Distributed Rail Traffic Management under Moving-Block Signalling

MSc Thesis

S.H.J. Lippes

TU Delft

Distributed Rail Traffic Management under Moving-Block Signalling

MSc Thesis

by

S.H.J. Lippes

to obtain the degree of Master of Science (Civil Engineering)
at the Delft University of Technology,
to be defended publicly on Friday May 24th 2024 at 14:00.

Student number:	4914147	
Project duration:	March 22, 2023 – TBD	
Thesis supervision:	Prof. Dr. R.M.P. Goverde,	TU Delft
	Dr. S. Sharif Azadeh,	TU Delft
	Ir. N.D. Versluis,	TU Delft

Cover image: ProRail (2021b)

TU Delft

Preface

After almost 14 months, I can finally say that my master thesis is finished. Completing it was undoubtedly the most difficult and labour intensive thing I have ever done. And although I am not fully satisfied with the result, I am still proud of all the hard work it took to get to this point. As I am now wrapping up this LaTeX file which I spend so many hours in, I realize that by doing so I am not only completing this thesis, or my master study, but my entire education of 8 years of primary school, 6 years of secondary education and almost another 6 years of university. Now that this chapter of my life now comes to a close, I wonder what lies ahead for me. I am excited to find out.

This thesis would not have been possible without the data provided by ProRail. I therefore would like to thank Dick Middelkoop for preparing and explaining this data for me. I would also like to thank my thesis supervisors for all the useful feedback and ideas they have given me. Finally, a special thank you goes to Nina for sitting down with me every two weeks to discuss my work and for keeping me optimistic.

Stijn Lippes
May 2024

Summary

With the ambition of many European policy makers to encourage a modal shift to rail, an increase in the demand for running trains can be expected. However, most of the rail infrastructure in Europe is already used at or near maximum capacity. A possible measure to increase capacity without the need to build new tracks is applying moving-block signalling. This alternative way of signalling makes it possible to run trains closer together than current signalling systems allow. If rail traffic increases however, the task of detecting and resolving potential conflicts between trains, for example if they want to pass a switch at the same time, can be expected to become more difficult. Rescheduling systems are currently being developed to support traffic managers who are burdened with this task. An increasing amount of trains to be taken into account by the conflict resolution models driving rescheduling systems does however lead to increases in the time they need to compute a solution. Since traffic management is a task performed in real-time, conflict resolution models lose their value to traffic managers if they are too slow. A way to decrease the computation time of large optimization models, which conflict resolution models often are, is to decompose them into multiple subproblems which are solved individually and coordinated in some way. Until now, no research has been performed combining these so-called non-centralized approaches for conflict resolution models with moving-block signalling. This research addresses this gap by developing a non-centralized moving-block conflict resolution model. The main research question this research focuses on is: what is the impact of a non-centralized real-time rail traffic management method on computation time and station arrival delay compared to a centralized method when applied to moving-block signalling?

To be able to develop a non-centralized moving-block conflict resolution model, a literature study of other research on conflict-resolution and non-centralized approaches is performed first. Based on this, decisions on how the model should be constructed are made. After the non-centralized moving-block model is formulated, it is verified in a small fictional case study. Finally, the model is applied to a real-world case study and its results are compared with those of a centralized model.

In the literature study it was found that the two main formulations used for conflict resolution models are the Alternative Graph (AG) and Mixed Integer Linear Programming (MILP). Moreover, non-centralized models can be divided in two approaches: distributed and decentralized. The difference between the two is that the distributed model has a specific separate sub-problem in charge of coordination between all other sub-problems, whereas in decentralized models sub-problems negotiate directly with their neighbours to coordinate their solutions. There are multiple ways to decompose the conflict resolution problem. The most common ones are geographical decompositions, train base decompositions and temporal decompositions.

For the development of the conflict resolution and non-centralized model, several decisions were made. Firstly, it was decided to formulate the conflict resolution model as a MILP since it is easier to make adjustments to and because it is best suited to be solved by a commercial solver. The challenges presented by the moving-block aspect of the model are dealt with by

partitioning the infrastructure into so called timing points and by discretizing the speed into two levels. The timing points are the switches and station platform tracks. The decision variables of the model are linked to these points. At these points, trains should maintain a minimum separation to their predecessor as determined by the blocking time. The blocking time is determined by several factors, of which a part is influenced by their speed, most notably the absolute braking distance. The speed itself is discretized in two levels as calculating it continuously would be too computationally intensive. The decision variables included in the model are the arrival time of trains at the timing points, the order in which the trains pass timing points and the speed level at which they pass the conflict points. The type of non-centralized model chosen is the distributed model. The model has a dedicated algorithm coordinating the sub-problems. The coordination is however primarily aimed at finding a feasible solution, and not so much an optimal solution. The conflict resolution model is decomposed into sub-problems for the distributed model geographically. The coordination works by applying constraints to the timing points shared by neighbouring sub-problems determined by the result of the first one of them solved. If these border constraints cause the neighbouring sub-problem to be infeasible, the constraints causing this infeasibility are identified and deleted one by one until a solution can be found. This is a process that can go back and forth between two areas until a feasible solution is found for both.

After the model was verified in a small fictional case study, it was applied to a real-world case study consisting of a large part of the rail network of the Dutch province Noord-Brabant. The distributed model found a feasible solution for 1202 out of 1208 attempts. In 98% of the cases the distributed model was solved faster than the centralized model. In 78% of cases the cumulative delay resulting from the distributed model at stations was smaller than or equal to the resulting delay for the centralized model. The difference in computation time is on average approximately 15 seconds, and the difference in cumulative station delay, if there is one, is usually smaller than 300 seconds. For all instances where the computation time of the distributed model exceeds that of the centralized model, an infeasibility was found. Finding an infeasibility does however not mean the computation time of the distributed model automatically exceeds that of the centralized model. It was found that an increase in the size of the initial delay or an increase in the number of initially delayed trains leads to a higher percentage of the runs having a larger cumulative station delay for the distributed model compared to the centralized model and a smaller percentage of runs resulting in equal cumulative station delay for both models.

It was concluded that a non-centralized real-time rail traffic management method applied to moving-block signalling can reduce computation time in almost all cases by a significant amount while for the majority of cases, the cumulative station delay decreases or remains the same compared to a centralized method. Since the model was only tested on only one case study, it can not be said with certainty that similar results would be found for a network with a different topology, size or schedule. It is therefore recommended to test the model in different cases in future research. Further recommendations include investigating why the computation time for certain runs with infeasibilities is significantly larger than other runs with infeasibilities and searching for a way to avoid the model failing in the limited amount of cases in which it can not find a feasibile solution.

Contents

1 Introduction

1.1. Background

Ever since its introduction in the 19th century, rail has played an important role in transportation. In its early days, the train revolutionized transport over land by providing unprecedented travel speed and efficiency. Transporting large quantities of goods became fast, reliable and affordable. This made rail one of the main factors paving the way for the industrial revolution, which completely changed society into how we know it today. However, after the invention of the automobile and the airplane in the early 20th century, the importance of rail steadily declined. Nevertheless, the mode kept playing an important role in many parts of the world up until the present day. For example, in 2018, the train accounted for 13% of the total distance traveled by individuals in the Netherlands, thereby being the second most used mode by distance. Moreover, about 4% of the total freight mass was transported by rail during the same period (ProRail, 2021a).

Rail has some significant benefits that still make it a preferable transport option in the present. It is foremost a highly energy-efficient mode. The specific energy consumption of rail is significantly lower than its main competitor: road. This is due to the relatively low friction that exists between steel wheels and track compared to rubber wheels and asphalt. Additionally, recent technological innovations such as regeneration of braking energy and driver advisory systems make rail even more energy efficient (Rail Freight Forward, n.d.). Moreover, through the high degree of electrification of railway infrastructure, rail is capable of utilising renewable energy sources such as wind and solar. As a result, rail is widely considered to be one of the most sustainable modes of transport. In addition to this, rail is able to transport large amounts of people and goods relatively space efficiently when compared to road based modes. A single train could for example easily equal the seating capacity of 100 cars while having a much smaller footprint. Rail is also considered a safe mode of transport. This is mainly due to various safety systems such as signalling systems, interlocking systems and automatic braking systems which aim to prevent accidents like collisions and derailments.

With the energy transition in mind, policy makers, like for example the European Commission, aim for a (partial) modal shift back to rail. By 2050, it aims for a doubling of freight rail traffic and a tripling of high speed rail traffic (Chebaro, 2020). Moreover, Europe's growing population (Eurostat, 2023) and urbanization (European Commission, 2020) will likely generate an increased demand for intercity transportation. This will inevitably create the need for more trains of all types to be operated. However, some of the rail infrastructure in Europe is already

used near or at maximum capacity, causing bottlenecks (Khadem Sameni and Landex, 2013; Rotoli et al., 2016). Measures therefore need to be taken to allow for more growth. The first possibility to create additional capacity would be to build new or expand existing infrastructure. This is however very costly and time consuming. The other possibility is to try to increase the capacity of existing infrastructure.

A measure capable of contributing to improving existing rail infrastructure capacity is changing conventional fixed-block signalling systems to moving-block signalling. Signalling exists to ensure safe operation of the railways. The way signalling systems are designed does however have a significant effect on the capacity of the infrastructure. The main goal of signalling is to maintain a safe separation between trains by providing enough space for a train to stop in case a preceding train suddenly fails. Under traditional fixed-block signalling, a stretch of track is partitioned in sections, not necessarily of the same length, where each section is known as a block. The length of a block should be at least the longest distance any train allowed to use the track requires to stop when emergency brakes are applied. Only one train at a time is allowed to occupy a block. The information telling the train driver whether he is allowed to proceed into a block or not, known as the movement authority, is communicated through a system of track side signals. Since the braking distance of most trains will be shorter than a block length, some of the potential capacity of the track remains unused. Moreover, the location of a train within a block is not taken into account. A leading train almost leaving a block will still require any following trains to stop at exactly the same position as when the train just entered the block. Moving-block signalling solves these inefficiencies. It makes it possible to run trains as close together as their individual absolute braking distances allow. This is achieved by calculating the minimum required safe separation between trains in real-time based on their current speeds. The movement authority is then communicated continuously instead of in intervals through trackside signals. This system is called moving-block signalling because the ‘blocks’ trains are occupying move along with the trains themselves.

A second measure to enable higher rail infrastructure capacity usage is the improvement of traffic management. One of the tasks of traffic management is the detection and resolution of conflicts. A conflict is for example a situation in which two trains are approaching a switch at the same time. Under normal circumstances when trains run according to their schedule conflicts should not occur. However, when a train is delayed there is a risk of it coming into conflict with another train, especially in busy areas with many switches. To resolve a conflict, measures need to be taken. It can sometimes be difficult to determine which measure or set of measures is best as their outcome can potentially lead to other conflicts and thus even more delay. These decisions are taken by traffic managers. For this task they rely primarily on their knowledge and experience. Some disturbances might occur more frequently than others. If a disturbance occurs that the traffic manager in charge has experienced before, they can reuse a solution they know worked well then. This does however not always guarantee the best possible solution. To help traffic managers make better decisions a new tool is being developed: rescheduling systems. These are software programs calculating the optimal solution strategy based on the current traffic situation. Rescheduling systems would be especially useful in complex situations in which it is difficult for a human traffic controller to oversee the impact certain potential measures have.

With the goal of encouraging a modal shift back to rail, the job of traffic managers can be expected to become more difficult. With potentially more trains in the network, possibly running closer together as a result of moving-block signalling, the risk of disturbances occurring

increases. Moreover, the probability of these disturbances leading to conflicts also increases. Furthermore, the effect of disturbances and conflicts could persist longer and affect more trains in the timetable. As a result, it becomes more important to make good conflict resolution decisions. Traffic managers have to think further ahead and take more factors into account for this. Advisory rescheduling systems could therefore be given an important role in the decision process of traffic managers. However, for the conflict resolution algorithms forming these rescheduling systems it is also true that the problem becomes more difficult with an increased amount of trains. Consequently, they will require more time to compute their optimal solution. Since traffic management is performed in real-time and potential solutions need to be implemented as soon as possible, rescheduling systems with a long computation time lose their value as an aid to traffic managers. A possible technique to reduce the computation time of large mathematical problems, which conflict and resolution algorithms often are, is to decompose them into multiple sub-problems. In this research the words decomposed and non-centralized are used synonymously for this. Conversely, solving the problem without decomposition is called centralized.

In the past, relatively much research has been done on centralized fixed-block real-time rail traffic management and corresponding models, for example by D'Ariano et al. (2007). Also non-centralized fixed-block traffic management has seen a fair share of research, like for example by Corman et al. (2010). Rarer is research on centralized moving-block traffic management. Versluis et al. (2023) is the best example of this. Despite being mentioned in the European COMBINE project as part of the architecture of future traffic management systems back in 2000 (Giuliari et al., 2000), there has, to the best of my knowledge, never been any concrete research performed combining the use of a moving-block signalling system with non-centralized rail traffic management. This thesis is intended to address this gap.

1.2. Research question

The aim of this research is to create a non-centralized real-time rail traffic management model under moving-block signalling which gives feasible results in an acceptable computation time. The main research question therefore reads:

What is the impact of a non-centralized real-time rail traffic management method on computation time and station arrival delay compared to a centralized method when applied to moving-block signalling?

To come to an answer to the main question the following sub-questions are posed:

1. What is the current state-of-the-art regarding centralized and non-centralized conflict resolution methods under moving-block and fixed-block signalling?
2. How to formulate a conflict resolution model for a non-centralized network with moving-block signalling?
3. How to coordinate the sub-problems of a decomposed conflict resolution model?
4. How does the proposed model applied to a case study compare to a centralized moving-block method regarding the computation time and delay reduction results?

The answers to these questions are found by analyzing literature on both similar centralized and non-centralized models for conflict resolution under both fixed-block and moving-block

signalling systems and by developing and testing a new non-centralized moving-block conflict resolution model.

1.3. Structure

This report is structured as follows. In chapter 2 a literature study is performed to provide theoretical background information relevant for making decisions regarding the construction of the model. These decisions are also described in this chapter. In chapter 3 the conflict resolution model is described and subsequently in chapter 4 the non-centralized model. Next, the models are tested on a small fictional network in chapter 5 to verify that they work correctly. In chapter 6, the model is applied to a case study network to validate it. Finally, in chapter 7, an answer to the research question is given, the results are discussed, and recommendations for future research are made.

2

Railway context

As stated in the introduction, this research aims to develop a non-centralized real-time rail traffic management model under moving-block signalling and to test whether this gives feasible results in an acceptable computation time. This chapter aims to give theoretical background information relevant for making decisions regarding the construction of the model. In essence, the model consists of two parts: a moving-block conflict resolution model and an algorithm to coordinate the sub-problems that result from the decomposition of the problem. Firstly, information relevant for understanding the moving-block conflict resolution model is covered. Next, the required background information regarding the coordination algorithm is given.

For the background information needed for the development of a moving-block conflict resolution model, three topics are distinguished: the mathematical formulation type, the particularities of the signalling system and the requirements in regards to traffic management. For the coordination algorithm, two topics are distinguished: decomposition forms and decomposition domains. The background information provided for these topics was gathered from literature. In section 2.1, the literature search which provided this literature is described.

2.1. Literature search

In this section the literature search is described. The focus of this search was primarily on the mathematical formulation types and the decomposition forms. The modeling decisions related to these topics namely have the greatest impact on the working of the model. The information used throughout the other topics in part also originates from the literature found in this search, but were thus not part of the focus of the search. The two types of mathematical formulation most found in literature are Alternative Graphs (AG) and Mixed Integer Linear Programming (MILP). These formulations are described in section 2.2. The two forms of decomposition found most are the distributed and decentralized approaches. They are described in section 2.5.

During the search for literature, attention was payed to achieving a fair balance between papers focusing on centralized and decomposed approaches. This was done because both the conventional centralized approach and the decomposed approach should be understood, as the former forms the baseline from which the latter was developed and to which it can be compared. Moreover, the decomposed approach is of course important to include since this research aims at developing a decomposed model. The papers on the centralized approach

were primarily selected on the number of times they got referenced in other papers and therefore can be considered as the core of the field. For the selection of papers on decomposed approaches the inclusion of concrete examples was the main selection criterion. The resulting selection of papers in Table 2.2 are thus not representative of the actual distribution of past research between centralized, distributed and decentralized approaches. The majority of past research focused on developing centralized approaches. In Figure 2.1, the number of hits in Scopus for literature on rail rescheduling in general per publication year are plotted in blue and the number of hits on decomposed rail traffic management specifically per publication year are plotted on top of this in orange. The number of hits for rescheduling in general were obtained using the following search query: '(rail* AND (rescheduling OR "conflict resolution"))'. The number of hits for the decomposed share were obtained using the same query while adding 'AND (distributed OR decentral* OR decompos*)' to it. The number of hits remaining after subtracting the number of hits for decomposed traffic management from the total number of hits can be assumed to be the number of papers on centralized traffic management. Not all publications resulting from either search are entirely relevant and precisely on the intended topic. However, the plot does give a good indication of the actual distribution of research between centralized and decomposed approaches. Additionally, this chart also shows that research on rail rescheduling has been increasing strongly over the past decade.

An overview of the literature classified by a number of relevant characteristics can be found in Table 2.2 at the end of the chapter.

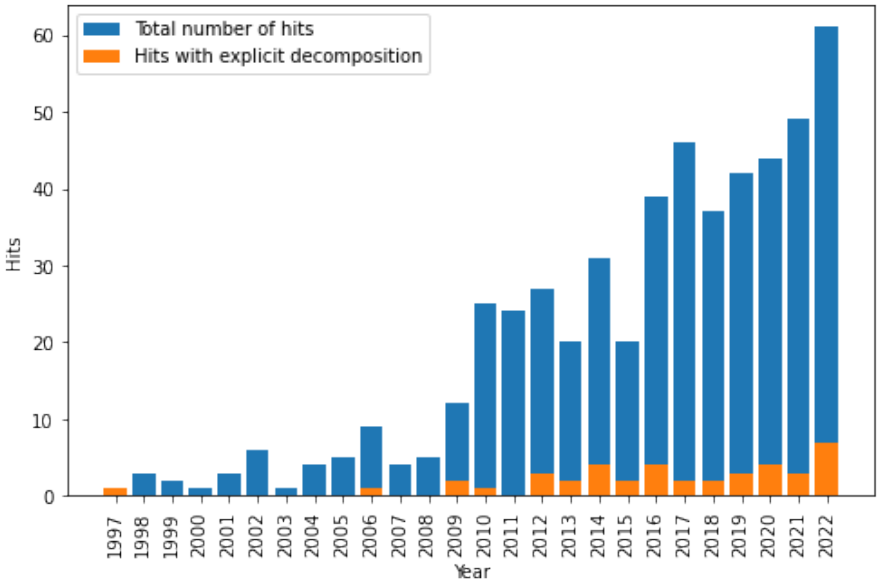


Figure 2.1: Number of Scopus hits for rail traffic management per publication year

2.2. Mathematical formulation types in literature

To convert a problem in the physical world to something computationally solvable, a conceptual model needs to be made. There are many different ways to make a model. The most suitable type depends on the relevant physical processes involved in the problem and on what the objective is. In case of the conflict resolution problem, modelling it as a mathematical optimization model is the most common option. Within mathematical optimization, multiple ways of formulating a problem exist. In the literature on rail conflict resolution, two main formulation

types are used: Alternative Graph (AG) and Mixed Integer Linear Programming (MILP). In this section these two formulations are described and compared. Based on this information it is decided what type of formulation will be used for the conflict resolution model.

2.2.1. Alternative graph

An alternative graph represents an optimization problem using a graph. This makes it easy to visualize the problem which can help to increase its interpretability. Nodes represent decision variables and edges represent constraints or relationships between variables. The alternative graph primarily uses binary and discrete variables. Complex relations that require continuous variables are thus less suitable to be modeled using AG.

The alternative graph formulation was first introduced by Mascis and Pacciarelli (2002) as a way to solve the job-shop scheduling problem. Although job-shop scheduling was developed with factory processes in mind, it can also be applied to train rescheduling (D'Ariano et al., 2007). Train services can be characterized as jobs and blocks as machines. The passing of a train through a block can therefore be seen as a machine completing an operation. An example of a centralized fixed-block CR model using an alternative graph formulation can be found in Appendix A. An advantage of the alternative graph method is that the problem can be written down relatively compactly. This helps understanding the model better and could be more efficient when being solved. Also helpful for the interpretability is that the problem can be represented visually in a graph.

The alternative graph method has been extensively applied to rail conflict resolution, all be it mostly by the same research group. The method was first proposed in a rail context in the EU project COMBINE 2 (Giannettoni and Savio, 2004). Subsequently, the idea was further researched by simultaneously D'Ariano et al. (2007) and Mazzarello and Ottaviani (2007). In D'Ariano et al. (2008), the alternative graph method developed in the previous two works was implemented in a real-time traffic management system called ROMA (Railway traffic Optimization by Means of Alternative graphs). Corman and Quaglietta (2015) tested ROMA more extensively using a realistic rail traffic simulation environment. In Corman et al. (2011) the alternative graph method was modified to take into account priority some trains might have over others.

2.2.2. Mixed integer linear program

A mixed integer linear program (MILP) is a problem with a linear objective function bounded by a set of linear constraints where the decision variables can either be continuous, integer or binary. The advantage of MILP over the alternative graph formulation is its flexibility in objective functions. Where the alternative graph formulation is primarily suitable for consecutive delay minimization, a MILP formulation also allows optimization for other objectives, given they are linear, without the necessity to modify the model. Additionally, whereas in an alternative graph formulation only limited local rerouting is possible, see for example D'Ariano et al. (2008) where the computation time of the alternative graph model significantly increases when rerouting is allowed, there is less restriction to rerouting in a MILP formulation. Although the problem instances obviously also increase in MILP when rerouting is included. As a result, solving large disruptions like track blockages also becomes possible using a MILP formulation, see for example Zhu and Goverde (2019). Problems formulated as an alternative graph can also be rewritten as a MILP. A disadvantage of MILP compared to alternative graph is the need to specify more constraints. Moreover, an alternative graph can be easily

visualised on, unsurprisingly, a graph, which is not always the case for a MILP but can help in understanding the problem. An example of a centralized fixed-block CR model using a MILP formulation developed by Törnquist and Persson (2007) can be found in Appendix B. As can be observed in Table 2.2, the papers using an alternative graph formulation almost exclusively use the branch-and-bound method to find their solution. Moreover, the papers formulating the problem using MILP never seem to use branch-and-bound. Instead, they favour commercial solvers or heuristics.

One of the first MILP models for the conflict resolution problem was the one described above by Törnquist and Persson (2007). It was developed around the same time as the first alternative graph models by D'Ariano et al. (2007) and Mazzarello and Ottaviani (2007). Several years later Caimi et al. (2012) created a similar model with only binary decision variables. Strictly this therefore is not a MILP, but a binary linear programming model. More recently Pellegrini et al. (2015) developed a model called RECIFE-MILP. The Törnquist and Persson (2007), Caimi et al. (2012) and Pellegrini et al. (2015) models were developed for a centralized system. Later, Yi et al. (2023) created a distributed version of RECIFE-MILP. Luan et al. (2020) and Cavone et al. (2022) created their own models for distributed systems. A MILP formulation can also be applied in combination with decentralization. Examples are Zhan et al. (2016) who decomposed temporally and Shang et al. (2018) with an entity-based decomposition. MILP models have also been developed for moving-block signalling. An example of this is the RECIFE-MILP adaptation of Versluis et al. (2023).

For the type of model formulation, it was decided to choose MILP. This is more easy to deal with during the construction of the model as it allows changes to the model to be implemented more easily as there is no need for predetermination of alternative arcs. Moreover, as can be concluded from the observations of Table 2.2, MILP models are better suited for the use of commercial solvers. This is an advantage as it thus not requires to create a custom branch and bound algorithm.

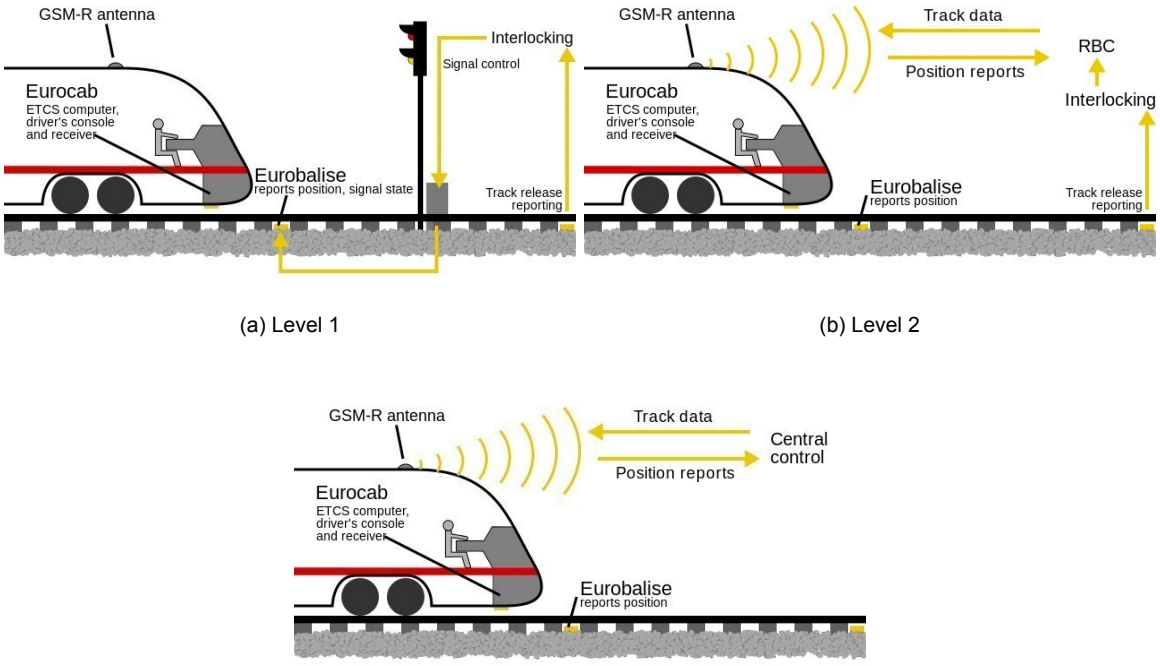
2.3. Signalling

In this section some more background information on the railway signalling concepts relevant for this research is given. First, the European Rail Traffic Management System (ERTMS), which enables moving-block signalling, is introduced. Next, blocking time is introduced and it is explained its components can be converted to moving-block signalling. The information provided in this section partially explains the headway constraints and parameters later included in the conflict resolution model.

2.3.1. The European Rail Traffic Management System (ERTMS)

One of the main measures the EU is taking to facilitate its goal of increasing rail traffic on the continent is the implementation of a single automatic train protection standard across the continent: the European Rail Traffic Management System (ERTMS). In time, ERTMS should replace each country's conventional national signalling systems. ERTMS consists of two main subsystems: the European Train Control System (ETCS) and the Global System for Mobile communications Rail (GSM-R) (European Commission, n.d.). GSM-R is a radio communication standard using exclusive radio frequency bands dedicated to rail. ETCS is a modern train control standard based on in-cab equipment. ETCS is subdivided into three levels, see Figure 2.2. The higher the level, the more reliant on onboard digital systems the train operation becomes. ETCS Level 1 is most alike conventional systems, relying on track side commu-

nication and safety equipment like train detection. Under Level 2, ETCS onboard systems and trackside equipment communicate via GSM-R. Train integrity monitoring, the checking whether trains remain intact and not lose carriages, is however still performed using track side equipment such as track circuits or axle counters. With ETCS Level 3 all information necessary for safe train operation is continuously exchanged between train and trackside systems via GSM-R, including train integrity monitoring. To enable trains to accurately determine their own position along the track, Eurobalises are introduced. These are essentially reference points for trains to calibrate their own velocity based position calculations with. Calibration is necessary as a train's own velocity measurements are not 100% accurate, which over time will cause the difference between a train's presumed position and actual location to steadily increase.



(a) Level 1

(b) Level 2

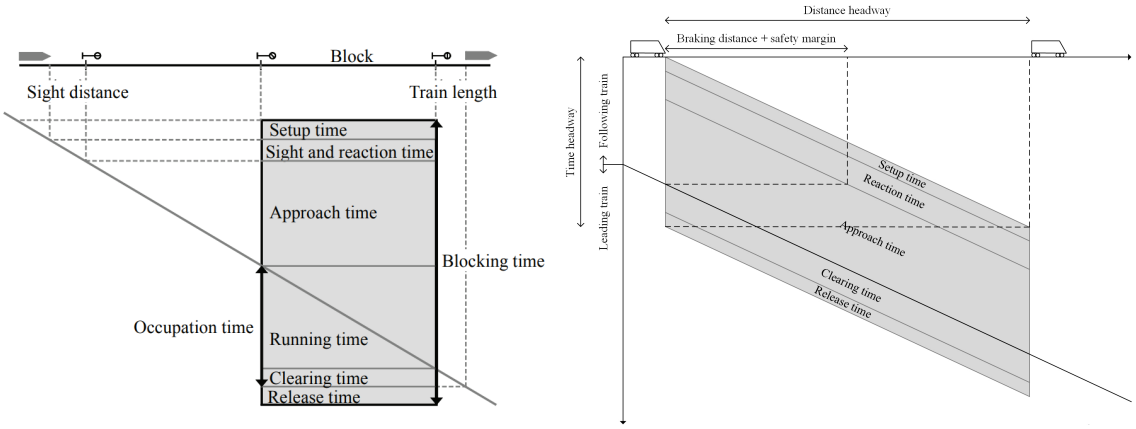
(c) Level 3

Figure 2.2: The three ETCS levels (European Commission, n.d.)

2.3.2. Blocking times

As was explained in the introduction, the goal of signalling is to ensure a safe separation between trains. In fixed-block signalling this is achieved by partitioning the infrastructure in blocks which are only allowed to be used by one train at a time. The length of a block is at least the largest distance that any train allowed to use the track requires to come to a full stop when emergency braking. The resulting time interval between two trains corresponding to their separation in distance is described by a concept called blocking time. Blocking time can be decomposed in a number of components, namely setup time, sight and reaction time, approach time, running time, clearing time and release time. Combined they can be visualized in a blocking time diagram, see for example Figure 2.3a. These times can also be converted to a distance by multiplying them with the train speed. This is visualized for fixed-block systems at the top of Figure 2.4. The setup time consists of requesting, setting and locking a route. The sight and reaction time consists of the time taken by the driver to observe and react to the indication of a signal. The approach time consists of the time needed to traverse the dis-

tance from the brake indication to the considered block. The running time consists of the time needed to traverse the considered block. The clearing time consists of the time needed for the train to traverse its own length and the release time is the time needed to release the route.



(a) Fixed-block (Goverde et al., 2013)

(b) Moving-block (sizes not to scale)

Figure 2.3: Blocking time diagrams

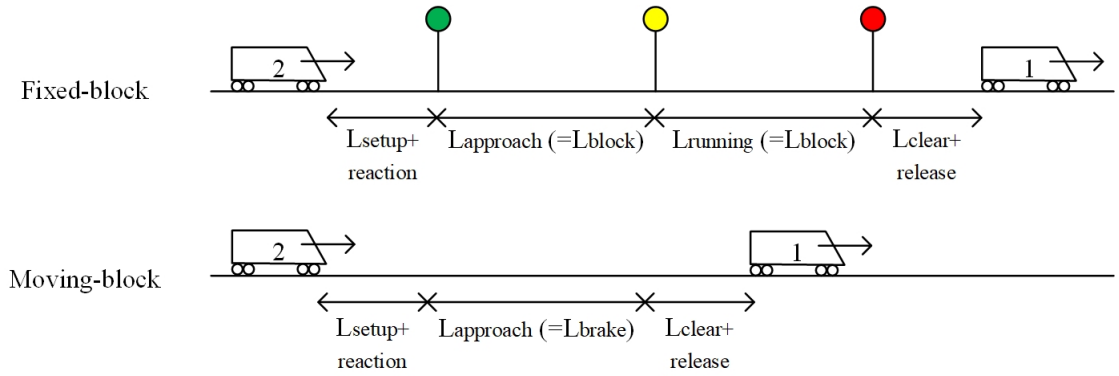


Figure 2.4: Blocking time components converted to distances Signalling (distances not to scale)

For moving-block signalling the parameters these components consist of change, as is described by Quaglietta et al. (2022) and summarized in Table 2.1. The most significant change when adapting blocking time to moving-block signalling is the absence of a running time. Trains essentially occupy blocks of infinitesimally small length, which means the running time of a train through this occupied ‘block’ is infinitesimally short as well. The approach time becomes the braking distance of the following train converted to the time it needs to traverse that distance with its current speed, increased by a safety margin. The sight and reaction time remains more or less the same but now also includes. Instead of reacting to a track-side signal, the train driver now reacts to an indication on his in-cab display. The reaction time now also includes the time it takes the train’s onboard computer to compute a dynamic speed profile from the provided movement authority The setup time and release time still include the time needed for setting and locking, or releasing, a route. Two significant components of the setup time and release time for moving-block are delays resulting from the direct communication between train and traffic control. Specifically, the delay from communicating the movement authority from traffic control to train, known as the Radio Block Center (RBC) communication delay and, in the opposite direction, the delay from communicating the Train Position Report (TPR).

These delays are the result of information being send in intervals, and thus not being fully continuous. The clearing time is the only component being exactly the same for fixed-block and moving-block signalling. Since the occupied 'blocks' are of infinitesimally small length, the blocking time diagram changes from a stairway to a slope, as can be seen in Figure 2.3b. A conversion of these times to distances can be found at the bottom of Figure 2.4.

Table 2.1: Blocking time parameters in fixed-block and moving-block (Quaglietta et al., 2022)

Component	**Fixed-block**	**Moving-block**
Setup time	Route + signals	Route + RBC communication
Sight and reaction time	Human reaction to trackside signal	Human reaction to cab display
Approach time	Block length	Braking distance + safety margin
Running time	Block length	Infinitesimal
Clearing time	Train length	Train length
Release time	Route + signals	Route + TPR delay

2.4. Traffic management

This section describes the role of traffic management. With the information provided in this section, the decision variables to be included in the conflict resolution model and the objective function are determined.

The goal of traffic management is to make sure the movement of trains through the network is as safe and efficient as possible. Since for rail, traffic is scheduled beforehand, traffic management for this mode is limited to resolving any unexpected events that occur during the operation. Events affecting the operation of rail traffic can be roughly divided into two categories: disturbances and disruptions (Cavone et al., 2022). Disturbances are small perturbations resulting in small delays. An example is a train departing a minute later due to a large amount of passengers boarding and alighting. Disruptions are larger perturbations leading to the need to reschedule all resources (routes, rolling stock and crews). An example of a disruption is the blockage of a track section due to for example malfunctioning rolling stock or trackside equipment.

Traditionally, traffic management is a task performed by humans. In the Netherlands, the whole network has one national control centre and twelve regional traffic control centres which are further subdivided in three to twelve dispatching areas (Hornung, 2023). The national control centre, aptly named Operational Control Centre Rail (OCCR), oversees and coordinates the work of the regional control centres. Moreover, it deals with national incidents, for example a central system failure, and checks whether local perturbations and their corresponding solutions influence traffic on a national scale. In charge of the regional control centres are traffic managers (verkeersleiders in Dutch). They are mostly involved with coordinating their underlying dispatching areas and managing disruptions like track blockages. In charge of each dispatching area is a dispatcher (treindienstleider in Dutch). They are responsible for detecting and resolving the smaller disturbances which require attention at a greater level of detail. Dispatchers and traffic managers manage traffic primarily based on their knowledge and experience and possibly in the future with the help of rescheduling systems (Corman et al., 2010; Hornung, 2023).

There are multiple objectives possible when solving conflicts. Some common objectives are minimization of delay, maximization of punctuality or minimization of energy consumption. One might assume minimization of delay and maximization of punctuality are the same. There is however a slight difference between them. A train is usually considered to be on time, or punctual, if it arrives at a station within a given margin of a couple of minutes, typically three or five, of its scheduled arrival time. For example, given there is a situation in which two possible solutions to a conflict exist. The first solution results in two trains both being delayed by four minutes and the second solution in having one train with no delay and one with six minutes delay. In this example trains are considered to be on time if they arrive within five minutes of their scheduled time. Optimizing for delay would then favour the last option, since the sum of delay across both trains is lowest. Meanwhile optimizing for punctuality would favour the first option since both trains in this scenario would be considered on time while in the other option the second train would be considered late.

There exist a number of measures to manage traffic in order to solve disruptions and disturbances. These are:

- **Retiming**: Delaying or speeding up a train, for example by increasing or decreasing dwell time at a station.
- **Reordering**: Changing the order at which trains pass a section of track, for example by letting a train pass a switch in a station area before a delayed train running in the opposite direction which would have otherwise gone first.
- **Rerouting**: Changing the route of a train. This could either be local, for example by changing platforms at a station, or global, for example by running a high speed train along a conventional line instead of a high speed line.
- **Short turning**: Changing the final destination of a train to one of its intermediate stops and thus canceling the remaining part of the service.
- **Stop skipping**: Deciding to skip a stop at a station.
- **Canceling**: Deciding not to run a train at all.

To solve a conflict it is possible to take one or a combination of multiple of these actions. Often they are closely linked with one another. For example, retiming and rerouting. When a train is forced to wait at a station, and is thus retimed, a subsequent train scheduled to arrive at the platform occupied by this train might need to be rerouted to a different platform. Similarly, retiming and reordering are closely linked. Allowing one train to overtake another namely often implies having to hold, and thus retime, the other train. Also reordering and rerouting can be linked as allowing a train to overtake another could require a different track than scheduled for one of the trains at the location of overtaking.

The latter three measures and global rerouting might only be applied to solve disruptions. Disturbances would usually suffice with retiming, reordering or local rerouting measures. Short turning and canceling are the most drastic measures to take. They lead to the trains themselves and the operating crew to end up at different locations than originally scheduled, which might result in parts of the remaining schedule to become practically infeasible. From a modeling perspective, taking into account rolling stock and crew rescheduling leads to an increased complexity. To keep the model relatively simple it is therefore decided not to include short turning or train cancelling. Since these measures might only be taken for larger disruptions

in which trains are significantly delayed, only disturbances will be considered for the model to resolve. Stop skipping will not be included in the model since it is also a measure usually only taken for disruptions. In a similar effort to keep the model simple, it will neither consider rerouting. This means the scheduled route of the trains through the infrastructure will be assumed fixed. This leaves the remaining two measures, retiming and reordering. They will be focused on and are included in the model.

The next decision to make is what the objective of the model should be. From the point of view of passengers, the most important task of rail is to get them from station A to station B reliably and as fast as possible. They thus care most about their trains being delayed as little as possible. It was therefore decided that the objective of the model will be to minimize the sum of arrival delays at all stations over all trains in the considered network. This objective was chosen over maximizing punctuality since the latter requires drawing a line of what is considered punctual or not, which would introduce some subjectivity, while minimizing delay does not. It was decided to not add weights to trains in the objective function if for example they are expected to carry more passengers. This because it would make it more difficult to verify if the model works correctly. For example, trying to figure out why the conflict resolution model made the decision to give priority of one train over another at a switch would require to take into account each of their weights.

2.5. Distributed and decentralized decomposition forms

In this section, the two main decomposition forms are described. Based on the information provided here, it is decided which form is chosen for the coordination algorithm used in this research.

Ideally, mathematical problems, like the conflict resolution problem, are solved as a whole. This is known as centralized solving. However, when the size of the problem grows, for example when more trains are added, the time needed to find the optimal solution generally increases exponentially (Luan et al., 2020). If the problem has a limit on the solution time allowed, like is the case for real-time rail traffic management, this increase in computation time can cause centralized solving of the mathematical model to become infeasible. When mathematical problems become too large to be solved conventionally, an effective technique to reduce complexity is to decompose the problem into smaller bits. Using this technique, the original large-scale problem is reformulated into a set of several smaller sub-problems which contain fewer variables and are thus faster to solve. The sub-problems are connected to their neighbours through a small number of variables that are shared between them. To make sure the solutions of the sub-problems are combinable and result in the shared variables having the same values, some form of communication between the sub-problems is required. There are two approaches to this: distribution and decentralization.

In the distributed approach, also known as the hierarchical approach, a separate sub-problem is created responsible for coordination. This sub-problem is known as the master problem. In this approach, communication only exists between the master problem on the one hand and the other regular sub-problems on the other (one-to-many communication). The regular sub-problems can be of different mathematical type and structure (Leutwiler and Corman, 2023). An analogy with the current state of practice can be made when looking at traffic managers and dispatchers. Dispatcher try to optimize train traffic in their own dispatching areas without paying to much attention to what is happening in neighbouring areas, these are the respon-

sibilities of other dispatchers after all. The job of the traffic managers is to make sure the measures taken by individual dispatchers do not cause issues in other areas or on a higher level. Traffic managers can therefore be said to solve a master problem, while dispatchers solve regular sub-problems.

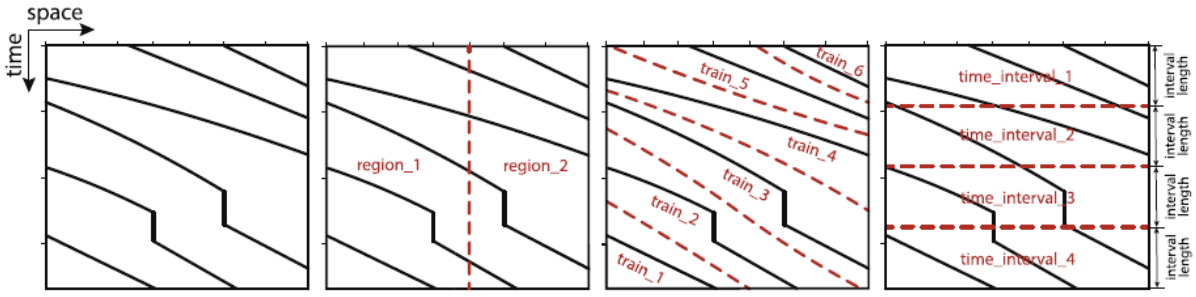
In the decentralized approach, communication links are created between sub-problems (many-to-many communication). Now the sub-problems themselves are responsible for creating a global feasible solution. Each sub-problem is trying to obtain the best solution for itself. Negotiation between sub-problems should eventually lead to a compromise which is globally feasible. In this approach the sub-problems need to be of the same mathematical type and structure (Leutwiler and Corman, 2023). When considering the analogy with traffic managers and dispatchers made for the distributed approach, translating it to a decentralized approach results in removing the traffic managers and letting the dispatchers coordinate their measures with the dispatchers of neighbouring dispatching areas themselves.

It is decided to use the distributed approach for the coordination algorithm in this research. The reasoning behind this is that it having a master problem is expected to provide a better overview of how the coordination is performed as opposed to the decentralized model where each sub-problem has their own communication links with their neighbours. Moreover, it is expected to be more easily scalable.

2.6. Decomposition domains

In this section, the domains in which the conflict resolution model can be decomposed are explained. Based on the information provided in this section it is decided which decomposition domain is used for this research.

In the case of rail networks, there are multiple domains in which the conflict resolution problem can be decomposed. Based on Leutwiler and Corman (2023) and Luan et al. (2020), a distinction is made between three types of decomposition domains: geographical, entity and temporal. In Figure 2.5 below, a visual illustration of these domains is shown. The original timetable which is being decomposed is shown in Figure 2.5a. All three types of decomposition are suitable for solving in both distributed and decentralized ways.



(a) Train timetable (b) Geographical (c) Entity (train) (d) Temporal

Figure 2.5: Illustration of decomposition domains (Luan et al., 2020)

The geographical decomposition is most easily comprehensible. Using this approach, the network is subdivided in distinct geographical areas, see for example Figure 2.5b. The total surface area or cumulative track length enclosed by each separate area does not necessar-

ily need to be the same for each sub-region. Coordination in this type of decomposition involves making sure there is consistency in trains crossing the border between two subdivisions (Leutwiler and Corman, 2023). There exists an analogy between geographical decomposition with the current state of practice. Rail networks are already subdivided in dispatching areas in which each dispatcher resolves conflicts while mostly considering their own allocated part of the network. Applying conflict resolution algorithms operating in the same geographically divided dimension as dispatchers both allows for a direct comparison between the two and makes the step of implementing the algorithms as aids to the dispatchers more straight forward. Alternatively, the network can be decomposed in multiple areas using dedicated algorithms. The advantage of this is that each area will be of a similar computational complexity and they are therefore all expected to have similar solution times. Two examples of algorithms to mathematically decompose networks into geographical areas are given in Lamorgese and Mannino (2015) and Luan et al. (2020). Papers considering the construction of algorithms to solve the conflict resolution problem for networks already decomposed geographically are Mazzarello and Ottaviani (2007), Corman et al. (2010), Corman et al. (2012) and Corman et al. (2014). All of these use an alternative graph formulation with distributed solving. An example of a geographically decomposed problem solved decentrally is Liu et al. (2018). Examples of geographically decomposed networks using MILP formulation are Lamorgese and Mannino (2015) and Yi et al. (2023). The idea to decompose networks geographically for conflict resolution algorithms was first proposed in the EU COMBINE and COMBINE 2 projects (Giuliari et al., 2000; Giannettoni and Savio, 2002; Giannettoni and Savio, 2004).

Entity-based decomposition involves subdividing the original problem in distinct entities. Usually these are separate trains, see for example Figure 2.5c. Other possibilities are groups of trains, for example categorized by priority or train type, or individual blocks (Leutwiler and Corman, 2023). Entity decomposed sub-problems lend themselves well for agent-based, so decentralized, solving. Examples of papers considering entity-based decompositions are Fay (2000), Shang et al. (2018), Perrachon et al. (2020) and Bretas et al. (2023). In the review paper by Marcelli and Pellegrini (2021) an entity-based decomposition was also suggested. Each of these papers proposes a decentralized agent-based solution approach.

In a temporal decomposition the original problem is subdivided in smaller time spans, see Figure 2.5d. The coordination between the sub-problems is similar to a geographic decomposition. However, instead of physical boundaries between areas, the boundaries between sub-problems in a temporal decomposition are time interval based. This means one sub-problem does consider the whole network, but only for a short period of time. Before coordination with other sub-problems, a single sub-problem will thus favour measures with the largest immediate delay reduction. An example of a temporal decomposition approach in literature can be found in Zhan et al. (2016).

It is decided to decompose the main problem geographically. This decision was made because it is the most easily comprehensible decomposition which can also be easily visualized, has resemblance with the current practice of dispatching areas, and mainly because it corresponds with earlier research by Hornung (2023) which focused on finding the best geographical decomposition for the case study network used in this research.

Author	Title	Centralized or decomposed	Signalling system	Level of detail	Problem formulation type	Solution method	Decomposition domain	Perturbation
Bretas et al. (2023)	A decentralised multi-agent system for rail freight traffic management	decentralized	fixed-block	line (2 branches)	-	multi-agent	entity (single train)	disturbance
Caimi et al. (2012)	A model predictive control approach for discrete-time rescheduling in complex central railway station areas	centralized	fixed-block	station	Binary linear (MPC)	CPLEX	-	disturbance
Cavone et al. (2022)	An MPC-Based Rescheduling Algorithm for Disruptions and Disturbances in Large-Scale Railway Networks	distributed	fixed-block	network	MILP (MPC)	bi-level heuristics	geographic	disturbance and disruption
Corman et al. (2010)	Centralized versus distributed systems to reschedule trains in two dispatching areas	distributed	fixed-block	area	AG	branch & bound, heuristics (FCFS, ARI)	geographic	disturbance
Corman et al. (2011)	Optimal multi-class rescheduling of railway traffic	centralized	fixed-block	station	AG	branch & bound	-	disturbance
Corman et al. (2012)	Optimal inter-area coordination of train rescheduling decisions	distributed	fixed-block	network	AG	branch& bound	geographic	disturbance
Corman et al. (2014)	Dispatching and coordination in multi-area railway traffic management	centralized and distributed	fixed-block	network	AG	branch & bound, heuristics	geographic	disturbance
Corman and Quaglietta (2015)	Closing the loop in real-time railway control: Framework design and impacts on operations	centralized	fixed-block	line	AG	branch & bound	-	disturbance
D'Ariano et al. (2007)	Conflict Resolution and Train Speed Coordination for Solving Real-Time Timetable Perturbations	centralized	fixed-block	area	AG	branch & bound, heuristics	-	disturbance
D'Ariano et al. (2008)	Reordering and Local Rerouting Strategies to Manage Train Traffic in Real Time	centralized	fixed-block	line	AG	branch & bound, local search	-	disturbance and disruption
Diaz de Rivera et al. (2020)	Potential for Moving Blocks and Train Fleets to Enable Faster Train Meets on Single-Track Rail Corridors	centralized	moving-block	line	spread sheet	spread sheet	-	disturbance
Dündar and Sahin (2013)	Train re-scheduling with genetic algorithms and artificial neural networks for single-track railways	centralized	fixed-block	line	genetic algorithm	genetic algorithm	-	disturbance
Fay (2000)	Decentralized Railway Control Based on Autonomous Agents	decentralized	fixed-block	-	-	multi-agent	entity (single train track segments)	disturbance and disruption
Giannettoni and Savio (2002)	Traffic management in moving block railway systems: the results of the EU project COMBINE	distributed	moving-block	line	-	-	geographic	disturbance
Giannettoni and Savio (2004)	The European project COMBINE 2 to improve knowledge on future rail Traffic Management Systems	distributed	mixed	area	AG	-	geographic	disturbance
Giuliani et al. (2000)	Moving block and traffic management in railway applications: the EC project COMBINE	distributed	moving-block	line	-	-	geographic	disturbance
Liu et al. (2018)	A Multi-agent Based Approach for Railway Traffic Management Problems	decentralized	fixed-block	line	-	multi-agent	geographic	disturbance
Luan et al. (2020)	Decomposition and distributed optimization of real-time traffic management for large-scale railway networks	distributed	fixed-block	network	MILP	3 algorithms	geographic, entity, temporal	disturbance
Mazzarello and Ottaviani (2007)	A traffic management system for real-time traffic optimisation in railways	distributed	mixed	areas, network	AG	local heuristics	geographic	disturbance
Perrachon et al. (2020)	Experimental study on the viability of decentralized railway traffic management	decentralized	fixed-block	line (2 branches)	-	multi-agent	entity (single train)	disturbance
Rodriguez (2007)	A constraint programming model for real-time train scheduling at junctions	centralized	fixed-block	line	constraint programming	branch & bound	-	disturbance
Shang et al. (2018)	Distributed Model Predictive Control for Train Regulation in Urban Metro Transportation	decentralized	metro	loop line	MILP (MPC)	single-shot	entity (single train)	disturbance
Törnquist and Persson (2007)	N-tracked railway traffic re-scheduling during disturbances	centralized	fixed-block	network	MILP	CPLEX	-	disturbance
Wang et al. (2016)	Rescheduling Trains Using Petri Nets and Heuristic Search	centralized	fixed-block	line	petri nets	2 algorithms	-	disturbance
Zhan et al. (2016)	A rolling horizon approach to the high speed train rescheduling problem in case of a partial segment blockage	decentralized	fixed-block	line	MILP	single-shot, CPLEX	temporal	disruption

Table 2.2: Literature

3

Conflict resolution model

The goal of this research is to develop a non-centralized moving-block conflict resolution model for rail traffic management. In this chapter the proposed conflict resolution model for moving-block signalling is described. The coordination is described in the next chapter. All general modelling decisions were described in the previous chapter. In subsection 3.1.1 and subsection 3.1.2, the challenges particular to developing a moving-block conflict resolution model in comparison to fixed-block models are explained and the chosen solutions to these challenges are introduced. In section 3.2, the resulting mathematical model is given.

3.1. Challenges of constructing a moving-block conflict resolution model

Fixed-block and moving-block differ mainly in the way the track is partitioned and the influence train speed has. These two differences and how they are dealt with for moving-block signalling will be explained in this section.

3.1.1. Track partitioning

In fixed-block signalling systems, the infrastructure is partitioned in blocks which may only be assigned to one train at a time. This allows for a conflict resolution model to assign each block to a specific train for a period of time. The decision variables of a fixed-block model thus describe the time instances each train starts and ends the occupation of a block and in which order they pass the blocks. Under moving-block signalling there is no fixed discretization of the track. Therefore, the timing and ordering of trains needs to be defined in a different way.

To be able to make timing decisions, some form of track discretization is still needed however. It would be infeasible to assign variables to each infinitesimal small point along the track after all. To determine whether a conflict occurs, a prediction needs to be made of when trains arrive at certain points in space. Under fixed-block signalling a conflict occurs when a block is requested by two trains at the same time. Under moving-block signalling a conflict occurs when a point is required by two trains at the same time. It is however not relevant to know the exact location where the conflict would occur. Take for example a train that unexpectedly slows down en route on a track between two stations with no switches due to a technical failure in the train. Another train following the same route and catching up to the slowed down train will then cause a conflict. The only possible solution to ensure the two trains do not crash into each other, assuming the malfunctioning leading train can not speed up anymore for sure, is to slow down the following train. Recall that only the arrival times at stations are relevant to

the performance of the model. Therefore it does not matter if the following train starts slowing down far before catching up to the leading train or if it slows down only just before catching up. It is only relevant that it arrives at the upcoming station as soon as possible. Or in other words, the exact points where the following train is retimed and by what amount are not relevant, as long as these retimings result in the optimal retiming of the arrival at the station. Station platform tracks, or more specifically the point where trains stop at station platforms, are therefore designated as points where retiming decisions can be made. Besides retiming the arrival time, the departure time of a train may also be retimed.

Conflicts can also occur between trains crossing paths at a switch. If there would be a switch in the track between the stations in the example above where another train is scheduled to cross the considered track, the way the following train spends its extra minute might become relevant. If the following train would change its scheduled timing before passing the switch, it could potentially lead to another conflict with the crossing train. Retiming decisions should therefore also be made at switches. A train scheduled to precede another train over a switch can for example also be forced to yield for that train and let it precede instead. Switches are therefore also the points where reordering decisions can be made. Examples of all possibilities of how a pair of trains can pass a switch and the corresponding minimum time separation between the trains can be found in Appendix C. Henceforth the places where decisions can be made, so switches and station platform tracks, will be collectively referred to as timing points.

3.1.2. The influence of train speed

Under fixed-block signalling the speed of a pair of trains has little influence on their distance separation. The separation is mainly determined by the length of the blocks. Only the distances related to the reaction, setup and release time are affected by the speed of the trains. The time separation on the other hand is influenced strongly by speed. The approach, running and clearing time are determined by it. Increasing the speed will therefore increase the distance separation by only a small amount while the blocking time reduces significantly. Under moving-block, the approach time is mainly determined by the following train's braking distance. In turn, a train's braking distance depends on its speed. Since the approach time forms the largest part of the time separation in moving-block signalling, increasing the speed thus increases the distance separation between two trains significantly. This increase in distance separation also increases the time separation. To determine the time separation in moving-block signalling, it is thus necessary to have knowledge of the speeds.

To be very accurate in determining train speed, a speed profile could be determined. This is a description of what speed a train is travelling at, at every location along its route. This would however require significantly more computations as for every change in train timings a new speed profile would need to be determined. Moreover, this changed speed profile could lead to a vicious cycle with different braking distances, thus different time separations and could then potentially require more retimings or cause more conflicts. Therefore it is decided to use discretized speed in two levels and introduce a binary decision variable for it at every timing point. The first discretized level is the scheduled speed, at which the train runs according to a speed profile resulting in running times between timing points as scheduled. The second level is the maximum speed, which when used leads to the shortest possible running times. For any train being assigned to take more, or the same amount of time, between two timing points as the scheduled speed running time, the braking distance resulting from the scheduled speed at the corresponding timing point is used. For any train being assigned to take less time than

the running time when running at scheduled speed, the braking distance resulting from the maximum speed at the corresponding timing point is used. This has the advantage of giving a finite number of braking distances at the expense of some capacity loss when the assigned time of a train between two timing points would allow for a slower speed than corresponding to the assigned speed level, thus resulting in a smaller braking distance and therefore time separation.

Since not every point in space is considered, but only the previously defined timing points, an assumption is made that if two trains pass two subsequent timing points with at least a minimum separation time between them, this will also be the case for all other not considered infinitesimal small points in the space between them.

3.2. Model formulation

The conflict resolution algorithm is based on the centralized fixed block model proposed in Törnquist and Persson (2007) and given as an example of a MILP model in Appendix B. Below, all model constraints are given.

3.2.1. Sets, sequences, parameters and variables

In the list below, the sets, parameters and variables used in the model are listed.

The main sets used in the model are the set of trains T and the set of timing points P. Both main sets have subsets providing important information. The subset $S \subset P$ contains the station platform tracks designated as timing points. Subset $K_i \subset P$ is a sequence containing the timing points along the route of train $i \in T$. This sequence is ordered chronologically according to the original timetable where the first element is the first timing point of the route of train $i \in T$. Subset $S_i \subset P$ contains the timing point at the station platform tracks where train $i \in T$ has a scheduled stop. It is thus also a subset of K_i. $B_i \subset P$ contains the first and last timing points of the route of train $i \in T$ when they are not at a station. Like S_i, it is thus also a subset of K_i. Subset $L_j \subset T$ is a sequence containing the trains passing by timing point $j \in P$ in chronological order. Finally, ID is the set of train and timing point combinations where an initial delay is applied.

The parameters are used as constant inputs to the model. The following parameters are related to the infrastructure. $n_i \in K_i$ is the last timing point along the route of train $i \in T$ in the considered area. $\pi_i^j \in K_i$ and $\sigma_i^j \in K_i$ indicate the previous and subsequent timing point along the route of train $i \in T$ at timing point $j \in K_i$ respectively. Next, the parameters related to the train motion. $rts_i^j \in \Re_+$ and $rtm_i^j \in \Re_+$ give the scheduled and minimum running time between timing point $j \in K_i$ and its consecutive timing point $\sigma_i^j \in K_i$ for train $i \in T$ respectively. $cts_i^j \in \Re_+$ and $ctm_i^j \in \Re_+$ give the scheduled and minimum clearing time at timing point $j \in K_i$ of train $i \in T$ when running at scheduled or maximum speed respectively. $ats_i^j \in \Re_+$ and $atm_i^j \in \Re_+$ provide the scheduled and minimum approach time to timing point $j \in K_i$ by train $i \in T$ when running at scheduled or maximum speed respectively, where the approach time is the time needed to cover the braking distance. Then, the parameters related to the timetable. $dw_i^j \in \Re_+$ is the minimum dwell time of train $i \in T$ at timing point station platform track $j \in S_i$. At all other timing points it has a value of 0. $a_i^j \in \Re_+$ is the scheduled arrival time of train $i \in T$ at timing point $j \in K_i$ and $d_i^j \in \Re_+$ is the scheduled departure time. Next, the parameters related to headways. $swt_{i,k}^j \in \Re_+$ indicates the time needed to change the direction of the switch at timing point $j \in P \setminus S$. This parameter has the value 0 if there is no need to change the

direction of the switch, for example if trains $i \in L_j$ and $k \in L_j \setminus \{i\}$ follow each other. $set^j \in \Re_+$ is the setup time of timing point $j \in P$. $rel^j \in \Re_+$ is the release time of timing point $j \in P$. $rct \in \Re_+$ is the reaction time which is the same across all trains and timing points. $sft \in \Re_+$ is a time safety margin used for moving-block headways and is also the same across all trains and timing points. Then penultimately, $indel_i^j \in \Re_+$ is the variable describing the initial delay applied to the combination of train and timing point $(i,j) \in ID$. Finally, the parameter M is a significantly large constant.

The model contains both continuous and binary variables. $x_i^j \in \Re_+$ holds the assigned arrival time of the head of train $i \in T$ at timing point $j \in K_i$. Directly related to the arrival time, $z_i^j \in \Re_+$ holds the delay of the head of train $i \in T$ at timing point $j \in S_i \cup B_i$: the timing points at station platform tracks and each trains first and last timing points. It is the difference between the assigned and scheduled arrival time. The two binary variables are $\lambda_{i,k}^j \in \{0,1\}$ and $v_i^j \in \{0,1\}$. $\lambda_{i,k}^j$ is related to the order of trains $i \in T$ and $k \in L_j \setminus \{i\}$ at timing point $j \in K_i$. If the order of the trains is as scheduled, it has value 1, if the order is reversed it has value 0. v_i^j indicates the speed level of train $i \in T$ between timing point $j \in K_i$ and subsequent timing point $\sigma_i^j \in K_i \setminus \{n_i\}$. It has the value 1 if the speed is according to schedule or lower and value 0 if it runs faster than scheduled.

Sets and sequences

T	Set of trains
P	Set of timing points
$S \subset P$	Set of timing points at station platform tracks
$K_i \subset P$	Sequence of timing points along the route of train $i \in T$ ordered according to the timetable
$S_i \subset S$	Set of timing points at station platform tracks where train $i \in T$ has a scheduled stop
$B_i \subset P$	Set of first and last timing points of the route of train $i \in T$ if these are not at station platform tracks
$L_j \subset T$	Sequence of trains to pass by timing point $j \in P$ ordered according to the timetable
ID	Set of combinations of timing points and trains with an initial delay

Parameters

$n_i \in K_i$	Last timing point along the route of train $i \in T$
$\pi_i^j \in K_i$	Previous timing point along the route of train $i \in T$ at timing point $j \in K_i$
$\sigma_i^j \in K_i$	Subsequent timing point along the route of train $i \in T$ at timing point $j \in K_i$
$rts_i^j \in \Re_+$	Planned running time between timing point $j \in K_i$ and its consecutive timing point $\sigma_i^j \in K_i$ for train $i \in T$
$rtm_i^j \in \Re_+$	Minimum running time between timing point $j \in K_i$ and its consecutive timing point $\sigma_i^j \in K_i$ for train $i \in T$
$cts_i^j \in \Re_+$	Clearing time at timing point $j \in K_i$ of train $i \in T$ when running at scheduled speed.
$ctm_i^j \in \Re_+$	Clearing time at timing point $j \in K_i$ of train $i \in T$ when running at maximum speed.
$ats_i^j \in \Re_+$	Time needed to cover scheduled speed braking distance to timing point $j \in K_i$ by train $i \in T$ when running at scheduled speed

$atm_i^j \in \Re_+$	Time needed to cover maximum speed braking distance to timing point $j \in K_i$ of train $i \in T$ when running at maximum speed
$dw_i^j \in \Re_+$	Minimum dwell time of train $i \in T$ at the timing point at station platform track $j \in S_i$. =0 for non station timing points.
$a_i^j \in \Re_+$	Scheduled arrival time of train $i \in T$ at timing point $j \in K_i$
$d_i^j \in \Re_+$	Scheduled departure time of train $i \in T$ at timing point $j \in K_i$
$swt_{i,k}^j \in \Re_+$	The time needed to turn the switch at timing point $j \in P \setminus S$. $\in \Re_+$, if there is a need for a change in switch direction. =0, if there is no need for a change in switch direction.
$set^j \in \Re_+$	Setup time of timing point $j \in P$
$rel^j \in \Re_+$	Release time of timing point $j \in P$
$rct \in \Re_+$	Reaction time
$sft \in \Re_+$	Time safety margin used for moving-block headways
$indel_i^j \in \Re_+$	The initial delay applied to train and timing point combination $(i, j) \in ID$
$M \in \Re_+$	A significantly large constant

Variables

$x_i^j \in \Re_+$	Arrival time of the head of train $i \in T$ at timing point $j \in K_i$
$z_i^j \in \Re_+$	Delay of the head of train $i \in T$ at timing point $j \in K_i$
$\lambda_{i,k}^j \in \{0,1\}$	Binary variable used to indicate whether train $i \in T$ or train $k \in L_j \setminus \{i\}$ passes by timing point $j \in K_i$ first. =1 if the order remains as scheduled
$v_i^j \in \{0,1\}$	Binary variable used to indicate whether train $i \in T$ runs according to scheduled or maximum speed between timing point $j \in K_i \setminus \{n_i\}$ and timing point $\sigma_i^j \in K_i$. =1 if the speed remains as scheduled

3.2.2. Objective function and constraints

Below, the model objective function and constraints are given.

$$\text{minimize} \sum_{i \in T} \sum_{j \in S_i \cup B_i} z_i^j \tag{1}$$

s.t.

$$x_i^j \le x_i^{\sigma_i^j} - rts_i^j \cdot v_i^j - dw_i^j - rtm_i^j(1 - v_i^j) \qquad \forall i \in T, \forall j \in S_i : j \ne n_i, \tag{2}$$

$$x_i^j > x_i^{\sigma_i^j} - rts_i^j(1 - v_i^j) - dw_i^j - Mv_i^j \qquad \forall i \in T, \forall j \in S_i : j \ne n_i, \tag{3}$$

$$d_i^j \le x_i^{\sigma_i^j} - rts_i^j \cdot v_i^j - rtm_i^j(1 - v_i^j) \qquad \forall i \in T, \forall j \in S_i : j \ne n_i, \tag{4}$$

$$d_i^j > x_i^{\sigma_i^j} - rts_i^j(1 - v_i^j) - Mv_i^j \qquad \forall i \in T, \forall j \in S_i : j \ne n_i, \tag{5}$$

$$x_k^j - x_i^j \ge cts_i^j v_i^j + ctm_i^j(1 - v_i^j) + ats_k^j v_k^j + atm_k^j(1 - v_k^j) + swt_{i,k}^j + set^j + rel^j + rct + sft + dw_i^j - M(1 - \lambda_{i,k}^j) \qquad \forall j \in P, \forall i,k \in L_j : k > i, \tag{6}$$

$$x_i^j - x_k^j \ge cts_k^j v_k^j + ctm_k^j(1 - v_k^j) + ats_i^j v_i^j + atm_i^j(1 - v_i^j) + swt_{i,k}^j + set^j + rel^j + rct + sft + dw_i^j - M\lambda_{i,k}^j \qquad \forall j \in P, \forall i,k \in L_j : k > i, \tag{7}$$

$$x_k^j \geq x_i^{\sigma_i^j} - M(1 - \lambda_{i,k}^j) \qquad \forall j \in P, \forall i, k \in L_j : k > i, \sigma_k^j = \pi_i^j \vee \sigma_k^j = \sigma_i^j, \qquad (8)$$

$$x_i^0 \geq a_i^0 \qquad \forall i \in T, \qquad (9)$$

$$x_i^j - a_i^j \leq z_i^j \qquad \forall i \in T, \forall j \in K_i, \qquad (10)$$

$$z_i^j = indel_i^j \qquad \forall (i,j) \in ID. \qquad (11)$$

Equation (1) is the objective function. It describes that the sum of the delay variables of all trains at timing points being part of the union of the set of timing points at station platform tracks where train i has a scheduled stop and the set of border points part of the route of train i should be minimized. The delays should be minimized at border points to prevent the distributed model picking randomly large arrival times at these points. This addition to the objective function is needed with an eye on the coordination between two adjacent areas. Otherwise, if the conflict resolution of the first area would produce random unnecessarily large arrival times at the border, the conflict resolution of the adjacent area will use them as external constraints, greatly increasing the delay of the late trains at the stations in that area.

There are two groups of constraints that are relatively similar in structure and variables involved. The first group consists of constraints (2) to (5). They ensure the time between the arrival times of a train at two consecutive timing points is within the right limits. The second group consists of constraints (6) and (7). These constraints ensure the headway between two consecutive trains at a timing point is sufficiently large.

The first group of constraints ensure that the arrival time of a train i at a timing point j is at least the running time between it (timing point j) and train i's first subsequent timing point (σ_i^j) earlier than its arrival time at that subsequent timing point, and, in case timing point j is at a station platform track, the minimum station dwell time and scheduled departure time are taken into account. If a trains runs according or slower than scheduled the minimum amount of time between its arrivals at two consecutive timing points is given by its scheduled running time rts. If a train runs faster than scheduled, the minimum amount of time between its arrivals at two consecutive timing points is given by its minimum running time rtm, which is achieved by running at maximum speed. For a train to run be considered to run faster than scheduled this also means the maximum amount of time between two consecutive timing points is given by the scheduled running time rts.

Constraint (2) ensures the running time of train i between timing points j and σ_i^j is set to the scheduled running time rts in case the train is set to run according to scheduled speed by the binary variable v_i^j being 1, or the minimum running time rtm if the train is set to run at its maximum speed by the binary variable v_i^j being 0. In case timing point j is at a station platform track, the minimum dwell time of train i at that station is added to the running time.

Constraint (3) is supplementary to (2) and makes sure the time between train i's arrivals at timing points j and σ_i^j is smaller than the scheduled running time in case it is decided for the train to run faster than scheduled.

Constraints (4) and (5) apply only to timing points j which are at station platform tracks. They are similar to the first two constraints. However, they ensure the departure of a train i from the timing point at station platform track j will not happen before its scheduled departure time d_i^j. If the constraints would be rewritten to describe the running time as the minimum difference

between two recorded time instances, the running time now describes the minimum difference between the scheduled departure at the first point and the arrival at the second point, instead of the minimum difference between the arrival at the first point and the arrival at the second point.

The second distinct group of constraints ensure a sufficiently large headway between trains i and k. The minimum headway between two trains consists of the following components: the clearing time of the leading train from timing point j, the release time of timing point j, the setup time of timing point j, the reaction time, the approach time of the following train to timing point j and the safety margin. In case the timing point is a switch of which the direction needs to be changed, the time it takes to do this is also included. The clearing time is defined as the time between the passage of the head and the tail of a train. This time depends on whether the train runs at scheduled or maximum speed. The approach time is defined as the time the head of a train needs to cover its braking distance while maintaining its current speed. Constraint (6) applies in case the two trains run in scheduled order, constraint (7) in case the order is reversed.

Four other constraints remain. Constraint (8) makes sure the order of trains remains the same at the subsequent timing point of train k for two trains following the same route in the same or exact opposite directions. Constraint (9) ensures the arrival time of a train i at the first timing point along its route is at least its scheduled arrival time a. Constraint (10) defines the delay z to be the difference between the assigned model arrival time x and the scheduled arrival time a. Finally, constraint (11) applies the initial delay to the model by setting the delay variable z equal to the initial delay parameter $indel_i^j$.

To summarize, this chapter outlines the development of the conflict resolution model for moving-block signalling used in this research. The objective of the model is to minimize arrival delays at stations. Moving-block signalling presents two particular challenges compared to fixed-block systems, namely in defining track partitioning and how train speeds are dealt with. The model addresses these challenges by partitioning the track in timing points at switches and station platform tracks where the retiming decisions are be made, and by introducing two train speed levels. The model is formulated as a Mixed Integer Linear Programming (MILP) problem. The output of the model contains a target of when each train involved in the considered network should aim to arrive at timing points along their route that when met will result in the minimum sum of station arrival delays.

4

Distributed model

Now that a conflict resolution model for moving-block signalling has been formulated, the second model that allows for non-centralized solving can be introduced. As can be deduced from the title of this chapter and as was explained in section 2.5, the decision was made to develop a distributed model. It can be characterized as such since there is a central coordination algorithm communicating to each of the sub-problems. The precise working of the coordination algorithm is explained in section 4.2. First however, the chosen geographical decomposition is explained in section 4.1.

4.1. Network decomposition

Like explained in section 2.6, it was decided to decompose the main problem geographically. In general for the geographical decomposition, each area should border at least one other area. Determining where to draw the border is outside the scope of this research. It is worth noting that the location of the border has an impact on the performance of the model. In Luan et al. (2020), an optimization model was developed to balance the size of the areas in the decomposition optimally. Hornung (2023) further enhanced this approach and applied it to the case study found in chapter 6 of this report. Besides computing where the borders should be, it is also possible to use existing boundaries, for example national or regional borders or existing dispatching areas.

The border of two areas consists of a set of timing points which are included in both areas. This set is called the collective border of the two areas. See for example Figure 4.1 and Figure 4.2 below. Here, the collective border of areas 1 and 2 consists of timing point B only. This point is present in the conflict resolution of both area 1 and area 2. If the border would be shifted to the left, timing points S1A and S1B would become the collective border. Two connected timing points can never be part of the same collective border. A border could for example never exist which simultaneously includes points D and E. Since each sub-problem in the distributed model consists of one geographic area as a result of the chosen distribution, the sub-problems will henceforth also be referred to simply as areas.

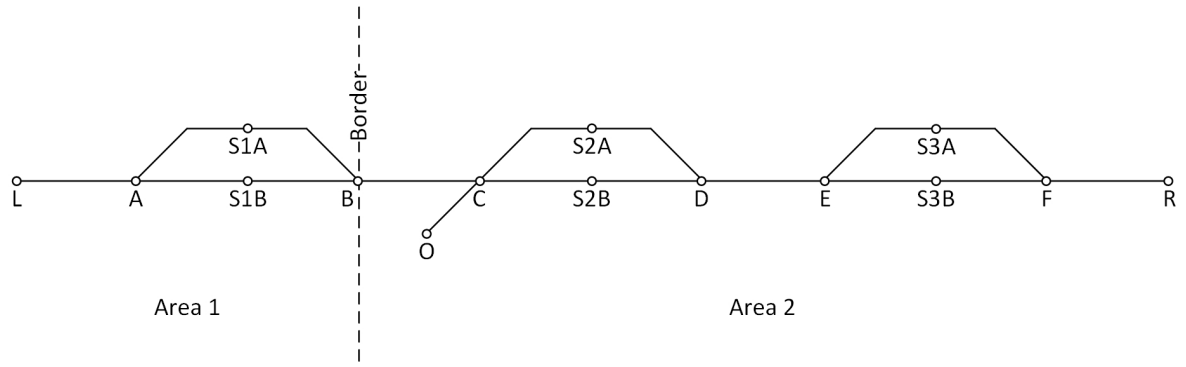
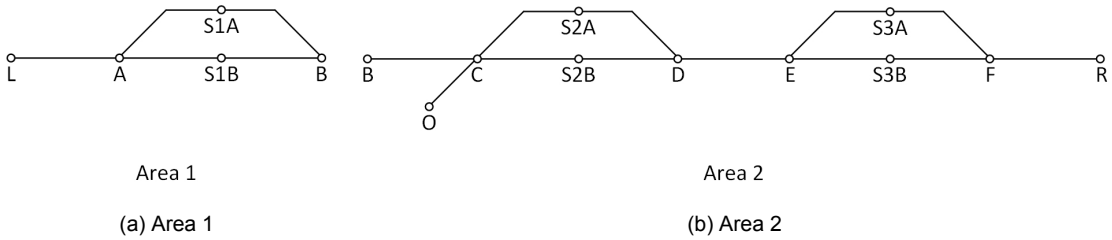


Figure 4.1: Example of a border between two areas



(a) Area 1

(b) Area 2

Figure 4.2: Areas separated

4.2. Coordination algorithm

The task of the coordination algorithm is to activate all the appropriate sub-problem conflict resolution algorithms, provide them with all the necessary data and make sure their local solutions can be combined to form a single feasible global solution. The algorithm does this by posing additional constraints on sub-problem borders. Coordination is only needed if the initial delay and any potential consecutive delays can not be resolved in the sub-problem of initial delay occurrence and there is delay (initial or consecutive) spilling towards one or multiple additional sub-problems under the supervision of the coordination algorithm.

To explain the coordination algorithm, it is divided in 4 separate parts: the initialization, the feasibility check, the feasible solution loop and the infeasible solution loop. Each part is explained individually below.

The coordination algorithm starts with the initialization when delay is detected. First, it locates in which area this delay occurs. Next, it runs the conflict resolution model for the area with the initial delay as input. If multiple initial delays were detected, the conflict resolution model of the area of the first inputted delay is run. The initialization is visualized in a flowchart in Figure 4.3.

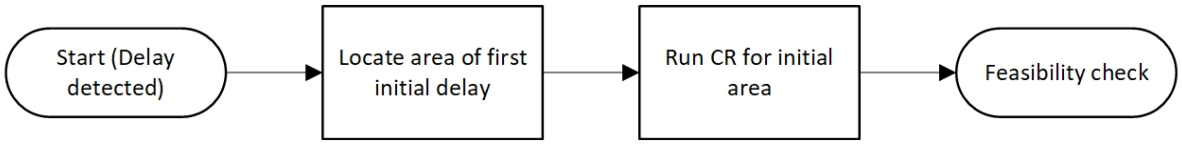


Figure 4.3: Flow chart of the initialization

The next steps of the algorithm consist of two main repeating branches. Which branch to continue with is decided in the feasibility check. One branch is followed if the conflict resolution

model was able to compute a feasible solution. The other branch is followed if no feasible solution could be found. This is the case when two constraints are contradicting, for example when two trains are constrained to use the same timing point at the same time. This would interfere with the minimal train separation constraints of the conflict resolution model, thereby making solving it impossible. A flowchart of the feasibility check can be found in Figure 4.4.

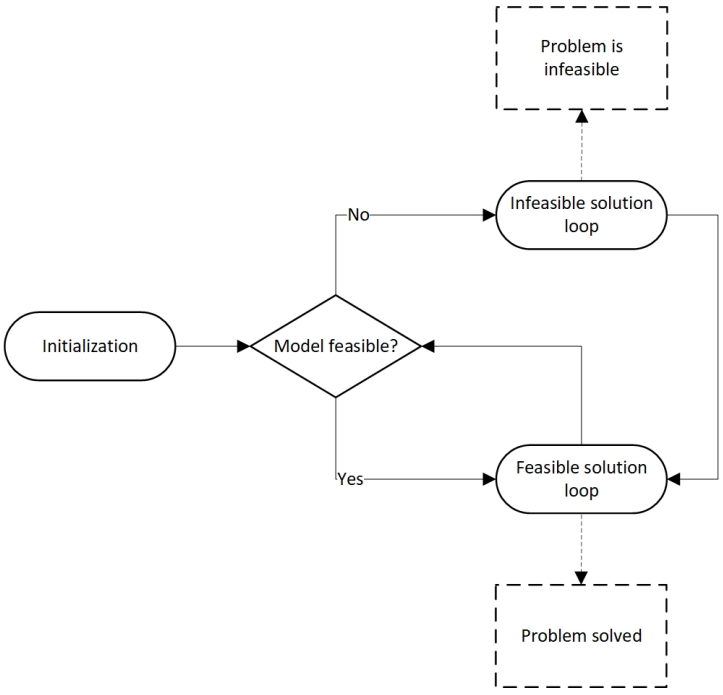


Figure 4.4: Flow chart of the feasibility check

If the model is feasible and the conflict resolution thus returns a solution, the algorithm advances to the feasible solution loop. A flow chart visualizing it can be found in Figure 4.5. First it checks whether there is any delay spilling over to another adjacent controlled area or if there are any changed train orders at the border. If this is the case the areas in question are added to the end of a queue of areas in need of solving. Next, the coordination algorithm will start the conflict resolution of the first area in the queue taking into account any already solved shared variables at the collective border as input. Then, the algorithm performs the feasibility check with the just computed conflict resolution result.

This process continues until one of the conflict resolution runs is unfeasible or until the queue is empty. If the queue is empty, a final check is performed to determine whether all areas where initial delays were inputted have been solved by a conflict resolution run at least once. If this is the case, the algorithm is done and a global solution was found. If not all areas with initial delay have been solved yet, the first of these is added to the queue and the algorithm continues.

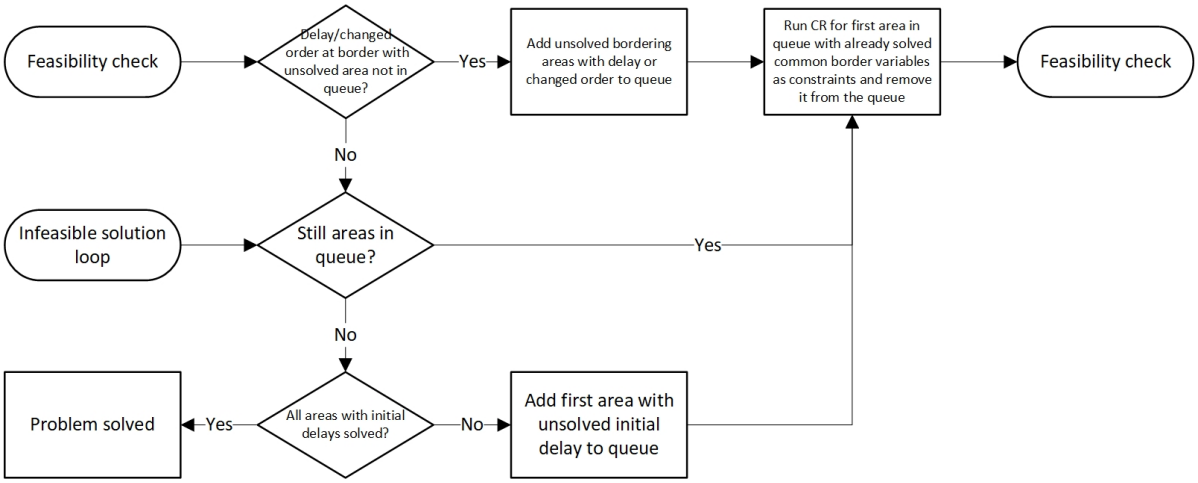


Figure 4.5: Flow chart of the feasible solution loop

When the feasibility check finds that the conflict resolution of an area is infeasible, the infeasible solution loop is started. First, the Irreducible Inconsistent Subset (IIS) is computed. This is a set of constraints that combined make the model infeasible. The IIS is computed using a method included in the conflict resolution model solver Gurobi (Gurobi Optimization, n.d.). If no constraints imposed from the border with other areas are in the IIS, the problem is infeasible as the result of an initial delay and is therefore unsolvable. If there are border constraints in the IIS, it is checked whether any of these constraints have not been removed before in any possible previous cycles of the loop. The check looks for exactly the same constraints, so the same combination of train, timing point and arrival time. The first constraint found that meets this criterion is removed from the model after which the conflict resolution model is started again. A feasibility check is then performed. If the model remains infeasible, the model returns to the first step of the infeasible solution loop. If the solution now is feasible, the resolving loop is started. If none of the border constraints in the IIS have never been removed before, it is checked whether this situation has occurred before during the current infeasible solution loop sequence. If this is the case, the last removed border constraint from the IIS is removed again and the conflict resolution model run again. If this is not the case it is established that the model has ended up in an infinite loop and therefore it is concluded that the problem is infeasible. The infeasible solution loop can be found in Figure 4.6.

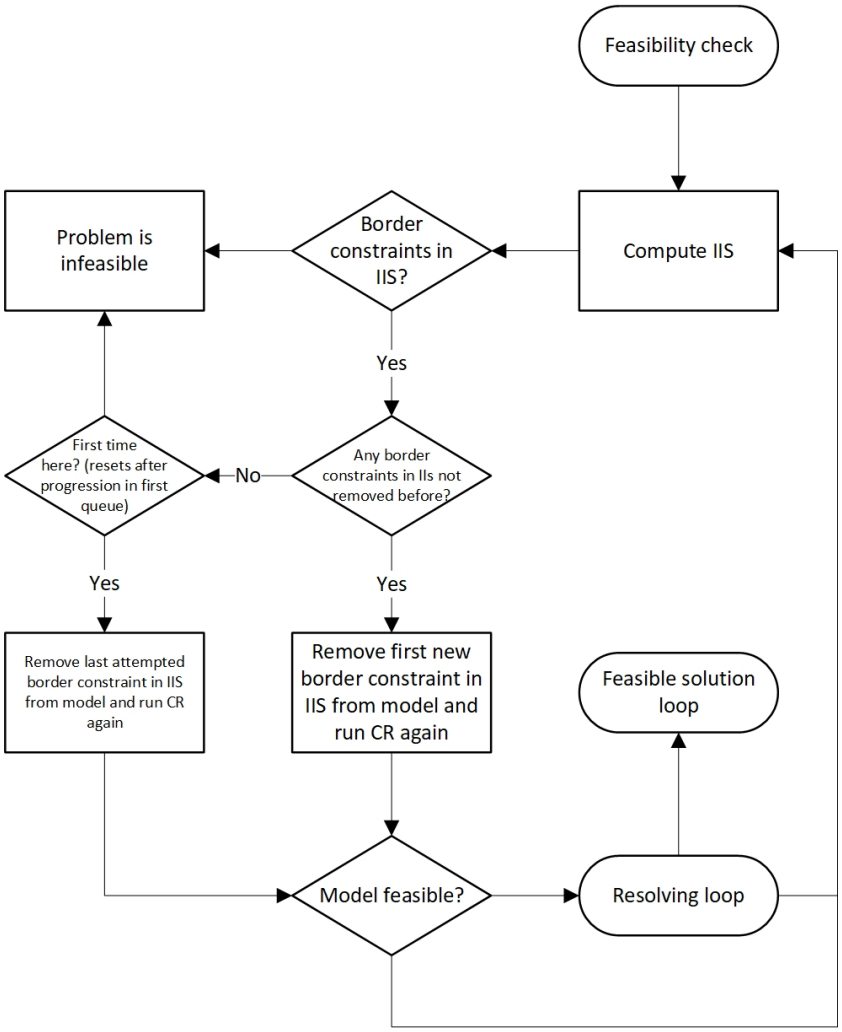


Figure 4.6: Flow chart of the infeasible solution loop

Since a feasible solution produced by the infeasible solution loop does by definition not align with at least one of its bordering areas anymore, these areas need to be resolved. This is done in the resolving loop. First any still unsolved bordering areas are added to the main queue. Second, the bordering areas which contributed at least one of the removed border constraints are added to a second queue for areas that need to be resolved. The first area from the second queue is then resolved with the new border constraints computed in the infeasible solution loop. If this area is feasible, the next area in the second queue is resolved. This continues until the second queue is empty. The coordination algorithm then proceeds with checking whether there are still areas left in the main queue and continues appropriately in the feasible solution loop. If any of the areas turn out to be infeasible after resolving, the infeasible solution loop for this area is started. If there are any other areas left in the second queue, they are removed.

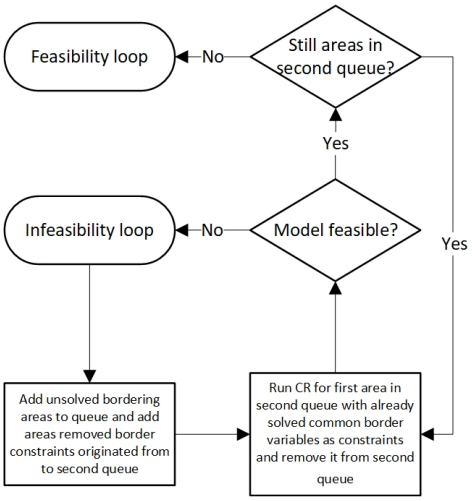


Figure 4.7: Flow chart of the resolving loop

In Figure 4.8 a flow chart of the entire model is given. The model was also written out in pseudocode. This can be found in Appendix D.

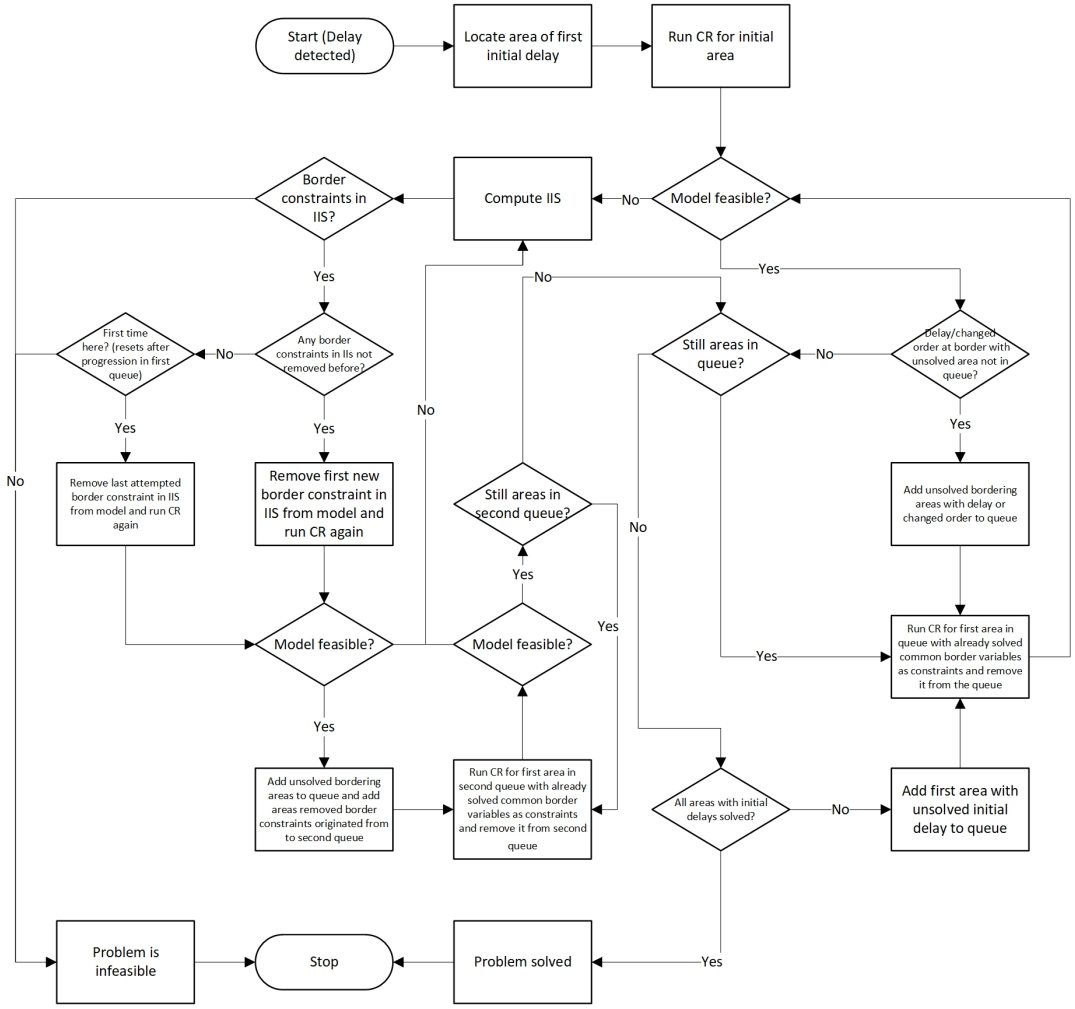


Figure 4.8: Flow chart of the coordination algorithm

4.3. Example of the working of the coordination algorithm

Below an illustrative example of the working of the coordination algorithm is given. The following scenario describes a situation in which one local dispatching problem returns an infeasibility. The steps from delay detection to the output of the final solution are described step by step.

The topology of the network considered is as follows. There are four dispatching areas ordered in a line. In other words, there are two areas on the edges both bordering one other area and two areas in the middle both bordering two on either side. From left to right the areas are numbered 1 to 4, see Figure 4.9. The delay occurs somewhere in area 2 at a station where a train is delayed by several minutes after issues with boarding passengers.

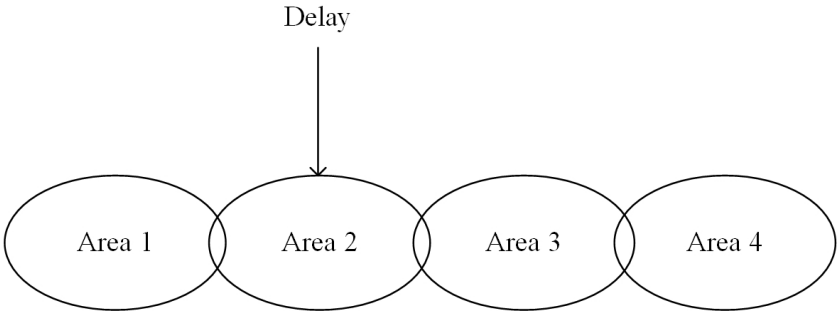


Figure 4.9: Network topology example

First, the conflict resolution algorithm of area 2 is activated. After solving the area, the solution shows some consecutive delays and changed orders at the border with both area 1 and 3. The model is constructed such that the lowest numbered adjacent area is solved first, so in this case area 1. After solving this area with the delay and order inputs from the solution of area 2, a solution for area 1 is found which is feasible. Next, area 3 is solved. Once again a feasible solution is found taking into account the delay and order constraints at the border with area 2. After assessing the output of area 3, some consecutive delays and order changes at the border with area 4 are found. Therefore, this area is solved next. However, with the delay and order constraints at the border with area 3, no feasible solution is able to be constructed. The model assesses which constraints lead to the infeasibility by computing the IIS. The first border constraint from area 3 found in the ISS is removed from the problem of area 4 which is solved again. Area 4 now has a feasible solution. Since the solution does not align with area 3 anymore, area 3 is solved again with the new border constraints from area 4 and the old constraints from area 2. No feasible solution is found which means the IIS of area 3 is constructed. A border constraint from area 2 is removed and the conflict resolution algorithm for area 3 is run again. A solution was found which means now area 2 will need to be recomputed using the new border constraints from area 3, the constraints from the solution of area 1 and the initial delay. Immediately a solution was found. This means all areas have a feasible solution which align with one another and thus the coordinated model is solved.

5

Model verification

To verify the correct working of the conflict resolution, a small fictional network with accompanying train schedule was created. To test each of the distinct constraint groups of the conflict resolution model (the running time and the headway constraints), specific scenarios were designed to trigger these constraints. Since the network is relatively simple, it is possible to manually predict the solutions the model should produce. If the solutions produced by the model after running it correspond with these expectations, it can be concluded that the model works as intended. On the same network a similar test is performed to check the working of the distributed model. The first section describes the test network and schedule. The second section describes the conflicts tested for and includes the results of the tests.

5.1. Network, schedule and parameter values

To test the ability of the conflict resolution model to perform its primary function, resolving conflicts, the network was created in a way such that any deviation from the schedule will likely result in conflicts. The network therefore consists of a single track used by trains in both directions with three stations providing room for trains in opposing directions to pass each other. A visualisation of the network can be seen in Figure 5.1. The stations are named *S1*, *S2* and *S3*, all with two platforms each named A and B, where the A platforms are used by trains running from left to right, and the B platforms by trains running from right to left. The network has three external points of entry and exit, namely points *L*, *R* and *O*. Trains are also allowed to start at any station along the line. A total of six trains are included in the schedule, named T1 to T6. Trains T1 to T3 run from left to right, T4 to T6 from right to left. Trains T1 and T2 start from the border point named *L*, T3 starts at station platform track *S2A* and T4, T5 and T6 start from point *R*. Trains T1, T2 and T3 exit the network at *R*, T4 and T5 run to *L* and train T6 exits at *O*. To be able to test the coordination algorithm the network needs to be divided in at least three areas. In Figure 5.2 the chosen border points between the three areas are indicated.

The scheduled arrival times of the trains can be found in Table 5.1 and are visualized in Figure 5.3. Note that the values given in this schedule might not be realistic for a real-world line with three stations. These values are only chosen to test the model's decisions and therefore have a non-specified unit of time. Trains do not stop at the stations in the test scenarios.

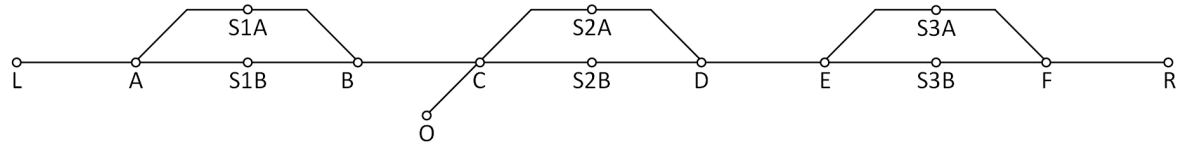


Figure 5.1: Visual representation of the test network

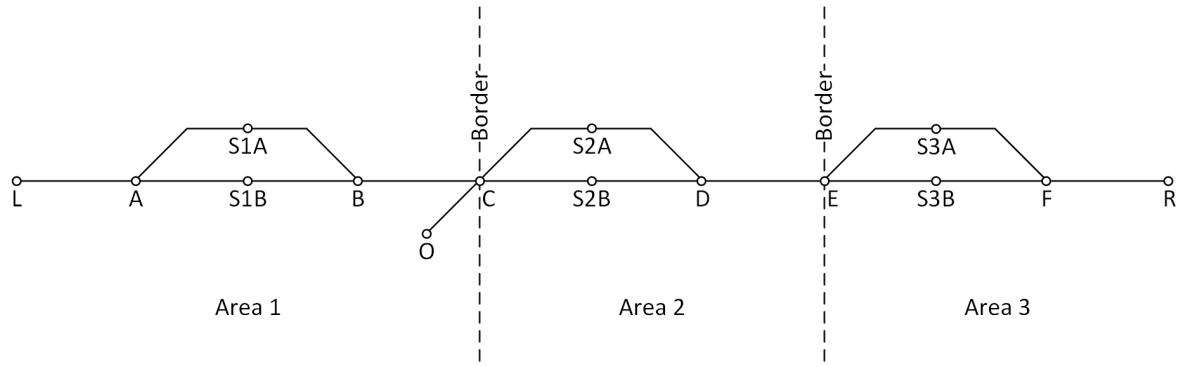


Figure 5.2: Visual representation of the test network with area borders

Table 5.1: Test network schedule: scheduled arrival times of trains at timing points

Timing point	Train					
	T1	**T2**	**T3**	**T4**	**T5**	**T6**
L	50	55		90	95	
A	65	70		75	80	
S1A	70	75				
S1B				70	75	
B	75	80		65	70	
O						75
C (border)	90	95		50	55	60
S2A	95	100	0			
S2B				45	50	55
D	100	105	5	40	45	50
E (border)	115	120	20	25	30	35
S3A	120	125	25			
S3B				20	25	30
F	125	130	30	15	20	25
R	140	145	45	0	5	10

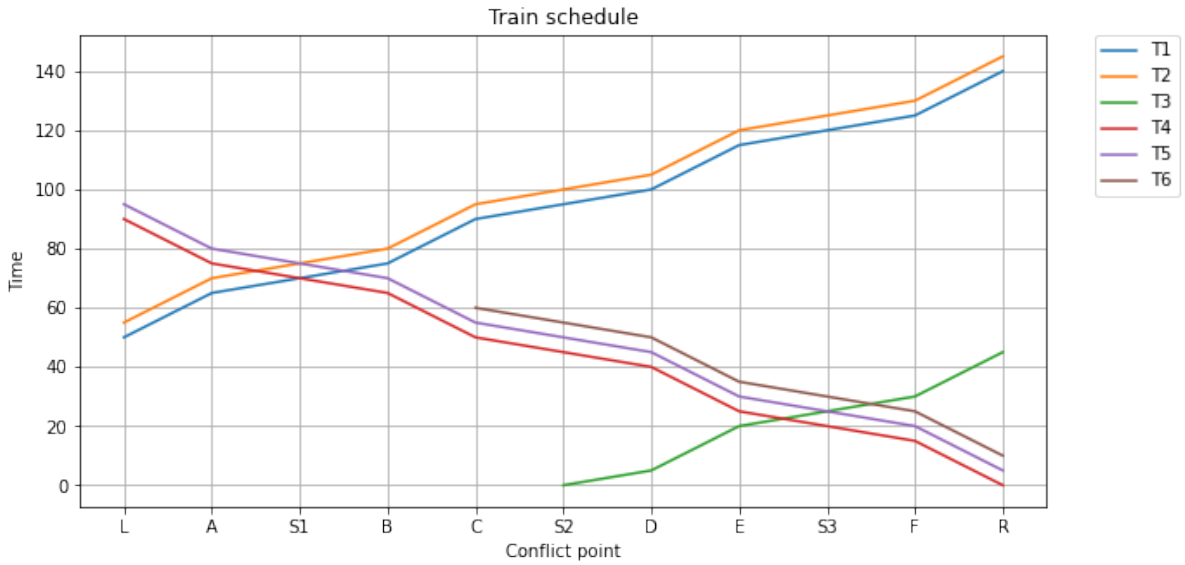


Figure 5.3: Test network schedule time-distance plot (Note, timing point O has been left out by mistake)

All trains in the test case have exactly the same characteristics. They have a length of 80 meters, at scheduled speed a clearing time of 2 time units and a approach time of 1 time unit and at maximum speed a clearing time of 1 time unit and a approach time of 5 time units. This applies to all timing points in the network. As also becomes clear from the schedule, the scheduled running time between two non-station timing points is 15 time units and between a non-station and a station timing point in either direction it is 5 time units. At maximum speed the running time between two non-station timing points becomes 13 time units. The running time between a non-station and a station timing point becomes 3 time units and between a station and a non-station timing point it becomes 2 time units. The distance between all timing points is set to 200 meters. This modeling choice is once again not realistic, as the running time between station and non-station timing points is shorter than between two non-station timing points yet the distance is the same. The time needed to turn a switch is set at 2 time units. For simplicity the formation time and the headway safety margin are set at 0 time units.

5.2. Delay scenarios

Three delay scenarios were created each aimed at testing a specific part of the model. The first two scenarios test the conflict resolution model part. The first scenario specifically focuses on the first group of constraints of the conflict resolution model, as defined in subsection 3.2.2, dealing with train running times. The second scenario subsequently focuses on the second group of constraints dealing with headways. The third scenario is designed to test the coordination model part.

5.2.1. Running times

To test whether the conflict resolution model appropriately speeds up trains, a scenario is created in which the best solution to resolve the delay of a train would be to consistently select the highest speed level for it. The selected delay is 40 time units for train T6 before starting at *R* (setting its start time to t=50). This exact delay makes sure T6 will not cause any conflicts with other trains when it is run at an average speed of either maximum, normal or slower than normal speed. Contrarily, if it is run on average at any speed between maximum and normal speed conflicts will occur with T1 and T2 at timing point *C*. Since this scenario focuses on

conflict resolution, and not coordination, the model is run centralized. The resulting solution can be found in Table 5.2 and Figure 5.4.

Table 5.2: Result running time test scenario: resulting arrival times of trains at timing points

Timing point	Train					
	T1	**T2**	**T3**	**T4**	**T5**	**T6**
L	50	55		90	95	
A	65	70		75	80	
S1A	70	75				
S1B				70	75	
B	75	80		65	70	
O						99
C	90	95		47	53	86
S2A	95	100	0			
S2B				45	50	84
D	97	105	2	40	45	81
E	115	120	20	25	30	68
S3A	120	125	25			
S3B				20	25	66
F	122	128	30	15	20	63
R	140	145	45	0	5	50

Figure 5.4: Result running times test scenario time-distance plot

As expected, the model runs train T6 at its maximum speed, thereby minimizing the delay as much as possible. It is therefore concluded that the running time constraints work as intended.

Notably some of the other trains are run at their maximum speeds along some parts of their route as well, even though they are not delayed or bound to come into conflict with another train otherwise. An example is train T1 arriving at *D* 3 time units ahead of schedule. This can be explained by the fact that speeding up some trains for short sections has no influence on the

delay at the timing points of interest, namely the stations and network edge. In other words, the value of some speed variables and arrival times at certain timing points has no influence on the the objective function value, thus allowing the model to make a free choice for their value. Preferably the solution would not include trains speeding up when they don't need to. This could for example be prevented by adding a small penalty in the objective function for any trains that have been sped up. However, this would change the meaning of the objective value away from being purely the sum of delay. It is therefore decided to accept that some trains are arbitrarily sped up and slowed down between stations when they are not involved in potential conflicts.

5.2.2. Headways

To make sure the model maintains the correct headway between trains, a scenario is created in which conflicts arise leading to a change in the running of both involved trains by the model. The model should then thus take into account the appropriate headway between them. To create this scenario, train T3 is delayed by 10 time units at its starting point *S2A*. This creates conflicts between it and trains T4, T5 and T6. Finding the solution with the least delay in this scenario is quite difficult. For example, if train T3 is set to run at its maximum speed it will arrive at timing point *E* at t=25, the same time as T4 is scheduled to arrive there. Speeding up train T4 allows it to arrive at *E* at t=22 at the earliest (as it is not allowed to leave station platform track *S3B* before its scheduled departure time). However, since it is now running at maximum speed, its approach time increases to 5 time units. Combined with the clearing time of T3 (1 unit) and the time needed to turn the switch (2 units), the minimum headway between train T3 and T4 would be 8 time units, which is more than the available 3 units between them. The model should recognise this and come up with a solution in which either T3 arrives at *E* later, or in which T4 is delayed such that it passes *E* after T3. Both options lead to another conflict with T5. If T3 is delayed by 5 time unit, thus arriving at *E* at t=30, a similar conflict as just discussed will occur between trains T3 and T5. If T4 is delayed to pass *E* after T3, it will arrive there at t=29, as the minimal headway between T3 and T4 now is 4 time units (1 unit clearing T3, 1 unit braking T4 and 2 units changing switch direction). This will then lead to a conflict between T4 and T5. The possible solutions keep branching as T6 will also be involved in conflicts in a similar manner. The final solution the model calculated can be found in Table 5.3 and Figure 5.5. For the same reason as given for the running times test scenario, this scenario is run centralized as well.

Table 5.3: Result headways test scenario: resulting arrival times of trains at timing points

Timing point	Train					
	T1	**T2**	**T3**	**T4**	**T5**	**T6**
L	50	55		90	95	
A	65	70		75	80	
S1A	70	75				
S1B				70	75	
B	75	80		65	70	
O						75
C	90	95		47	53	60
S2A	95	100	10			
S2B				45	51	55
D	100	105	12	42	47	50
E	115	120	25	29	32	35
S3A	120	125	28			
S3B				24	27	30
F	122	128	33	19	22	25
R	140	145	46	0	5	10

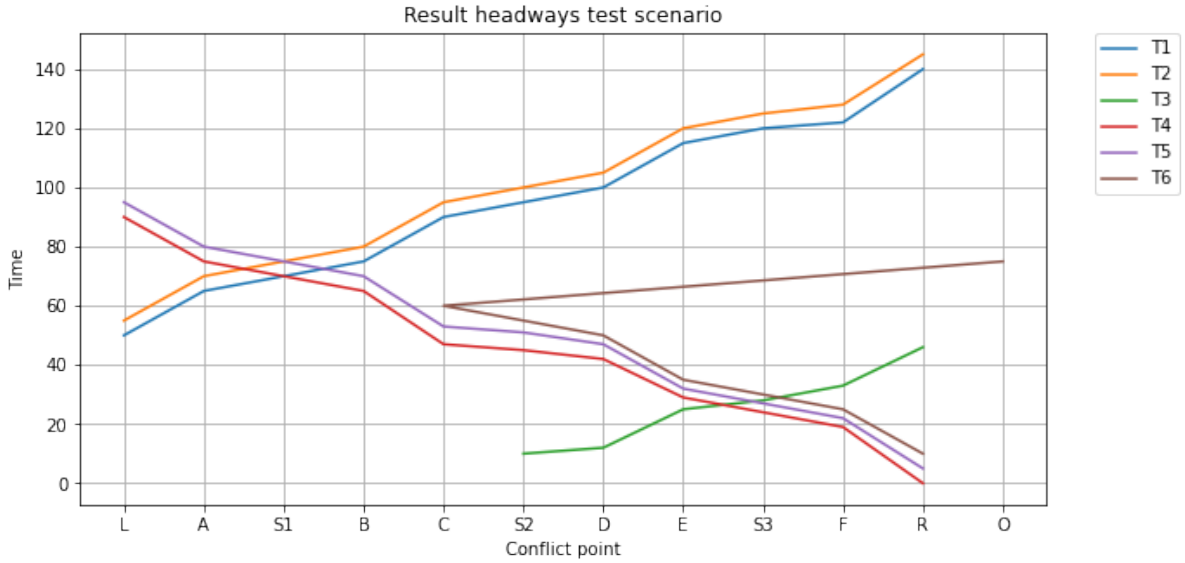


Figure 5.5: Result headways test scenario time-distance plot

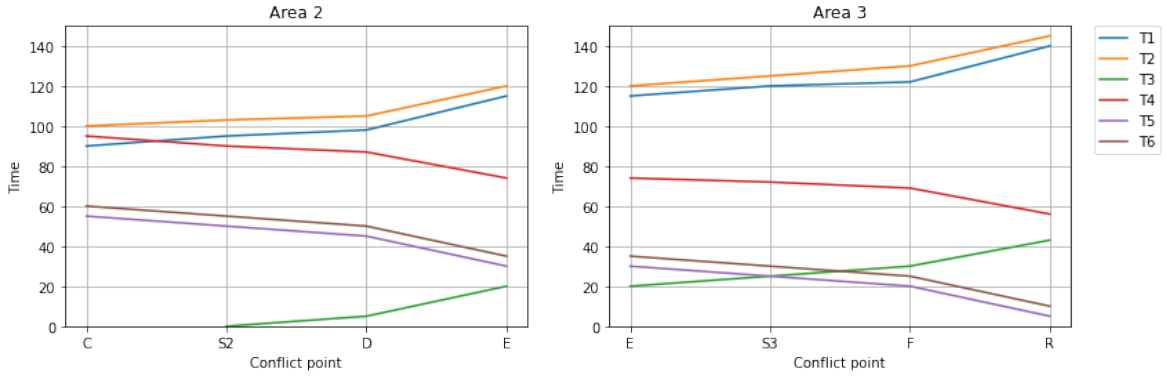
A check of the results confirms that all headways between trains are sufficient for the chosen speed levels. It can thus be concluded that the conflict resolution part of the model performs as intended.

An interesting observation of the result is that train T3 does not run at its maximum speed for the whole route. This means it is still delayed when reaching *R*. Between *S3A* and *F* it runs according to its scheduled speed. This can be explained by the fact that if T3 would run at maximum speed on this section, the minimum headway between it and T6 at *F* would be 8 time units. This is not possible as T6 can be at *F* at t=23 at the earliest. This shows that the model indeed takes into account headways and possible headways when determining a

solution.

5.2.3. Coordination

To test the working of the coordination algorithm, delays were chosen such that an infeasibility arises in one of the areas. Specifically, this was achieved by delaying a train entering from the right of the network. At timing point *R* train T4 was delayed by 56 time units. To reduce the delay of this train as much as possible, the conflict resolution model of areas 3 and 2 will try to run it as fast as possible. This will however lead to a situation in which the trains at the border between areas 1 and 2 are ordered such that solving area 1 becomes impossible. At timing point *C* train T4 will be set to arrive between trains T1 and T2 running in the opposite direction. Since there is only one track between timing points *B* an *C*, area 1 becomes infeasible and will require resolving areas 2 and 3 such that train T4 is no longer ordered between trains T1 and T2. Solving areas 3 and 2 without coordination (the border arrival times determined for area 3 were used as input to area 2) results in the following time-distance plots: (see Figure 5.6)



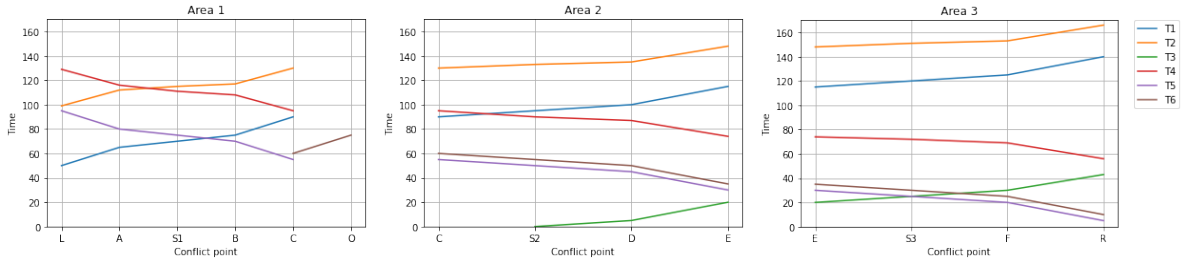
(a) Area 2 (b) Area 3

Figure 5.6: Test scenario uncoordinated resulting time-distance plots

By applying the coordination algorithm, one of the border constraints placing train T4 between trains T1 and T2 should be removed from the conflict resolution algorithm of area 1. This could either be achieved by delaying train T4 more such that it arrives at timing point *C* after train T2, by speeding up trains T1 and T2 such that they both arrive at timing point *C* before train T4 or by delaying train T2 such that it has no conflict with train T4 between timing points *B* and *C* anymore. The solution resulting from the coordination algorithm is given in Table 5.4 and is visualized in Figure 5.7. As can be seen, the option to delay train T2 is chosen by the algorithm. Since this option was one of the options expected before running the model, it is concluded that the algorithm works as intended.

Table 5.4: Coordination test scenario resulting schedule: resulting arrival times of trains at timing points

Timing point	Train					
	T1	**T2**	**T3**	**T4**	**T5**	**T6**
L	50	99		129	95	
A	65	112		116	80	
S1A	70	115				
S1B				111	75	
B	75	117		108	70	
O						75
C (border)	90	130		95	55	60
S2A	95	133	0			
S2B				90	50	55
D	100	135	5	87	45	50
E (border)	115	148	20	74	30	35
S3A	120	151	25			
S3B				72	25	30
F	125	153	30	69	20	25
R	140	166	45	56	5	10



(a) Area 1 (b) Area 2 (c) Area 3

Figure 5.7: Test scenario coordinated resulting time-distance plots

6

Model validation by means of a case study

In this chapter the case study will be described. The case study consists of testing a variety of delay scenarios on a section of the Dutch railway network and comparing the results of the centralized and distributed models.

6.1. Network, schedule and parameter values

This section describes the network considered for the case study, the corresponding timetable and the specific parameter values used.

6.1.1. Network

The selected network is a section of the Dutch rail network cut off at Roosendaal in the west, Lage Zwaluwe in the northwest, 's-Hertogenbosch in the northeast and Liempde in the southeast. The network borders the rest of the network at Roosendaal at the lines towards Bergen op Zoom and Antwerp in Belgium, at Lage Zwaluwe at the line towards Dordrecht, at Breda at both the connections to the highspeedline (HSL) to Rotterdam and Antwerp, at 's-Hertogenbosch at the lines towards Utrecht and Nijmegen and at Liempde at the line towards Eindhoven. The choice to use this (sub)network was made mainly because Hornung (2023) already performed an analysis on how to partition this specific network most optimally for distributed conflict resolution. Moreover, the network and corresponding timetable provide plenty of opportunities for conflicts to occur in case of delays as there are several locations where trains cross paths. A schematic map of the network including the borders between sub-areas proposed by Hornung (2023) can be found in Figure 6.1. The network is divided in a total of four sub-areas, each with a major intercity station of which the area bears the name: Roosendaal (Rsd), Breda (Bd), Tilburg (Tb) and 's-Hertogenbosch (Ht). The borders between the sub-areas are at the nearest timing points to where the borders in Figure 6.1 are indicated. Similarly, the borders between the considered network and the rest of the Dutch (and Belgian) network are at the nearest timing point to where the borders are indicated. At locations where the border does not intersect the tracks in the figure, namely at Lage Zwaluwe, 's-Hertogenbosch and the HSL near Breda, the border is located at the timing point closest to the end of the track in the figure. Some border points are station platform tracks. These are located at the stations of Etten-Leur and Tilburg Universiteit.

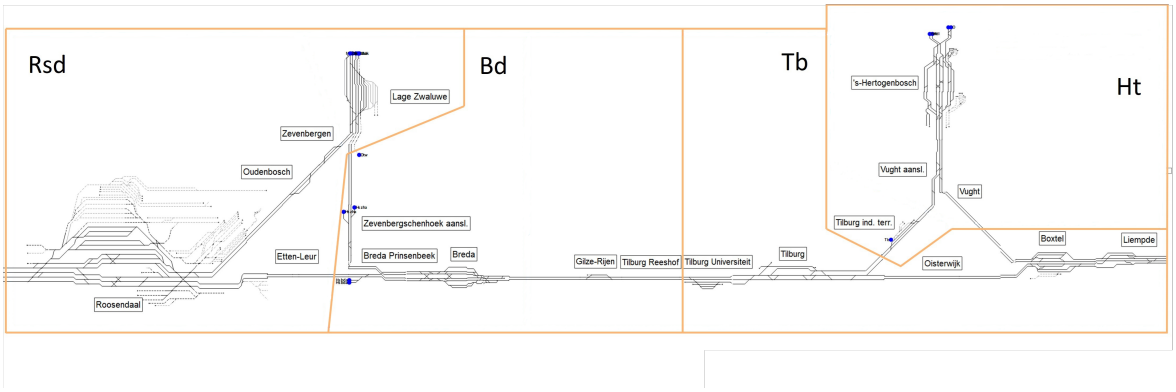


Figure 6.1: Case study network with decomposition borders (Hornung, 2023).
Rsd=Roosendaal, Bd=Breda, Tb=Tilburg & Ht='s-Hertogenbosch

The timing points included in the network are all switches and station platform tracks used by trains included in the timetable. The specific location of the timing points is determined by their relative distance to each other as given in the dataset. Whether a switch timing point is located exactly at the center of a switch or at one of its far ends is not very relevant as the difference is at most a couple of meters, which is negligible when comparing it to the distances between timing points of potentially multiple kilometers. A total of 330 timing points is included, of which 56 are station platform tracks and 274 are switches. There are some differences between the actual number of platforms and the number of timing points introduced in the case study. Whether a station platform track is given a timing point is determined as follows. There exist station platforms long enough to accommodate two trains simultaneously. In the Netherlands the lowercase letters *a* and *b* are often added to a platform number to distinguish between them. Both are given a separate timing point if each end is used by at least one train in the timetable as a stop. For a small number of platforms which are not long enough to service two trains simultaneously, two timing points were created as well. This is the result of the exact location of stopping of some trains at the platform being different from some other trains in the used dataset. To keep consistency in running times and distances between switches and these platforms, it was decided to add two separate platform timing points (denoted with *.1* and *.2* with the former for the first point in the direction of running, which is always the same for the platforms split like this). The route of each train through a station, and thus the platform they use, is determined from the case study dataset by tracing the switches they pass. In a small number of cases both the last switch before the platform and the first switch after the platform are the same for two platforms. These platforms are therefore combined into a single timing point (denoted with a */* between the two platform numbers). No additional conflicts were noted as a result of this.

In Table 6.1 below, an overview of the number and type of timing points per area is given. Note that the sum of the number of timing points reported in the table does not match the total. This is the result of the border points being included in both areas they form the border of. Border points can be both station platform tracks or switches.

Table 6.1: Overview of timing points per area

	Area					
	Rsd	**Bd**	**Tb**	**Ht**	**Sum**	**Total**
Number of station platform tracks	16	16	17	10	59	56
Number of switches	84	70	64	67	285	274
Sum	100	86	81	77	344	330

6.1.2. Timetable

For the creation of the timetable used in this case study, a 2022 dataset provided by ProRail was used. This dataset contains the possible variations of schedules for each train number series passing through the case study network. Train number series are collections of trains operating (mostly) the same route throughout the day. Although the timetables in the Netherlands are generally periodic with 1-hour cycle times, some slight differences can exist between trains of the same series running during for example the early morning and the rest of the day. The dataset includes all such possible variations. From the dataset, a selection of train number series variants was made to best resemble the timetable of a regular weekday (Monday to Thursday) during the middle of the day. Most scheduled trains are included in the case study timetable. The trains to and from Belgium and freight trains were however not included as a result of a lack of sufficient data on them. The Sprinter train between Roosendaal and Vlissingen with train series number 6100 is completely absent from the dataset and therefore not included. Trains running without passengers, known as empty trains, are excluded from the timetable as well. An overview of the included train number series can be found in Table 6.2. This table also shows the route of the trains through the case study network on the basis of their stops, the number of trains operated per hour per direction and the type of train operating the route. More details on the characteristics of each train type can be found in subsection 6.1.3. For the routes indicated in the table, note that for trains entering or leaving the network from the network border, the first station stop outside the network is given and thus not necessarily the whole route from start to finish.

Table 6.2: Overview of train number series included in the case study timetable

Train series number	**Route (stations with stops)**	**Number per direction per hour**	**Train type**
800 (IC)	Utrecht - 's-Hertogenbosch - Eindhoven	2	VIRM
900 (IC)	HSL - Breda	2	BR186+ICR
1100 (IC)	HSL - Breda - Tilburg - Eindhoven	2	BR186+ICR
1900 (IC)	Dordrecht - Breda - Tilburg - (Boxtel) - Eindhoven	1	ICMm
2200 (IC)	Dordrecht - Roosendaal - Bergen op Zoom	2	VIRM
3500 (IC)	Utrecht - 's-Hertogenbosch - Eindhoven	2	VIRM
3600 (IC)	Roosendaal - Etten-Leur - Breda - Tilburg - 's-Hertogenbosch - Oss	2	DDZ
3900 (IC)	Utrecht - 's-Hertogenbosch - Eindhoven	2	VIRM
4400 (SPR)	's Hertogenbosch Oost - 's-Hertogenbosch - Vught - Boxtel - Best	2	SLT
5900 (SPR)	Dordrecht Zuid - Lage Zwaluwe - Zevenbergen - Oudenbosch - Roosendaal	2	SLT
6000 (SPR)	Zaltbommel - 's-Hertogenbosch	2	SLT
6400 (SPR)	Tilburg Universiteit - Tilburg - Oisterwijk - Boxtel - Best	2	FLIRT
6600 (SPR)	Dordrecht Zuid - Lage Zwaluwe - Breda-Prinsenbeek - Breda - Gilze-Rijen - Tilburg Reeshof - Tilburg Universiteit - Tilburg - 's-Hertogenbosch - 's-Hertogenbosch Oost	2	FLIRT

For the case study, a time window of one hour is considered, starting and ending on the hour. Trains thus enter the network at either the external border at the time the train is scheduled to cross into the network, at any timing point within the network at t=0, or at a station within the network where the start of a train service is scheduled, at the time it is scheduled to depart. Similarly, the route of a train ends at the external border at the time a train crosses out of the network, at a station within the network which is the final destination of a train, or at any timing point at t=60min. Trains leaving the network before the end of the hour are split into two, where the part falling outside the hour is moved to the start of the hour. This way, the entire time window is filled with trains. This results in a total of 68 trains being considered.

In Figure 6.2 below, a time distance plot is shown as an example. The plot focuses on a train from the 2200 series, in this case variant H-2. In the plot all timing points this train passes are indicated. All other trains using some of the same infrastructure are plotted as well. Since the 2200 series runs with a frequency of two per hour, a second variant of the series can be seen as well, namely variant H-1. This trains passes the hour boundary within the network. Therefore it has been split into two separate parts, one ending at Rsd4b a few minutes before the end of the time window, and another one starting one timing point further, at Rsd101B, just after the start of the time window.

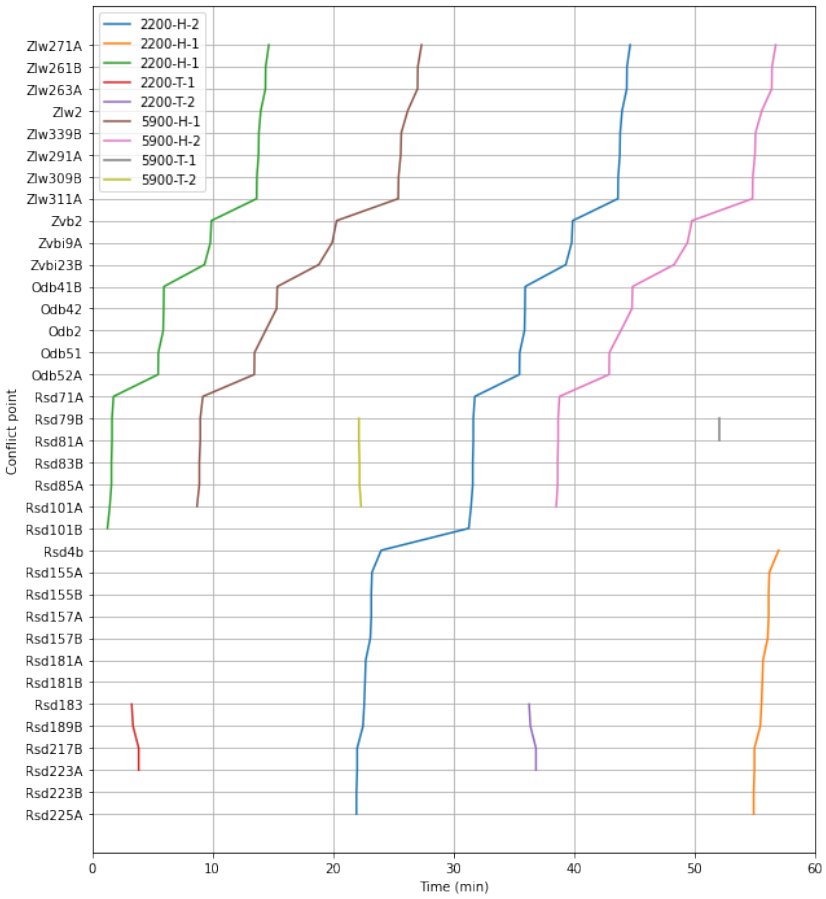


Figure 6.2: Time distance plot for train 2200-H-2

By combining the number of timing points in each areas with the timetable, the number of events, which are arrivals of a train at a timing point, can be determined. The result can be found in Table 6.3. In addition, the total number of variables, the number of continuous variables, the number of binary variables and the number of constraints included in the conflict resolution model of each area as well as in the centralized model are given. Note that this are all the variables an constraints created. Not necessarily every variable is also used in a constraint and not necessarily every constraint is relevant. The comparatively huge number of binary variables are the result of an oversight in the programming of the model where an $\lambda_{i,k}^{j}$ variable was created for every combination of train pairs and timing points. This lead to the creation of redundant variables with for example a pair of trains for a timing point where one of the trains never uses that timing point.

Table 6.3: Overview of events, variables and constraints per area

	Area					
	Rsd	**Bd**	**Tb**	**Ht**	**Sum**	**Centralized**
Number of events	327	383	478	432	1620	1572
Number of decision variables	107583	147455	229440	187488	671966	2474328
Number of continuous variables	654	766	956	864	3240	3144
Number of binary variables	106929	146689	228484	186624	668726	2471184
Number of constraints (without initial delays)	2954	4568	6441	5893	19856	19384

6.1.3. Parameters

Besides the scope of the network and the timetable, several more parameters are needed to run the model. They can be grouped into train specific and general parameters.

Train specific

The 68 train services included in the timetable are performed using six different train types. Of these types, two occur with more than one length. All have a maximum speed of 160 km/h, except for one which has a maximum speed of 140 km/h. In Table 6.4 below, all different types with their relevant lengths and maximum speeds as well as the number of trains operated are shown.

Table 6.4: All included train types with their respective lengths, maximum speeds and number of them operated

Train type	VIRM	ICMm	DDZ	BR186+ICR (loc+coach)	SLT	FLIRT
Length(s) (m)	162/270	81	154	276	138/170	144
Max. speed (km/h)	160	160	140	160	160	160
Number operated	5/16	3	8	10	6/8	12

To determine the braking, clearing and running times of the trains, some assumptions had to be made. Like explained in subsection 3.1.2, the model does not determine an exact speed trains should be travelling at when passing a timing point, but instead chooses one of two speed levels: scheduled speed or maximum speed. For both speed levels, simulation data from ProRail was used containing the speeds and corresponding braking distances of all trains at the timing points, as well as their arrival and departure times there.

To determine the approach time of a train at a timing point, its braking distance was divided by its speed at the timing point. Recall that the approach time is not the time the train needs to stop, but the time it needs to cover the braking distance when it is not braking. Or in other words, if a train should stop before a timing point, it should start braking a braking distance before it. If it is however clear to continue and is allowed to maintain its current speed, the time the train needs to cover the braking distance with that speed is the approach time. The assumption underlying this method is that the speed of a train does not change in the period it needs to approach the timing point. For example, if the braking distance of a train at a certain timing point is 200 m and its speed is 10 m/s, its approach time would be 20 s. However, it is now assumed that 20 seconds earlier, or 200 meters back, the train had the exact same

speed and braking distance. This assumption will lead to trains having a potentially larger than realistically required headway when slowing down and a potentially smaller than realistically required headway when accelerating.

Similar to the approach time, the clearing time was determined by dividing the train length by its speed. Here it is assumed that the train is not accelerating or decelerating while parts of the train are still located at the timing point. The running times are calculated by either taking the difference between the departure time at a timing point and the arrival time at the consecutive point if the first of these points is a station platform track with a scheduled stop, or the difference between the arrival times at both points otherwise.

Regarding the dataset used for the scheduled speed level, it should be noted that the running times between two stations do not add up to the difference between the actual scheduled departure and arrival times given in the timetable. Consequently, the speeds are higher and the resulting braking and clearing times thus longer. This is the result of, what ProRail calls slack, not being included in the simulation that generated the dataset. This slack allows train drivers some room to not always drive exactly according to the scheduled speed and also allows small disturbances to be recovered without inevitably causing a delay. As most timing points are switches in station areas, where maximum track speeds are generally low, the speed simulated for the scheduled speed level is often equal to the maximum track speed and therefore also to the speed in the maximum speed dataset. The only timing points where differences between scheduled and maximum speeds therefore occur are in places where the maximum track speed is high. In total 25 passages of trains at timing points, which is 1.59% of the total number, have a different speed and therefore braking and clearing time for the two levels.

Because the running times of the scheduled speed dataset are thus shorter than the difference between the departure and arrival times in the timetable, conflicts would occur by default if trains would actually adhere to them. Therefore, the running times for the scheduled speed level needed to be adjusted. Since the timetable only provides the scheduled departure and arrival times at stations, an assumption had to be made for determining the running times between other timing points. Namely, that the percentage a running time between two timing points takes of the total running time between the last and next station remains the same. Take for example two stations A and B with two switch timing points between them. If the running time from station A to the first switch was 10% of the total running time from A to B before, the new running time between A and the first switch will remain 10% of the total running time between A and B. If the total running time between A and B was originally 10 minutes and is now 15 minutes, the running time between station A to the first switch timing point changes from 1 minute to 1.5 minutes.

It was decided to not adjust the braking and clearing times. Primarily because it is difficult to predict how the speeds would change in switch areas, if they change at all. Moreover, it is better to be on the conservative side in regards with the parameters that influence headways, which assuming higher speeds accomplishes.

The minimum dwell times at stations were calculated for each train at every scheduled stop by subtracting its arrival time from its departure time using the maximum speed dataset. The longest minimum dwell time obtained is 60 seconds, while the shortest is 42 seconds.

General

Some parameters are the same across all trains and timing points. These are the setup, reaction and release time. As explained in subsection 2.3.2, the setup time consists of the time needed to set the route plus the RBC communication time. The reaction time refers to the time it takes a human to react to a stimulus and also includes the time it takes the train's onboard computer to compute a dynamic speed profile from the provided movement authority. The release time consists of the time needed to release the route plus the TPR delay. For the RBC communication time and the TPR delay, a value of 0.5 seconds is assumed. The reaction time is assumed to be 4 seconds. These values are the minimum as also used by Quaglietta et al. (2022). By choosing the minimum values for the RBC communication time and TPR delay, the most optimistic scenario with the best performing communication times for the implementation of moving-block signalling is assumed, resulting in the smallest possible headways. Furthermore, it is assumed train drivers thus have a quick reaction time. The time needed for setting and releasing the routes is separated from the setup and release times as they are dependent on whether for example a change of direction of a switch is required. The setup, reaction and release time are then constant for all timing points with a combined value of: 0.5+0.5+4=5 seconds. The time needed to change the direction of a switch (the switch setting time) is estimated to be 10 seconds and is thus only applied in case the direction actually changes. This value is purposefully estimated relatively high to avoid any headways possibly becoming to small, potentially resulting in a dangerous situation. Furthermore, a safety margin of 5 seconds is added to all headways, also at timing points where changing the direction of a switch is not required. Finally, the linear model value of M is set at 999,999. This is a random high number which is assumed to be sufficiently large. No consideration went into determining this number.

In Table 6.5 below, an overview of all parameter values described in this section can be found.

Table 6.5: General model parameters

Parameter	Value
RBC communication time	0.5 s
TPR delay	0.5 s
Human reaction time	4 s
Switch setting time	10 s
Safety margin	5 s
M	999,999

6.2. Delay scenarios

Five different delay scenarios are created to test the model. Each scenario is based on disturbances that could occur in reality. The five scenarios are as follows:

1. Single train delayed at the border
2. Single train delayed at a station
3. Single train delayed at any other arbitrary timing point
4. Group of trains delayed at the same border point
5. Group of trains delayed at different arbitrary timing points

The first scenario represents a single train being delayed outside the network. The second scenario represents a single train being delayed at a station in the network, for example due

to a large number of boarding and alighting passengers. It was decided to create a distinct scenario for station delays, instead of including them with the third scenario, because stations are occupied for a greater percentage of the time than a switch, thus conflicts are more likely to occur there. The third scenario represents any delays occurring during the running of a train, for example due to a speed restriction as a result of a malfunctioning level crossing. The fourth scenario represents a disturbance outside the network causing all trains from a certain direction to be delayed. The fifth scenario represents different trains being delayed at the same time, without necessarily any relation between them. Contrary to the third scenario, in the fifth scenario, trains are also allowed to be delayed at the border or at a station.

To be able to test the coordination model several delays resulting in infeasibilities need to be included. Since finding specific delays that result in infeasibilities by hand is quite a challenge, it was decided to test for a large systematically generated amount of delays. One would then statistically expect infeasibilities to occur in at least some of the tests. For the first, second and fourth scenarios, delays at all possible combinations of trains and scenario specific timing points are therefore tested. For the first scenario this thus entails testing each train that enters the network from the border, for the second scenario testing for all stops of all trains and for the fourth scenario testing at all border points. Each of these tests is performed with three amounts of initial delay: 5, 10 and 15 minutes. A minimum of 5 minutes of delay was chosen as this is the smallest delay to possibly cause any conflicts. A maximum of 15 minutes was chosen as it is still small enough to be counted as a disturbance (and not as a disruption) while also being large enough to cause enough conflicts in the network to require the distributed model to run all four areas in some attempts. For the third and fifth scenarios, arbitrary trains receive a random delay between 5 and 15 minutes at an arbitrary point along their routes. Points close to the end of the time window receive a delay of at most the time they are separated from it. In the fifth scenario the number of trains given a delay is randomly decided with a minimum of two, and a maximum of four trains. For both of these last scenarios 400 draws were generated.

In Table 6.6 below, an overview of the delay scenarios and the number of runs attempted is shown. In the rightmost column, the number of attempted ‘runs’ is indicated. One run entails solving one combination of delays once using the centralized model and once using the distributed model. In scenario 1 there is one initially delayed train per run. This train has its initial delay imposed at the border. For each train entering the network through the border, up to three runs are performed: one with an initial delay of 5 minutes, one with 10 minutes and one with 15 minutes. This initial delay is only imposed if the planned arrival time plus the imposed delay does not exceed the time window barrier of one hour. If it does exceed the time window the run is not performed. In the first row of the table one can read that for scenario 1, there is one initially delayed train per run, a total of 46 initial delays of 5 minutes were imposed, a total of 43 of 10 minutes delay and a total of 39 of 15 minutes of delay. The total number of runs performed in this scenario is the sum of this, namely 128 runs. In scenarios where more than one train was initially delayed, the sum of initial imposed delays can exceed the total of attempted runs. This is because in these scenarios, multiple initial delays are imposed per run, namely on every train which should have an initial delay.

Table 6.6: Number of delays and runs attempted per scenario

Scenario	Number of initially delayed trains	Number of initial delays imposed of				Number of attempted runs
		5 min	10 min	15 min	Total	
1	1	46	43	39	128	128
2	1	96	83	77	256	256
4	2-9	38	38	38	114	24
		300-500 s	**500-700 s**	**700-900 s**	**Total**	
3	1	178	120	102	400	400
5	2	120	83	71	274	137
5	3	185	98	83	366	122
5	4	259	162	143	564	141

The delays are added to the conflict resolution model as constraints. For all scenarios except scenario 2, the delay is added as a constraint of the arrival time at the corresponding timing point. A delay of 5 minutes for train 1100-H-1 at switch Lpe1257B would be added to the model as: $x_{1100-H-1}^{Lpe1257B} = a_{1100-H-1}^{Lpe1275B} + 300$. As a reminder, $x_{1100-H-1}^{Lpe1257B}$ is the arrival time variable assigned by the model and $a_{1100-H-1}^{Lpe1257B}$ the scheduled arrival time. For the delays at station platform tracks in scenario 2, the minimum dwell time is increased. This is done by changing the input parameter associated with the dwell time (dw_i^j). To allow the distributed model to detect which area to run, an external constraint does however need to be added to the model. Therefore, a constraint is added setting the arrival time at the station where the delay occurs to be equal to the scheduled arrival time, for example: $x_{1100-H-1}^{Tb2} = a_{1100-H-1}^{Tb2}$. This also prevents the delayed train from speeding up before the station where the delay occurs in order to arrive there earlier to sit out (part of) the added dwell time.

6.3. Results

With all model parameters set and all scenarios created, the results are generated. To shortly recap, the aim of this thesis is to determine to what extent a distributed conflict resolution model applied to moving-block signalling can reduce computation time compared to a centralized model. As the main question already implies, the expectation is that the distributed model is able to reduce computation time. At the same time, like was found and explained in chapter 5, it is expected that the quality of the solution, that is the cumulative reduction of delay at stations, decreases for some combinations of delays. If these expectations come true, a trade-off between computation time and solution quality will occur. The results reported in this section should provide insight into the quantitative aspect of both sides of this trade-off such that the decision whether this trade-off would be worth it could be made.

It should be noted that although the quality of the solution is based on the sum of arrival delays of all trains at their scheduled station stops, the objective function of the conflict resolution model also includes the delay at border points. See subsection 3.2.2 for the explanation why. When the resulting delay of a run is mentioned in this section, it is about the solution quality, so the cumulative station delay, and thus not the objective function value.

All results were generated using a laptop with a 2.20 GHz CPU and 16 GB of RAM. The software versions used are Python 3.7.6 and Gurobi 9.5.1. The number of successfully completed runs can be found in Table 6.7 below. In scenario's 1 to 4, all attempted runs were successfully completed. However, in scenario 5, six runs failed to find a feasible solution. The reason for

these failures is explained in Appendix E. Since the number of failed runs is relatively small, and a fix for the model was expected to take too long to implement, it was decided to ignore them going forward and focus on the results of the runs that did solve successfully. The results of the remaining 1202 runs which were solved are analyzed for their computation time and cumulative station delay separately in subsection 6.3.2 and subsection 6.3.3 respectively. The computation time and cumulative station delay results are then combined in subsection 6.3.4.

Table 6.7: Number of delays and successful runs per scenario

Scenario	**Number of initially delayed trains**	**Number of initial delays imposed of**				**Number of successful runs**
		5 min	**10 min**	**15 min**	**Total**	
1	1	46	43	39	128	128
2	1	96	83	77	256	256
4	2-9	38	38	38	114	24
		300-500 s	**500-700 s**	**700-900 s**	**Total**	
3	1	178	120	102	400	400
5	2	120	83	71	274	137
5	3	185	98	80	363	121
5	4	249	156	139	544	136

6.3.1. Characteristics of the results of the models with no initial delays

To put the effects of delays on the computation time into perspective, the resulting parameters of running the models without delays is shown first. The results were generated by adding an initial ‘delay’ of 0 seconds to a single train. Four runs were performed, where in each run the initial ‘delay’ was applied to a timing point in a different area. This allows for testing the distributed model for all four areas. These baseline results can be found in Table 6.8. Besides the computation time (c.t.) of each run, the total cumulative station delay is also included in the table to confirm the timetable and models do not contain any conflicts resulting in delays at stations. In practice, running a conflict resolution model would of course not be necessary if no delays were detected like in this case.

Table 6.8: Baseline results

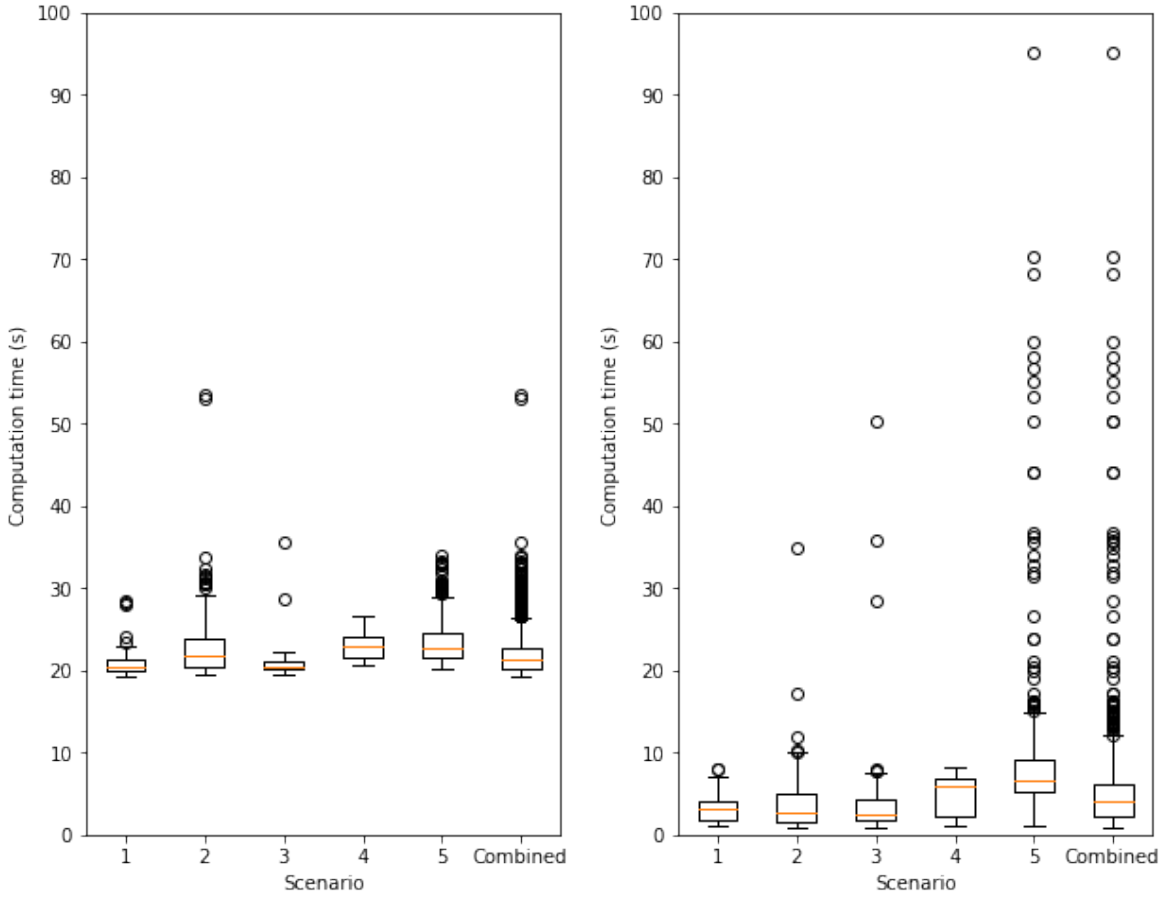
Area of initial ‘delay’	**Centralized c.t. (s)**	**Distributed c.t. (s)**	**Cumulative station delay centralized (s)**	**Cumulative station delay distributed (s)**
Rsd	22.6	1.1	0	0
Bd	20.4	1.6	0	0
Tb	21.6	2.3	0	0
Ht	20.7	3.4	0	0

The first and most important observation of these results is that the centralized model takes significantly longer to compute than the distributed model. Secondly, it can be observed that there is a difference in computation times between the four different locations where the ‘delay’ is applied for both the centralized and the distributed model. The difference in computation time between the longest and shortest computation time for both models is at most approximately 2 seconds. Interestingly, where the longest computation time for the centralized model is found when the ‘delay’ is applied in the Rsd area, the delay in this area results in the shortest computation time found for the distributed model. A possible explanation is that the computer is performing other tasks in the background as well leading to some spread in the computation

times.

6.3.2. Computation time

The computation time results are reported as follows. In Figure 6.3, boxplots of the computation times for each individual scenario as well as one of the data of all the scenarios combined are given for both the centralized and the distributed model. In Table 6.9, the number of runs where the computation time of the distributed model is longer than the centralized model is given for every scenario.



(a) Centralized

(b) Distributed

Figure 6.3: Boxplots of the computation time per scenario

Table 6.9: Number of runs where the computation time of the distributed model is longer than the centralized model

Scenario	Distributed c.t. >centralized c.t.	Percentage of scenario total
1	0	0%
2	1	0.39%
3	3	0.75%
4	0	0%
5	20	5.08%
Combined	24	2.00%

The main thing to be observed from these results is that the distributed model is almost always

faster than the centralized model. The boxplots in Figure 6.3 clearly show that the computation time of the centralized model is on average approximately 20 seconds, with only some outliers taking more than 30 seconds to compute. Meanwhile, the distributed model has an average computation time of approximately only 5 seconds. On average, the distributed model is thus 15 seconds faster. Some outliers found for the distributed model are more extreme than the centralized model however and can even exceed the computation time of the centralized model. This is especially the case for scenario 5. This can also be observed from Table 6.9, where it can be seen that in scenario 5 20 runs took longer to compute using the distributed model compared to the same run using the centralized model.

All runs where the computation time of the distributed model is longer than the computation time of the centralized model have in common that an infeasibility was detected at some point during the coordination process. This does however not mean that the need to solve an infeasibility automatically makes the distributed model slower than the centralized model. In fact, 25 runs with a solved infeasibility still had a shorter computation time for the distributed model. Despite this, the need to solve an infeasibility still does significantly increase the computation time of the distributed model. To illustrate this, the computation time boxplots for the distributed model with the runs requiring solving infeasibilities excluded are given in Figure 6.4 below.

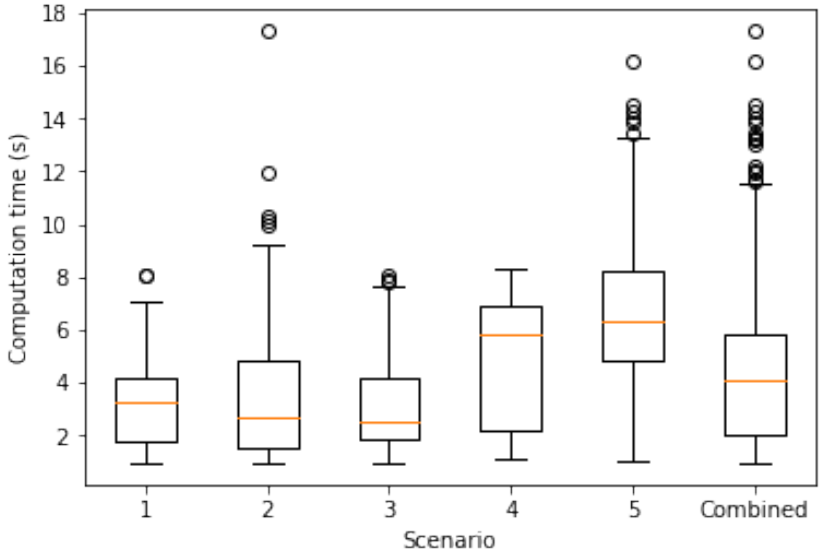
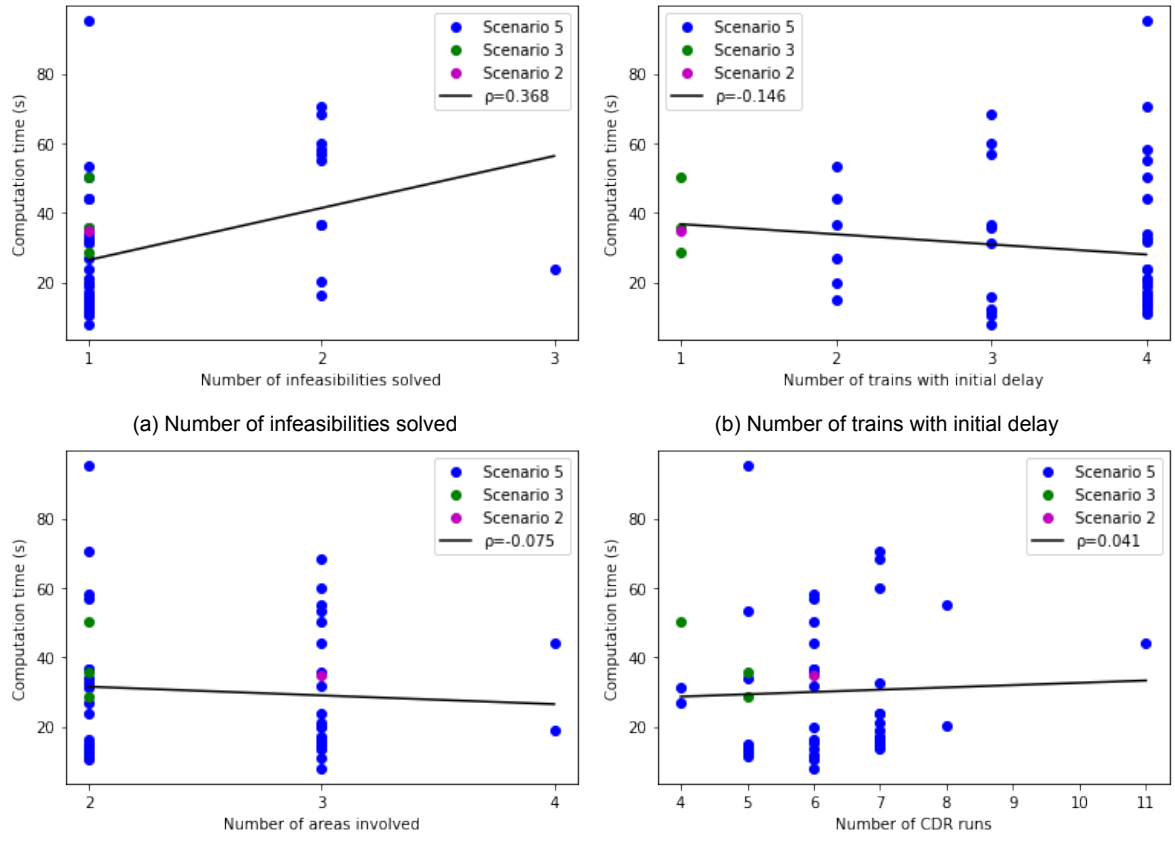


Figure 6.4: Boxplots of the distributed computation time with runs requiring solving infeasibilities excluded

Comparing Figure 6.4 with Figure 6.3b shows the significant impact on computation time the need to solve infeasibilities has. To understand why the computation time of the distributed model sometimes exceeds that of the centralized model, some additional possible factors influencing the size of it are researched. The aim of doing this is to potentially be able to predict which model would be solved faster in case an infeasibility is detected when attempting to solve the distributed model. Knowing the distributed model has a high probability to take more time to compute when continuing than switching to the centralized model, enables the choice to actually switch. Therefore, four possible relations were investigated: a possible relation between the computation time of runs where infeasibilities are solved and:

1. the number of infeasibilities solved during the run
2. the number of trains with an initial delay
3. the number of areas involved by the coordination algorithm
4. the number of conflict resolution (CR) runs started by the coordination algorithm

If a relation would exist between the computation time and any of these parameters, it is expected that the relation would be linear. For example, one additional CR run would mean an additional area needs to be (re)solved, which is expected to take about as much time as any other run. In Figure 6.5 below, scatter plots are displayed for each of the relation candidates. To test the possible relation, a regression line is added. The corresponding Pearson correlation coefficients are reported in the figure legends (Pearson, 1895).



(a) Number of infeasibilities solved

(b) Number of trains with initial delay

(c) Number of areas involved

(d) Number of CDR runs

Figure 6.5: Possible relations

From these plots it can be concluded that there is no obvious relation between the computation time of the distributed model and the number of trains with an initial delay, the number of areas involved or the number of CR runs. This because the Pearson correlation coefficients of the regression lines fitted to these relation candidates are close to zero. Moreover, the slopes of the regression lines of the second and third candidates are negative, while logically one would expect them to be positive as finding a solution with more delayed trains or finding a solution while having to coordinate more areas should be harder. Based on its Pearson correlation coefficient, a relation between the computation time and the number of infeasibilities solved is more probable, although still unlikely. The fact that there are not many runs where more

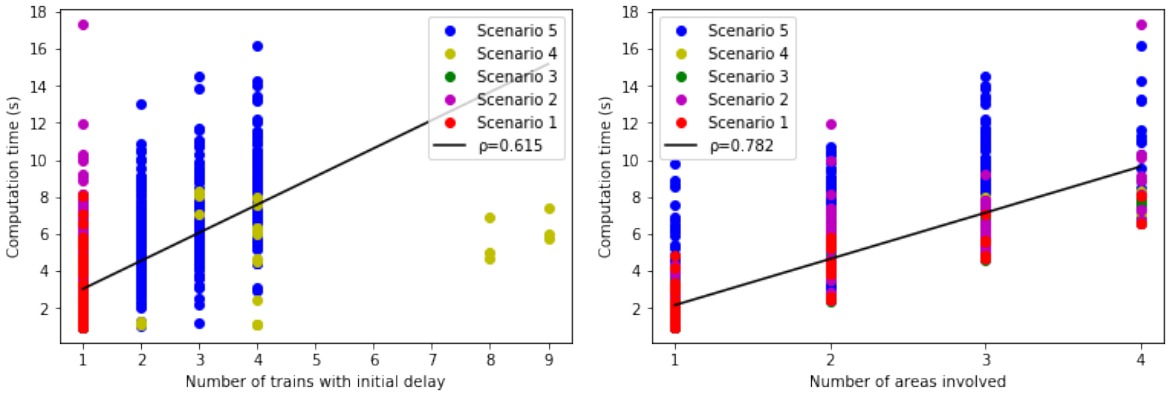
than one infeasibility was solved increases the uncertainty of this possible relation even further.

The only other factor remaining that could possibly explain the large computation times of the distributed model in case of an infeasibility is the calculation of the Irreducible Inconsistent Subset (IIS), the set of constraints that make the model infeasible. Presumably, certain infeasibilities make the IIS harder to calculate. It is however not known how the Gurobi function used to determine the IIS precisely works and which factors influence it. The question why some infeasibilities take longer to solve therefore remains open.

To explore whether the computation time of the distributed model for runs without the need to solve an infeasibility could theoretically exceed that of the centralized model, even though no such cases were observed in the case study runs, some possible relations with other parameters are investigated as well. For the distributed model three relations which were also analyzed for the distributed runs with infeasibilities to solve are considered: between the computation time and

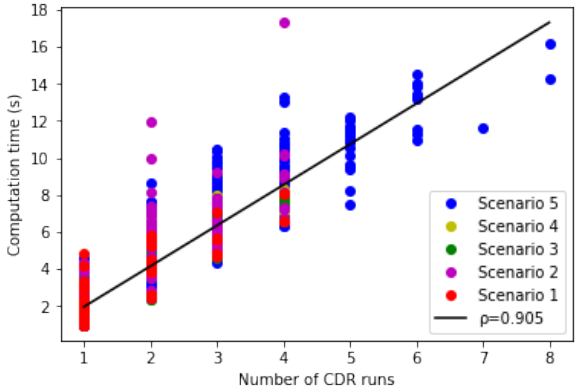
1. the number of initially delayed trains
2. the number of areas involved by the coordination algorithm
3. the number of conflict resolution (CR) runs started by the coordination algorithm

Like for the distributed model with infeasibilities to solve, the results are displayed in scatter plots with a regression line added. Again, it is assumed that any possible relations would be linear. The results can be found in Figure 6.6 below.
The plots show very clear linear relations between the computation time and the number of areas involved and the number of CR runs. Also the Pearson correlation coefficient of the regression line plotted for the relation between the computation time and the number of trains with an initial delay suggests a relation exists with that parameter. It should be noted that the number of CR areas and the number of areas involved are very much related with each other. Having one more area involved in the coordination does mean one more CR run needs to be performed. Likewise, the amount of initially delayed trains and the number of CR runs are related. Adding one more train makes it more likely that an additional area is involved and thus that one or multiple additional CR runs need to be performed. Since the clearest relation with the highest Pearson correlation coefficient is found for the number of CR runs, this is assumed to be the primary parameter for predicting the computation time. Using this relation, it can be predicted that the average computation time of the distributed model would equal that of the centralized model at about 10 CR runs. This also happens to be the theoretical maximum number of CR runs without any infeasibilities occurring for the case study network, see Figure 6.7. Since the distributed model was tested for only one network partition, it remains unclear if a partition with more areas would also result in the theoretical maximum number of CR runs having a computation time similar to the centralized model. The maximum number of CR runs would go up by 5 for one additional area. It remains to be seen however if the additional computation time per CR run (which would now involve areas with less variables) would decrease by 33% however (now: 2 sec / CR run, with add. area: 1.33 sec / CR run).



(a) Number of trains with initial delay

(b) Number of areas involved



(c) Number of CR runs

Figure 6.6: Possible relations distributed model with no infeasibilities

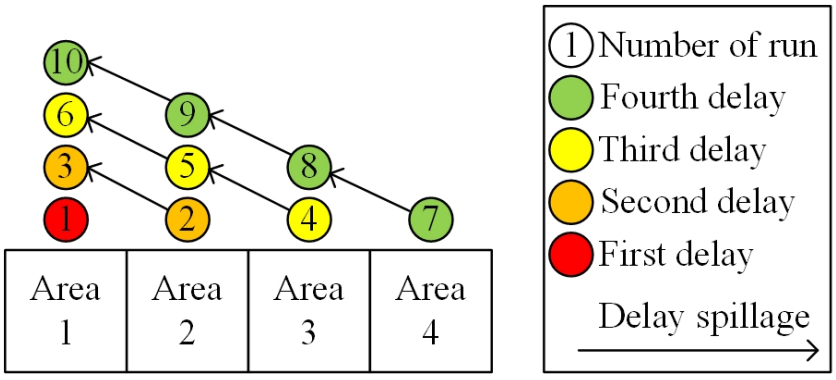
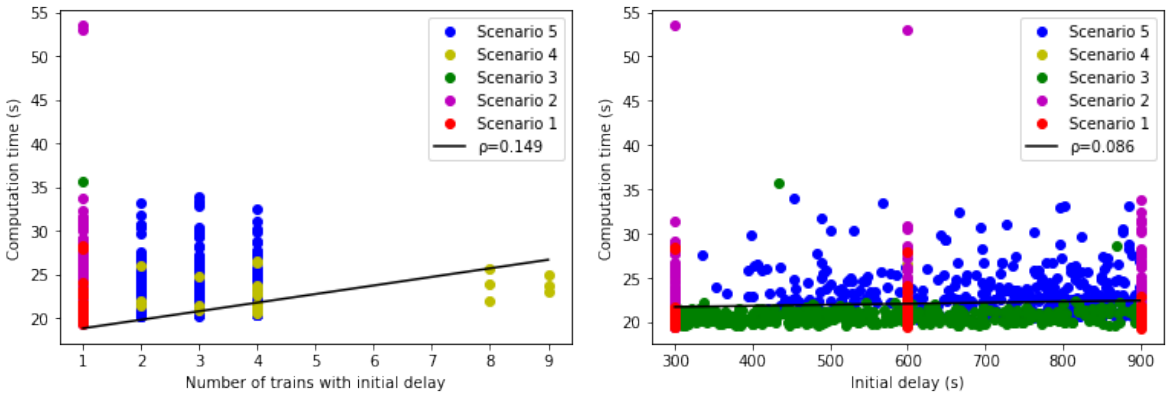


Figure 6.7: Visualization of the maximum number of CR runs without an infeasibility

Since there does seem to be a relation between the computation time of the distributed model and the 'complexity' of the delay input if there are no infeasibilities, two possible relations between the computation time and parameters representing the 'complexity' are considered for the centralized model as well: between the computation time and the number of initially delayed trains and between the computation time and the size of the initial delay. These possible relations are considered because the centralized model does not deal with multiple areas and because by definition only one CR run needs to be performed. If the centralized computation time does turn out to increase with the number of initially delayed trains or the size of the initial delay, this would mean the hypothesis that there is a point at which the distributed

model computation time exceeds the centralized computation time becomes less likely. The results are plotted in Figure 6.8. For the size of the initial delay of scenario 5, the largest applied value in the run is used.



(a) Regression number of trains with initial delay vs computation time

(b) Regression size of initial delay vs computation time

Figure 6.8: Possible relations centralized model

From these figures and the corresponding Pearson coefficients, it can be concluded that there is no significant relation between the computation time and the number of delayed trains or between the compuatation time and the size of the initial delay in the centralized model. For the computation time of the centralized model it can therefore be said that it is more or less the same for any size or number of initial delays added to it.

6.3.3. Cumulative station delay

In Table 6.10, the number of runs per scenario for which the cumulative station delay is either larger in the distributed model, larger in the centralized model, or equal in both models is given. Figure 6.9 shows the frequency of the magnitude of the difference between the cumulative station delay of the centralized and distributed model for all scenarios combined. In this figure, a positive difference means the distributed model has a larger cumulative delay.

Table 6.10: Number of runs per scenario for which the cumulative station is larger in the distributed model, larger in the centralized model, or equal in both models

Scenario	**Number of runs with the cumulative station delay largest for**		
	Distributed	**Centralized**	**Equal**
1	40	14	74
2	99	25	132
3	116	39	245
4	8	10	6
5	168	164	62
Sum	431	252	519

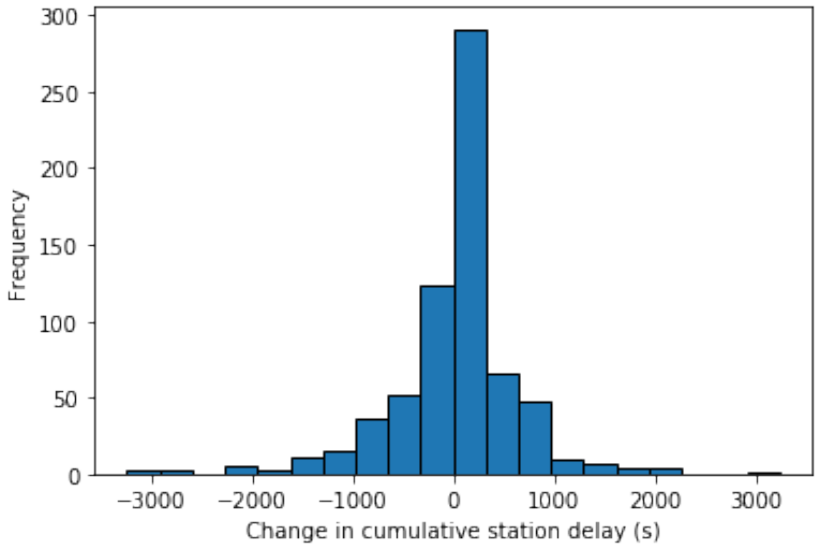
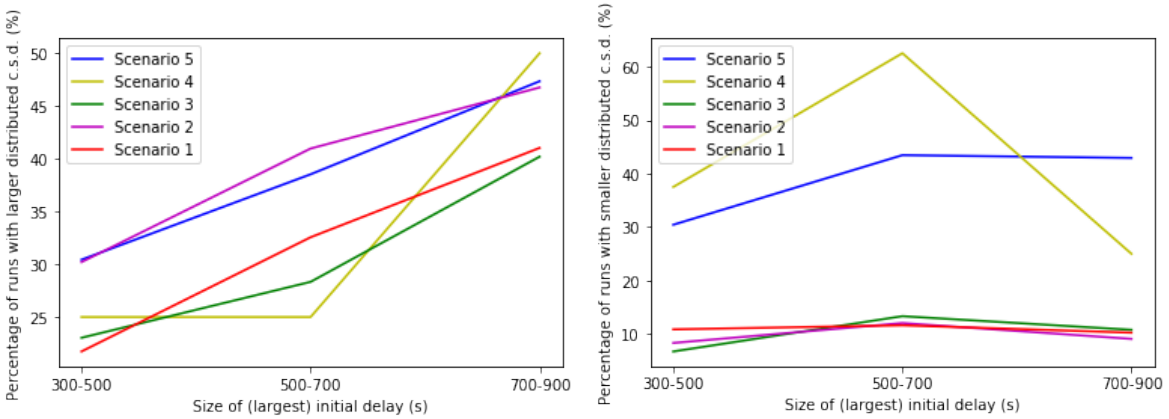


Figure 6.9: Histogram of the difference of cumulative station delay going from the centralized to the distributed model (excluding the runs with no difference, bin size = 323.7)

As explained before, the expectation was that the distributed model would result in longer or equal cumulative delays compared to the centralized model only. From Table 6.10 it can be concluded however that for some combinations of initial delays the distributed model results in less cumulative station delay than the centralized model. If there is a difference, the cumulative station delay is however still larger in the distributed model in about two thirds of the cases. Figure 6.9 shows that the difference, if there is one, is usually relatively small.

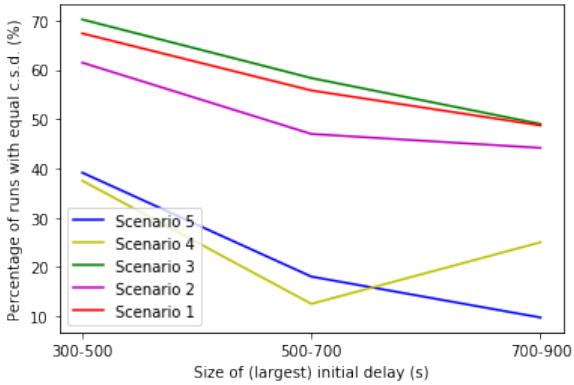
In the following, like was done in the previous subsection, an attempt will be made to find possible relations between the cumulative station delay and two input parameters. In contrast to the possible relations analyzed for the computation time, the relations with the cumulative station delay will primarily focus on determining whether the distributed model results in more, equal or less cumulative station delay than the centralized model. The size of this difference is thus less of interest. It was not necessary to have this focus for the computation time as most of the runs resulted in a smaller computation time for the distributed model and the runs for which the computation time was larger could be explained by the presence of an infeasibility.

The first possible relation analyzed is between the cumulative station delay and the size of the initial delay. In Figure 6.10, the three plots show per scenario the percentage of runs where the distributed model results in a larger (Figure 6.10a), smaller (Figure 6.10b) or equal (Figure 6.10c) cumulative station delay relative to the centralized model. For scenario 5, the largest initial delay per run is used to represent the whole run.



(a) Larger cumulative station delay in the distributed model

(b) Smaller cumulative station delay in the distributed model

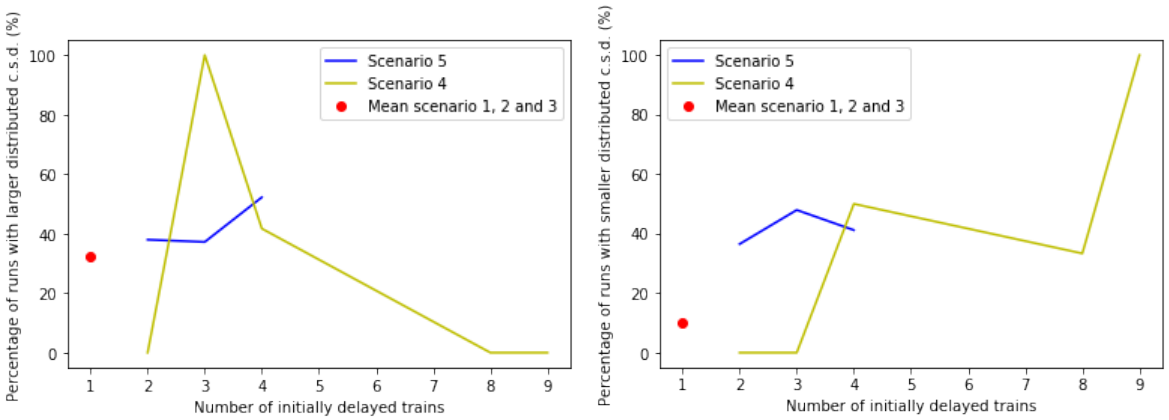


(c) Equal cumulative station delay in both models

Figure 6.10: Percentage of runs with larger, smaller or equal cumulative station delay for the distributed model relative to the centralized model for three initial delay ranges per scenario

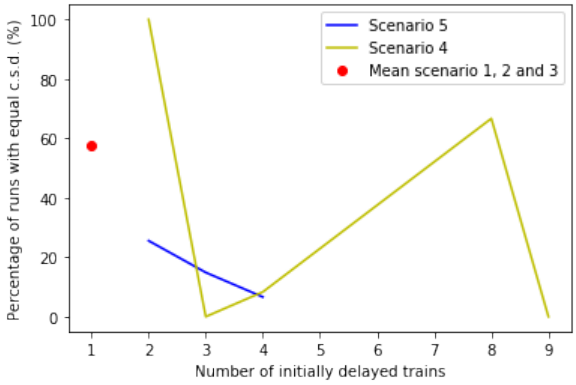
From these plots it seems that the larger the initial delay, the larger the number of runs resulting in a larger cumulative station delay for the distributed model. This seems to go at the expense of the number of runs for which the resulting cumulative station delay is equal among both models. The number of runs for which the cumulative station delay is smaller for the distributed model compared to the centralized model seems to remain about constant regardless of the size of the initial delay. Moreover, it is noticeable that for runs with multiple initially delayed trains (in scenarios 4 and 5), there are significantly less runs resulting in equal cumulative station delay and significantly more runs resulting in less cumulative station delay for the distributed model.

Following on from that, the second possible relation analyzed is between the cumulative station delay and the number of initially delayed trains. In a similar fashion, the plots for the shares of larger, smaller and equal cumulative station delays for the distributed model compared to the centralized model can be found in Figure 6.11. For these plots, all three scenarios with one initially delayed train are aggregated into a single percentage.



(a) Larger cumulative station delay in the distributed model

(b) Smaller cumulative station delay in the distributed model



(c) Equal cumulative station delay in both models

Figure 6.11: Percentage of runs with larger, smaller or equal cumulative station delay for the distributed model relative to the centralized model for the number of initially delayed trains per scenario

From these plots, any possible relations become less clear. For scenario 4, this is mainly due to the very small number of runs. This leads to some 100% and 0% data points. For scenario 5, this is due to the fact that the runs only included numbers of initially delayed trains in the range of 2 to 4. Disregarding the results for scenario 4 due to the small amount of data, there does seem to be a decrease in the share of runs with an equal cumulative station delay for both runs the more initially delayed trains there are. Meanwhile, the percentage of larger cumulative station delays for the distributed model increases. The percentage of runs with smaller distributed cumulative station delays seems to stay more or less the same. These observations are in line with what was seen for the size of the initial delay, in the sense that the greater the size of the delay or the number of delayed trains become, the less runs result in an equal amount of cumulative station delay and the more runs result in an increase of the cumulative station delay for the distributed model compared to the centralized model. Furthermore, it seems that runs with a single initially delayed train more often result in equal cumulative station delay for both models and less often result in less cumulative station delay for the distributed model compared to the centralized model.

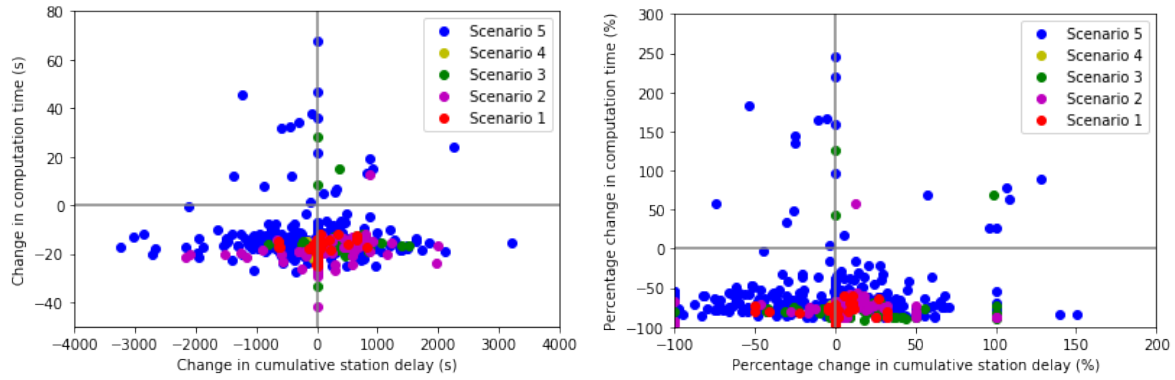
6.3.4. Computation time versus cumulative station delay

By combining the results of the computation time and the results of the cumulative station delay, insight into the trade-off between the two can be given. In Table 6.11, the number of

runs for which the distributed model has an either faster or slower computation time and either more or less resulting cumulative station delay compared to the centralized model is shown per scenario. In Figure 6.12, the change of the computation time and cumulative station delay when going from a centralized to a distributed model is plotted for both the change in true value and percentage.

Table 6.11: Number of runs per scenario with both smaller computation time (c.t.) and cumulative station delay (c.s.d.), smaller c.t. and larger c.s.d., larger c.t. and smaller c.s.d. and both larger c.t. and c.s.d. for the distributed model compared to the centralized model

	c.t. dist. ... c.t. cent.	**≤**	**≤**	**>**	**>**
	c.s.d. dist. ... c.s.d. cent.	**≤**	**>**	**≤**	**>**
Scenario					
1		88	40	0	0
2		157	98	0	1
3		283	114	1	2
4		16	8	0	0
5		215	159	11	9
Sum		759	419	12	12



(a) Value

(b) Percentage

Figure 6.12: Change of a run in computation time and cumulative station delay going from the centralized to the distributed model

This table and plot show that, firstly, there is no relation between computation time and the resulting cumulative station delay. A smaller computation time does not lead to necessarily a smaller or larger cumulative station delay and similarly, an increase in computation time does not imply either either. Secondly, the percentage change in Figure 6.12b shows that the computation time does not increase by more than 3.5 times and the resulting cumulative station delay by more than 2.5 times for the tested runs. Finally, the table and plot confirm that indeed most runs resulted in a shorter computation time in the distributed model while in about two third of cases the resulting cumulative station delay is equal or smaller. This share does seem to decrease however with more trains being included with an initial delay.

6.4. Summary

By applying the centralized and distributed models to five initial delay scenarios representing a total of 1208 model runs, the following results were found:

- 1202 runs were successfully solved by both the centralized and distributed model while

six runs failed to solve for the distributed model. Since the number of failed runs is relatively small, it was decided to ignore these runs and focus on the runs which did successfully solve.

- Of the runs that were successfully solved, 98% did so with a shorter computation time for the distributed model than for the centralized model. All distributed runs which took longer to solve have in common that they all required solving at least one infeasibility. Among the distributed runs with an infeasibility solved, no parameter was found having a direct relation with the length of the computation time. Therefore it was deduced that the only factor primarily influencing the computation time of these runs is the computation of the IIS. Since it is unknown how the IIS is computed, or what factors influence it, the reason why some runs with infeasibilities take considerably longer to compute than others remains unknown. Among all centralized model runs, no clear relation between the computation time and any other parameter was found either. For the distributed runs without infeasibilities a significant relation was found however. Namely, between the computation time and the the number of conflict resolution runs. Using this relation it can be predicted that the computation time of the distributed model would approximately be the same as the centralized model for the theoretical maximum number of conflict resolution runs possible with four areas, thus making it very unlikely the distributed model would need more time to compute as long as there are no infeasibilities. This poses the question whether this would also be the case for partitions with a different number of areas or areas with a different size.
- Of the runs that were successfully solved, 22% had a larger cumulative station delay for the distributed model, 15% for the centralized model and in the remaining 63% of runs, the cumulative station delay was equal for both models. If there is a difference in the cumulative station delay between the two models, it is most often relatively small with a difference smaller than approximately 300 seconds. It was found that an increase in the size of the initial delay leads to a higher percentage of the runs having a larger cumulative station delay for the distributed model compared to the centralized model and a smaller percentage of runs resulting in equal cumulative station delay for both models. A similar relation was found for increasing the number of initially delayed trains: more delayed trains yield a larger percentage of runs with a larger cumulative station delay for the distributed model and a smaller percentage of runs with equal computation time. The percentage of runs resulting in a smaller cumulative station delay for the distributed model does not seem to be affected by a change in the size of the initial delay or an increase in the number of delayed trains. It does seem however, that having more than one initially delayed train leads to a higher percentage of the runs resulting in a smaller cumulative station delay for the distributed model at the expense of the percentage of runs with an equal computation time.

Considering these results, it can be said that, at least for this particular case study, the distributed model is the more efficient option in most of the cases. Only when infeasibilities are present, there is a risk of the distributed model taking longer or not being able to complete at all. In most of the considered runs the resulting cumulative station delay was the same for both the centralized and distributed models. In case the models give different solutions the centralized model has the smallest cumulative station delay in most cases. There are however plenty of instances where the distributed model outperformed the centralized model in this metric. The probability of a run resulting in more cumulative station delay for the distributed than the centralized model increases the larger the initial delay or the more trains are initially delayed. For the largest considered initial delays, still over 50% of the runs result in shorter

or equal cumulative station delays for the distributed model however. Overall, it is therefore concluded that there is not much of a decrease in cumulative delay performance when switching to a distributed model, while there is a significant decrease in computation time. However, a decrease in relative solution quality and an increase in computation time of the distributed model is observed when there are more trains initially delayed. A decrease in relative solution quality of the distributed model is also observed for an increase in the size of the initial delay.

7

Conclusions

7.1. Conclusions

This research aims to answer the following main question: *What is the impact of a non-centralized real-time rail traffic management method on computation time and station arrival delay compared to a centralized method when applied to moving-block signalling?* To come to an answer to this question, four sub-questions were posed. The answers to these sub-questions are given before the main question is answered.

1. *What is the current state-of-the-art regarding centralized and non-centralized conflict resolution methods under moving-block and fixed-block signalling?*
   Conflict resolution models are often written as mathematical optimization models. The two main formulations used are the Alternative Graph (AG) and Mixed Integer Linear Programming (MILP), although other formulations exist as well. Non-centralized models can be divided in two approaches: distributed and decentralized. The main difference between the two is that the distributed model has a specific separate sub-problem in charge of coordination between all other sub-problems, whereas in decentralized models sub-problems negotiate directly with their neighbours to coordinate their solutions. There are multiple ways to decompose the main conflict resolution problem. The main ones are geographical decompositions, train based decompositions and temporal decompositions. Most previous research focused on conflict resolution for fixed-block signalling and centralized solving. Few research has been performed for conflict resolution for moving-block signalling and/or non-centralized solving.

2. *How to formulate a conflict resolution model for a non-centralized network with moving-block signalling?*
   The decision was made to formulate the conflict resolution model as a MILP since it is easier to make adjustments to and because it is best suited to be solved by a commercial solver. The challenges presented by the moving-block aspect of the model are dealt with by partitioning the infrastructure into so called timing points and by discretizing the speed into two levels. The timing points are the switches and station platform tracks. The decision variables of the model are linked to these points. At these points, trains should maintain a minimum separation to their predecessor as determined by the blocking time. The blocking time is determined by several factors, of which a part is influenced by their speed, most notably the absolute braking distance. The speed itself is discretized in two levels as calculating it continuously would be too computationally intensive. The

decision variables included in the model are the arrival time of trains at the timing points, the order in which the trains pass timing points and the speed level at which they pass the conflict points.

3. *How to coordinate the sub-problems of a decomposed conflict resolution model?*
   The type of non-centralized model chosen is the distributed model. The model has a dedicated algorithm coordinating the sub-problems. The coordination is however primarily aimed at finding a feasible solution, and not so much an optimal solution. The conflict resolution model is decomposed into sub-problems for the distributed model geographically. The coordination works by applying constraints to the timing points shared by neighbouring sub-problems determined by the result of the first one of them solved. If these border constraints cause the neighbouring sub-problem to be infeasible, the constraints causing this infeasibility are identified and deleted one by one until a solution can be found. This is a process that can go back and forth between two areas until a feasible solution is found for both.

4. *How does the proposed model applied to a case study compare to a centralized moving-block method regarding the computation time and delay reduction results?*
   The model was applied to a case study consisting of a large part of the rail network of the Dutch province Noord-Brabant. The distributed model found a feasible solution for 1202 out of 1208 attempts. In 98% of the cases the distributed model was solved faster than the centralized model. In 78% of cases the cumulative delay resulting from the distributed model at stations was smaller than or equal to the resulting delay for the centralized model. The difference in computation time is on average approximately 15 seconds, and the difference in cumulative station delay, if there is one, is usually smaller than 300 seconds. For all instances where the computation time of the distributed model exceeds that of the centralized model, an infeasibility was found. Finding an infeasibility does however not mean the computation time of the distributed model automatically exceeds that of the centralized model. It was found that an increase in the size of the initial delay or an increase in the number of initially delayed trains leads to a higher percentage of the runs having a larger cumulative station delay for the distributed model compared to the centralized model and a smaller percentage of runs resulting in equal cumulative station delay for both models.

Considering this, an answer to the main question can be formed: At least for cases similar to the case study performed here, a non-centralized real-time rail traffic management method applied to moving-block signalling can reduce computation time in almost all cases by a significant amount while for the majority of cases, the cumulative station delay decreases or remains the same compared to a centralized method. The more delays are initially applied or the larger the initial delays are, the more often the cumulative delay resulting from the distributed model will be larger than for the centralized model and the more likely an infeasibility will occur potentially leading to a longer computation time of the distributed model compared to the centralized model.

7.2. Discussion

In this section, this thesis will be discussed. The discussion is split in three categories: the limitations of the models, the limitations of the results found for the case study and the way the results relate to other research.

7.2.1. Limitations of the models

Models are a simplification of reality by definition. The most major deficiencies of the conflict resolution model and the coordination algorithm used for the distributed model are described here, starting with those of the conflict resolution model. Firstly, the number of traffic management measures available for the model to take is limited. The most impactful measure missing is rerouting. It is not hard to imagine that many potential conflicts could be solved much more efficiently by locally rerouting trains. Especially near stations where multiple alternative routes through switch areas could be taken or alternative platforms could be used. This lacking from the model thus excludes many possible solutions which could result in much smaller consecutive delays. If rerouting were to be added, it would require a substantial change of the conflict resolution model. The second limitation originates from how the conflict resolution model deals with train speed. Since the model discretizes speed into two levels, scheduled speed and maximum speed, the speed is often estimated larger than required for the desired running times. As a result, braking distances and clearing times are estimated too large most of the time. This causes the minimum headways to be larger than required as well. The full capacity potential of moving-block signalling is thus not achieved. Multiple speed levels could potentially be added to the model however. The final limitation of the conflict resolution model mentioned here is that the length of trains are not taken into account except for determining the clearing time. This could potentially lead to problems at timing points located closer together than the length of a train. If a train has to hold in front of a timing point, but the rear of the train is still occupying another timing point, the model does not recognize this. Other trains scheduled to pass the timing point occupied by the rear of the first trains are allowed to do so by the model, even if in reality the first train is still occupying it. A similar problem could occur if a queue of trains appears between two timing points located further apart. If the leading train of this queue is being held in front of a timing point for a long time, the distance between this and the previous timing point could be filled up entirely by other trains. This is also not being considered by the model. This limitation could be resolved by adding an additional constraint to the model.

The coordination algorithm forming the core of the distributed model also comes with a fair share of limitations. Firstly, it solves sub-problems in a sub-optimal order. For example, the first area it solves is that of the first initial delay based on the order in which the initial delays are given in the input. In a situation like visualized in Figure 6.7, the most optimal order would be to start solving the initial delay applied to area 4. This would result in the model needing only 4 CR runs, while solving it in the order of the input would require 10 runs, as shown in Figure 6.7. A smarter way of determining what area to start with would be tracing the number of remaining timing points of the initially delayed train and choosing the one with the most timing points remaining in other areas. This way it would be more likely that there is delay spilling over to other areas with initial delays, reducing the probability any areas need to be resolved. Likewise, the order in which constraints are removed from the model in case an infeasibility is found is sub-optimal and similarly based on removing the first constraint included in the IIS. Moreover, the inclusion of the IIS itself is potentially a limitation. As found in section 6.3, the IIS is most likely the reason certain distributed model runs take considerably longer to compute. A self-developed method to remove constraints could make the model faster.

7.2.2. Limitations of the case study

The main limitation of testing the models on only a single case study, is that there is a fair chance the model results are biased towards that particular case study. It can not be said with certainty that similar results would be found for a network with a different topology, size or schedule. Comparing the results of chapter 6 with those of chapter 5, already show that

a problem with a smaller size completely changes the relative results of the centralized and distributed models.

Considering the data used for the case study network, some other limitations arise. Firstly, not all trains occurring in the full timetable are included. Most importantly, freight trains are excluded. This makes the model easier to solve. Secondly, there is barely any difference between the speed of the two discretized speed levels. The speed, and with that also the braking distances and clearing times, are the same across both levels for almost all timing points. Although the running times do differ, the lack of differences in braking distances and clearing times result in the moving-block aspect of the conflict resolution model not being fully utilized. Because the timetable used was designed for a fixed-block system, this does not directly lead to additional conflicts. The headways realized would namely still be smaller than those scheduled for fixed-block. The effect of the lack of differences in speed levels on computation times for the two models is therefore expected to be relatively small. The cumulative station delays are potentially a little higher, as there is less room for trains to catch up on delays.

The final limitation of the case study is the lack of interaction between trains in the chosen time window and trains in the previous and subsequent periods. Realistically, a train delayed at the end of the time window influences subsequent trains outside the window and potentially causes consecutive delays for them. These trains are however assumed to not exist for the case study. The amount of registered cumulative station delay caused by trains towards the edges of the time window is therefore often incomplete and thus too small. Similarly, there is a lack of interaction between trains in the considered network and trains outside of the border.

As a side-note to the limitations mentioned above, a mistake in the programming of the conflict resolution model was found near the completion of this research. This caused an ordering variable to be created for every combination of train pairs and timing points. Many redundant variables with for example a pair of trains for a timing point where one of the trains never uses that timing point were created as a result of this. This could have had a significant impact on the computation time of the model.

7.2.3. Results in relation to other research

To place the results of this research into perspective with other research, it is compared with the conclusions of some closely related papers found in the literature review. Most of interest is research which performed a comparison between a centralized and a distributed model. These are: Corman et al. (2010), Corman et al. (2014), Luan et al. (2020) and Cavone et al. (2022). They all compared a fixed-block centralized model with a fixed-block distributed model.

The results found for the case study are mostly in line with other research on distributed models. They all concluded that the distributed model is faster. Corman et al. (2010) also explicitly mentions the size of the average delay for both models. Their results showed that for some of the tested scenarios the centralized model yielded smaller average delays, and that for other scenarios the distributed model yielded smaller delays. This is also in line with the results found in this research, where sometimes the distributed model has smaller resulting cumulative station delays, and sometimes the centralized model.

7.3. Recommendations

Based on the findings from the discussion and several observations throughout this research, several recommendations for future research can be made. The most pressing recommenda-

tion is to apply the models to more case studies with different sizes, topologies and schedules. Interesting would be to find the size of the network for which the computation time of the distributed and centralized models are about equal. A network topology the distributed model has not been tested for is the orientation of three areas in a triangle. It would be interesting to see how well the model would work here. More dense schedules could be tested to fully test the capability of the moving-block aspect of the conflict resolution model. This could then be compared to a distributed fixed-block model. It can also be observed whether the number of infeasibilities for which the model fails increases when applying a denser schedule.

Another recommendation would be to try to find a way to avoid runs failing. One suggestion to try to achieve this is to remove the computation of the IIS and to replace it with another system that determines which constraints should be removed. On the topic of the IIS, another recommendation for future research would be to investigate why there is a relatively large difference in computation time for calculating it for different infeasibilities. For some infeasibilities the IIS is calculated quickly, for others it can take a significant amount of time.

In regard to the conflict resolution model, the following recommendations can be made. Firstly, it could be extended to also deal with disruptions. For this the lacking traffic management measures would need to be added to the model. This does however require a complete reformulation of the model. Secondly, the length of trains can be taken into account in the model to avoid problems at timing points located closer together than the length of a train.

A final recommendation, or more of a suggestion, would be to modify the model to work with virtual coupling. Virtual coupling allows trains to run even closer together than moving-block. It would be interesting to see how a conflict resolution model performs with a dense schedule while running trains even closer together.

Bibliography

Bretas, A. M. C., Mendes, A., Jackson, M., Clement, R., Sanhueza, C., & Chalup, S. (2023). A decentralised multi-agent system for rail freight traffic management. *Annals of Operations Research*, *320*(2), 631–661. https://doi.org/10.1007/S10479-021-04178-X/FIGURES/9

Caimi, G., Fuchsberger, M., Laumanns, M., & Lüthi, M. (2012). A model predictive control approach for discrete-time rescheduling in complex central railway station areas. *Computers & Operations Research*, *39*(11), 2578–2593. https://doi.org/10.1016/J.COR.2012.01.003

Cavone, G., Van Den Boom, T., Blenkers, L., Dotoli, M., Seatzu, C., & De Schutter, B. (2022). An MPC-Based Rescheduling Algorithm for Disruptions and Disturbances in Large-Scale Railway Networks. *IEEE Transactions on Automation Science and Engineering*, *19*(1), 99–112. https://doi.org/10.1109/TASE.2020.3040940

Chebaro, S. (2020, 30-3-2023). New European mobility strategy aims to triple high-speed rail traffic. https://www.railtech.com/policy/2020/12/10/new-european-mobility-strategy-aims-to-triple-high-speed-rail-traffic/#:~:text=The%5C%20European%5C%20Commission%5C%20aims%5C%20to,have%5C%20doubled%5C%20by%5C%20that%5C%20time.

Corman, F., D'Ariano, A., Pacciarelli, D., & Pranzo, M. (2014). Dispatching and coordination in multi-area railway traffic management. *Computers & Operations Research*, *44*, 146–160. https://doi.org/10.1016/J.COR.2013.11.011

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. https://doi.org/10.1016/J.TRC.2015.01.014

Corman, F., D'Ariano, A., Hansen, I. A., & Pacciarelli, D. (2011). Optimal multi-class rescheduling of railway traffic. *Journal of Rail Transport Planning & Management*, *1*(1), 14–24. https://doi.org/10.1016/J.JRTPM.2011.06.001

Corman, F., D'Ariano, A., Pacciarelli, D., & Pranzo, M. (2010). Centralized versus distributed systems to reschedule trains in two dispatching areas. *Public Transport*, *2*(3), 219–247. https://doi.org/10.1007/S12469-010-0032-7/METRICS

Corman, F., D'Ariano, A., Pacciarelli, D., & Pranzo, M. (2012). Optimal inter-area coordination of train rescheduling decisions. *Transportation Research Part E: Logistics and Transportation Review*, *48*(1), 71–88. https://doi.org/10.1016/j.tre.2011.05.002

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. https://doi.org/10.1287/TRSC.1080.0247

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. https://doi.org/10.1109/TITS.2006.888605

Diaz de Rivera, A., Tyler Dick, C., & Evans, L. E. (2020). Potential for Moving Blocks and Train Fleets to Enable Faster Train Meets on Single-Track Rail Corridors. *Journal of Transportation Engineering, Part A: Systems*, *146*(8), 1–13. https://doi.org/10.1061/JTEPBS.0000403

Dündar, S., & Sahin, I. (2013). Train re-scheduling with genetic algorithms and artificial neural networks for single-track railways. *Transportation Research Part C: Emerging Technologies*, *27*, 1.

European Commission. (n.d.). Rail. https://transport.ec.europa.eu/transport-modes/rail_en

European Commission. (2020). Urbanisation in Europe. https://knowledge4policy.ec.europa.eu/foresight/topic/continuing-urbanisation/urbanisation-europe_en

Eurostat. (2023). Population projections in the EU. https://ec.europa.eu/eurostat/statistics-explained/index.php?oldid=497115#Population_projections

Fay, A. (2000). Decentralized Railway Control Based on Autonomous Agents. *IFAC Proceedings Volumes*, *33*(9), 83–88. https://doi.org/10.1016/S1474-6670(17)38128-4

Giannettoni, M., & Savio, S. (2002). Traffic Management In Moving Block Railway Systems: The Results Of The EU Project COMBINE. *WIT Transactions on The Built Environment*, *61*, 953–962. https://doi.org/10.2495/CR020941

Giannettoni, M., & Savio, S. (2004). The European Project COMBINE 2 To Improve Knowledge On Future Rail Traffic Management Systems. *WIT Transactions on The Built Environment*, *74*, 695–704. https://doi.org/10.2495/CR040701

Giuliari, C. M., Pellegrini, F., & Savio, S. (2000). Moving block and traffic management in railway applications: the EC project. *WIT Transactions on The Built Environment*, *50*, 11–20. https://doi.org/10.2495/CR000011

Goverde, R., 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 and Management*, *3*, 78–94. https://doi.org/10.1016/j.jrtpm.2013.12.001

Gurobi Optimization. (n.d.). Model.computeIIS(). https://www.gurobi.com/documentation/current/refman/py_model_computeiis.html

Hornung, G. (2023). *Hourly Traffic Density Based Decomposition in Non-Centralised Railway Traffic Conflict Resolution*. http://resolver.tudelft.nl/uuid:f9b31c5a-2977-4df9-8c50-0a4c40631257

Khadem Sameni, M., & Landex, A. (2013). Capacity Utilization in European Railways: Who Is the Fairest of Them All? *Proceedings of the Transportation Research Board (TRB) 92nd Annual Meeting Transportation Research Board*.

Lamorgese, L., & Mannino, C. (2015). An Exact Decomposition Approach for the Real-Time Train Dispatching Problem. *Operations Research*, *63*(1), 48–64. https://doi.org/10.1287/opre.2014.1327

Leutwiler, F., & Corman, F. (2023). A review of principles and methods to decompose large-scale railway scheduling problems. *EURO Journal on Transportation and Logistics*, *12*, 1–17. https://doi.org/10.1016/J.EJTL.2023.100107

Liu, J., Chen, L., Roberts, C., Li, Z., & Wen, T. (2018). A Multi-agent Based Approach for Railway Traffic Management Problems; A Multi-agent Based Approach for Railway Traffic Management Problems. *2018 International Conference on Intelligent Rail Transportation (ICIRT)*. https://doi.org/10.1109/ICIRT.2018.8641621

Luan, X., De Schutter, B., Meng, L., & Corman, F. (2020). Decomposition and distributed optimization of real-time traffic management for large-scale railway networks. *Transportation Research Part B: Methodological*, *141*, 72–97. https://doi.org/10.1016/J.TRB.2020.09.004

Marcelli, E., & Pellegrini, P. (2021). Literature Review Toward Decentralized Railway Traffic Management. *IEEE Intelligent Transportation Systems Magazine*, *13*(3), 234–252. https://doi.org/10.1109/mits.2020.2970180

Mascis, A., & Pacciarelli, D. (2002). Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, *143*(3), 498–517. https://doi.org/10.1016/S0377-2217(01)00338-1

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. https://doi.org/10.1016/J.TRB.2006.02.005

Pearson, K. (1895). Note on Regression and Inheritance in the Case of Two Parents. *Proceedings of the Royal Society of London*, *58*, 240–242.

Pellegrini, P., Marlière, G., Pesenti, R., & Rodriguez, J. (2015). RECIFE-MILP: An Effective MILP-Based Heuristic for the Real-Time Railway Traffic Management Problem. *IEEE Transactions on Intelligent Transportation Systems*, *16*(5), 2609–2619. https://doi.org/10.1109/tits.2015.2414294

Perrachon, Q., Chevrier, R., & Pellegrini, P. (2020). EXPERIMENTAL STUDY ON THE VIABILITY OF DECENTRALIZED RAILWAY TRAFFIC MANAGEMENT. *WIT Transactions on The Built Environment*, *199*, 337–344. https://doi.org/10.2495/CR200311

ProRail. (2021a). *Integrale Mobiliteitsanalyse 2021 Deelrapportage Spoor en BTM* (Report). https://zoek.officielebekendmakingen.nl/blg-990633.pdf

ProRail. (2021b). Via Data verkent nieuwe toepassingen voor het spoor. https://www.prorail.nl/nieuws/via-data-verkent-nieuwe-toepassingen-voor-het-spoor

Quaglietta, E., Versluis, N., Goverde, R., Pellegrini, P., Nardone, R., Vittorini, V., Mazini, A., Garcia, M., & Usman Sanwal, M. (2022). *Real-Time Traffic Rescheduling Algorithms for Perturbation Management and Hazard Prevention in Moving-Block Operations* (Report). Shift2Rail. https://www.performingrail.com/_files/ugd/aca342_7dc5557aee7f4f7d93dc4a7d80531e9e.pdf

Rail Freight Forward. (n.d.). *30 by 2030 Rail Freight Strategy to boost modal shift* (Report). Rail Freight Forward. https://www.railfreightforward.eu/sites/default/files/downloadcenter/whitepaperldupdated.pdf

Rodriguez, J. (2007). A constraint programming model for real-time train scheduling at junctions. *Transportation Research Part B: Methodological*, *41*(2), 231–245. https://doi.org/10.1016/J.TRB.2006.02.006

Rotoli, F., Navajas Cawood, E., & Soria, A. (2016). *Capacity assessment of railway infrastructure – tools, methodologies and policy relevance in the eu context*. Publications Office of the European Union. https://doi.org/10.2791/037759

Shang, F., Zhan, J., & Chen, Y. (2018). Distributed Model Predictive Control for Train Regulation in Urban Metro Transportation. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. https://doi.org/10.1109/ITSC.2018.8569630

Törnquist, J., & Persson, J. A. (2007). N-tracked railway traffic re-scheduling during disturbances. *Transportation Research Part B: Methodological*, *41*(3), 342–362. https://doi.org/10.1016/J.TRB.2006.06.001

Versluis, N. D., Pellegrini, P., Quaglietta, E., Goverde, R. M. P., & Rodriguez, J. (2023). An Approximate Conflict Detection and Resolution Model for Moving-Block Signalling by Enhancing RECIFE-MILP. *RailBelgrade 2023*.

Wang, P., Ma, L., Goverde, R. M. P., & Wang, Q. (2016). Rescheduling Trains Using Petri Nets and Heuristic Search; Rescheduling Trains Using Petri Nets and Heuristic Search. *IEEE transactions on Intelligent Transportation Systems*, *17*(3), 726–735. https://doi.org/10.1109/TITS.2015.2481091

Yi, X., Marlière, G., Pellegrini, P., Rodriguez, J., & Pesenti, R. (2023). Coordinated train rerouting and rescheduling in large infrastructures. *RailBelgrade 2023*.

Zhan, S., Kroon, L. G., Zhao, J., & Peng, Q. (2016). A rolling horizon approach to the high speed train rescheduling problem in case of a partial segment blockage. *Transportation Research Part E: Logistics and Transportation Review*, *95*, 32–61. https://doi.org/10.1016/J.TRE.2016.07.015

Zhu, Y., & Goverde, R. M. P. (2019). Railway timetable rescheduling with flexible stopping and flexible short-turning during disruptions. *Transportation Research Part B: Methodological*, *123*, 149–181. https://doi.org/10.1016/j.trb.2019.02.015

A

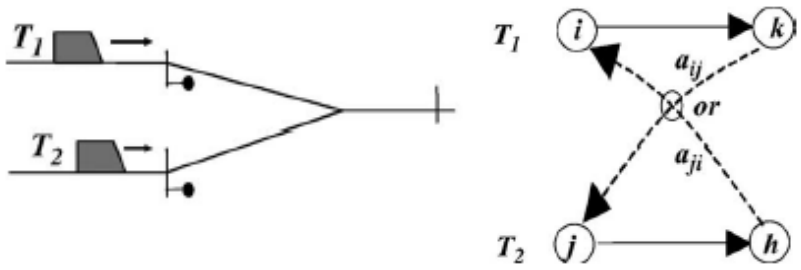
Example of a centralized fixed-block CR model using an alternative graph formulation

In this appendix an example of a centralized fixed-block CR model using an alternative graph formulation is given. It is taken from the work of D'Ariano et al. (2007).

An alternative graph consists of a set of nodes N, a set of directed arcs F and a set of pairs of alternative arcs A. A node represents one of n operations o_i with $i \in N$. Two special nodes are o_0 and o_n. These are dummy 'operations' with no processing time representing the start and finish of the set of operations. It is assumed o_0 precedes and o_n follows all other operations of all trains. An arc, denoted by two nodes as (i, k), represents a precedence relation. In this case, operation $i \in N$ comes before operation $k \in N$. p_i is the processing time of operation i. If t_i is the starting time of operation o_i, the starting time of operation o_k is given by $t_k \geq t_i + p_i$. The arcs in the set of directed arcs F are fixed. In other words, they always occur. For example on a stretch of track consisting of two blocks and no intersections, a train traversing the first block will inevitably traverse the second block next. Once two jobs require the same machine, in the case of rail once two trains require the same block, a potential conflict arises which requires a processing order to be determined. For this a pair of alternative arcs $((k, j), (h, i))$ contained in set A is used. From this pair exactly one arc should be chosen. The headway arc between two operations is denoted by $a_{j,i}$ or $a_{k,h}$. This arc is only part of the total headway time between two operations. The other part being the running time over the block. This results in a second constraint for the starting time of operation o_i: $t_i \geq t_h + a_{j,i}$. The selected set of alternative arcs from set A is set S. The objective is to minimize the starting time of dummy operation o_n. As this is the last 'operation', minimizing its starting time consequently means minimizing the starting time of all preceding operations and thus the total delay. Below the resulting mathematical model is given.

$$
\begin{array}{lll}
\min & t_n - t_0 & \\
\text{s.t.} & t_j - t_i \geq p_i & \forall (i,j) \in F \\
 & (t_j - t_i \geq a_{ij}) \vee (t_h - t_k \geq a_{hk}) & \forall ((i,j),(h,k)) \in A
\end{array}
$$

A visualisation of the alternative graph method is given in Figure A.1. The alternative arcs are represented by dotted lines, the fixed arcs by solid lines and a pair of alternative arcs is indicated by a small circle connecting the arcs. If train T_1 gets priority, alternative arc (k, j) needs to be selected.



(a) (b)

Figure A.1: Pair of alternative arcs for a conflict at an intersection (D'Ariano et al., 2007)

B

Example of a centralized fixed-block CR model using a MILP formulation

In this appendix an example of a centralized fixed-block CR model using a MILP formulation is given. It is a slightly simplified version taken from the work of Törnquist and Persson (2007). The model has the same objective as the example given for the alternative graph model: minimizing the sum of delays at the end of each train service (also known as final or exit delays).

This model uses a simplistic network structure where the network is divided in so-called track segments. Each track segment consists of n parallel blocks. At the border of each segment there are switches connecting all blocks in one segment to all blocks in the other (if there is more than one block in a segment). A running train thus moves from a block in one segment to a block in the next and not between multiple blocks within the same segment.

Sets
Set T contains all trains, set B all track segments and set E all events. An event is the occupation of a block by a train. K_i, a subset of E, is the set of events ordered according to the timetable for train i. Similarly, subset L_j contains the events of track section j ordered according to the timetable. Finally, set P_j contains all parallel blocks in track segment j.

Parameters
The minimum occupation time of a train for event k is denoted as d_k. This is the time the train belonging to the event needs to traverse the block belonging to the event. If the event contains a station, it also includes the minimum dwell time. The scheduled start and end time of an event k according to the timetable are $b_k^{initial}$ and $e_k^{initial}$ respectively. The minimum separation time, the time required between a train leaving and a subsequent train entering track segment j, is denoted as Δ_j^F for two trains following each other. Similarly, the minimum separation time between two trains meeting each other for section j is Δ_j^M. Note that this model assumes separation times as a constant. In practice they would be different for every train pair depending on speeds and train specific characteristics. The length of a block t in segment j is denoted as g_{jt}^{track}. The length of train i is denoted as g_i^{train}. If a stop is planned at a station in event k, the binary parameter h_k has value 1. Its value is 0 otherwise. Parameter o_k indicates the point of origin of event k. n_i and m_j denote the last events in subsets K_i and L_j respectively. $k < \hat{k}$ is used to indicate that $\hat{k}$ is any event following event k in the original timetable. M is a significantly large constant.

Variables

A total of six decision variables is used in this model, three continuous and three binary. The continuous variables are the following: x_k^{begin} contains the start time of an event k, x_k^{end} contains the end time and z_k contains the delay of the start of an event. The binary variables are q_{kt}, $\gamma_{k\hat{k}}$ and $\lambda_{k\hat{k}}$. q_{kt} equals 1 if event k uses track t. $\gamma_{k\hat{k}}$ equals 1 if event k occurs before event $\hat{k}$. Contrarily, $\lambda_{k\hat{k}}$ equals 1 if event k occurs after event $\hat{k}$, and is thus rescheduled.

Objective and constraints

$$\min \quad \sum_{i \in T} z_{n_i}$$

s.t.

$$x_k^{end} = x_{k+1}^{begin} \qquad \forall k \in K_i, \forall i \in T : k \neq n_i, \tag{B.1}$$

$$x_k^{end} \geq x_k^{begin} + d_k \qquad \forall k \in E, \tag{B.2}$$

$$x_k^{begin} \geq b_k^{initial} \qquad \forall k \in E : h_k = 1, \tag{B.3}$$

$$x_k^{end} - e_k^{initial} \leq z_k \qquad \forall k \in E, \tag{B.4}$$

$$\sum_{t \in P_j} q_{kt} = 1 \qquad \forall k \in L_j, \forall j \in B, \tag{B.5}$$

$$q_{kt} + q_{\hat{k}t} - 1 \leq \lambda_{k\hat{k}} + \gamma_{k\hat{k}} \qquad \forall k, \hat{k} \in L_j, \forall t \in P_j, \forall j \in B : k < \hat{k}, \tag{B.6}$$

$$x_{\hat{k}}^{begin} - x_k^{end} \geq \Delta_j^M \gamma_{k\hat{k}} - M(1 - \gamma_{k\hat{k}}) \qquad \forall k, \hat{k} \in L_j, \forall j \in B : k < \hat{k}, o_{\hat{k}} \neq o_k, \tag{B.7}$$

$$x_{\hat{k}}^{begin} - x_k^{end} \geq \Delta_j^F \gamma_{k\hat{k}} - M(1 - \gamma_{k\hat{k}}) \qquad \forall k, \hat{k} \in L_j, \forall j \in B : k < \hat{k}, o_{\hat{k}} = o_k, \tag{B.8}$$

$$x_k^{begin} - x_{\hat{k}}^{end} \geq \Delta_j^M \lambda_{k\hat{k}} - M(1 - \lambda_{k\hat{k}}) \qquad \forall k, \hat{k} \in L_j, \forall j \in B : k < \hat{k}, o_{\hat{k}} \neq o_k, \tag{B.9}$$

$$x_k^{begin} - x_{\hat{k}}^{end} \geq \Delta_j^F \lambda_{k\hat{k}} - M(1 - \lambda_{k\hat{k}}) \qquad \forall k, \hat{k} \in L_j, \forall j \in B : k < \hat{k}, o_{\hat{k}} = o_k, \tag{B.10}$$

$$\lambda_{k\hat{k}} + \gamma_{k\hat{k}} \leq 1 \qquad \forall k, \hat{k} \in L_j, \forall j \in B : k < \hat{k}, \tag{B.11}$$

$$g_i^{train} q_{kt} \leq g_{jt}^{track} \qquad \forall k \in (K_i \cap L_j), \forall t \in P_j, \forall j \in B, \forall i \in T \tag{B.12}$$

$$x_k^{begin}, x_k^{end}, z_k \geq 0 \qquad \forall k \in E. \tag{B.13}$$

From the decision variables used in this model, the available rescheduling measures are easily identifiable. Retiming is regulated through changes in the x_k^{begin} and x_k^{end} variables, reordering through the binary $\lambda_{k\hat{k}}$ variable and local rerouting is done through changing the q_{kt} variable. Note that the model assumes a train can change tracks between all track segments, i.e., there are switches everywhere. For a detailed explanation of the constraints, the reader is referred to the source paper by Törnquist and Persson (2007). In this paper a second possible objective function is formulated. Instead of optimizing for minimal delay, this second objective function aims at optimizing punctuality. This was achieved by translating delay into a cost and introducing an additional penalty for trains delayed more than a pre-specified amount of minutes. This only required a change of the objective function and one additional constraint and variable. Furthermore, in the paper additional optional constraints are formulated to regulate the model outcome. For example, introducing a maximum number of order swaps at specific track segments. This relatively easy mutability proofs the flexibility of the MILP formulation.

C

Type of switch timing points

There are five different scenarios for two subsequent trains passing a switch. In Figure C.1 a visualisation of the different types is given. Firstly, trains could be following each other. In this situation the origins (the previous timing point) and destinations (the next timing point) of the two trains are the same. When this is the case, the two trains could pass the timing point with a minimum headway of the braking distance of the following train based on its speed when passing the timing point. Since the model is based on passage times to describe the train operation, the braking distance is converted to a braking time by dividing it by the train's velocity at the timing point. Switches do not need to change direction so no set-up time is needed.

Secondly, there is the case of two trains merging onto the same track, diverging from the same track to different tracks or two trains crossing paths. In these situations the minimum headway of the trains should be at least the braking time of the second train based on its speed at the moment the switch has changed direction and is clear to pass plus the set-up time of the switch. The set-up time of the switch needs to be added to the braking time because while the switch is changing positions the train approaching it needs to be able to stop in case the switch happens to fail. For these first two situations the components the headway between the two trains consists of stay the same when their order is reversed. The only difference in the size of the headway after the order is reversed would be due to a different braking time of the new following train.

This is different in the third and fourth situation. In the third situation there is a train diverging from a shared bidirectional track before another train merges onto that track at this timing point. In the fourth situation the diverging happens after another train merged onto the shared bidirectional track. The fourth situation is the reversal of the third situation and vice versa. If a timing point of the third type occurs, a timing point of the fourth type always occurs too at the end of the section of shared bidirectional track between them. For a timing point of the third type the minimum headway between the two trains is the same as timing points of the second type. For a timing point of the fourth type the minimum time between the two trains at this point now needs to be at least the minimum running time of the first train from this timing point to the next timing point of the third type plus the minimum running time of the second train from this third type timing point to the timing point in consideration plus the headway needed between the trains at the timing point of the third type.

Between timing points of the third and fourth type there could be other switches. These switches are the fifth and final type of timing points. Since the two trains moving past these

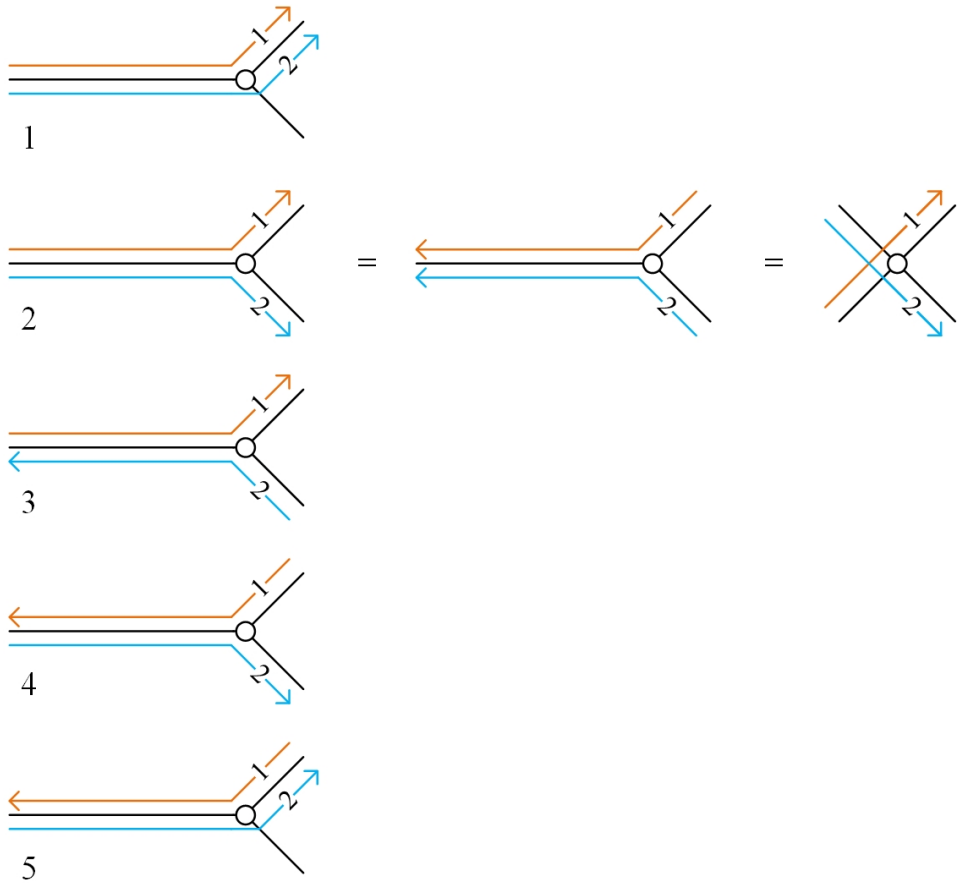


Figure C.1: Timing point types (1: following, 2: both diverging *or* both merging *or* crossing, 3: diverging before merging 4: diverging after merging, 5: meeting)

timing points are always between a third and fourth type timing point, they meet head on. At this type of timing points no reversal of order is possible. Hence, the order of trains should be the same as at the corresponding third and fourth type timing points.

A pair of third and fourth type timing points with any number of fifth type timing points essentially form a single block for trains in opposite directions. When any number of trains enters at one side, no train can enter from the other side until all other trains have left it. The order of trains passing through the shared section should be the same at both ends of the section. As a result, this means that only one ordering decision for the whole section needs to be made. This happens at the timing point of the third type, i.e., the point where the first train leaves the shared bidirectional section. For the other points in the section the train order as decided at the third type point is fixed. The train passage times at these points follow from the first constraint.

D

Pseudocode of the coordination algorithm

```
function Coordination(InitialDelays)
    determine areas with initial delays as InitialDelayAreas
    set ProblemSolved to 0
    set Solution to empty list
    set AreaQueue to empty list
    set AreaQueueRedo to empty list
    set AreasSolved to empty list
    set RemovedConstraints to empty list
    set CurrentArea to InitialDelayAreas[0]
    set CurrentSolution to ConflictResolution(CurrentArea)
    set FirstTimeInLoop to true
    while ProblemSolved = 0 do
        if CurrentSolution is feasible then
            add CurrentArea to AreasSolved
            add CurrentSolution to Solution
            determine unsolved areas bordering CurrentArea with delay at the border result-
            ing from CurrentSolution as UnsolvedBorderingAreasWithDelay
            if length(UnsolvedBorderingAreasWithDelay) > 0 then
                add UnsolvedBorderingAreasWithDelay to AreaQueue      * to back of queue
        else
            determine constraints in CurrentSolution.IIS which are part part of the internal
            border as PotentialRemovableConstraints
            if length(PotentialRemovableConstraints) > 0 then
                set i to 0
                while PotentialRemovableConstraints[i] in RemovedConstraints and i <
                length(PotentialRemovableConstraints) - 1 do
                    set i to i + 1
                if PotentialRemovableConstraints[i] not in RemovedConstraints then
                    add PotentialRemovableConstraints[i] to RemovedConstraints
                    remove PotentialRemovableConstraints[i] from CurrentArea
                else
                    if FirstTimeInLoop = true then
                        set FirstTimeInLoop to false
```

```
                    set i to length(RemovedConstraints) - 1
                    while RemovedConstraints[i] not in PotentialRemovableConstraints
                    do
                        set i to i - 1
                    remove RemovedConstraints[i] from CurrentArea
                else
                    set ProlemSolved to false
                    return “Problem is infeasible: stuck in loop”
            set CurrentSolution to ConflictResolution(CurrentArea)
            if CurrentSolution is feasible then
                add CurrentArea to AreasSolved
                add CurrentSolution to Solution
                determine unsolved areas bordering CurrentArea with delay at the border
                resulting from CurrentSolution as UnsolvedBorderingAreasWithDelay
                determine previously solved areas bordering CurrentArea with different
                delay at the border resulting from CurrentSolution as SolvedBorderingAr-
                easWithDelay
                if length(UnsolvedBorderingAreasWithDelay) > 0 then
                    add UnsolvedBorderingAreasWithDelay to AreaQueue * to back of
                    queue
                if length(SolvedBorderingAreasWithDelay) > 0 then
                    add SolvedBorderingAreasWithDelay to AreaQueueRedo
                    remove solutions belonging to the areas of SolvedBorderingAr-
                    easWithDelay from Solution
                    remove areas of SolvedBorderingAreasWithDelay from AreasSolved
                while length(AreaQueueRedo) > 0 and CurrentSolution.feasibility = fea-
                sible do
                    set CurrentArea to AreaQueueRedo[0]
                    set CurrentSolution to ConflictResolution(CurrentArea)
                    remove CurrentArea from AreaQueueRedo
                    if CurrentSolution is feasible then
                        add CurrentArea to AreasSolved
                        add CurrentSolution to Solution
                    else
                        set AreaQueueRedo to empty list
        else
            set ProblemSolved to false
            return “Problem is infeasible: no border constraints in IIS”
    if CurrentSolution is feasible then
        if length(AreaQueue) = 0 then
            for all area in InitialDelayAreas do
                if area not in AreasSolved and length(AreaQueue) = 0 then
                    add area to AreaQueue                          * to back of queue
        if length(AreaQueue) = 0 then
            set ProblemSolved to true
            return Solution
        else
            set CurrentArea to AreaQueue[0]
            set CurrentSolution to ConflictResolution(CurrentArea)
```

remove CurrentArea **from** AreaQueue
set FirstTimeInLoop **to true**

E

Analysis of a failed case study run

As can be observed by carefully comparing Table 6.6 and Table 6.7, 6 out of the 400 model runs of scenario 5 failed to successfully find a solution. In this appendix, an explanation for these failures is given.

All of the 6 failed runs failed in the distributed model. The centralized model was still able to find a solution for all these 6 runs. In the runs that failed, infeasibilities were found during the calculation of the distributed model. This shows that it is most likely that the issue is caused by the coordination algorithm while the conflict resolution works as intended. Closer inspection of the failed distributed models shows that they ended up in repeatedly trying to remove the same constraint from the IIS which was re-added to the model again by the conflict resolution of the bordering area. Below an example is given of one of the failed runs.

The initial delays of this failed run are as follows: Train 33 delayed by 494 seconds at timing point Ht2149, train 102 delayed by 883 seconds at timing point Ht2117, train 43(2) delayed by 448 seconds at timing point Gz4 and train 117 delayed by 435 seconds at timing point Ht2223. Below, all the steps of the coordination can be retraced.

1. Detect that first delay inserted is located in area Ht
2. Solve conflict resolution area Ht
3. Delay detected at the border between area Ht and area Tb: solve conflict resolution area Tb with already solved border variables as input
4. Delay detected at the border between area Tb and area Bd: solve conflict resolution area Bd with already solved border variables as input
5. Area Bd infeasible with current external constraints: compute IIS
6. Remove first constraint from IIS from model (arr. time train 43(2) at border point Tbu2 = 1800 seconds) and reattempt to solve area Bd
7. Attempt successful, resolve area Tb
8. Area Tb infeasible with current external constraints: compute IIS
9. Remove first constraint from IIS from model (arr. time train 43(2) at border point Tbu2 = 2390 seconds) and reattempt to solve area Tb

10. Attempt successful, resolve area Bd
11. Area Bd infeasible with current external constraints: compute IIS
12. No constraints in IIS not already removed before (only arr. time train 43(2) at border point Tbu2 = 1800 seconds)
13. Remove this constraint for the final time, resolve area Bd and then resolve area Tb
14. Error: No constraints in IIS not already removed before (only arr. time train 43(2) at border point Tbu2 = 2390 seconds): conclude stuck in loop

Like can be seen in this example, the IIS of both areas Bd and Tb only returned the constraint setting the arrival time of train 43(2) at border point Tbu2. This means the value for this variable for which a feasible solution exist is neither the value leading to an optimal solution in area Bd or Tb. Indeed, when looking at the solution found using the centralized model, a time between the two area solutions of 2348 seconds is found. Finding this value in one of the areas for the distributed model in its current form is not possible because the solution would not be optimal. Changing the constraints to greater or smaller than versions would also not help since the optimizer would then just return the same value mutated by the smallest possible amount. A possible solution would be to divert from exclusively removing constraints from the IIS, for example by developing one's own method to determine the most promising constraints to remove.