| 17         | 37093    | 37090     | 37093 |
| 18         | 37090    | 37088     | 37090 |
| 19         | 37090    | 37087     | 37090 |
| 20         | 37092    | 37088     | 37086 |

# Appendix F

## Figures

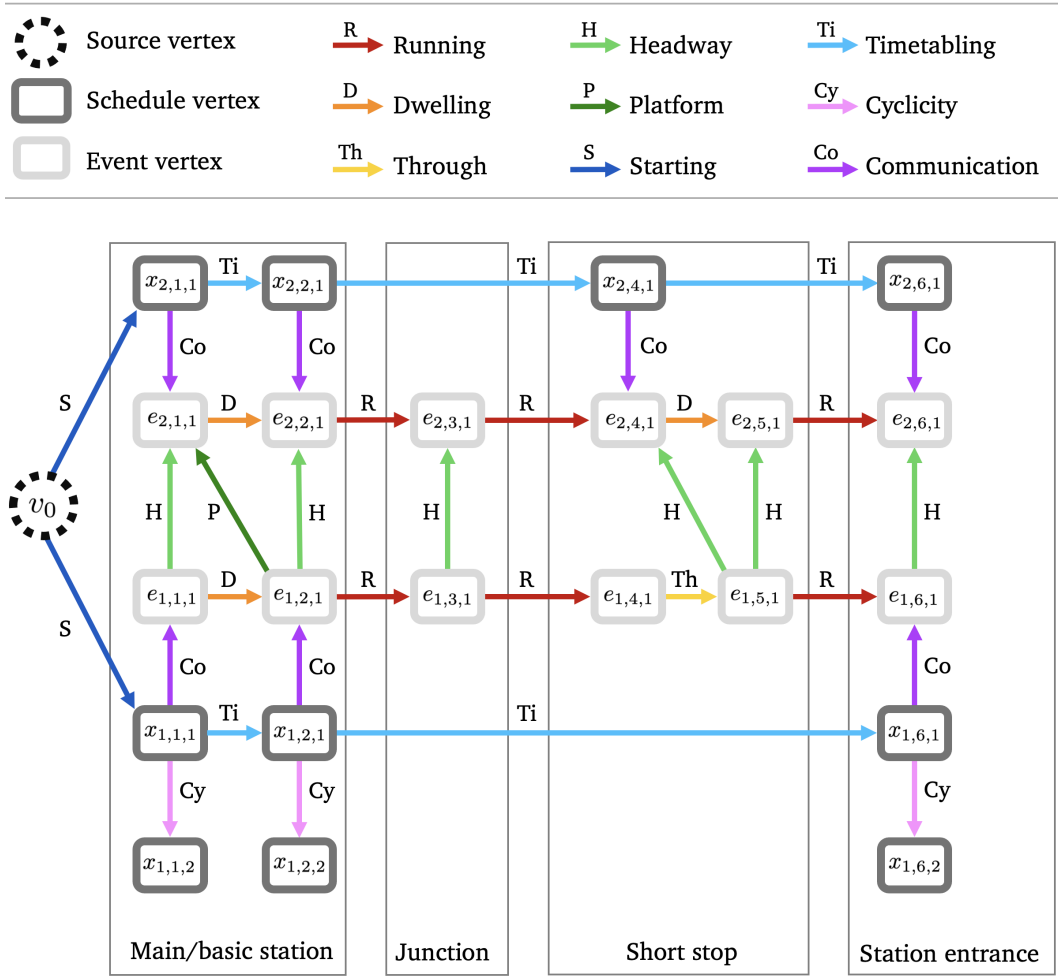


**Figure F.1:** Colour-blind-friendly version of the constructed graph from Figure 3.1. In this graph, train  $z_{2,1}$  follows behind train  $z_{1,1}$  throughout a network. There are six activities. Trains start by entering a basic or main station. The entrance of this station is activity  $q_1$ , while the exit is activity  $q_2$ . At this station, the platform arc is only present if  $z_{1,1}$  and  $z_{2,1}$  use the same platform at this station. The trains then pass a junction, which is activity  $q_3$ . Subsequently,  $q_4$  and  $q_5$  are a short stop entrance and exit, only train  $z_{2,1}$  makes a stop here. The network ends at the station entrance  $q_6$ . To show the cyclicity arcs, also the schedule vertices for train  $z_{1,2}$  are given.

# Time-distance diagram Utrecht-Arnhem corridor

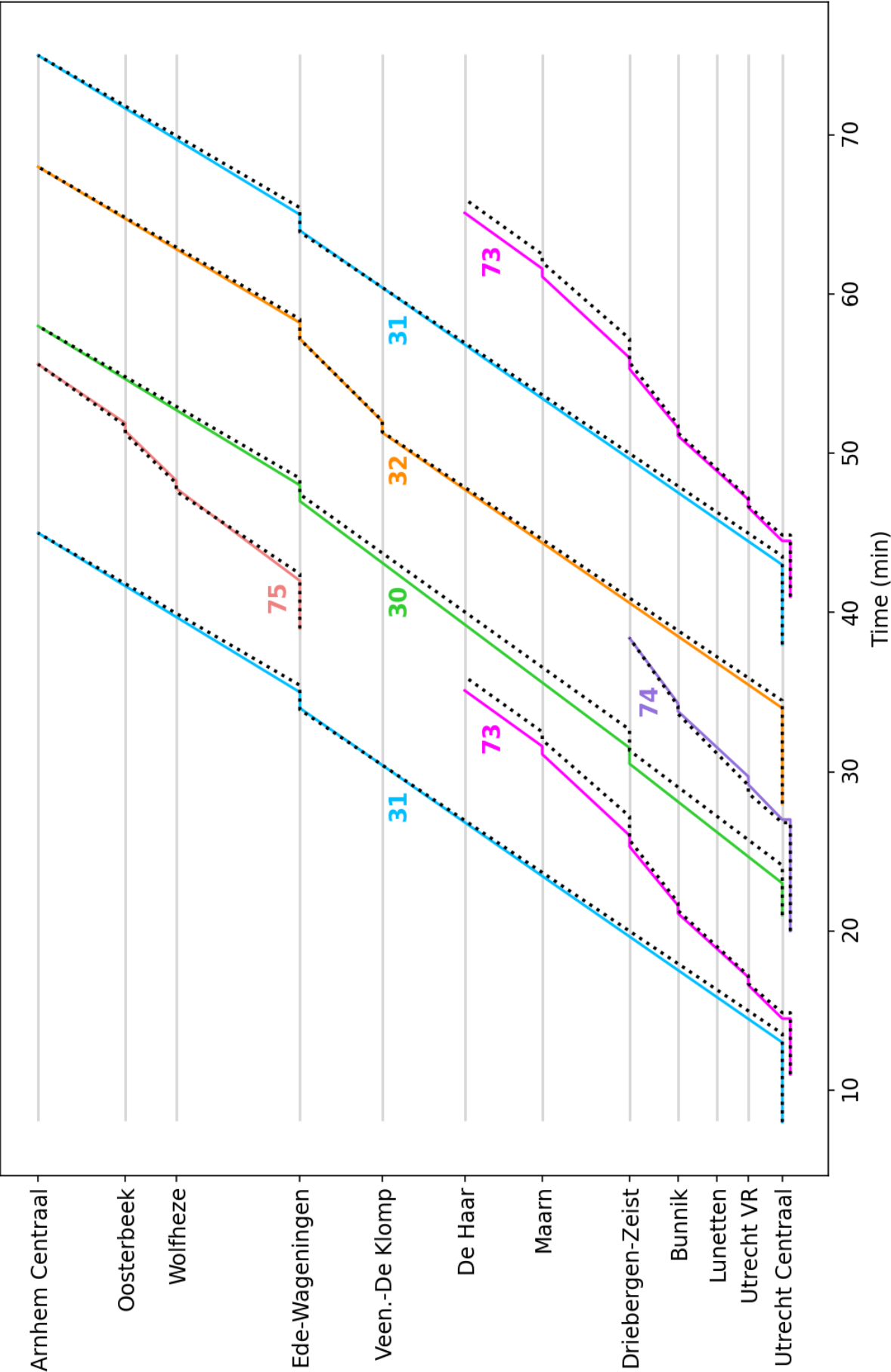


**Figure F.2:** Enlarged version of Figure 8.1. The black dotted lines represent the timetable obtained by optimising over delay scenarios without fluctuations in running time. These are overlaid on the original timetable, as depicted in Figure 6.3.

# Time-distance diagram Utrecht-Arnhem corridor

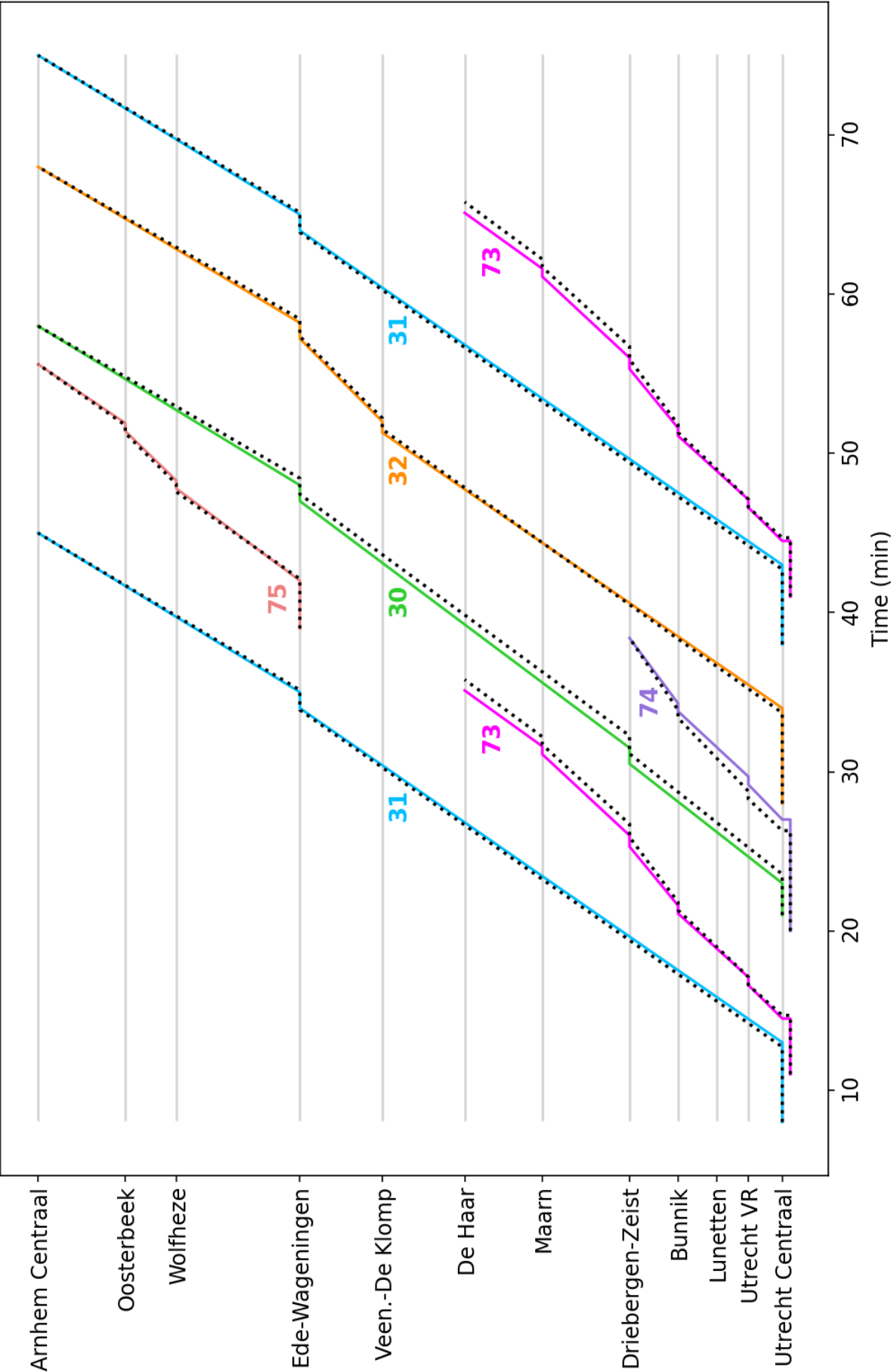


**Figure F.3:** Enlarged version of Figure 9.1. The black dotted lines represent the timetable obtained by optimising over the dependent delay factors. These are overlaid on the original timetable, as depicted in Figure 6.3.