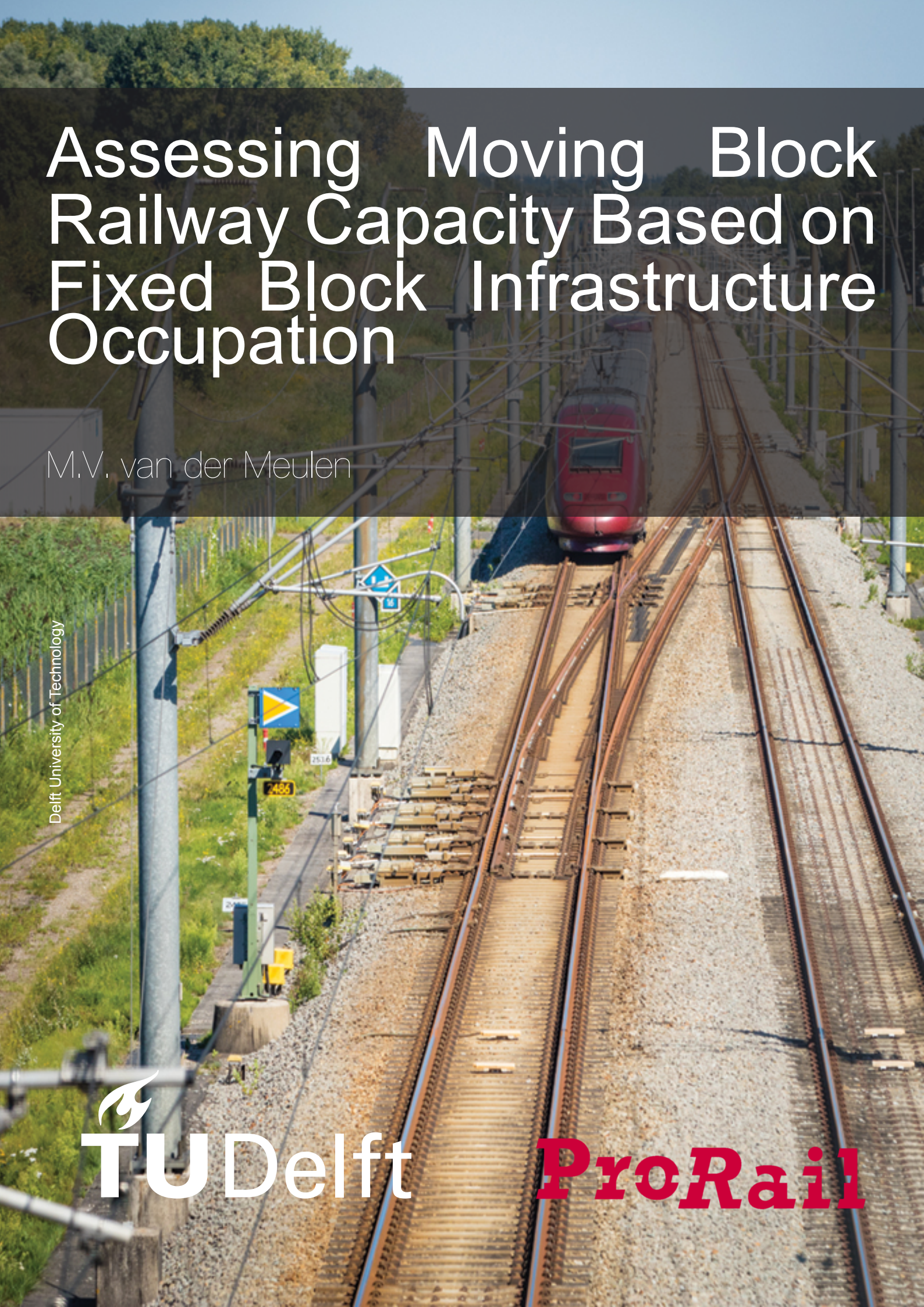
Assessing Moving Block Railway Capacity Based on Fixed Block Infrastructure Occupation

M.V. van der Meulen



Delft University of Technology

TU Delft

ProRail

Assessing Moving Block Railway Capacity Based on Fixed Block Infrastructure Occupation

by

M.V. van der Meulen

to obtain the degree of Master of Science in Civil Engineering
at the Delft University of Technology, department of Transport and Planning
to be defended publicly on March 11, 2022 at 16:00 PM.

Student number:	4218337	
Project duration:	March 12, 2021 – March 11, 2022	
Thesis committee:	Prof. dr. R.M.P. Goverde	TU Delft (chair)
	Dr. ir. E. Quaglietta	TU Delft (daily supervisor)
	Dr. ir. P.K. Krishnakumari	TU Delft (external supervisor)
	A.D. Middelkoop	ProRail (daily supervisor)

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Cover photo: A Thalys train on the HSL-Zuid nearby Hazeldonk, made by Stefan Verkerk (2020), retrieved from ProRail Beeldbank.

TU Delft Delft University of Technology

ProRail

Preface

During my final year at high school, I was thinking about what to study after my high school graduation. Doubting between Architecture and Building Engineering, and because of my lack of feeling with the English language also between a research university and university of applied sciences , I decided to start a bachelor Civil Engineering at the TU Delft. It was a good choice!

During my bachelor I discovered the beauties and challenges of transportation engineering, especially of Public Transport. I got interested in public transport, because of its importance for society: it keeps cities accessible, it assures movability for certain groups in society who don't have access to car and/or bike and it is better for our environment. Besides, there is always room to make it more efficient.
I would like to thank Rob Goverde and Egidio Quaglietta for introducing me into the world of railways. I enjoyed your lectures about all kind of topics within railway operations. Also, thanks for supervising me during my graduation project. Due to Covid-19 most of our meetings were online, but still it was possible to discuss my progress, helping me with the model definition and guiding me in the right direction.

Special thanks to Dick Middelkoop from ProRail. Unfortunate we couldn't work together in the Inktpot due to the restrictions, but I would like to thank you for your time and sharing your knowledge about railway operations in the Netherlands during our online meetings. I liked our discussions about my knowledge from the lectures at the TU Delft and your expertise. It was nice to gain experience with the simulation software that is being used at ProRail. I would also like to express my gratitude to the rest of the I&TV team from ProRail. Thanks for welcoming me, helping me and the fun quarterly meetings.

My time as a student was what life is: it has it's ups and downs. I had an amazing time at my student association. There I had the opportunity to grow as a person and develop myself. Also I did make some great friends. Thanks for all the breaks in the library between studying. I appreciated it! But struggling with courses and family circumstances didn't make life easy. Thanks to my Father in heaven for His blessings and taking care of me.

My master's study Transport & Planning is finalized by this thesis. Fully in English, which still surprises me a little bit. Although, it wouldn't surprise me if you would spot some grammatical errors, so please forgive me for that. I'm looking forward to start my career in the railway industry.

I wish you a pleasant reading.

Martijn van der Meulen
Delft, January 2022

Summary

Research problem
Over the past decade the number of travelled kilometres by train has increased significantly in the Netherlands. This has also a consequence: the Dutch railway network is getting full. One of the solutions for this might be a new signalling system. ETCS L3 Moving Block, one of the signalling types of ERTMS, promises an increase of capacity. With a dynamic speed profile (just as ETCS L2) and without fixed blocks, the occupation times will be shorter compared to NS'54/ATB-EG. However, the capacity gains may differ between certain locations.

To assess rail capacity, the timetable compression method is arguably the most widely used method. An issue with this method is defining boundaries of line sections. It is up to the infrastructure manager to interpret the guidelines presented by the UIC, but a misinterpretation will lead to different results. Therefore there is need for an alternative method to identify capacity bottlenecks, for example a method which could make use of track occupation data of a moving block system. However, ETCS L3 Moving Block is still in development, so gathering data out of daily operations is not possible. Also gathering moving block data out of simulation isn't always a convenient solution. In simulation software FRISO, used at ProRail, ETCS L3 Moving Block is not (yet) implemented. So to assess capacity gains of moving block, first a model is needed which estimates the blocking times of trains operating under moving block signalling.

This research aims to define a novel method to assess capacity of moving block. It uses trajectory data from a fixed block system to estimate the track occupation of a future moving block system. Therefore, the following main research question will be answered:

How can capacity gains of Moving Block be assessed with data for both railway corridors and complex nodes?

Methodology
To assess the capacity gains of moving block, a model has been defined to estimate blocking times for moving block. As input it uses infrastructure characteristics, rolling stock parameters and planned time-distance data of trains operating under fixed block signalling. The blocking time for fixed- and moving block can be divided into six time components: route setup-, sight and reaction-, approach-, running-, clearing- and release time. The length of the route setup time and release time depend on system characteristics and safety margins. In the model presented in this research these are fixed values. The sight and reaction time mainly depend on the reaction time and the level of attention of the train driver. Also this time component is modelled as a fixed value.

The approach time is defined as the time the train runs through the absolute braking distance. The approach time can be estimated with the braking characteristics of the train, gradient of the tracks and trajectory data of the train. With the braking characteristics and the gradient of the track the possible braking rate of a train can be determined. From the point of interest (EoA) a braking curve will be constructed backwards in time and distance per timestep Δt. Once the braking curve has the same speed as the free flow speed of the trajectory data, the indication point is found. The time it takes for the train to run from the indication point to the EoA will be the approach time.

For the running time there are three different situations. First, on a normal track line, the running time is equal to the time period between two requests for movement authority (MA). Second, when a virtual block is applied, it is the time it takes to run through that block. Third, for a stop at a station, the running time is equal to the dwelling time. The clearing time is the time it takes a train has fully passed a location. Then, the total length of the blocking time is the sum of all six time components. The start time is the passage time minus the approach-, reaction- and setup time. The end time is the passage time plus the running-, clearing- and release time. Once the blocking times are known, the buffer times can be calculated by subtracting the end of blocking time of the leading train from the start of the blocking time of the following train.

Two methods are described to identify bottlenecks. A first approach is to sum the blocking times of all trains per block during a time period. The block with the highest summed blocking time indicates a bottleneck. A disadvantage is that this approach works only for homogeneous traffic. However, most (heavy) rail networks have also lines with heterogeneous traffic. When a line with homogeneous traffic would be assessed with this method, it should be decomposed from the heterogeneous traffic lines. A second approach is to analyse the buffer times between two trains. Bottlenecks can be identified using the following steps: first, determine all the start- and end times of the blocking time of a section. Second, calculate the buffer times at that section. Third, make a list of all buffer times of the study area and sort it from short to long. Fourth, keep all unique train pair combinations that appears first in the list. These are the shortest buffer times for each train pair. The section that are kept can be identified as bottlenecks. This approach works for both homogeneous as heterogeneous traffic. Also, with this method it easy to assess a complete network at once and it is not needed to decompose corridors.

Verification

The defined model to estimate blocking times for moving block has been verified using data out of the microscopic railway simulation model EGTRAIN. A portion of the South West Main Line between London Waterloo and Surbiton has been modelled. This corridor consists of two sprinter services, both operating 6 times an hour. First, the British fixed-block AWS/TPWS three aspect signalling has been used, which is very similar to the dutch ATB-EG signalling system. The output of this run had been used as input for the model to estimate the blocking time for a moving block signalling system. Then the same corridor was simulated with ERTMS L3 Moving block. The output of the second simulation was compared to the results of the defined model.

Between the estimated and simulation approach-, clearing- and blocking time was a correlation of respectively 0.98, 0.99 and 0.99. This shows that the model is very precise. Between the estimated and simulation approach-, clearing- and blocking time was an absolute difference of respectively 0.79s, 0.53s and 0.87s. The 95%-confidence interval is respectively (-4,4), (0,1) and (-3,3) seconds. Given that ProRail plans with an precision of 6 seconds, it is verified that the model is also accurate.

Case study

The described methods were applied to a case study in the Netherlands. Four corridors were selected as study area, which included 174 trains. These corridors, operating under NS'54/ATB, were simulated with the event driven microscopic simulation model FRISO, owned by ProRail.

For both fixed block as moving block the buffer times were calculated and analysed using the defined method. By keeping the same timetable, the buffer time between two trains increases on average by 75 seconds (60%) using moving block. Three different train pairs with the shortest buffer time were analysed in detail. These showed that the blocking time reduces between 45% and 60%. The critical location between these trains moved towards the first or last switch they shared, depending on the train order.

In this thesis a bottleneck is defined as the location with the shortest buffer time between two trains, also called a critical location, section or block. With fixed block signalling, most bottlenecks are located in a station area (69%), followed by a junction (13%), open line (13%) and cargo yard (5%). In general, with moving block the critical locations move towards switches, more bottlenecks will be located in station areas (74%) and junctions (17%) and less in open line areas (4%).

Conclusion

Moving block signalling promises a significant increase of capacity compared to a fixed block system, such as NS'54/ATB. This is mainly caused by a strong reduction of the approach- and running time.

With infrastructure data, rolling stock parameters and planned time-distance data, the blocking times for a moving block signalling system can be estimated. Given the blocking times, buffer times between trains can be calculated. With the buffer times, capacity bottlenecks in a network can be identified for both homogeneous as heterogeneous traffic without splitting corridors into line sections. With moving block, bottlenecks will move towards switches in junctions and station areas. On average, moving block will result in a significant increase of capacity in bottlenecks.

Samenvatting

Onderzoeksprobleem
Over de afgelopen 10 jaar is in Nederland het aantal gereisde kilometers per trein significant toegenomen. Dit heeft een gevolg: het Nederlandse spoornetwerk raakt vol. Eén van de oplossingen zou een nieuwe trein beveiligingssysteem kunnen zijn. ERTMS/ETCS L3 Moving Block (bewegend blok) beloofd een toename van de capaciteit op het spoor. Met een dynamisch snelheidsprofiel (net als in ETCS L2) en zonder vaste blokken, zal de spoorbezettingstijd korter zijn vergeleken met NS'54/ATB-EG. Echter, de capaciteitswinst zal verschillen tussen bepaalde plekken.

Om de capaciteit te beoordelen is de methode van gecomprimeerde dienstregelingen (timetable compression method, UIC leaflet 406) de meest gebruikte methode. Echter, een probleem met deze methode is het definiëren van de grenzen van een lijn sectie. Het is aan de infrastructuur manager om de richtlijnen te interpreteren, zoals ze door de UIC gegeven zijn. Echter, een mis-interpretatie zal lijden tot een verschillend resultaat. Daarom is er een wens voor een alternatieve methode om capaciteit knelpunten te identificeren, bijvoorbeeld een methode die gebruik maakt van spoorbezettingstijden van een bewegend blok systeem. Echter, ETCS L3 Moving Block is nog in ontwikkeling, dus het verzamelen van data uit de dagelijkse praktijk is niet mogelijk. Ook het verzamelen van bewegend blok data uit een simulatie is niet altijd een gemakkelijke opgave. In de simulatie software FRISO, gebruikt bij ProRail, is ETCS L3 Moving Block (nog) niet geïmplementeerd. Dus om de capaciteitswinst van de bewegende blokken te kunnen beoordelen, is er eerst een model nodig die de bloktijden schat van treinen die opereren onder een bewegend blok systeem.

Dit onderzoek heeft als doel om een methode te definiëren die de capaciteit van een bewegend blok systeem beoordeelt. De methode zal trein data gebruiken werkend onder een vast blok systeem om de spoorbezetting te schatten van een toekomstig bewegend blok systeem. In dit onderzoek zal deze hoofd onderzoeksvraag worden beantwoord:

Hoe kan de capaciteitswinst van Bewegend Blok worden beoordeeld met data voor zowel spoor corridors als knooppunten?

Methodologie
Om de capaciteitswinst van een bewegend blok systeem te kunnen beoordelen, is er een nieuw model gedefinieerd die de bloktijden van een bewegend blok systeem kan schatten. Als input gebruikt het model infrastructuur karakteristieken, parameters van materieel en geplande tijd-afstand data uitgevoerd onder een vast blok systeem. De bloktijden van een vast- en bewegend blok systeem kunnen worden onderverdeeld in zes componenten: route set-up, zicht- en reactietijd, aanrijtijd, rijtijd, uit-rijtijd en tijd om het blok vrij te geven. De lengte van de route set-up en de tijd om het blok vrij te geven zijn afhankelijk van de systeem karakteristieken en veiligheidsmarges. In het model is aangenomen dat dit constante waardes zijn. De zicht- en reactietijd is voornamelijk afhankelijk van de reactie tijd en de mate waarin de machinist oplet. Ook dit tijdscomponent is gemodelleerd als een constante waarde.

De aanrijtijd is gedefinieerd als de tijd die de trein nodig heeft om zijn remafstand te overbruggen. De aanrijtijd kan worden geschat met behulp van de rem karakteristieken van de trein, de helling van het spoor en tijd-afstand data van de trein. Met de rem karakteristieken en de helling van de spoorbaan kan de remming van de trein worden bepaald die de trein kan bereiken. Vanaf het punt waarvoor de aanrijtijd moet worden berekend (EoA) wordt de remcurve terug in de tijd en afstand berekend per tijdstap Δt. Als de rem curve dezelfde snelheid heeft bereikt als dat de trein heeft op die locatie, dan is het indicatie punt gevonden. De tijd die nodig is om van het indicatie punt naar de EoA te rijden is gelijk aan de aanrijtijd.

Voor de rijtijd zijn er drie verschillende situaties. Ten eerste, voor een normaal stuk spoor is de rijtijd gelijk aan de tijd tussen twee verzoeken om een gebied te mogen betreden. Ten tweede, wanneer er een virtueel blok is toegepast, is de rijtijd gelijk aan de tijd die nodig is om door het blok te rijden. Ten derde, als de trein een geplande stop maakt, is de rijtijd gelijk aan de tijd die de trein stil staat. De uit-rijtijd is gelijk aan de tijd die de trein nodig heeft om een locatie volledig te passeren. Uiteindelijk

is de totale lengte van een bloktijd de som van de zes tijdscomponenten. De start tijd is het passage tijdstip min de aanrijtijd, reactietijd en route set-up tijd. De eind tijd is het passage tijdstip plus de rijtijd, uit-rijtijd en de tijd om een locatie vrij te geven. Als de bloktijden zijn uitgerekend, kunnen de buffer tijden worden uitgerekend. Dit is de eindtijd van de leidende trein min de start tijd van de opvolgende trein.

Er zijn twee methodes gedefinieerd om knelpunten te identificeren. De eerste aanpak is het sommeren van de bloktijden van alle treinen die het blok passeren gedurende een tijdsperiode. Het blok van de hoogste gesommeerde bloktijd indiceert een knelpunt. Een nadeel is dat deze methode alleen werkt voor homogeen treinverkeer. Daarbij komt dat de meeste spoornetwerken ook trajecten hebben met heterogeen verkeer. Wanneer een traject zou worden beoordeeld met deze methode, zal deze dus moeten worden gescheiden van de traject met heterogeen verkeer. Een tweede aanpak is het analyseren van de buffer tijden. Knelpunten kunnen worden geïdentificeerd met de volgende stappen: eerst moeten alle begin- en eindtijden van de bloktijden worden bepaald. Als tweede kunnen de buffer tijden worden uitgerekend. Als derde wordt er een dataset gemaakt met alle buffer tijden in een gebied, oplopend in lengte met daarbij de bijbehorende trein combinatie. Als laatste stap worden alle niet-unieke trein combinaties verwijderd uit de dataset, waarbij de eerst voorkomende wordt behouden. Met andere woorden: de laagste buffer tijd tussen twee trein combinaties wordt behouden. Deze blokken kunnen worden geïdentificeerd als knelpunten. Deze tweede aanpak werkt voor zowel homogeen als heterogeen verkeer. Ook is deze methode makkelijk toe te passen om een heel netwerk in één keer te analyseren en is het niet nodig om trajecten te scheiden.

Verificatie

De gedefinieerde methode om de blok tijden van een bewegend blok systeem te schatten kan worden geverifieerd met behulp van data uit het microscopisch trein simulatie model EGTRAIN. Daarin is een deel van het Zuid-West hoofdspoor tussen Londen Waterloo en Surbiton gemodelleerd. Dit corridor bestaat uit twee sprinters die beide 6 keer per uur opereren. Als eerste is het Britse vast blok systeem AWS/TPWS gebruikt, die erg lijkt op het Nederlandse ATB-EG treinbeveiligingssysteem. De uitvoer van deze simulatie wordt als invoer gebruikt voor het model om de blok tijden voor een bewegend blok systeem te schatten. Vervolgens is hetzelfde studiegebied nogmaals gesimuleerd met ETCS L3 Moving Block. De uitkomst van deze simulatie is vergeleken met de geschatte bloktijden.

Tussen de geschatte en gesimuleerde aanrijtijd, uit-rijtijd en blok tijd zit een correlatie van respectievelijk 0.98, 0.99 en 0.99. Dit laat zien dat het model erg nauwkeurig is. Tussen de geschatte en gesimuleerde aanrijtijd, uit-rijtijd en bloktijd zit een absoluut verschil van respectievelijk 0.79s, 0.53s en 0.87s. Het 95%-betrouwbaarheidsinterval is respectievelijk (-4,4), (0,1) en (-3,3) seconden. Gegeven dat ProRail de dienstregeling plant met een nauwkeurigheid van 6 seconden, laat zien dat het model ook erg accuraat is.

Case studie

De beschreven methode is toegepast op een case studie in Nederland. 4 corridors zijn geselecteerd als studie gebied waarin in totaal 174 treinen opereren. Deze corridors, die opereren onder NS'54/ATB, waren gesimuleerd met een gebeurtenis-gedreven microscopisch simulatie model FRISO, eigendom van ProRail.

Voor zowel vaste- als bewegende blokken zijn de buffer tijden berekend en geanalyseerd met behulp van het gedefinieerde model. Als de dienstregeling wordt behouden, dan zal de buffer tijd tussen twee treinen toenemen met een gemiddelde van 75 seconden (60%) met een bewegend blok systeem. De drie verschillende trein combinaties met de kortste buffer tijd zijn ook in detail geanalyseerd. Deze lieten zien dat de bloktijden met 45% tot 60% afnemen ten opzichte van NS'54/ATB. De kritieke locaties bij deze treinen verplaatste zich naar de eerste of laatste wissel die ze beide gebruikte, afhankelijk van de trein volgorde.

In deze scriptie is een knelpunt gedefinieerd als de locatie met de kortste buffer tijd tussen twee treinen, ook wel een kritieke locatie, -sectie of -blok genoemd. Met vaste blokken zijn de meeste knelpunten te vinden in een stations gebied (69%), gevolgd door aansluitingen (13%), vrije banen (13%) en goederenemplacementen (5%). Omdat met bewegende blokken over het algemeen de kritieke plekken zich verplaatsen naar wissels, zijn er meer knelpunten in stations gebieden (74%) en aansluitingen (17%) en minder in vrije banen (4%).

Conclusie

Een bewegend blok systeem beloofd een significante toename van capaciteit vergeleken met een vast blok systeem, zoals NS'54/ATB. Dit wordt hoofdzakelijk veroorzaakt door een sterke afname van de aanrijdtijd en rijtijd.

Met infrastructuur data, materieel karakteristieken en geplande tijd-afstand data kunnen de bloktijden van een bewegend blok systeem worden geschat. Met de bloktijden kunnen de buffer tijden tussen trein worden uitgerekend. Met de buffer tijden kunnen capaciteitsknelpunten in een netwerk worden geidentifieerd voor zowel homogeen als heterogeen verkeer zonder het splitsen van corridors in lijn secties. Met een bewegend blok system zullen knelpunten zich verplaatsen naar wissels in aansluitingen en stationsgebieden. Over het algemeen zal een bewegend blok systeem zorgen voor een significante toename van capaciteit in knelpunten.

Glossary

L	Length of the train [m].
T	Timestamp [s].
a	Deceleration of the train [m/s^2].
t	Time length [s].
v	Speed of the train [m/s].
x	Position of the train [m].
AIS	Automatic Identification System for maritime transport.
ATB-EG	Automatische Trein Beveiliging Eerste Generatie.
ATO	Automatic Train Operation.
ATP	Automatic Train Protection system.
AWS	Automatic Warning System.
BC	Braking curve.
BT	Buffer time.
DRP	Name of area, Dienstregelpunt.
EBD	Emergency Brake Deceleration braking-curve in the ETCS braking model.
EBI	Emergency Brake Intervention.
EGTRAIN	Microsimulation model for railways.
EoA	End of Authority.
ERTMS	European Rail Traffic Management System.
ETCS	European Train Control System.
EU	European Union.
FRISO	Flexible Rail Infrastructure Simulation of Operations.
GSM-R	International standard for wireless railway communication, part of ERTMS.
HWT	Headway time.
IC	Intercity.
IM	Infrastructure Manager.
L2	ERTMS/ETCS Level 2.
L3	ERTMS/ETCS Level 3.
LRIT	Long Range Identification and Tracking system for maritime transport.
MA	Movement Authority.
MB	Moving Block.
MTPS	Rolling stock Position Service, Materieel Trein Positie Service.
NS'54	Dutch Legacy block signalling system, seinstelsel 1955.
OBU	On-Board Unit.
ORBIT	Warning system for a red signal in a train, OOGST RemcurveBewaking In Trein.
ORR	Driving on braking distance; Op Remweg Rijden.
PR	Position Report.
QATS	System to monitor and troubleshoot malfunctions of ERTMS.

RBC Radio Block Centre.
RFD Rail Fundamental Diagram.
ROT Runway Occupation Time.
RTM Realtime Train Monitoring.

SBI Service Brake Intervention Limit.

TI Train Integrity.
TPWS Train Protection and Warning System.
TROTS Train detection & tracking system.

UIC Union Internationale des Chemins de Fer, International Union of Railways.

Contents

Preface i

Summary iii

Samenvatting v

Glossary ix

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Problem definition . . . 1
- 1.3 Research objectives and scope . . . 2
- 1.4 Research questions . . . 2
- 1.5 Report outline . . . 3

2 Background 5
- 2.1 Blocking time theory . . . 5
- 2.2 Timetable compression method . . . 8
- 2.3 Braking curve . . . 10

3 Literature review 13
- 3.1 Terminology . . . 13
- 3.2 Shortcomings and limitations in UIC 406 Timetable compression method . . . 13
- 3.3 Other capacity assessment methods in railways . . . 14
  - 3.3.1 Analytical . . . 14
  - 3.3.2 Optimisation . . . 15
  - 3.3.3 Simulation . . . 15
  - 3.3.4 Other approaches . . . 16
- 3.4 Data-driven methods in other transportation modes . . . 17
  - 3.4.1 Aviation . . . 17
  - 3.4.2 Road traffic . . . 18
  - 3.4.3 Maritime transport . . . 19
- 3.5 Conclusion . . . 19

4 Methodology 21
- 4.1 A mathematical approach to assess moving block track occupation times . . . 21
- 4.2 Using track occupation data to identify capacity bottlenecks . . . 26
  - 4.2.1 Summing blocking times . . . 26
  - 4.2.2 Buffer times . . . 28

5 Verification 29
- 5.1 EGTRAIN . . . 29
- 5.2 Verification case study setup . . . 30
- 5.3 Approach to assess moving block track occupation times . . . 32
  - 5.3.1 Data collection and processing . . . 32
  - 5.3.2 Results . . . 33
  - 5.3.3 Discussion and conclusion . . . 36
- 5.4 Using track occupation data to identify capacity bottlenecks . . . 37

6 Case Study **41**
6.1 FRISO . . . 41
6.2 Model setup . . . 43
6.2.1 Study area . . . 43
6.2.2 Timetable and rolling stock . . . 43
6.2.3 Infrastructure . . . 43
6.2.4 Route setup times . . . 44
6.3 Data collection and processing . . . 44
6.3.1 Fixed block blocking times . . . 45
6.3.2 Planned time-distance data . . . 46
6.3.3 Rolling stock parameters . . . 47
6.3.4 infrastructure characteristics . . . 47
6.4 Model application . . . 48
6.4.1 Approach time . . . 48
6.4.2 Clearing time . . . 50
6.4.3 Running time . . . 50
6.5 Buffer time calculations . . . 51

7 Results **53**
7.1 Summing blocking times . . . 53
7.2 Buffer times analysis . . . 54
7.2.1 D800-B7400 (Amsterdam CS - Amsterdam Bijlmer ArenA) . . . 56
7.2.2 H7400-B3900 (Duivendrecht - Amsterdam Bijlmer ArenA) . . . 57
7.2.3 A4400-C3500 (Boxtel - 's-Hertogenbosch) . . . 57
7.2.4 Critical sections . . . 58
7.3 Roberto . . . 60
7.3.1 D800-B7400 (Amsterdam CS - Amsterdam Bijlmer ArenA) . . . 62
7.3.2 H7400 - B3900 (Duivendrecht - Amsterdam Bijlmer ArenA) . . . 64
7.3.3 A4400-C3500 (Boxtel - 's-Hertogenbosch) . . . 66

8 Discussion **69**
8.1 Limitations of the model . . . 69
8.2 Calculation of approach time . . . 70
8.3 Model application with realised data . . . 71

9 Conclusion & recommendations **73**
9.1 Conclusion . . . 73
9.2 Recommendations . . . 76
9.2.1 Future work and research . . . 76
9.2.2 Practical improvements to FRISO and Roberto . . . 77

Bibliography **79**

A Different shapes of the fundamental diagram **87**

B Added code lines in EGTRAIN **89**

C Python scripts verification study **93**

D Python scripts case study **97**

E Case study: Rolling stock **111**

F Case study: Infrastructure **117**

G Case study: trajectories **133**

1 Introduction

1.1. Background

Over the past decade the number of travelled kilometres by train has increased significantly in the Netherlands. This is good news for environmentalists, however it has also a consequence: the Dutch railway network is getting full [27]. Pre-Covid-19, Rogier van Boxtel, at that time CEO of the NS, warned that the capacity limit will be reached in 2027 [45]. During the corona pandemic, the number of travellers has dropped massively, but it is expected that the growth will continue after the crisis [46]. Building new infrastructure can be a solution for the railway industry to cope with the demand, however this is expensive and there isn't always enough space available to build new tracks. Therefore a solution may be found in another corner: a new signalling system.

On the 17th of May 2019 the government decided to replace the complete block signalling and Automatic Train Protection (ATP) with European Rail Traffic Management System (ERTMS) before the end of 2050 [67]. The current system, NS'54/ATB-EG block signalling, has been designed more than 60 years ago and the hardware is outdated, so this needs to be replaced. To meet the agreements made with the European Union (EU), it is chosen to build ERTMS Level 2 on the Dutch railway network, instead of investing in new hardware of the current ATB-EG system. Except advantages such as ERTMS is safer compared to ATB-EG and it is interoperable across European borders, ERTMS has another advantage: the infrastructure capacity is expected to increase. With ATB-EG the blocklengths and signal positions are based on trains with the worst braking rate. In ETCS Level 2 the system computes a dynamic speed profile, based on the characteristics of the train, whereby trains can brake later for a red signal.

Although in ETCS Level 2 wayside signalling makes place for in-cab signalling, the system still uses fixed blocks. In ETCS Level 3 Moving Block, one of the next steps in the development of ERTMS, this will become moving blocks. With fixed blocks the complete network has been split up in sections (blocks), where each block can be occupied by one train. When a certain block is occupied, no other train can enter that block. With moving blocks, the occupied block moves with the train. It is not needed to reserve time to run through a block, resulting in shorter occupation times and more efficient use of the infrastructure [32]. Also it is expected that maintenance costs with ERTMS L3 Moving Block will reduce, since there will be less equipment alongside the tracks.

1.2. Problem definition

Just like ETCS L2, ETCS L3 Moving Block promises an increase of capacity. However, how big this promise is, is part of some unanswered questions. This may differ between certain locations. ProRail wants to know this and has a strong interest in the development of ETCS L3 Moving Block, because it can increase the reliability and availability of their network [27]. With knowing capacity bottlenecks of ATB-EG, ETCS L2 and ETCS L3 Moving Block, it can support ProRail's planning and long-term decision making.

To assess rail capacity, the timetable compression method [37] is arguably the most widely used method [100]. An issue with this method is defining the boundaries of line sections. For example, should a line be split into smaller line sections or should it be seen as a long-distance service. Both choices will lead to different results [10, 40]. Leaflet Code 406 [37] by the UIC gives some guidelines, but in the end it is up to the infrastructure manager's expertise to make the right choice.

Therefore there is a need for an alternative method, for example a method which could make use of track occupation data collected during operations or produced with simulation. However, gathering moving block data from daily operations is not possible, since ETCS L3 Moving Block is still in development. Also gathering moving block data out of simulation isn't always a convenient solution, especially for ProRail. In the simulation software FRISO, used at ProRail, ETCS L3 Moving Block is not (yet) implemented. Implementing Moving Block in FRISO, or building the Dutch rail infrastructure in another simulation tool, is expensive and a time consuming process. This makes it harder for ProRail to investigate the effects of Moving Block on the capacity and bottlenecks of the Dutch railway network. So to assess capacity gains of moving block, first a model is needed which estimates the blocking times of trains operating under moving block signalling.

1.3. Research objectives and scope

This research aims to define a novel method to assess capacity of moving block. It uses track occupation data from a fixed block system to estimate the track occupation of a future moving block system. Therefore, the following main research question will be answered:

How can capacity gains of Moving Block be assessed with data for both railway corridors and complex nodes?

Once the method has been defined, simulation data will be gathered with EGTRAIN. First, a corridor will be simulated with a fixed block signalling system. Then, the defined method will be applied to estimate the blocking times for a moving block system. The outcome will be compared to moving block simulation data from EGTRAIN, to validate the model. The defined method will be applied to a case study in the Netherlands. Four corridors in the Dutch railway network will be simulated and assessed to see how moving block affects bottlenecks in corridors and complex nodes.

For this research two assumptions are made. First, the timetable is given. The goal is to compare two signalling systems and find the benefit in infrastructure occupation. Therefore the timetable, and thus the train order, will be a constant factor. Second, it is assumed that moving block has been fully developed. If and when moving block will be integrated in the dutch railway network, it will be ERTMS Level 3 Moving Block. In the current state of ERTMS Level 3 Moving Block the verification of train integrity – performed by an on-board device called Train Integrity Monitoring (TIM) – for trains with variable composition such as freight trains is still an open challenge [4]. Besides this, communication failures could lead to an emergency brake. This results in a capacity drop, which has a big impact on the operations' performances. This thesis will not discuss these two problems.

1.4. Research questions

Based on the problem definition and main research question, sub questions are formulated. These will be answered in this research and will help to answer the main research question. The sub questions are:

1. **What are the shortcomings in the timetable compression method and which solutions do exist?**
   This question will be answered through a literature study. It is known that the UIC Timetable compression method has some limitations and shortcomings. Since the publication of the 2nd edition in 2013, researchers have proposed a couple of solutions to these limitations.

2. **Which potential data-driven methods to assess capacity do exist?**
   Besides railways, also other transportation modes have to deal with capacity constraints. The literature study will give an overview of data sources in other transportation modes and how this data is being used to assess capacity.

3. **How can the blocking times under moving block signalling be estimated?**
   In this thesis a method will be defined on how to estimate the blocking times under moving block signalling using train data operating under fixed block signalling.

4. **How can bottlenecks be identified without splitting lines?**
   This method should bypass the limitations in the UIC timetable compression method, but give similar results.

5. **How does Moving Block affect bottlenecks?**
   Capacity is constrained because of bottlenecks in the railway network. Especially at these locations it is interesting to know how much moving block affects the buffer times and minimal headways, since these locations limit the maximum number of trains on a corridor. There is also a possibility that the bottleneck will move to another location on a corridor.

6. **How can the defined method support ProRail and the rail industry in assessing capacity impacts of Moving Block signalling?**
   In the last part of this study already existing tools at ProRail, FRISO and Roberto, will be used next to the defined method. It will be shown how these can complement each other.

1.5. Report outline

Chapter 2 contains a more extensive description of background information. First the blocking time theory will be explained and how the differences between fixed- and moving block affect the blocking times. In section 2.2 the timetable compression method by the UIC is explained. Section 2.3 gives information about the principles of the braking curve under ERTMS. These principles will be taken into account in the model definition in chapter 4.

Chapter 3 is a literature review, starting with explaining some terminology. This is followed by a discussion of the shortcomings and limitations in the UIC Timetable compression method (sec.3.2 and other capacity assessment methods in railways used in literature (sec.3.3. In section 3.4 a broader scope is taken at data driven capacity assessment methods in other transportation modes.

In chapter 4 the model to estimate the blocking times under moving block signalling is explained, followed by an approach to identify capacity bottlenecks using the blocking times. These methods will be verified in chapter 5 using simulation data of EGTRAIN.

Chapter 6 applies the methods that are explained and verified in the previous chapters. First the FRISO simulation tool and the model setup are stated (sec.6.1 and 6.2), followed by an explanation on how the model is being applied and the calculations are made. In chapter 7 the results of the case study are given.

This thesis is being closed by a discussion on the used methods and results (Ch. 8) and a conclusion (Ch.9), which answers the research questions. In section 9.2 recommendations for future research and improvements to FRISO will be given.

2

Background

This chapter provides relevant background information about infrastructure occupation in railways and will be used as starting point in chapter 4. In section 2.1 the Blocking time theory is explained, including the differences between a Fixed and Moving Block signalling system in this theory. Section 2.2 summarises the Timetable compression method by the UIC [37], which is one of the most used capacity assessment methods in railways. In section 2.3 the principle of the braking curve in ERTMS is told.

2.1. Blocking time theory

In general, blocking time – also called occupancy time – is the total time a section of a track is allocated exclusively to a train movement. It starts as soon as the preparations to allocate a train's Movement Authority (MA) demand for exclusive occupation of a route's element. This request should be made before the train reaches the element, in case the request will be denied and the train has still sufficient time and space to stop. The blocking time ends when the train has completely left the section and all signalling components have been reset to normal position [11, p.221]. Thus, the blocking time of a track section is much longer than the time a train occupies the section.

In figure 2.1 the blocking time of a **Fixed Block** signalling system is shown, including the specifications of all attributes. The length of a block is the distance between two signals, which can be a physical signal along the tracks (e.g. NS'54/ATB-EG or ETCS L1) or an in-cab signal (e.g. ETCS L2). Each time component is explained in table 2.1. The length (in time) of each attribute depends, among other things, on the infrastructure, the signalling system that has been used and train characteristics. The blocking time for a fixed block signalling system is always distance-discrete, which means that an occupied block always starts and ends at a fixed location. Drawing the successive blocking times for a train over a railway line in a time-distance diagram leads to a Blocking Time Stairway. This represents the operational use of a railway line by a train [30].

Compared to fixed block, **Moving Block** has the same time components. However, instead of the occupied block having a fixed position and length, it moves along with the train. Therefore one will –- on the first sight –- not observe the typical stairway in a timetable as with fixed block signalling, but a bandwidth around the trajectory of the train. However, because the subsystems (geometric interlocking, Radio Block Centre, Train Position Report) work periodically – and not continuous, Moving Block is a time-discrete system. One will still get steps in his timetable, but much smaller ones (see figure 2.2). The step sizes are defined by the frequency of the subsystems, T_CYCLOC. During the time it takes to request and get the MA, the train moves. Considering this, the running time is equal to T_CYCLOC, instead of the time it takes to run through a block. All in all, it is accepted to speak about a bandwidth, but in simulation it should still be considered that it is discrete [11, p.226].

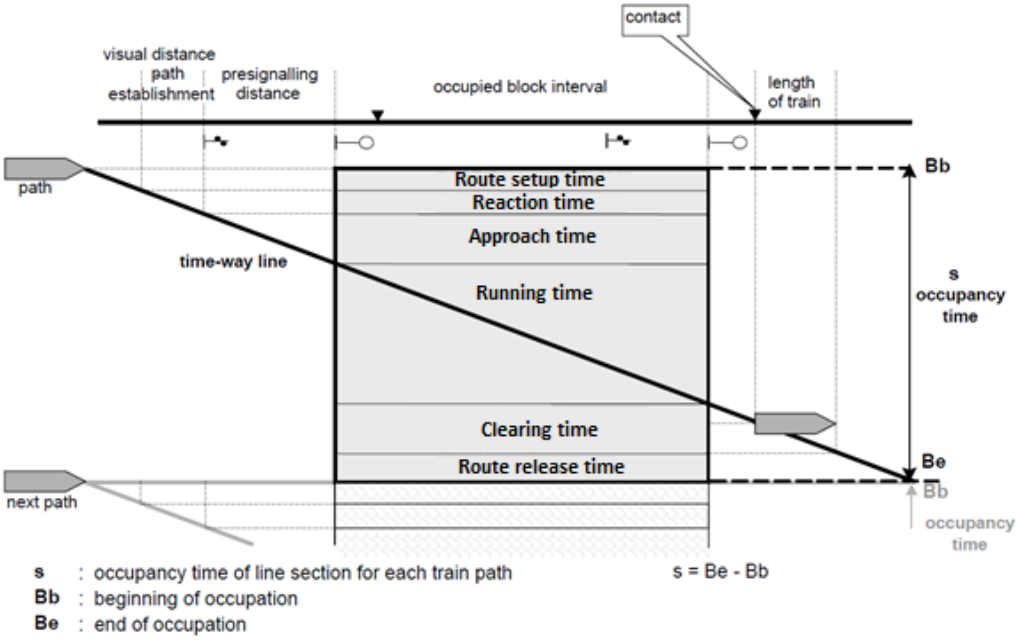


Figure 2.1: Physical attributes of a block section (Source: International Union of Railways (UIC) [37], edited by author)

Table 2.1: Blocking time components [11]

Time component	Remark
Route setup	Preparation of the MA has to start sufficiently early that the signal aspect changes at latest with the start of the reaction time. The preparation covers moving switches, locking route elements, commanding the signal aspect.
Reaction	A reaction offset to interpret the distant signal aspect is granted to the train driver. It may be defined as a time or as a distance (in correspondence to minimum sighting distances). In case of a scheduled stop ahead of the track section the reaction time has to be replaced by the time demand for the departure process, which may usually be triggered just after the opening of the exit signal.
Approach	The approaching time is related to the approach indication in approach to the block. It may either a separate distant signal (one-block signalling) or a combined main/distant signal (two-block signalling). If in two-block signalling the braking distance takes more than one block section, multiple-aspect signalling has to be applied granting multiple block sections for braking.
Running	The time for the head of the train to run through the block
Clearing	The clearing time is the time it takes to completely leave the section. It starts the moment the head of the train left the block and ends as soon the rear-end has left the block.
Route release	The route release is the time it takes to reset all signalling components to normal position, so that another train can enter that block.

Virtual blocks

Although Moving Bock is a time-discrete signalling system, at some locations it works like a distance-discrete system [11, p.230-231]. Depending on the specific infrastructure design, certain sections have to be operated in (virtual) blocks in any case, as trains should not come to a standstill for various reasons:

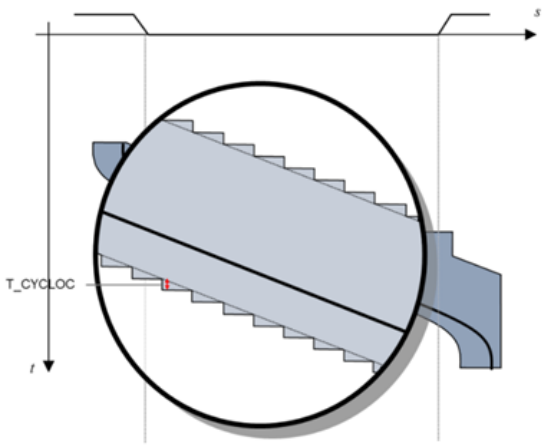


Figure 2.2: Blocking time of moving block being discrete with intervals of T_CYCLOC (Source: Büker et al. [11])

1. Moveable elements (e.g. switches, bridges) can only be occupied as a whole
2. Overhead catenary design does not allow standstill
3. Level crossings, to prevent a blockage for road traffic, which could lead to dangerous situations
4. Initial traction effort is too low to ensure re-acceleration (e.g. slopes)
5. Maximum coupling forces may avoid re-acceleration

Trains will get permission to approach these sections completely, or not at all, so it can be seen as distance-discrete instead of time-discrete. However switches have to been reported as shifted and locked before the train is authorized to approach that switch. Therefore extra time is needed to reserve a switch, in the contrary to for example level crossings, which is a fail-safe system. In practise path conflicts and capacity bottlenecks occur most frequently in stations and junctions rather than on open line sections [5] [59]. This is being confirmed in section 7.2.4. At stations and junctions switches are used to make it possible to change tracks. Since these switches are operated as (virtual) blocks, these become often decisive for minimum headway times and decrease the benefits of the moving block principle (see figure 2.3). In consequence, this could have a huge impact on the capacity consumption.

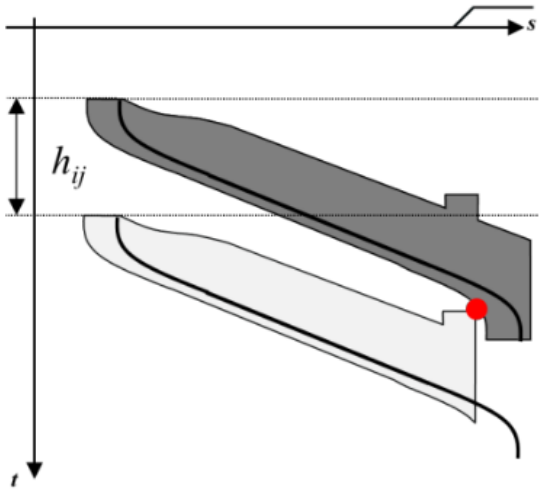


Figure 2.3: Moving block being interrupted in switchpoint (Source: Büker et al. [11])

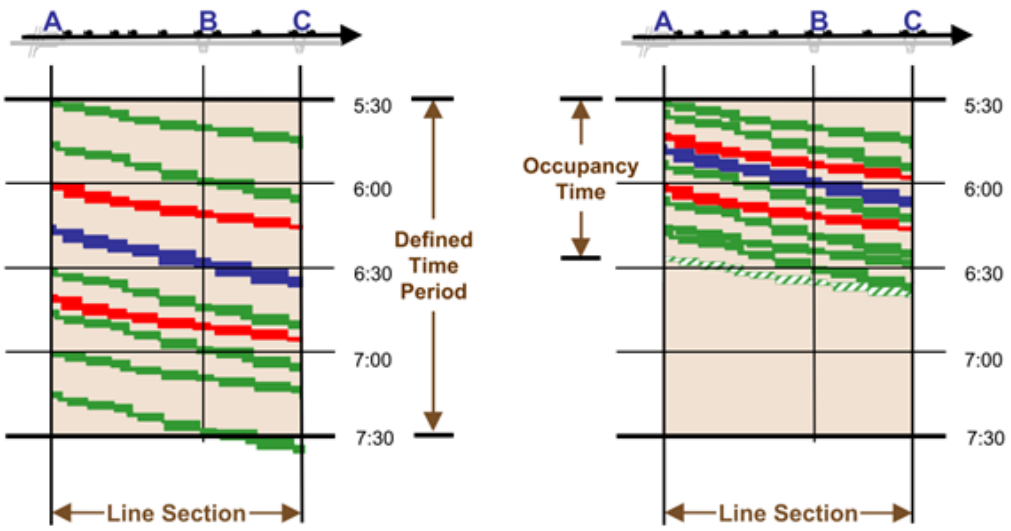


Figure 2.4: Timetable on a double-track line section before (left) and after (right) compression (Source: International Union of Railways (UIC) [37])

2.2. Timetable compression method

The timetable compression method [37] is arguably one of the most used methods to assess capacity of railways, see for example [29, 50, 59]. In this section the method will be explained for general cases, where in section 3.2 the shortcomings and limitations are elaborated. For special cases, one can read leaflet UIC Code 406 [37]. The timetable compression method follows out of the blocking time theory. When the blocking time has been calculated for each block, one gets a graph such as figure 2.4. The original purpose of the method is to measure capacity occupation of a given timetable, which is achieved by compressing the train blocking time stairways [10]. The compression and evaluation can be summarised in four steps:

1. **Defining infrastructure and timetable boundaries**
   First, a corridor should be defined. A corridor represents the main (inter)national connections and consists of multiple lines. The chosen corridor excludes hump and storage yards, terminals and other adjacent properties, since they operate independently. However, their impact on the corridor should be included (e.g. a train path to/from a terminal).

2. **Defining section for evaluation**
   There are two types of sections to consider:

   - **Train path line sections**, referring to market conditions, are used for measuring capacity by inserting or excluding train paths. They are those parts of the line in which sequences of long-distance trains are defined within a timetable
   - **Line sections** are a subset of a Train path line section and are used to measure capacity consumption by compressing the timetable.

   Train path line sections should be broken down into line sections if any of the following criteria are applicable:

   - Establish areas where the infrastructure conditions differ significantly:
     - Signalling system
     - Number of tracks in the line section
     - Branching lines
   - Establish areas where significant timetable or traffic operation differences occur:
     - Beginning or ending services

  - Different number of trains
  - Train mixture and/or train sequence
  - Crossing trains
- Line section along single-track lines are defined by the corresponding adjacent interlockings where trains can cross or overtake

It should be noted that (train path) line sections may overlap. For example, when a train path line section will be broken down into a line section at a station area, this station area will be included in the analysis of both line sections. Once the line section has been defined, the analysed time period will be defined. This could be 24h, to calculate the occupancy rate over a whole day, or only for peak hours. However, it is advised to use time periods not shorter than 2 hours.

3. **Calculating capacity consumption**
Once the sections and time period are defined, the timetable can be compressed. This will be done by compressing the train paths (blocks) as close as possible, without any overlap between two blocks. For example, see figure 2.4. The occupancy time rate can be computed by:

$$\text{Occupancy time rate } [\%] = \frac{\text{Occupancy Time}}{\text{Defined Time Period}} \times 100 \tag{2.1}$$

As a guideline, the calculated occupancy time rate should not exceed the proposed occupancy time rates given in table 2.2.

Table 2.2: Proposed occupancy time rates (Source: International Union of Railways (UIC) [37, p.29])

Type of line	**Peak hour**	**Daily period**
Dedicated suburban passenger traffic	85%	70%
Dedicated high-speed line	75%	60%
Mixed-traffic lines	75%	60%

To analyse a timetable, the capacity consumption can be used. This can be calculated by:

$$\text{Capacity consumption } [\%] = \frac{\text{Occupancy Time}(1 + \text{Additional Time Rate})}{\text{Defined Time Period}} \times 100 \tag{2.2}$$

Where the additional time rates are given in table 2.3. These time values are added to secure quality of operation (e.g. buffer time, quality time, etc.)

Table 2.3: Proposed additional time rates for lines (Source: International Union of Railways (UIC) [37, p.30])

Type of line	Peak hour	Daily period
Dedicated suburban passenger traffic	18%	43%
Dedicated high-speed line	33%	67%
Mixed-traffic lines	33%	67%

4. **Evaluating capacity consumption**
In order for capacity consumption values to best represent the corresponding infrastructure, the following conditions – given by leaflet code 406 [37, p.30-31] – can be used as a guideline:

- The capacity consumption values reflect the infrastructure characteristics of the defined train path line sections.
- The line section with the highest capacity consumption value along the train path line section is the representative line section for the train path line section.
- Acceptable quality of service is represented by capacity consumption values of up to and including 100%.

- Capacity consumption values beyond 100% represent a bottleneck, which means a lower quality of service, and should be subject to timetable or infrastructure improvement measures.
- Capacity consumption values below 100% represent available capacity and thus the potential for additional train paths along the defined train path line section.

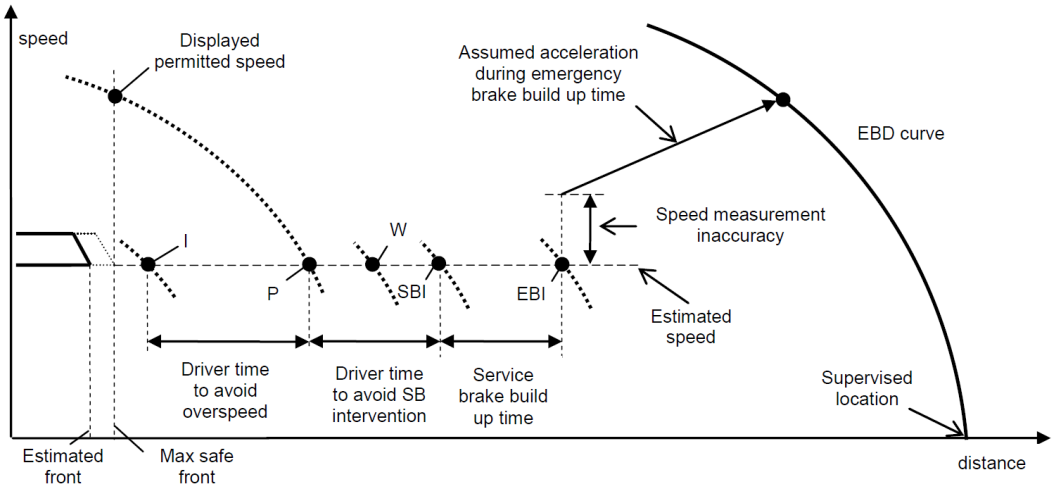


Figure 2.5: Overview of the EBD braking curve and its related supervision limits (Source: European Railway Agency [25])

2.3. Braking curve

The braking curve is defined by the European Railway Agency as the prediction of speed decrease versus distance [25]. From this prediction the ETCS on-board computer calculates the braking distances. These will also be used to assist the train driver and to allow him to drive comfortably, by maintaining the speed of the train within the appropriate limits.

The braking curve related to the speed decrease due to an emergency is called Emergency Brake Deceleration (EBD) curve. From the EBD and the measured speed of the train, the ETCS calculates in real time the distance necessary to stop. This distance before the stop location is called Emergency Brake Intervention (EBI) location. It is the point beyond which ETCS will bypass the driver, intervenes and stops the train. However, before this point is reached, the system calculates several other supervision limits: Indication (I), Permitted speed (P), Warning (W) and Service Brake Intervention (SBI). These locations are indicated in figure 2.5 and have the following function:

- Indication: the ETCS gives a indication signal to the driver. The driver will have enough time to act on the service brake, so that the train does not overpass the Permitted speed.
- Permitted speed: in case of overspeed, the ETCS leaves the driver an additional time to act on the service brake so that the train will not overpass the point beyond which ETCS will trigger the command of the brakes. The distance from the indication to the supervised location is called the perturbation distance.
- Warning: the moment when an additional audible warning will be given after the Permitted speed has been overpassed.
- Service Brake Intervention: This limit takes into account the service brake build up time so that the EBI supervision limit is not reached. The SBI is an optional feature which can be installed to avoid too frequent emergency braking. Emergency brakings can be damaging for both the rolling stock and the track.
- Emergency Braking Intervention: when this limit will be reached, the system applies the emergency brakes.

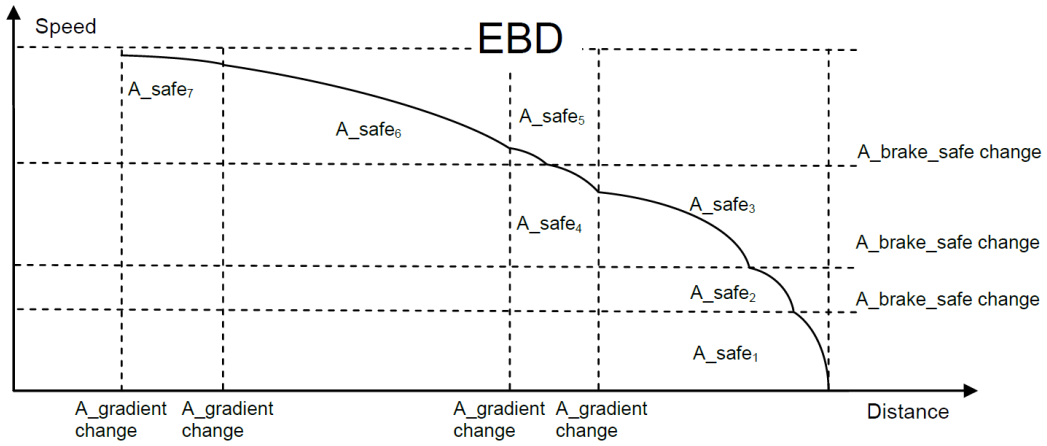


Figure 2.6: Construction of the EBD (Source: European Railway Agency [25])

Calculation of braking curves

The ETCS braking curve algorithms need the following parameters to perform their real time supervision and advisory functions [25]:

- Physical parameters, which result from the real time measurements by the ETCS on-board equipment: the position, speed and acceleration of the train
- ETCS fixed values. They mostly relate to the ergonomics of the braking curve model itself (e.g. driver reaction times)
- ETCS trackside data, such as signalling- and infrastructure data. These parameters are under full responsibility of the Infrastructure Manager.
- On-board parameters, such as information of the rolling stock.

The EBD is a piecewise parabolic shaped curve that starts from the target location and is computed with the deceleration from the emergency brake system and the deceleration/acceleration due to the uphill/downhill slopes:

$$A_{safe}(v,d) = A_{brake\_safe}(v) + A_{gradient}(d) \tag{2.3}$$

where: v = Speed [m/s]
d = Distance [m]

During the time the emergency brakes are released, both deceleration factors might change. The EBD has to adapt to that, which can be seen in figure 2.6. Every time one the factors changes, a new braking partial curve will be calculated, to be sure the train stops on the designated location.

$A_{gradient}$ is the acceleration/deceleration due to slope of the tracks. It can be calculated by:

$$A_{gradient} = \frac{g \cdot grad}{1000 + 10 \cdot M_{rotating}} \tag{2.4}$$

where: g = gravitational acceleration ($9.81\ m/s^2$)
$grad$ = gradient value in ‰[m/km]
$M_{rotating}$ = Compensation factor due to rotating mass of the train [%]

The gradient information is given to the on-board equipment in form of a gradient profile, see figure 2.7. The information is piecewise constant between two defined locations, so for every piece of track the gradient is known, but it is not continuous. It is continuous in a way that for every piece of track it is known what the gradient is, but constant between two defined locations [24, p.79].

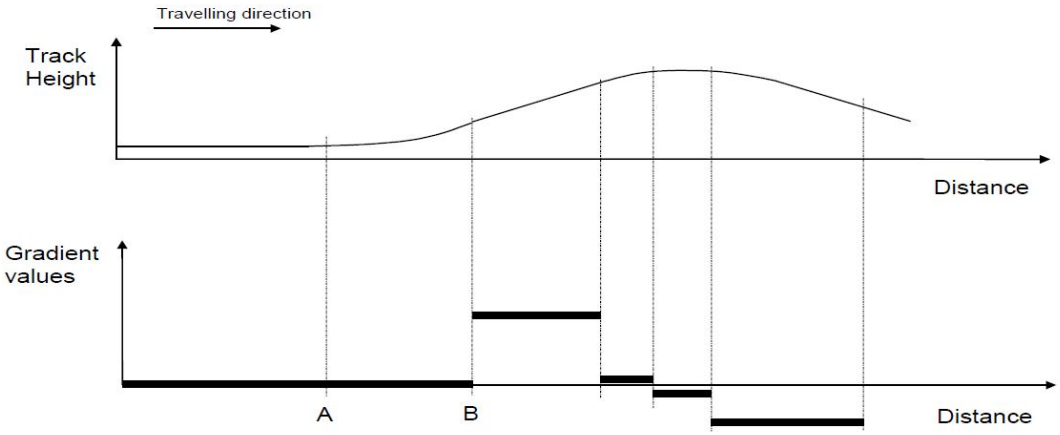


Figure 2.7: Gradient profile in ERTMS (Source: European Railway Agency [24], p.80)

As input parameters for A_{brake_safe} the EBD makes a differentiation between two situations. First, when the ETCS on-board equipment is fitting a train with a predefined composition, all corresponding rolling stock parameters can be preconfigured in the system. The train for which it is possible to store such predefined data are called **Gamma trains** [25]. For these trains, A_{brake_safe} can be calculated by:

$$A_{brake\_safe} = A_{brake\_emergency} \cdot K_{dry\_rst}(M\_NVEBCL) \cdot \{K_{wet\_rst} + M\_NVAVADH \cdot (1 - K_{wet\_rst})\} \quad (2.5)$$

In case of variable composition, it is neither possible to directly express nor to define the braking performance with deceleration data. These trains are so-called **Lambda trains** and a conversion model is needed. In this case, A_{brake_safe} is replaced by A_{brake_tuned} and can be calculated by:

$$A_{brake\_tuned} = A_{brake\_converted}(\lambda) \cdot K_{v\_int}(Train\ type) \cdot K_{r\_int}(L\_train) \quad (2.6)$$

where λ is the braking percentage, calculated according to UIC leaflet 544-1. For Lambda trains, equation 2.3 results in:

$$A_{safe}(v, d) = A_{brake\_tuned}(v) + A_{gradient}(d) \quad (2.7)$$

In equations 2.5 and 2.6 specific national values are used, which can differ from country to country. K_{dry_rst} and K_{wet_rst} quantify the braking performance on dry and wet rails. In the Netherlands, however, the factor for wet rails is not applied, so this is zero. Also the speed correction factor K_{v_int} and train length correction factor K_{r_int} are national variables. These integrated correction factors are actually used as tuning factors, allowing the infrastructure manager to tweak the ETCS braking curves to their national legacy signalling system [25].

Besides the deceleration factor A_{safe}, also the brake build up time T_{brake} affects the time when the emergency brakes should be released. For Gamma trains, T_{brake} is known by the railway undertaking. For lambda trains, the conversion model [24, p.197] dictates equation 2.8.

$$T_{brake} = Kt_{int} \cdot \left(a + b \cdot \left(\frac{L_{train}}{100} \right) + c \cdot \left(\frac{L_{train}}{100} \right)^2 \right) \quad (2.8)$$

where: Kt_{int} = Integrated correction factor
a, b, c = factors depending on brake settings and emergency/service brake intervention
L_{train} = Train length in m

3

Literature review

The literature review gives an overview of capacity assessment in railways and other transportation modes. First some terminology of capacity is explained (sec. 3.1), followed by an review of the UIC 406 Timetable compression method in section 3.2. Sections 3.3 and 3.4 give an overview of other capacity assessment methods. Section 3.5 provides a conclusion to this literature review.

3.1. Terminology

In literature capacity has been widely discussed. There is also a wide variety of definitions of capacity, all referring to different situations and/or phases in planning. Bešinović and Goverde and Jensen et al. discusses the terminology around capacity used in literature:

- **Theoretical capacity** (Synonyms: Design capacity, Absolute capacity, Capacity throughput): The maximum number of trains that can traverse a given part of the network in a predefined time period. The operation must be completely homogeneous with a single ideal train type with no supplements of any kind and represents an upper limit for infrastructure capacity. [10] [41]
- **Maximum capacity**: The maximal set of trains that can be handled in a given time period while considering the actual train mix where train types may have different stopping patterns and maximum speeds resulting in heterogeneous operation. Buffer times are not included in headways between trains. [41]
- **Practical capacity** (Synonyms: Achievable capacity, Effective capacity): The maximum capacity that can actually be used in a given time period to obtain a robust and stable operation. Buffer times are added to the minimum headway between trains to obtain planning headways that provide stability and robustness. [10] [41]
- **Capacity occupation** (Synonyms:Infrastructure occupation, Occupancy time, Consumed capacity, Carrying capacity, Used capacity): The time a set of trains in a given sequence occupies the infrastructure. [41]
- **Capacity occupation rate** (Synonyms: Utilization rate): The ratio between capacity occupation and a predefined time period, in which the set of trains should operate. [41]

3.2. Shortcomings and limitations in UIC 406 Timetable compression method

The timetable compression method is a capacity assessment method proposed by the International Union of Railways (UIC). As explained in section 2.2, the approach is to calculate capacity consumption by compressing a timetable and to evaluate the number of possible train paths for a line, node and corridor [37].

As explained in the UIC leaflet 406, the method should be seen as a guideline for infrastructure managers (IM), instead of strict rules. The occupancy- and additional time rates for lines are *proposed* as a fixed value. For nodes the occupancy rates are given as a range, since there was not enough data available when it was published. To interpret the guidelines correctly by the infrastructure manager, experience is needed. Misinterpretation can easily be made, for example should a line be split into smaller line sections or should it be seen as a long-distance service. Both choices will lead to different results [40]. Another argument is given by Jamili [39]: "the UIC 406 capacity method can determine "only" the capacity consumption on the line sections. The capacity consumption of larger parts of single track infra- structures may be estimated by considering only the crossing stations that are actually operated. This often leads to an increase of capacity on the railway line which is not found by simply 'compressing' the train graphs. Furthermore, this might result in the paradox situation where adding an extra train to the timetable reduces the capacity consumption. Due to the extra trains, the UIC 406 method requires the line section to be divided into more line sections. Compressing the timetable graphs for the line sections appeared by adding these extra trains results in less capacity consumption than in the case without the extra trains."

Bešinović and Goverde [10] reviewed the timetable compression method and made a literature review. This resulted in a list of four limitations of this method. First, according to the UIC 406 method the network should be decomposed in train path lines or line sections. Due to this decomposition, certain train dependencies, which overlap line sections, are neglected and result in an underestimated capacity occupation. A second limitation is the length of the decomposed line sections, which affect the resulting capacity occupation significantly. To overcome these issues, they proposed a network model for capacity assessment that preserves microscopic details of the infrastructure and all train dependencies using max-plus algebra. Third, the given saturation rates are a rough guideline and are highly dependent on the infrastructure layout, train characteristics and level of service. Besides, they may vary significantly for different national networks. A fourth limitation is one of the remaining limitations of the UIC 406 method. For the capacity assessment in nodes the UIC proposes to decompose a node in switch areas and platform track areas, and evaluate each segment independently. More on this can be read in section 3.3.1.

Furthermore there is the proposed method by the UIC on how to calculate occupancy time of switch areas. According to Veselý and Bazant [94] a lot of manual calculations are needed and it could be very time consuming. To solve this, they designed the method in simulation tool Villon [83], so the process can be automated. The tool was tested with a simple intersection, where a two line track and a one line track arrive at a station. The tool was verified by performing the manual calculations according to the UIC 406 leaflet.

3.3. Other capacity assessment methods in railways

Besides the timetable compression method, other methods exist to assess railway capacity. Some methods are an extension of the timetable compression method, while others can be seen as an alternative approach. In literature [1] [40] [57] most methods can be put into the categories analytical, optimisation and simulation. An overview will be given in the next subsections, followed by an subsection where queuing models and the rail fundamental diagram are elaborated.

3.3.1. Analytical

Analytical methods are aimed at determining a preliminary solution and are characterized for modelling the railway environment by means of a mathematical formula. In 1996 the International Union of Railways (UIC) published an analytical method in code 405 [36], which was officially replaced by the timetable compression method in 2004 as standard measure of capacity. However, it is still being used by some researchers, since according to Rotoli et al. "it offers an efficient estimation of the capacity of a line" [78]. They analysed the UIC 405 code and propose, in extension to this, a simplified approach where detailed information about the signalling system is not needed [77, p.19]. The input variables will be the data such as distance, scheduled travel time and number of train between consecutive nodes. A disadvantage is that this approach could underestimate the capacity occupation. To overcome this, Bešinović and Goverde [10, p.33] introduces a max-plus automata model for capacity assessment in

nodes.

Weik et al. [100] use a hybrid approach consisting of analytic timetable compression and stochastic simulation to provide insights to the area of timetable-independent modeling of train path correlation effects. Their work includes three major aspects: they (1) present a versatile Matlab-based simulation toolbox for timetable independent compression in station areas, (2) discuss the problem of accounting for correlations within the observation area on double-track railway lines and (3) explore approaches to incorporate interactions between line segments and stations. Their approach is more suitable to evaluate the effect of the underlying infrastructure on the capacity for strategic planning rather than the capacity occupation of a specific timetable. To cover the timetable effect, they calculate the infrastructure occupation for at least 100,000 different generated and compressed timetables.

In long term strategic planning, when only limited knowledge is available about the future timetable, the analytical method of Schwanhäußer [80] [81] can be used to evaluate capacity [55]. The method, also called STRELE formula, aims to determine capacity of a railway line by calculation of expected waiting times. In literature it has been referred to multiple times (see e.g. [78], [108] and [55]). The STRELE formula is implemented in software tools such as LUKS®[96] , which is the standard tool for capacity calculation in Germany. Besides this, it takes less computation time to determine the capacity of a railway line compared with simulations.

3.3.2. Optimisation

Optimisation methods are generally heuristic algorithms possibly based on mathematical programming tools, with the purpose of finding an optimal timetable. Often the throughput of a section will be maximised, considering constraints such as train mix, vehicle circulation, infrastructure layout and safety principles. Liao et al. [57] state that most optimisation approaches focus only on infrastructure resources. On the contrary, they include also candidate train services and fleet size to find a saturated timetable with maximum transportation performance running in a unit of time. A limitation in their current model is that station capacity is not included. There is a possibility to include this in a future research, as also vehicle maintenance constraints.

Jovanović et al. [43] focus specifically on railway station design analysis and capacity determination. For a given station layout, they determine the infrastructure occupation time based on the execution of all predicted train routes, and on the minimal follow-up times between incompatible routes. From a case study, they concluded the effects of the different conceptual designs were immediately noticeable, even in the case of very small changes in station design. The UIC 406 method (see section 2.2) was used to verify the obtained results. The results indicated that the occupation time rate could be reduced by 8% by adopting the most favorable design for a future station.

3.3.3. Simulation

Simulation methods are intended to provide a model as close as possible to reality in order to i.a. validate a given timetable, verifying feasibility and analyse robustness. Often simulation software is applied, such as RailSys [76], FRISO [64] and EGTRAIN [72]. Goverde et al. [30] used stochastic simulation in combination with the timetable compression method to propose a new concept of dynamic infrastructure occupation. The concept was applied to make a comparison study between the legacy Dutch NS'54/ATB signalling system (current and optimized situation) and ETCS Level 2 (current and shorter block lengths).

Vieira et al.[97] show how a circulation planning tool can aid railroads fully utilize their infrastructure in order to improve capacity utilization. Their simulation approach took just seconds to run, resulting in a much more powerful tool to assess capacity limits. The planning tool quantifies the maximum operational capacity and returns the throughput of the infrastructure and related efficiency.

On the contrary to most capacity assessment methods which are on a microscopic level, Lindfeldt [58] developed a model on a macroscopic level. He argues that for long term capacity planning microscopic simulation may represent a level of detail that is not necessary. Macroscopic simulation needs less computation time, so that may be more suitable for these cases. The model, where the core consists of a scheduler and dispatcher, considers the infrastructure and rolling stock on a macroscopic level, while timetables and perturbations have a more detailed representation. For the simulation of

Table 3.1: Literature overview of approaches to assess capacity in railways

Author(s)	Year	Method	Output
Goverde et al.	2013	Simulation	Capacity occupation
Lindfeldt	2015	Simulation	Macroscopic simulation model
Rotoli et al.	2016	Analytical	Practical capacity
Bešinović and Goverde	2018	Analytical	Capacity occupation
Vieira et al.	2018	Simulation	Theoretical capacity
Weik et al.	2020	Analytical	Capacity occupation
Widyastuti and Budhi	2020	Analytical	Theoretical capacity
Jovanović et al.	2020	Optimisation	Theoretical capacity
Bychkov et al.	2021	Queuing	Train dispatch
Diaz de Rivera and Dick	2021	Rail Fundamental Diagram	Impact of bottlenecks
Liao et al.	2021	Optimisation	Transportation performance

the timetables, the model uses a Monte Carlo approach. In the validation RailSys was used reference. Lindfeldt concluded that, despite the simplicity of the model, it is accurate enough to use it in a capacity analysis.

The framework of Jensen et al. [40] determines the capacity consumed by a set of trains and can be used in the strategic planning phase. The framework extends and improves the UIC 406 method because (1) there is no predefined timetable needed, only the service intention, (2) it can handle networks, lines and line sections and (3) it estimates the needed buffer times to achieve a robust capacity utilisation. In later (tactical) planning – when the timetable is more certain – it is able to calculate the capacity consumption more precise than the UIC 406 method. However, more calculations are needed, so when there is no model available to automate the calculations, the 406 method would be more suitable.

3.3.4. Other approaches

Besides capacity assessment methods which can be categorized as analytical, simulation or optimisation, it is worth to highlight two other specific approaches: Queuing theory and the Rail Fundamental Diagram (RFD).

Although in transport industry optimisation models are most popular, Bychkov et al. [13] argues that they are not always suitable to apply. If there are significant random factors in the system - which can be the case in railways - one has to deal with a stochastic optimization problem. Such problems are too complex to be solved and do not always allow to obtain meaningful results for applications. In these situation, queuing theory models can be considered. Bychkov et al. [13] build on their earlier work [54, 107] and propose a new method of modelling micro-logistics transport systems based on the queuing theory. The potential of this model is in the field of multimodal transport hubs, freight railway stations and marshalling yards. However, as they pointed out, it is not suitable for well-scheduled processes, such as unimodal public transportation or corporate cargo stations, where the influence of random factors needs to be minimized. In other works queuing theory has been used for stations [99], Marshalling Yards [21] and junctions [79].

With the development of moving block signalling systems, possibilities arise where car-following models can be implemented in railways. Originating from the fundamental diagram in road traffic (see section 3.4.2), the Rail Fundamental Diagram (RFD) is a relative new concept. In such a diagram the density of traffic [veh/km] will be plotted against traffic flow [veh/h]. One of the first RFDs was drawn by Sogin [85]. Corman et al. [17] used simulation to create RFDs for a line with moving block, discussing how different aspects such as train length, weight, and speed affect the shape of the curve. Building on that, Diaz de Rivera and Dick [19] showed the effect of vertical grades, which highly changes the braking distances. Besides, they showed how much more the flow (Trains/hour) can be with a higher density (Trains/kilometer) for Moving Block compared to multi aspect signalling. Since this is a relative new concept, there is still a lot more to investigate, such as the effect of station capacity on RFDs and how this method relates to other capacity assessment methodologies. A possible application of this method could be to verify the potential capacity of Automatic Train Operation (ATO) [17].

3.4. Data-driven methods in other transportation modes

Besides looking into existing literature in railways, we might learn something from data-driven approaches in other transportation modes. Although differences exist between transportation modes, such as data sources, travel purpose of the transportation modes, and research gaps and other problems, it would be self-righteous to only look at knowledge that exists within the railway industry. In this section a look is taken into data sciences, methods and approaches in aviation, road traffic and maritime transport.

3.4.1. Aviation

In aviation, data science has become a very popular topic in the last decade. Where in 2009-2012 only a few studies were in data science, in 2018 more than 400 works were published related to ”data science and analytics in aviation” [16]. Chung et al. divided these studies in four respective areas: big data, forecasting, machine learning and air logistics.

The term Big Data refers to the big volume of the data sets in large numbers, their variety, and the velocity requirements of the provision of the data [12]. These studies typically describe the (type of) data, rather than the application of it. Forecasting with data is one of the latest prevalent applications [15]. However, the goal can vary a lot: fuel usage and emission [14], runway incursions [86], delay and demand forecasting [15]. Machine learning can be applied to discover and extract meaningful patterns from large datasets [12]. Tong et al. proposed a deep learning method to predict aircraft speed during landing, which is more accurate and effective than the state-of-the-art method. Last but not least is air logistics. Examples can be found in a wide variety, such as the design of runways [9, 38, 86], decision making of the air traffic controllers [6, 74], and aviation maintenance risks [63]. For the future, Chung et al. see four important research directions with data in aviation: Blockchain, Industry 4.0 [105], sharing economy and multi-method analysis.

Since a small error in aviation can lead to a fatal accident, a large number of sensors are embedded to increase safety. These sensors produce large amount of data, which results also in a lot of different data sources that can be used for research. First of all, there is the Flight Data Recorder (FDR), which is mandatory in all aircraft [14]. Besides the FDR, most airplanes have also a Quick Access Recorder (QAR), which records most of the parameters during a flight, but is - in the contrary to the FDR - not designed to survive a crash [90]. But also on the ground there are recorders, for example to the open source trajectory recorder ADS-B [89] or meteorological data [9, 69]. An overview with examples of used data sources and methodologies are given in table 3.2.

In aviation, typically, runway capacity is the most stringent constraint on growth, ahead of airspace, aircraft parking or terminal capacity [22]. Because runway expansion is often opposed by (local) government, as they are likely to limit pollution and noise, research is focused on how to maximise capacity of existing runways. Runway capacity can be defined as a number of aircraft movements per defined time period. On one hand theoretical capacity does not take delay into account, on the other hand the practical capacity does take delay into account, whose value depends on the level of service that is aimed for [44]. To estimate the maximum runway capacity, there are three relevant factors: The wake turbulence categories, the levels of automation support and the Runway Occupation Time (ROT) [82]. The wake turbulence categories standardise the distance between preceding and succeeding aircraft in the air. The levels of automation support have an influence on the safety margin. In aviation different support systems exists, which have their own safety margin, depending on the error they have in measuring the location of the aircraft. The ROT is the time between the touchdown and the moment the aircraft leaves the runway, which depends on the runway exit. It is interesting to see that these three factors more or less also exist in railways: the wake turbulence distance is comparable to the braking distance, the automation support system to the signalling system and the ROT to the Blocking times. Sekine et al. [82] put these three factors in an AirTOp model to perform a data-driven simulation and analysis. In their research they concluded how these three factors affects the en-route and terminal delay time.

On the ROT a relative simple analysis was performed by Kumar et al. [49]. They used surface (trajectory) data to calculate the Runway Occupancy Time (ROT). For future work, they would like to include weather data and cover different seasons to capture the effect of wind and rain on the ROT. Later, Herrema et al. [33] included the weather effects. They provided a machine learning approach to predict the runway exit utilisation based on actual movements at airports. They used a variety of

Table 3.2: Literature overview of data-driven methods in aviation

Author(s)	Year	Data source	Methodology	Goal
Ramanujam and Balakrishnan	2015	Operational data	Discrete choice modelling	Quantifying the utility function of the air traffic controllers
Chati and Balakrishnan	2014	Flight Data Recorder (FDR)	Statistical analysis	Estimate operational values during landing and take off
Ghalebsaz-Jeddi et al.	2009	Multilateration surveillance system data	Statistical analysis	Analysing landing capacity of a runway
Tong et al.	2018	Quick Access Recorder (QAR)	Deep learning algorithm	Aircraft landing speed prediction
Sun et al.	2016	Runway ADS-B Data	Statistical analysis	Estimate aircraft mass
Jackson et al.	2015	Satellite imagery	Remote sensing	Extracting runway geometry
Baspinar et al.	2016	ALLFT+ (historical air traffic)	Queuing network model	Analyse delay propagation and capacity drop
Kumar et al.	2009	Track surface data	Statistical analysis	Calculate ROT
Herrema et al.	2019	Various (a.o. Radar and wind)	Machine learning	Predict runway exit and ROT
Sekine et al.	2021	Trajectories (Simulated)	Statistical analysis	Calculate runway capacity
Levy et al.	2004	DROMS	Statistical analysis	Estimating runway capacity

data sources to have in total 15 input variables, giving information about the trajectory, runway design and weather circumstances. Their output was a model which could predict if an aircraft would take a procedural or non-procedural runway exit and therefor a flight will experience a longer ROT or not. On a macroscopic level, Baspinar et al. [8] used a queuing network model to analyse delay characteristics of European air traffic. They found critical capacity values for different airports, which are tipping points between stable operations and unstable operations where the capacity drops and delays in the network occur.

3.4.2. Road traffic

In road traffic there are in general two different types of technology to collect traffic data: ”in-situ” technologies and Float Car Data [53]. ”In-situ” technologies refer to traffic data measured by detectors located along the roadside. This type of technologies can be split into two categories: intrusive and non-intrusive methods.

The intrusive methods basically consist of a data-recorder and a sensor placing on or in the road, for example pneumatic road tubes, piezoelectric sensors and magnetic loops. These sensors can be used both for real-time traffic information or be saved in a data-base and used later. Non-intrusive techniques are based on remote observations, such as manual counts, infra-red (passive and active), passive magnetic sensors, microwave radar, acoustic sensors and video image detection. This is a big difference with railways, where the numbers of trains are known before operations start.

Float Car Data is a principle which collects real-time traffic data by locating vehicles via mobile phones or GPS over the entire road network. For example, this can be used by a taxi or delivery company which tracks their vehicles [106]. However, it is also used by navigation companies, such as TomTom and Google Navigation. Google uses position data of mobile phones running on Android and uses this in Google Maps to give live traffic information [42].

Besides live traffic information, traffic data can also be used to calculate road capacity or optimize intersection signalling. With Edie’s definition [23] of flow, density and speed for an area in space and time, the data can be configured to a fundamental diagram [31] (see Appendix A). Key in the fundamental diagram is that *flow* is equal to *density* times *average speed*: $q = k \cdot u$. Since it has three different variables, the fundamental diagram can be draw in three different planes: flow-density, speed-flow, speed-density. Knoop and Daamen [47] proposed a method to automatically fit a fundamental diagram through simulation and real-world detector data.

3.4.3. Maritime transport

In maritime transport the Automatic Identification System (AIS) is a great resource of data. Since 2004, all passenger ships and ships over 300 gross tonnage have been fitted with AIS transponders [34], and since 2014 this system became compulsory for all fishing vessels of length above 15 meters in the EU [68]. In maritime transport it is applied as a navigation system to improve safety on waterways, but it is also a great source for research. An AIS terminal transmits marine traffic data in both static (e.g. length, width, type and MMSI number of the ship) and dynamic data (e.g. timestamp, position, speed and course). Other vessels and coastal authorities can pick up these signals, display it in their navigation system and manoeuvre safely through ports and waterways.

Liu et al. [60] gave an overview of studies where AIS data is used, for example marine safety, traffic management, sustainability, ship grounding and near-miss detections. Xin et al. [104] used AIS data as input of a microscopic simulation model for ship navigation in the ”Xiazhimen” waterway. Wang et al. [98] propose a data-driven approach for a lock scheduling problem, which is a typical optimizing and decision-making problem.

However, using AIS data is not without risk. AIS messages are vulnerable to manipulation and subject to hacking [75]. It is known that vessels involved in illegal activities commit identity fraud (false shipping numbers), have obscured destinations or manipulate the GPS [102]. Besides this, 50% of AIS static data transmissions have errors of any kind [102]. Of the dynamic data, 27% of ships do not transmit data at least 10% of the time (’go dark’).

Besides AIS, ships are also obligatory to have a Long Range Identification and Tracking (LRIT) system onboard. The output is government centring – in Europe centralized via the EU LRIT Cooperative Data Centre (CDC) [2]. The essence of the LRIT system is that participating countries get information, at least every 6 hours, on their own flagged ships wherever they are in the world [51]. Since the signal frequency is much lower compared to AIS, data is used of a longer time period and larger study area. Alessandrini et al. [2] used LRIT data to estimate shipping emissions on the waters around the Europe continent. Two years later, Alessandrini et al. [3] proposed a model to estimate the time of arrival in ports using LRIT data. In their case study they used shipping data from the Mediterranean Sea. Vespe et al. [95] did a statistical analysis on piracy on maritime transport in the Indian Ocean, using LRIT data of 5 years from the EU database.

To the knowledge of the author data-driven capacity assessment methods are not (much) used in maritime transport for ships. An extensive search for papers and articles, with keywords – including variations – like ”data-driven waterway capacity”, ”data-driven berth capacity” or ”data-driven port capacity” didn’t result in literature that was wanted. It could be that other methods are more suitable to solve the capacity problems in maritime transport. In the process of searching for literature, articles were found about capacity issues at container terminals. Mar-Ortiz et al. [62] tackled this with a data-driven method. However, this is another type of capacity problem than in railways, so for this research it is not very suitable.

3.5. Conclusion

The timetable compression method is arguably the most widely used method to assess rail capacity. Since it was published in 2013, multiple shortcomings and limitations are addressed, but also solutions are proposed. It can be concluded that it is up to IMs experience on how to interpret the guidelines presented in Code 406. This mainly concerns selecting (train path) line sections, occupancy times rates and additional time rates. Besides the timetable compression method, there are other approaches to assess capacity. These can be divided into analytical, optimisation and simulation approaches.

Other transportation modes show a couple of opportunities which can be used in the railway industry. In aviation sensors are widely used, which create big data sources for research. This research has a wide range of purposes, such as forecasting of delay and demand, fuel usage and runway incursions. To analyse the Runway Occupation Times and capacity mostly trajectory data has been used. In road traffic the fundamental diagram is an important tool to assess capacity. This gave Song et al. and Diaz de Rivera et al. the idea to introduce this concept in railways. Also in maritime transport there is trajectory data available, but it is less used for capacity analysis.

4

Methodology

In this chapter the methods are explained that are used in this research. First, in section 4.1, a mathematical approach to estimate blocking times for moving block signalling is described. Second, in section 4.2 a method is defined which should determine bottlenecks based on the calculated blocking times.

4.1. A mathematical approach to assess moving block track occupation times

In figure 4.1 the model framework of this research is presented. In this section a mathematical approach will be described to estimate blocking times for a moving block signalling system. This model, in figure 4.1 presented as function f, uses three input datasets: infrastructure characteristics, rolling stock parameters and planned time-distance data. Out of these datasets four input variables for the model can be extracted – the position (x) and speed (v) of the train, the deceleration capabilities of the train (a), length of the train (L) – and a couple of fixed input parameters (β), which are the length of the setup, reaction and release time.

The output of the model will be the blocking times for Moving Block (MB). An important difference between fixed- and moving block is that with a MB signalling system (in regular operations) fixed blocks do no longer exist, so one can not speak about the occupation or blocking times of blocks with a MB system. However, when the term 'blocking time for moving block' is used in this thesis, it is meant 'the time between a movement authority request and release time for a geographical location or infrastructural element'.

As explained in section 2.1 the blocking time consists of six time components: route setup-, (sight and)

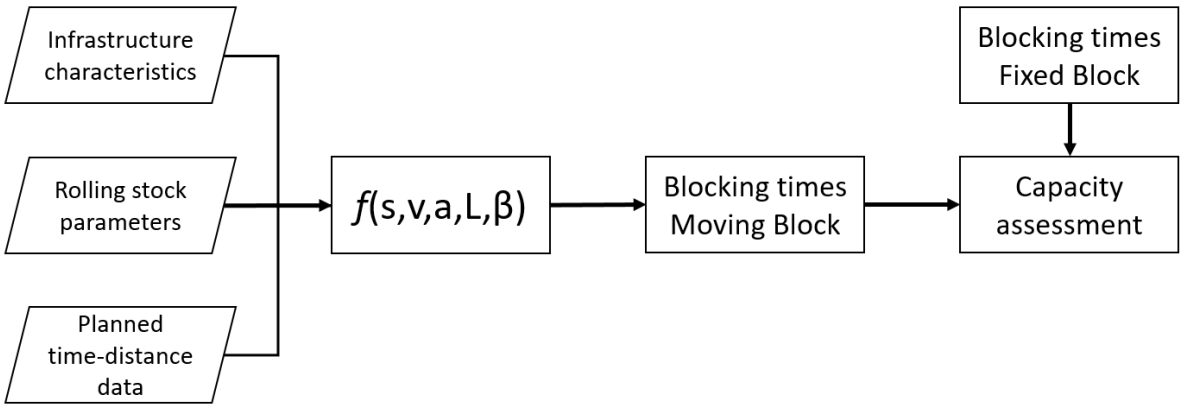


Figure 4.1: Model framework, with s:position, v:speed, a:deceleration, L:Length of the train, β:fixed parameters

reaction-, approach-, running-, clearing- and route release time. The setup-, reaction- and release time are independent of the behaviour of the train and are constant over time for a given block/location. The setup and release time depend on the characteristics of the signalling system and other safety margins. When the train passes a switch, additional setup time is needed to check the switch' position and if needed to shift and lock the switch. The reaction time mainly depend on the reaction time and the level of attention of the train driver. In this model these three time components are fixed input parameters. The other three components – approach-, running- and clearing time – are dependent on the behaviour of the train. Because they are independent of each other, they can be modelled all three individually. In the following subsections the calculations of these time components are explained for a certain location b.

Approach time

Büker et al. 2019 define the approaching time for in-cab signalling as "the time the train runs through the indication distance that is signalled by the cab signal" [11]. For Moving Block the indication distance is equal to the absolute service braking distance. With a constant deceleration, the braking distance can be defined with:

$$s(t) = \frac{1}{2}a\Delta t^2 + v_0 t \tag{4.1}$$

where: t = Time period
a = deceleration (see eq. 2.3)
v_0 = Speed before the train brakes

It is known for which location the approach time should be estimated – for Moving Block it is the End of Authority (EoA), but the length to brake to that location is unknown. Therefore, the model will build a braking curve from the EoA location backwards. The braking curve will be made until it reaches the free flow speed of the train. However, this cannot be done with a constant deceleration rate. As explained in section 2.3, the deceleration depends on the braking characteristics of the train and the gradient of the track:

$$A(s) = A_{train} + A_{gradient}(s) \tag{4.2}$$

$$A_{gradient} = g \cdot \sin\big(\arctan(\alpha)\big) \approx g \cdot \alpha \tag{4.3}$$

where: A_{train} = braking rate of the train [m/s^2]
g = gravitational force (9.81 m/s^2)
α = gradient at s [m/m]

The braking rate of the train is constant during braking, but the track gradient may change over the route and hence affect the actual distance needed by the train to brake. Therefore the deceleration is not constant and equation 4.1 cannot directly be used. The braking curve will be built in multiple steps from the EoA location, backwards in time and distance.

Let I be the trajectory dataset of a certain train – gathered by simulation or real-time operations – and let $i \in I$ be a logging (datapoint) of that dataset. From that logging the following information is known: time (T_i), location (S_i), speed (V_i) and possible deceleration that the train can perform from that location (A_i) until it reaches the next logging (A_{i+1}). The deceleration is calculated using eq. 4.2. The location for which the approach time should be calculated (b) is defined as s_{EoA}, where s_{EoA}. Let $\mathbb{W}$ be a set of whole numbers, [0,…,$i-1$]. When $S_i = s_{EoA}$ and the braking curve will be build, index $p \in \mathbb{W}$ will be used to run backwards through I from i, notated as I_{i-p}. This all has been visualised in figure 4.2.

Let Δt be the length of a timestep [s] for which the braking curve will be constructed, n the number of timesteps and v_n the speed [m/s] at the end of the braking curve that is constructed until then. For

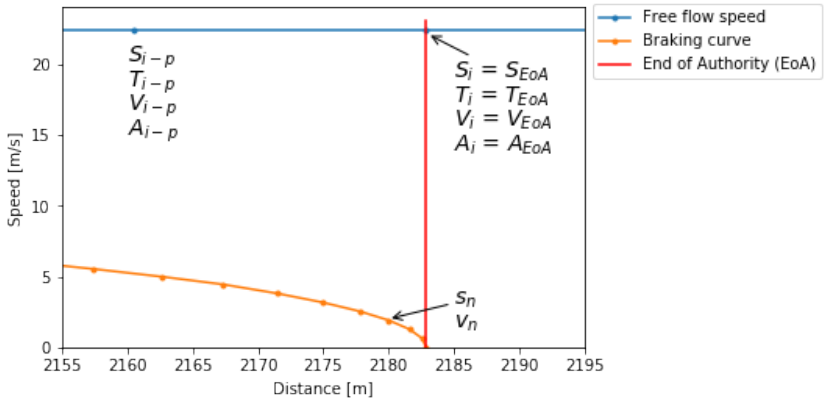


Figure 4.2: Speed-Distance diagram with the braking curve zoomed in on the EoA (zoomed version of figure 4.3a)

every timestep, the model starts with a check to see if the braking curve has reached a higher speed than the free flow speed:

$$V_{i-p} \leq v_n \tag{4.4}$$

When this is true, the model breaks out of the loop (equations 4.4-4.8) and starts to calculate the approach time (eq.4.9). When it is false, it will continue. The braking curve is constructed from s_{EoA} backwards per timestep Δt with:

$$s_n = -\frac{1}{2}A_{i-p}\Delta t^2 - v_{n-1}\Delta t + s_{n-1} \tag{4.5}$$

$$v_n = A_{i-p}\Delta t + v_{n-1} \tag{4.6}$$

where: s_n = Location of braking curve [m], with $s_0 = s_{EoA}$
v_n = Speed of braking curve at s_n [m/s], with v_0 = 0 m/s
A_{i-p} = Maximum possible deceleration between S_{i-p} and S_{i-p+1}

Since the braking curve is constructed backwards, the deceleration has a positive sign in equations 4.5 and 4.6. Equations 4.4 until 4.6 will loop as long as:

$$S_{i-p} \leq s_n \tag{4.7}$$

When this is true, V_{i-p}, S_{i-p} and A_{i-p} will be updated by increasing p by 1. Then the model will continue at eq. 4.4. For example, in figure 4.2 s_0 to s_8 will be calculated with A_{i-1}. $A_{i-1} > s_9$, so p will be increased by 1 and the model continues at eq. 4.4. In an event-driven dataset (simulated or real-time), the time gaps between $i-p$ and $i-p-1$ can be relative large compared to a time-driven dataset. So when an event-driven dataset is applied, V_{i-p} will be changed to:

$$V_{i-p,m} = V_{i-p} - A_{i-p}\Delta t \cdot m \tag{4.8}$$

where m is the number of loops between $i-p$ and $i-p-1$, starting at 0 every time V_{i-p} is updated. An example is given in section 6.4

Once 4.4 is true, the approach time can be estimated by:

$$t_{approach;b} = T_{EoA} - T_{i-p} \tag{4.9}$$

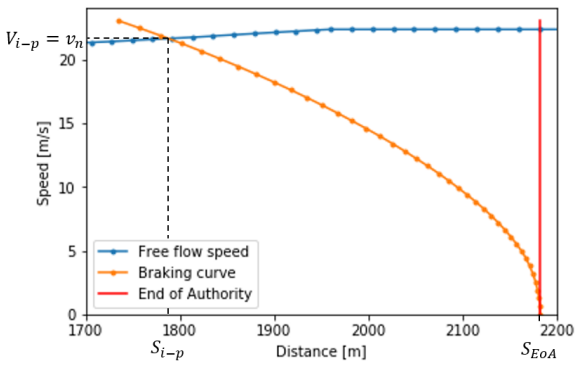
or in case of an event-driven dataset:

$$t_{approach;b} = T_{EoA} - T_{i-p} + \Delta t \cdot m \tag{4.10}$$

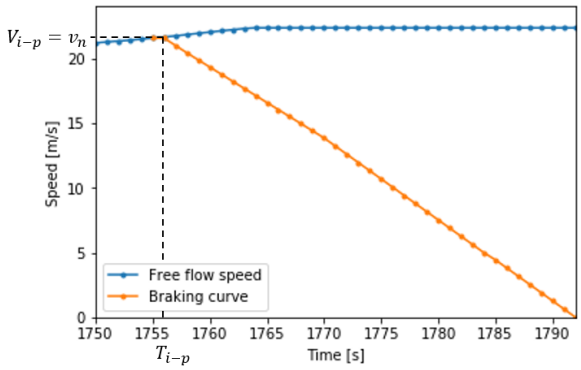
Although the indication point, where $V_{i-p} = v_n$, could be between $i-p$ and $i-p-1$, it is chosen to not add a correction factor. Without a correction factor, the model will have the same precision as the

input trajectory dataset. For example, when the dataset has a time interval of 1 second, the approach time will also have a precision of 1 second. Also, during the iteration process of defining the model and the verification (see chapter 5), it was found out that the current described model has a high precision and accuracy, so it is not needed to add a correction factor. For a event-driven dataset this is different, because there is a variation in time intervals between datapoints. To overcome this, eq. 4.8 was added to the model and $\Delta t \cdot m$ was added to the approach time. This way the aimed precision is equal to Δt.

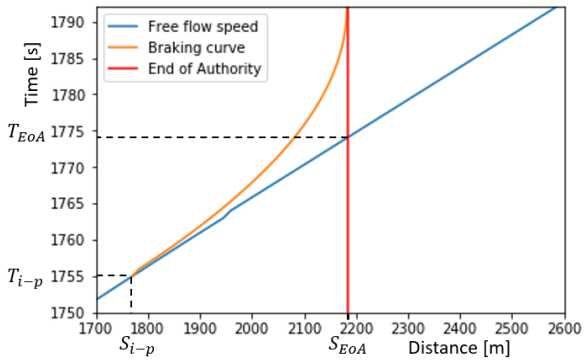
In figure 4.3 an example is given of a complete braking curve in a speed-distance, time-speed and time-distance diagram. The blue line is the free flow trajectory of the train before it reaches the End of Authority at 2180m. In this example, the indication point is just before 1800m. For the train it takes 18 seconds to run through the indication distance. When the train doesn't get permission to pass the EoA point, the train will brake at the indication point and reach the EoA point 37 seconds later.



(a) Speed-Distance diagram



(b) Speed time diagram



(c) Time Distance diagram

Figure 4.3: Construction of the braking curve in different diagrams

All in all, to calculate the approach time, these input variables are needed:

- Braking characteristics of the train (A_{train})
- Gradient of the tracks (α)
- Trajectory data of the train (S, T, V)

Running time

The running time is the time needed for a train to run through a block. However, there are only (virtual) blocks at specified locations. On the rest of the line blocks do no longer exist. For the calculation of the running time, there are three different situations:

1. On a normal track line, the running time is equal to the time period between two requests for MA (see section 2.1). The exact length depends on the type of system;

2. When b is in a virtual block (see section 2.1), the running time is the time it takes to run through the virtual block;
3. When a train stops, e.g. at a station, the running time is equal to the time it stands still (v = 0).

For virtual blocks, the following will be applied: Let B be a set of virtual blocks that a certain train passes and let $k \in B$ be the index to indicate a block. For each block, the train has an entrance- and exit location and -time: $l_k^{entrance}$, l_k^{exit}, $T_k^{entrance}$ and T_k^{exit}. Given that $l_k^{entrance} \leq b \leq l_k^{exit}$, the running time will be $T_k^{exit} - T_k^{entrance}$.

Mathematically this can be written as:

$$t_{running,b} = \begin{cases} 1/f & ;\text{For regular MB operations, depending on system characteristics} \\ T_m^{exit} - T_m^{entrance} & ;\text{For virtual blocks, } l_k^{entrance} \leq b \leq l_k^{exit} \\ t_{standstill,b} & ;\text{Standstill time at } b \end{cases} \tag{4.11}$$

where: f = Frequency of MA requests per second
b = Location for which the blocking time needs to be calculated
k = index of a block

Clearing time

In a fixed block situation the clearing time is the time it takes to completely leave a block. For moving block it is the time it takes for a train to cross a point in its full length. Hence the principle is not different compared to fixed block, but the location for which the clearing time should be calculated is different. Therefore an estimation model is needed. When a train runs with constant speed, the clearing time can be calculated by:

$$t_{clearing;b} = \frac{L}{v_b} \tag{4.12}$$

where: L = Length of the train [m]
v_b = Speed of the train at b [m/s]

However, this would lead to a large mis-estimation around stations, because at these locations trains accelerate/decelerate over a longer distance. Therefore, when the loggings of the trajectory dataset have a short interval, e.g. one second, the clearing time should be estimated by:

$$t_{clearing;b} = T(b + L) - T(b) \tag{4.13}$$

where: b = Location for which the blocking time wanted to be known [m]
L = Length of the train [m]
$T(b)$ = Time when the train has reached b [s]
$T(b + L)$ = Time when the train has reached b plus L [s]

Note that the trajectory dataset is not continuous, but discrete. So to estimate the clearing time, let $j \in \{0, \dots, \mathbb{W}\}$ and $S_i = b$. Location b is cleared when:

$$S_i + L \leq S_{i+j} \tag{4.14}$$

Starting at $j = 0$, j will be increased by 1 until this is true. When it is true, the clearing time can be estimated by:

$$t_{clearing;b} = T_{i+j} - T_i \tag{4.15}$$

Blocking time

The blocking time of a location or block on the track can be calculated by summing the length of each time component:

$$t_{block;b} = t_{setup} + t_{reaction} + t_{approach;b} + t_{running;b} + t_{clearing;b} + t_{release} \tag{4.16}$$

The start- and end time of the blocking time can be calculated by:

$$t_{Start;b} = T_{Passage\ time;b} - t_{setup} - t_{reaction} - t_{approach;b} \tag{4.17}$$

$$t_{End;b} = T_{Passage\ time;b} + t_{running;b} + t_{clearing;b} + t_{release} \tag{4.18}$$

Once all input variables and datasets are collected, the model can be executed. In the appendices C and D the python code can be found which was used to apply to model to the verification and case study.

4.2. Using track occupation data to identify capacity bottlenecks

In this section two methods to identify capacity bottleneck are described. Both methods uses track occupation data, for example the blocking times calculated in the previous section. In chapter 5.4 the methods will be verified.

4.2.1. Summing blocking times

As alternative for the timetable compression method, which compresses the blocking time stairways of train paths of a line section, this method sums the blocking times of a block over a defined period of time. The goal of this method is to find blocks in a network which have a high blocking time. Tracks with higher blocking times are likely to be capacity bottlenecks which require attention if service capacity need to be increased. Mathematically the method can be written as:

$$\tau_B = \sum_{i=1}^{n} t_{i,B} \tag{4.19}$$

where: i = Train *i*
n = Total amount of trains during passing a period of time
$t_{i,B}$ = Blocking time of train i at B
τ_B = Summed blocking time

This method can be applied for both fixed- as moving block signalling system. Depending on the system, B is a block, infrastructural element or location in the network.

The method will be explained via an example. In figure 4.4 a blocking time diagram is given, which is equal to the diagram used in the verification study in chapter 5. This study is about a portion of the South West Main Line between London Waterloo and Surbiton, passing Clapham Junction, Wimbledon and Raynes Park. This line has been modelled in EGTRAIN using the British fixed-block AWS/TPWS three aspect signalling. More details about the model and the example study can be read in chapter 5.

The method will sum all the blocking times in a time period for each block. In this example there are twelve trains, so each block will be occupied twelve times for a short period of time i.e. when a train passes by. The output is the summed blocking time of each block during one hour.

When the occupation times are summed of all the trains simulated in the reference period, T_B can be plotted in a bar graph for all blocks B, such as figure 4.5. In figure 4.5a the summed blocking times are plotted against the distance of the route. In this example all peaks are at stations where the trains stop. In figure 4.5b the bars are sorted based on the occupation times, with the most occupied block of the line on the left side. To keep the BlockID's readable, only blocks with an occupation time higher than 1000s are shown.

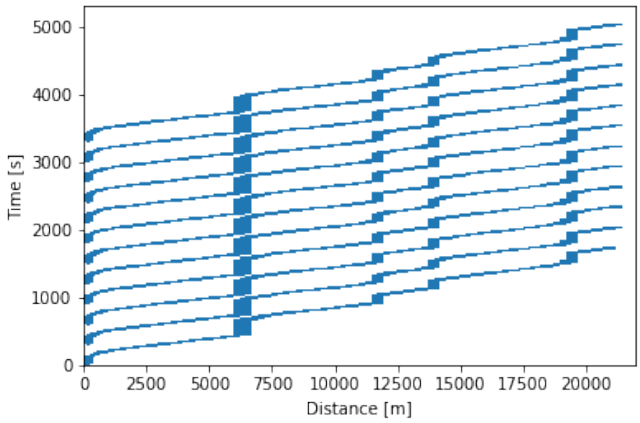
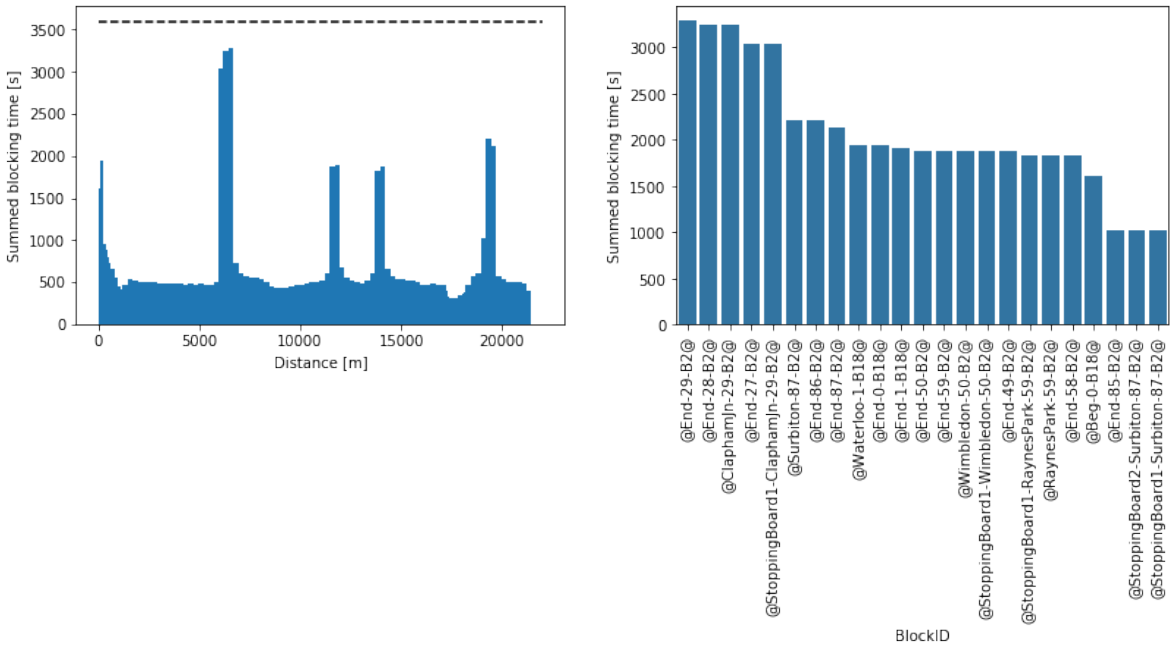


Figure 4.4: Blocking time diagram of a fixed block signalling system



(a) Sorted by distance

(b) Sorted by occupation time

Figure 4.5: Summed occupation times of figure 4.4

Looking at the summed blocking time has the following advantages:

- There is no boundary definition needed. The study area can be a whole network or just a line section;
- Practitioners do not need to pass through the often time consuming process of compressing the timetable by shifting train paths, which usually requires the support of dedicated software tools.
- The method results in a quick overview of the most critical locations or infrastructure elements;
- The train order is not needed to know, only the frequency and trajectories.

Besides this, the method has also a couple of limitations:

- One cannot see if there is room for an extra trainpath in the schedule.
- Just as almost any other capacity assessment method, the method strongly depends on the infrastructure layout. For example, one cannot compare a one-directional railway line compare to a two-directional railway line.

4.2.2. Buffer times

Another approach to find bottlenecks in a network would be to look at the time a track is not being occupied. In the timetable compression method blocking time stairways will be compressed as close as possible, without any overlap between two blocks. The location where two blocks touch each other, the critical block, will be the bottleneck of a line section. However, the compression is a time consuming process, so this needs to be avoided. A solution would be to find the location of the minimal time between the end of the blocking time of the leading train and the start of the blocking time of the following train.

Let J be the set of trains in a network and let $i, j \in J$ be a train. Let Q_{ij} be a set of blocks where train j is a direct successive train of i, so without any trains between them. $b \in Q_{ij}$ is used as index to indicate a specific block. $T_i^{Start}(b)$ and $T_i^{End}(b)$ are the start and end of the blocking time of train i of block b, so that a bottleneck δ_{ij} can be identified by:

$$\delta_{ij} = \min_{b \in Q_{ij}} \left(T_i^{\text{end}}(b) - T_j^{\text{start}}(b) \right) \tag{4.20}$$

One could also say that it is the headway time minus the blocking time, also known as the buffer time. Later on in this research a comparison will be made in the bottlenecks with fixed- or moving block signalling. To do this, it is expected that only looking at headway time will not be suitable, since – without compression – the headway times will not change, but the blocking times will. This might result in other bottlenecks between fixed- and moving block.

This approach can be explained with an example. In figure 4.6 a blocking time diagram of two intercity trains is plotted. Halfway, a stop train merges in and makes two stops before it splits again. On this line there are three different train combinations of a train following another train: Intercity-Intercity, Intercity-Stop train, Stop train-Intercity. This results also in three different bottlenecks, numbered in figure 4.6.

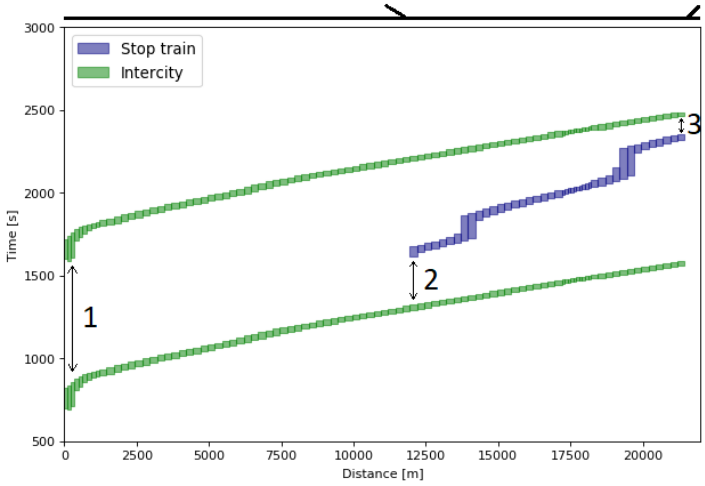


Figure 4.6: Identifying capacity bottlenecks using minimal buffer times between trains

A big advantage is that it is not needed to define boundaries and split corridors into line sections. A disadvantage is that the train order needs to be known. When also the exact timetable is known, bottlenecks can be ranked based on the length of the buffer times.

The defined approaches in this section will be tested and verified in section 5.4 with a homogeneous and heterogeneous traffic pattern

5 Verification

In chapter 4 a model to estimate the occupancy times of a moving block system was defined. In this chapter this model will be verified with a study in the microsimulation model EGTRAIN. First, a short introduction to EGTRAIN will be given, followed by a explanation of the used study area. Then the data collection and processing is elaborated. This section is closed by the results and conclusion of the verification.

5.1. EGTRAIN

EGTRAIN is a synchronous microscopic railway simulation model developed by Quaglietta [71]. It can be characterized by four general features:

1. Flexibility: The model has been designed in C++ and has an open structure, which means that inner model functions can be manipulated or further methods can be added without compromising or varying the original source code. This feature is also directly used in this verification.
2. Accuracy: in the design it was chosen for an object-oriented concept. By doing this, the model is able to produce in detail the behaviour of each railway component, such as rail vehicles, signalling equipment and infrastructure.
3. General applicability: the model is composed of four main modules: infrastructure, rolling stock signalling system and timetable (see figure 5.1). With this modular structure the model can be adapted to any case-study.
4. Computational efficiency: the model makes use of the multi-cores in computers, which makes it computational efficient and having a lower computation time.

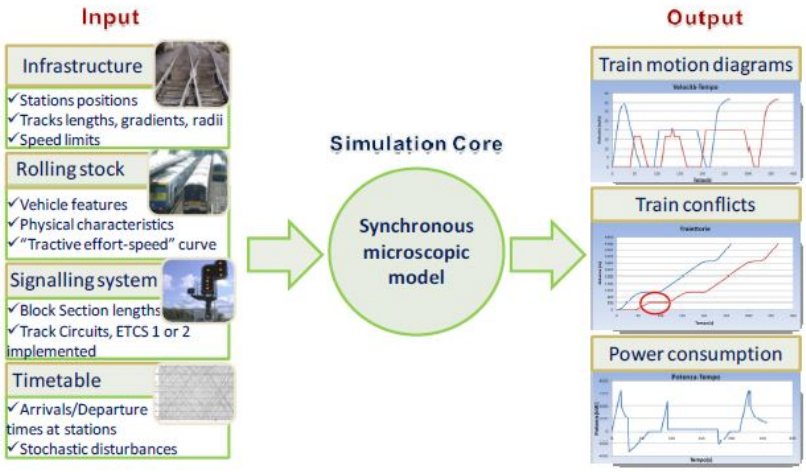


Figure 5.1: Architecture of EGTRAIN (Source: Quaglietta [71])

As input parameter, each module needs it own input files, such as infrastructure layout, train characteristics, signalling layout (e.g. block section index and lengths) and the timetable. The simulation core consists of three steps. First, for timestep t the forces on the train, the speed, position and power consumption are calculated. Second, after the speeds and position have been determined, both the signalling and the interlocking systems are updated. Third, the simulation clock goes ahead at instant $t+1$, and the simulation restarts again from step 1, until the whole simulation period has been simulated.

The microscopic model returns a database containing all information about the dynamic evolution of a train's state during the whole simulation period. For this research train data about the simulation time instant $t[s]$, train speed $v[m/s]$ and the (nose) position of the train $s[m]$ is used, but also the Mechanical power en Energy consumption is returned. The data can be printed out as a text file, so it can imported in software for data analysis.

A big advantage of EGTRAIN is that in the model fixed and moving block signalling systems are included. This means that EGTRAIN can provide track occupation data for both systems, while the rest of the model stays constant. Since the defined method in chapter 4 uses track occupation times of fixed block to estimate track occupation times of moving block, EGTRAIN can provide both the input data as the control data to verify the output of the defined method. This makes the verification much more reliable than when two different data sources (i.e. simulation tools) where used.

5.2. Verification case study setup

For the verification of the defined model a portion of the South West Main Line between London Waterloo and Surbiton has been modelled in EGTRAIN. The corridor is from Waterloo (Wtl) station to Surbiton (Sbn), with stops at Clapham Junction (CpJ), Wimbledon (Wbn) and Raynes Park (RnP).
In this verification two different scenarios are simulated. The first scenario is a homogeneous traffic situation where two different services are in operation, A2_WtlSurbiton and A3_WtlSurbiton. They follow respectively route A and B which can be seen in figure 5.2. The second scenario is a heterogeneous traffic situation and will only be used for the verification of the method to identify capacity bottlenecks. Here train A3_WtlSurbiton will be alternated by an intercity service, A4_WtlSurbiton. This train departs from London Waterloo and will drive the same route as A2_WtlSurbiton but without stopping at the other stations. The rolling stock characteristics of all three services can be seen in table 5.1. The time-distance and blocking time diagrams can be found in figure 5.3.

Table 5.1: Characteristics of the trains

Train	traintype	max. speed	braking rate	Jerk rate	Length
A2_WtlSurbiton	Stopping train	33.6 m/s	0.6 m/s^2	0.75 m/s^3	161.84 m
A3_WtlSurbiton	Stopping train	33.6 m/s	0.6 m/s^2	0.75 m/s^3	161.84 m
A4_WtlSurbiton	Intercity	55.8 m/s	0.8 m/s^2	0.75 m/s^3	93.34 m

In the homogeneous traffic situation the simulation period is 1 hour (3600s). During this period 6 trains of both services will depart alternately from London Waterloo station. In the heterogeneous traffic situation also 6 trains of both services will depart, but the simulation period here is 1.5 hours (5400s). To collect the fixed block data, the British fixed-block AWS/TPWS three aspect signalling has been used, which is very similar to the dutch ATB-EG signalling system. For the Moving block data, ETCS Level 3 Moving Block has been used. Other parameters and settings can be seen in table 5.2.

Table 5.2: Parameter and setting values

Setup time FB	3 s.
Setup time MB	0 s.
Switch setup time	5 s.
Release time FB	1 s.
Release time MB	1 s.
Reaction time	8 s.
Buffer times	0 s.
Recovery times	0 s.

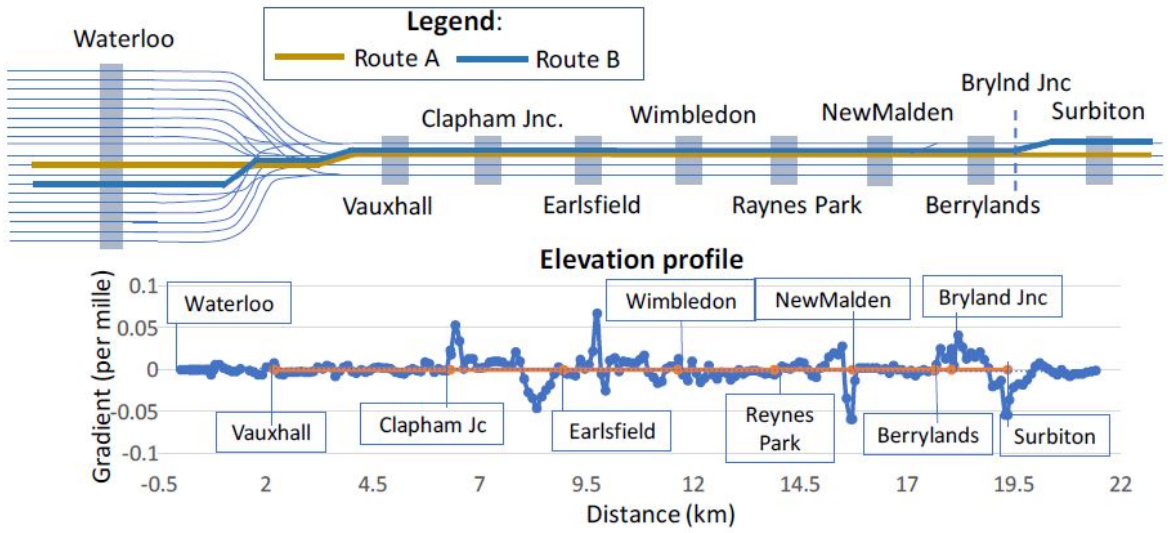
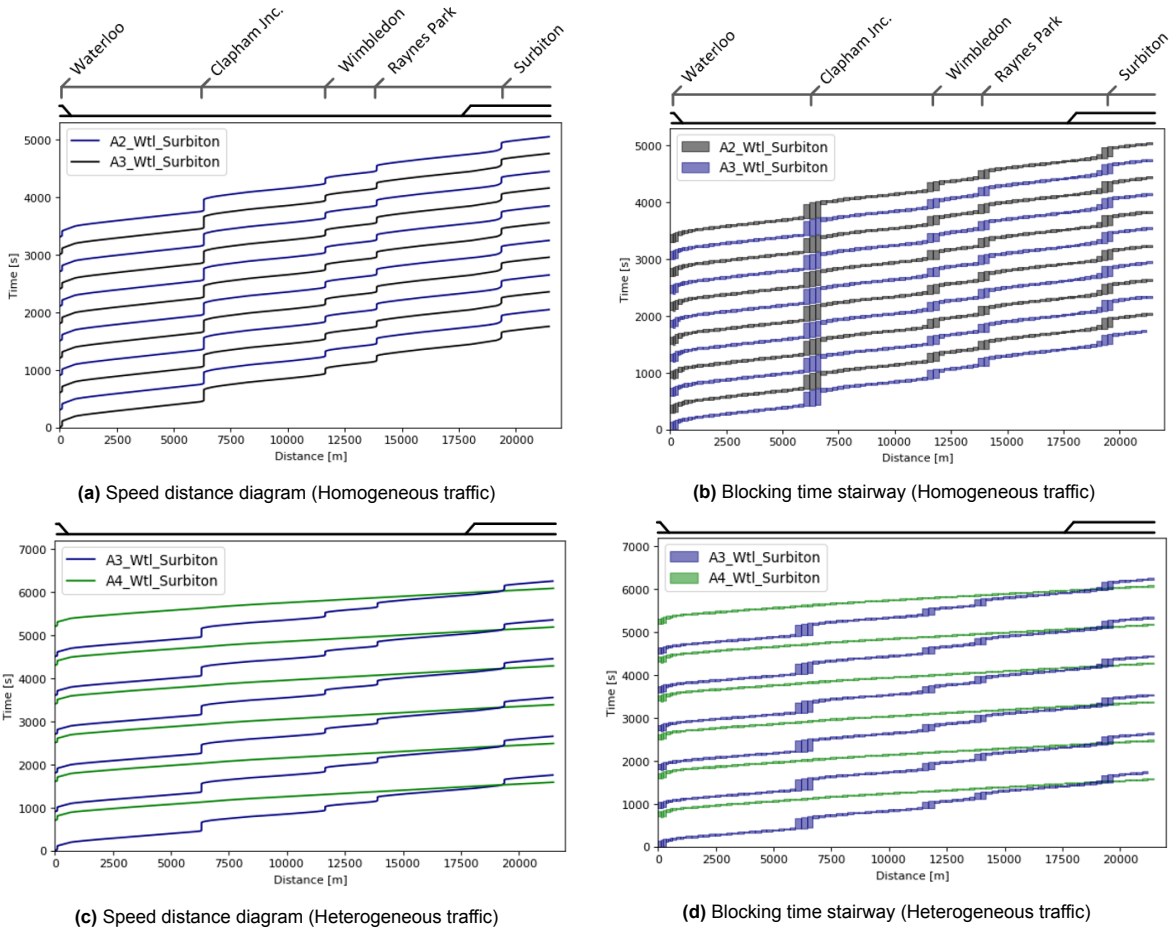


Figure 5.2: Layout and elevation profile of the Waterloo - Surbiton corridor on the South West Main Line in the UK (Source: Quaglietta and Goverde [73])



(a) Speed distance diagram (Homogeneous traffic)

(b) Blocking time stairway (Homogeneous traffic)

(c) Speed distance diagram (Heterogeneous traffic)

(d) Blocking time stairway (Heterogeneous traffic)

Figure 5.3: Trajectories of the route Waterloo-Surbiton operating with AWS/TPWS fixed block signalling

5.3. Approach to assess moving block track occupation times

5.3.1. Data collection and processing

As explained in section 4.1, the proposed model to estimate blocking times for a moving block signalling system uses three datasets: planned time-distance (trajectory)-, infrastructure- and rolling stock data.

The trajectory data is an output of the EGTRAIN simulation. For the model in this research the speed and location of the train at every timestep is needed. To collect this, a couple of extra code lines were added to the model, which can be seen in appendix B. These lines created a Time-Distance-Speed dataset of the twelve trains. As example, a piece of this dataset has been given in table 5.3.

Table 5.3: Time-Speed data set

Time [s]	Speed [m/s]	Position [m]
⋮	⋮	⋮
506	15.6478	832.858
507	15.8728	848.506
508	16.0863	864.379
⋮	⋮	⋮

The second input variable is the infrastructure dataset, which gives the gradient of the tracks. The gradient is an input variable for EGTRAIN. To make this information usable for this research, two datasets are needed: a link database (table 5.4) and a node database (table 5.5). A trackline, which is used by the train, consists of multiple links. These links are connected by nodes. By merging the node database with the link database, it is known how long each link is. This can be merged with the trajectory dataset, so it can be determined under which gradient the train drives for every timestep (see appendix C).

Table 5.4: Example of the link dataset

LinkID	Start node ID	End node ID	Curve radius [m]	Gradient [-]	Max. Speed [m/s]
⋮	⋮	⋮	⋮	⋮	⋮
128	29	30	10000	-0.0012	26.8224
129	30	31	10000	0.0011	26.8224
130	31	32	10000	-0.0007	17.8816
⋮	⋮	⋮	⋮	⋮	⋮

Table 5.5: Example of the node dataset

NodeID	X[km]	Y[km]
⋮	⋮	⋮
29	1.307592	0
30	1.408176	0
31	1.609344	0
⋮	⋮	⋮

Last but not least is the information of the rolling stock. From the rolling stock the length of the train and the service braking rate are needed. In this verification study both A2_WtlSurbiton and A3_WtlSurbiton service use the same rolling stock. However, this model should be suitable to be used for different kind of rolling stock, so these will be used as input variables.

The output of the model is an estimation of the different time components (approach, clearing and running time) of the track occupation under moving block. Remember that the model could estimate the blocking times (and each time component) for any location a train passes. It is chosen to calculate to the blocking times for every geographical location a fixed block would have started, since EGTRAIN will

do the same. Now, the same corridor will be simulated with EGTRAIN, but for a moving block signalling system – in this study ETCS L3 Moving Block. This gives the length of each time component in a simulated environment. The time components of the estimated and simulated results can be compared in a scatterplot.

Example: The third A2 train departs at 1604s at Waterloo Station. During its journey it passes block 15, starting at 2699.93m. The estimated approach time of this block for moving block is 20,59s. The simulated approach time is 20s. This creates one datapoint (20,59 ; 20) in the scatterplot.

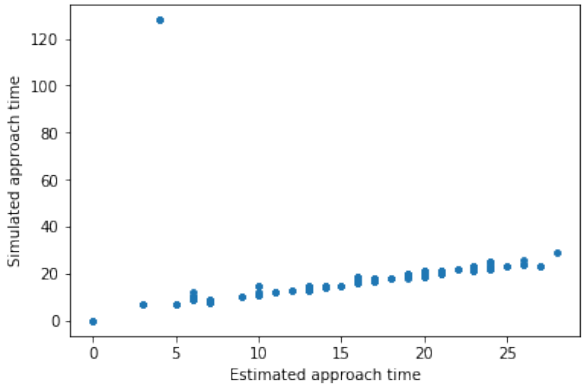


Figure 5.4: Scatterplot of the approach time without clearing the data

In total there are 1157 datapoints for each time component. In figure 5.4 the datapoints for the approach time are shown. As it can be seen, it consists of an outlier[1], at (5,128). Just before that block Raynes Park Station is located. In the simulated environment, the approach time started before the train stopped at Raynes Park, so the stopping time at that station is included in the approach time. This is an exception on blocks after the other stops. At the other stations the first block after the stop is being reserved just after the train departs. This last approach is implemented in the model. Because of this, there is a large difference between the estimated and simulated approach time at the block after Raynes Park. Since this is an exception compared to all the other results, this datapoint is removed from the dataset. The datapoints at (0,0) are kept, because these are a correct estimation compared to the simulated results.

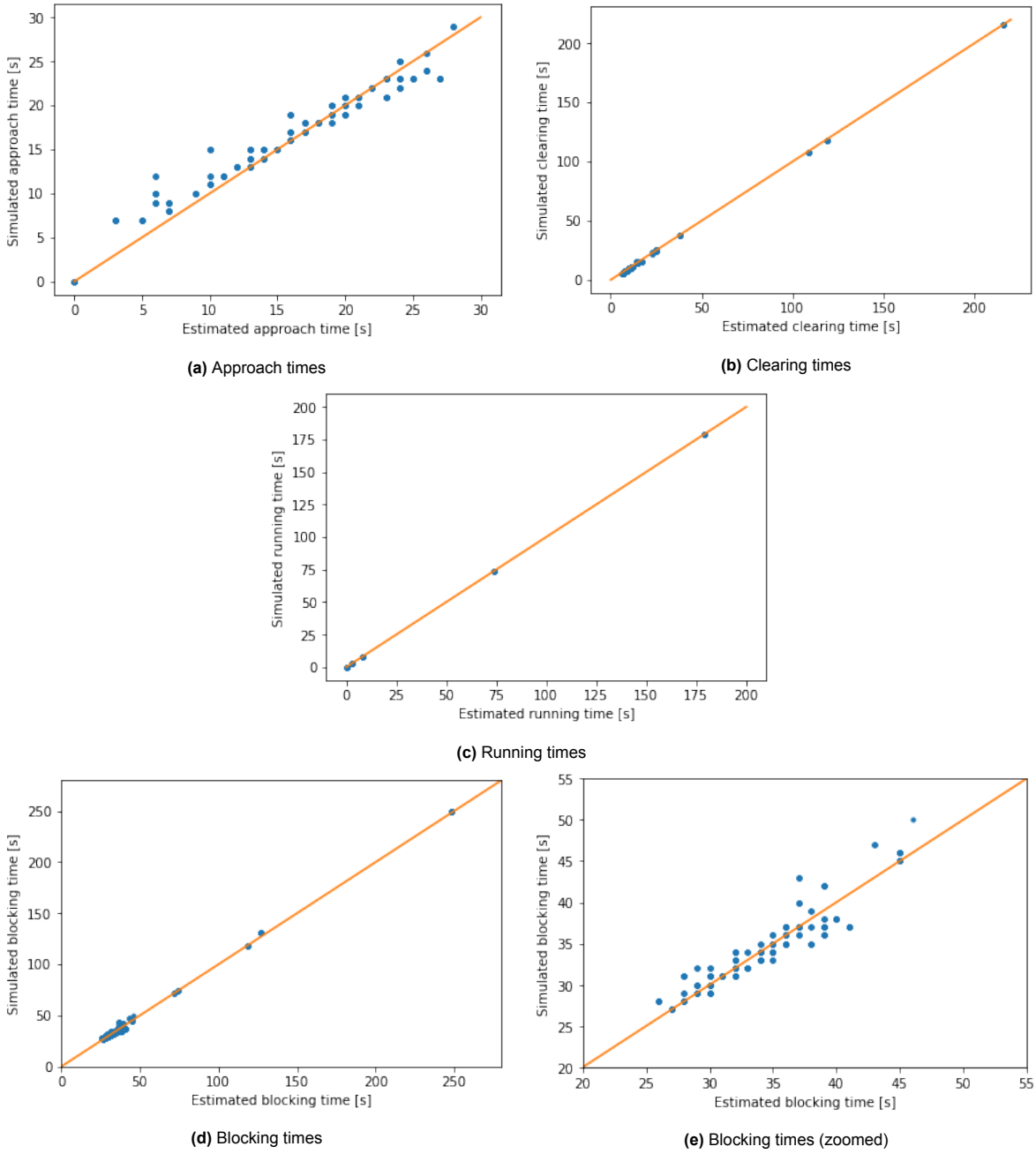
5.3.2. Results

The results of the approach time, clearing time, running time and blocking time (plus a zoomed version) can be seen in figure 5.5. The other three components are a fixed value, depending on the system and safety margins. These values are not shown in the figures, but are included in the blocking times (figures 5.5d and 5.5e). With a perfect model, an estimated time component would be equal to the simulated value, which means that all datapoints are on the diagonal in the figure. Therefore the diagonal line x=y is added to the graphs, to visualise how the models perform.

The strength of the model will be verified based on its precision and accuracy. The precision is how close the estimated values are to each other. This can be measured by finding the correlation between the estimated and the simulated values. Although we are not looking for a dependency between two variables, a high correlation indicates how precise the estimation is compared to the simulated results. With Pearson's correlation formula the correlation can be calculated:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{5.1}$$

[1] Actually, those are twelve datapoints, one of each train, which are located at the same spot



(a) Approach times

(b) Clearing times

(c) Running times

(d) Blocking times

(e) Blocking times (zoomed)

Figure 5.5: Scatterplots of estimated vs. simulated time components, including the aimed line x=y

where: n = sample size
x_i, y_i = individual sample points indexed with i
$\bar{x}, \bar{y}$ = sample mean

Knowing the precision alone is not enough the know the strength of the model. For example, the model should give a 1:1 ratio between the estimated and simulated datapoints – the diagonal line x=y. When it has a 1:2 ratio, it can still be precise, but it is not accurate. Accuracy is how close the estimated value is to the simulated value. It can be measured by calculating the absolute and relative differences between the estimated and simulated datapoints:

$$Abs.difference = \frac{1}{N}\sum_{i=1}^{N}|Y_i - \hat{Y}_i| \tag{5.2}$$

$$Rel.difference = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_i - \hat{Y}_i|}{\hat{Y}_i} \tag{5.3}$$

where: Y_i = Estimated value
$\hat{Y}_i$ = Simulated value
N = Sample size

Before the precision and accuracy will be calculated, the estimated running times should be highlighted. In 5.5c it looks like that the running times are perfectly estimated. However, remember that this sub-model includes three different situations: normal line sections, virtual blocks and stops at station areas. At normal line sections the running time – which is in this study about 95% of the datapoints – the running time is always zero, so this is indeed a perfect estimation. However this is not the case for the virtual blocks (to protect moving infrastructure elements) and stops at station areas. Because there are only a couple of datapoints which can be categorised in one of these two, this makes it harder (or impossible) to perform a statistical analysis on them. This does not mean that it is not possible to say anything about the quality of the estimation of the running times. The estimation of the running times through virtual blocks is very similar to the estimation of the clearing time: in both cases it is the time needed to cover a fixed distance. The distances are independent of the train's behaviour and the method to calculate the time is the same. So this means that it will give similar results in terms of precision and accuracy. The standstill time at a station can easily be measured in the trajectory dataset. Because there are only 4 stops in this study area, excluding starting station London Waterloo, there is not enough data to perform a proper statistical analysis. However, based on figure 5.5c it is possible to say that there are not large deviations.

Pearson's correlation formula is applied to the approach time, clearing time and the blocking time. The results can be seen in table 5.6. The results show a very high correlation between the estimated and simulated datapoints.

Table 5.6: Correlation between the estimated and simulated time components

Time component	Correlation $\left(\rho(t_{est}, t_{sim})\right)$
Approach time	0.98
Clearing time	0.99
Blocking time	0.99

The absolute and relative differences can be seen in figure 5.7. On average there is an absolute difference of 0.87 seconds between the estimated and simulated blocking time, which is a relative difference of 2.33% (see table 5.7).

Table 5.7: Average differences between the estimated and simulated time components

Time component	Absolute difference [s]	Relative difference [%]	95% Confidence interval
Approach time	0.79	6.14	(-4,4)
Clearing time	0.53	6.74	(0,1)
Blocking time	0.87	2.33	(-3,3)

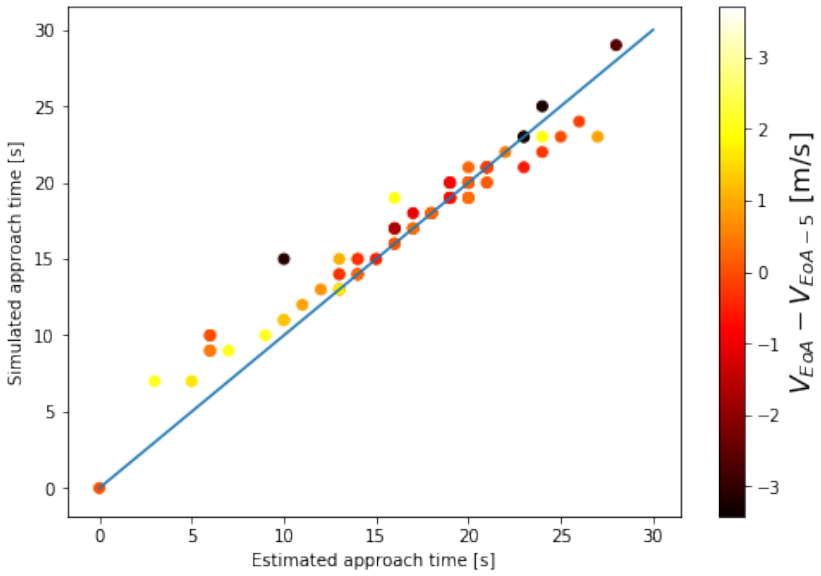


Figure 5.6: Estimated vs simulated approach time, where the colors indicate an acceleration or deceleration of the train

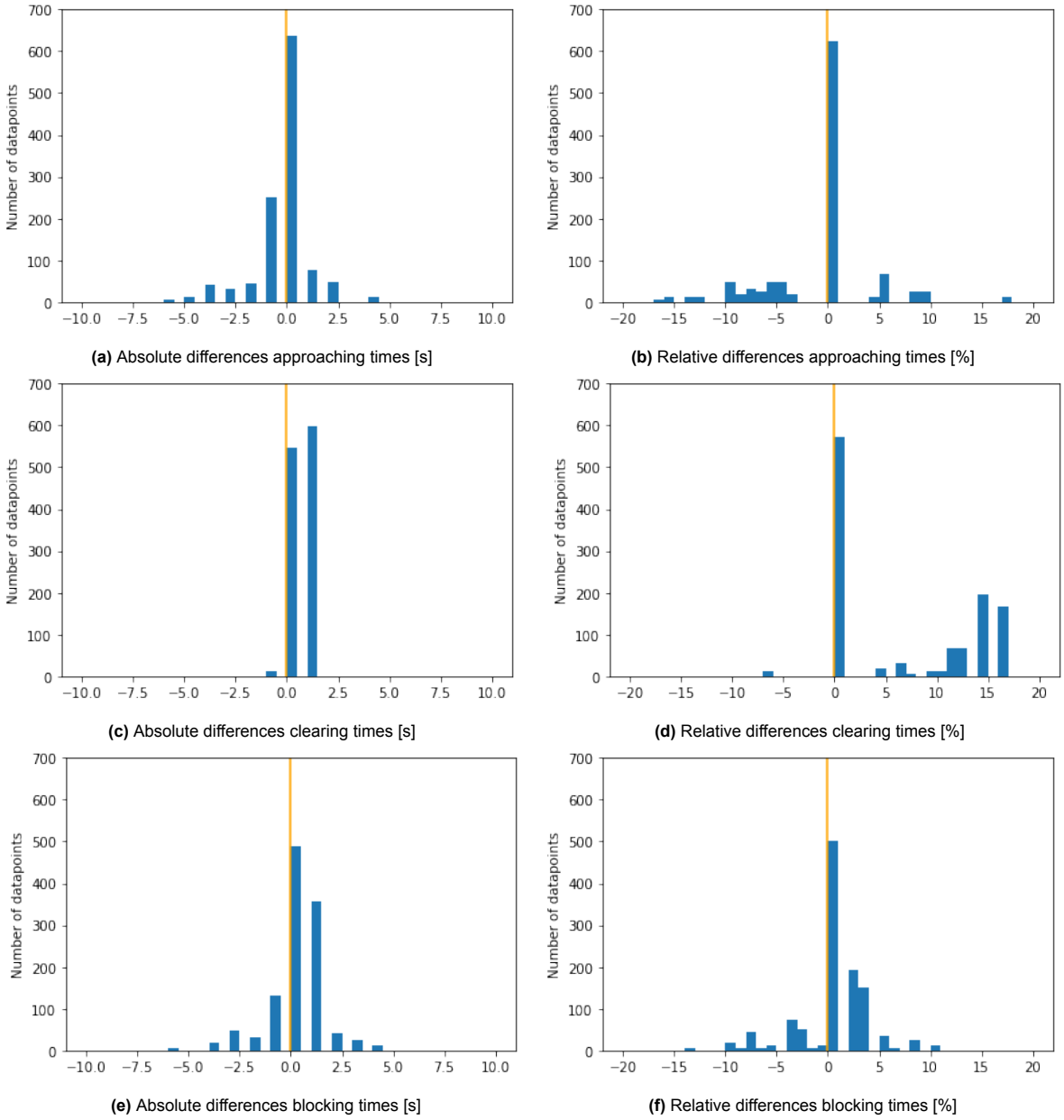
5.3.3. Discussion and conclusion

The largest deviation of all time components between the estimated and simulated results can be found in the approach time. Although on average the deviation is less than a second, some datapoints show a difference of a couple of seconds, considering figure 5.5a. It seems that the model slightly underestimates the approach time of EoA-locations with short approach times, and overestimates the approach time for EoA-locations with larger approach times.

By definition, the indication point depends on the speed and braking capabilities of the train. However, the approach time is the time it takes to cover the absolute braking distance. When the train would slow down when covering the braking distance, the approach will increase. The opposite applies too: when the train would accelerate, the approach time will decrease. This has been made visible in figure 5.6. Here the colors of the scatterplot indicate if the train slowed down or accelerated during 5 seconds before it reached the EoA. One would expect to have (mostly) yellow dots in the lower segment ($\leq$10s) and dark red and black dots in the upper segment (>20s). But there are also a couple of orange/red dots in the lower segment and orange/yellow dots in the upper segment. Also, there is a mix of all colors in the middle segment. This means that one cannot say with full certainty that the model performs worse when the train brakes or accelerate.

Besides this conclusion, one can also find an outlier at (10,15). Because the maximum allowed speed at this location is much lower than on the rest of the line, the approach time is also much lower, compared to the other black dots.

Looking at the computed differences between the estimated and simulated data in figure 5.7, about half of the estimated approach times are perfectly estimated. For the clearing time a similar result is achieved. A majority of the other clearing times is one second too high estimated. This results of an average differences of 0.53 seconds. In general clearing times are shorter than approach times, so this results in larger relative differences – up to 16% – in the clearing time than in the approach time. When considering all sixe time components together, e.g. the blocking time, in 95% of the cases the estimated blocking time is within a range of -3 to +3 seconds of the simulated blocking time. Currently ProRail plans with an precision of 6 seconds [92]. All in all, it can be concluded that the model estimates the blocking time very well, with a high accuracy and high precision.



(a) Absolute differences approaching times [s]

(b) Relative differences approaching times [%]

(c) Absolute differences clearing times [s]

(d) Relative differences clearing times [%]

(e) Absolute differences blocking times [s]

(f) Relative differences blocking times [%]

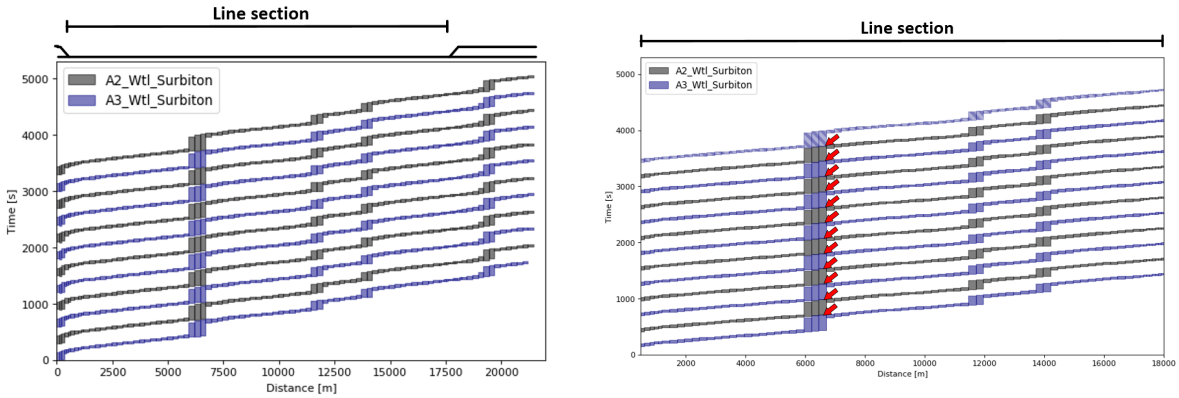
Figure 5.7: Histograms of the differences between the estimated and simulated time components

5.4. Using track occupation data to identify capacity bottlenecks

In section 4.2 two methods were described to identify capacity bottlenecks on a line section. In this verification study the timetable compression method described in UIC 406 is used to verify these methods. Since the study area is relative simple, with only one line section, and it is not needed to calculate the infrastructure occupation, the shortcomings of the timetable compression method will not hinder the usage of this method for the verification. Remember that we had a homogeneous and a heterogeneous traffic situation, as explained in section 5.2. The blocking time diagrams of both situations are shown in figures 5.8 and 5.9.

When the line sections are compressed, the critical block in the homogeneous situation is 30-B2 (marked by the red arrows in figure 5.8b). In the heterogeneous situation there are two critical blocks, which are 4-B18 (500-620m.) and 80-B2 (17900-18000m.). Those blocks are the bottlenecks of this line section.

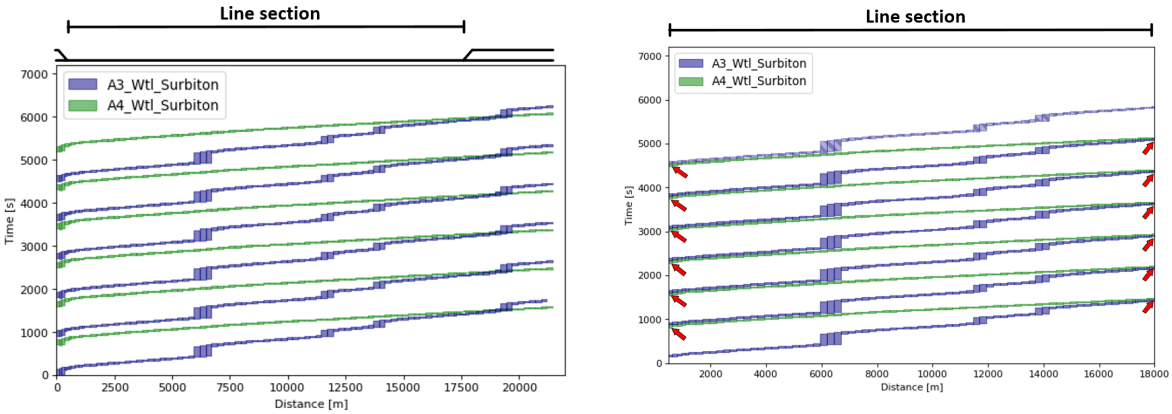
In figures 5.10 and 5.11 the summed blocking times are plotted for the homogeneous and hetero-



(a) Uncompressed

(b) Compressed

Figure 5.8: Blocking time diagrams of homogeneous traffic between London Waterloo and Surbiton



(a) Uncompressed

(b) Compressed

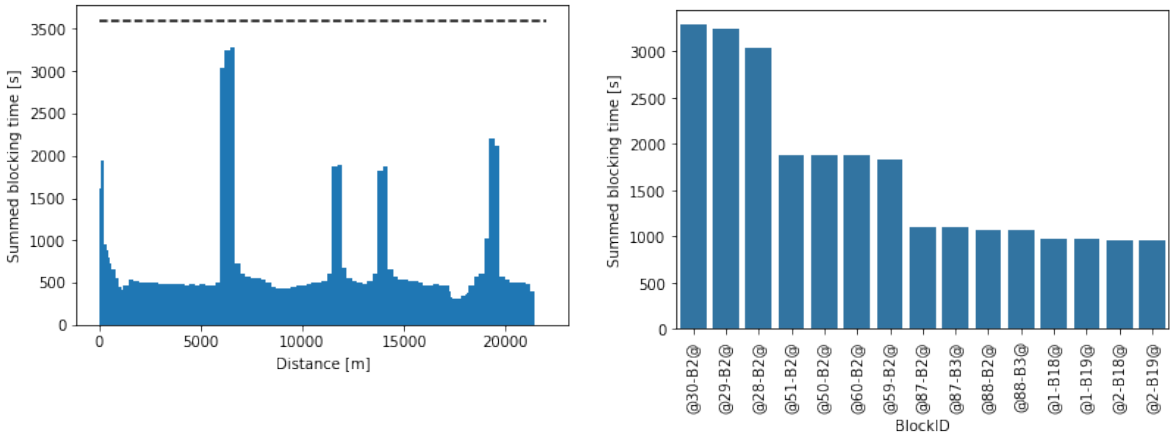
Figure 5.9: Blocking time diagrams of heterogeneous traffic between London Waterloo and Surbiton

geneous traffic situations, sorted over the length of the shared infrastructure and sorted per highest blocking time. In both figures there are clearly three peaks visible, which correspond to the first three stops of the sprinter trains. The peak at block 30-B2 (6450m.) is the highest, so this should indicate that this block is the bottleneck of this line section. However only for the homogeneous situation this matches with the result of the UIC 406 method. In the heterogeneous situation the highest summed blocking times does not match with the bottleneck in the UIC 406 method.

In figure 5.12 the buffer times of both traffic situations are plotted. In the homogeneous traffic situation the lowest buffer time can be found at block 30-B2. Since the traffic is homogeneous, this is also the buffer time between every two trains that pass that block. In the heterogeneous traffic situation the lowest buffer time can be found at block 80-B2 between a sprinter (A3) and a following intercity train (A4). Note that the next lowest buffer times in figure 5.12b are of the same pair of trains, so these cannot be seen as a bottleneck location. When a sprinter follows the intercity train, the lowest buffer time is at block 4-B18, which is the other bottleneck. The outcome of this approach matches with the bottlenecks after applying the timetable compression method.

When this is applied for a study area which consists of multiple corridors, one will get a list of (possible) bottlenecks. This list can be organized by sorting on the lowest buffer time to find the most urgent one. Other possibilities are to organize it by name of the area, type of area and to see how often a section is critical for between two trains. A more detailed application can be read in section 7.2.4.

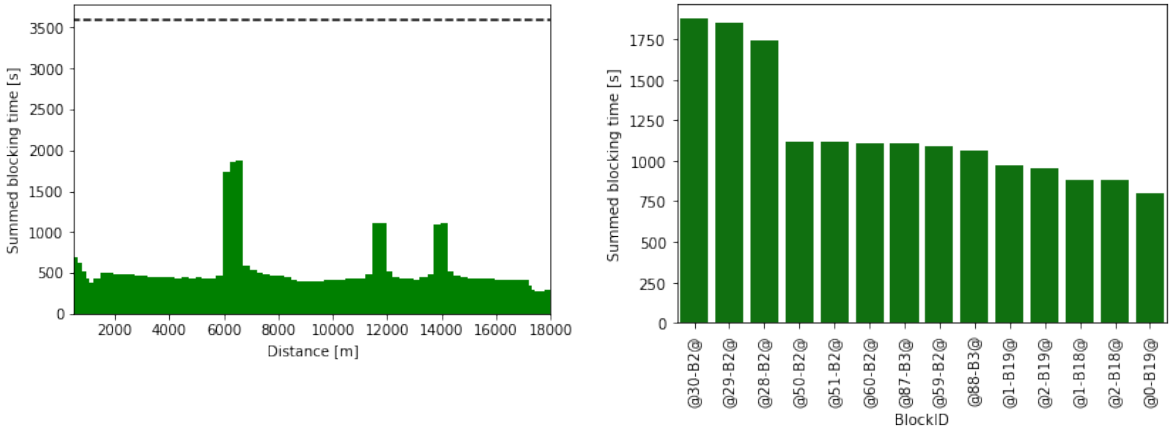
To conclude, summing the blocking times to find a bottleneck can be used in only a homogeneous traffic situation. Looking for the block with the minimal buffer time between two trains to identify a bottleneck is a suitable approach for both traffic situations.



(a) Sorted over the length of the shared infrastructure

(b) Sorted over highest summed blocking time

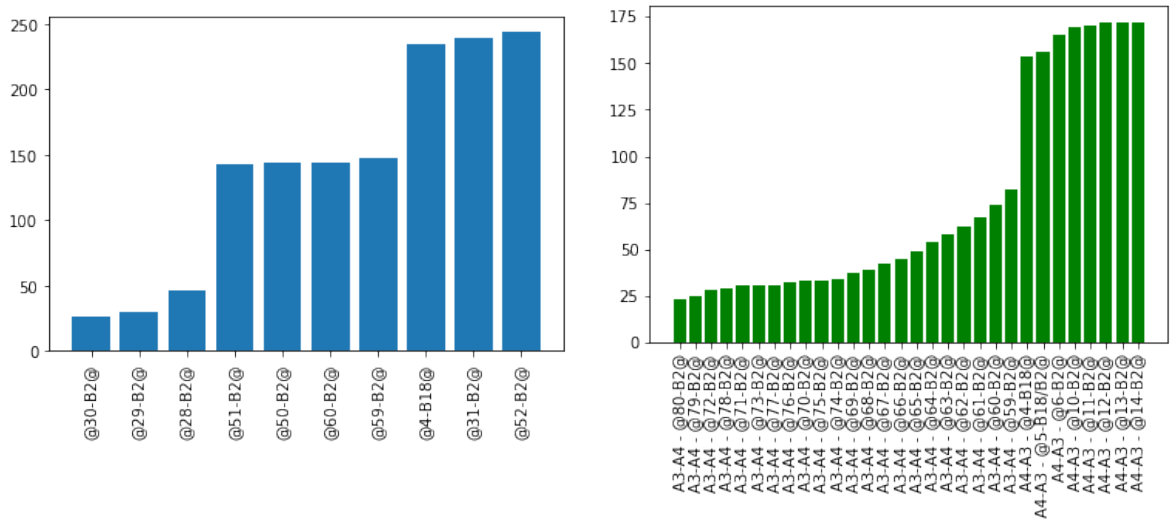
Figure 5.10: Summed blocking times of the heterogeneous traffic situation



(a) Sorted over the length of the shared infrastructure

(b) Sorted over highest summed blocking time

Figure 5.11: Summed blocking times of the homogeneous traffic situation



(a) Homogeneous traffic

(b) Heterogeneous traffic

Figure 5.12: Lowest buffer times between two following trains

6

Case Study

The defined method in chapter 4 will be applied to a case study in the Netherlands. The input data will be generated with the simulation tool FRISO, which will be introduced in section 6.1. In section 6.2 the model setup is presented. The data collection and processing steps are discussed in section 5.3.1. In the last two sections the model is applied and it is explained how the buffer times are calculated. The results of the case study are presented in chapter 7. .

6.1. FRISO

Flexible Rail Infrastructure Simulation of Operations, in short FRISO (Dutch: Flexibele Rail Infra Simulatie Omgeving), is an event driven microscopic railway simulation model owned by ProRail and developed by Incontrol Simulation Software [35]. FRISO has a couple of features from which it can distinguish itself from other simulation software [65, 66, 88]:

- It can be used from a wide variety of studies, such as analysing timetables, comparing train dispatching variants, tracing and quantifying bottlenecks in the infrastructure and calculations of travel times of trains;
- One of the key features of the tool is its flexibility. However, this is expressed in another way than in EGTRAIN. In EGTRAIN it was possible for the user to change the simulation core and decide what the output would be. In FRISO the way to make changes to the infrastructure, such as adding/(re)moving tracks, switches and signals, has been made very user friendly. This, in combination with smart editors for making timetable variants, reduces the time efforts significantly for implementing a simulation study.
- It can be connected to an external Traffic Management System (TMS). In these cases it can be used to test and compare different kind of TMS's;
- FRISO can be connected to other simulators to test and interfere with human behaviour, such as PRL (dispatch manager), VKL (traffic manager, e.g. TRINITY) and Morpheus (train driver simulator). This helps to improve the operations of the company.

FRISO has five main components, visible in figure 6.1:

- FRISO Input database (FRISO database)
- FRISO Simulation Engine (FRISO)
- FRISO Incontrol Center (FIC)
- FRISO Output Database
- FRISO Results Manager (FRM)

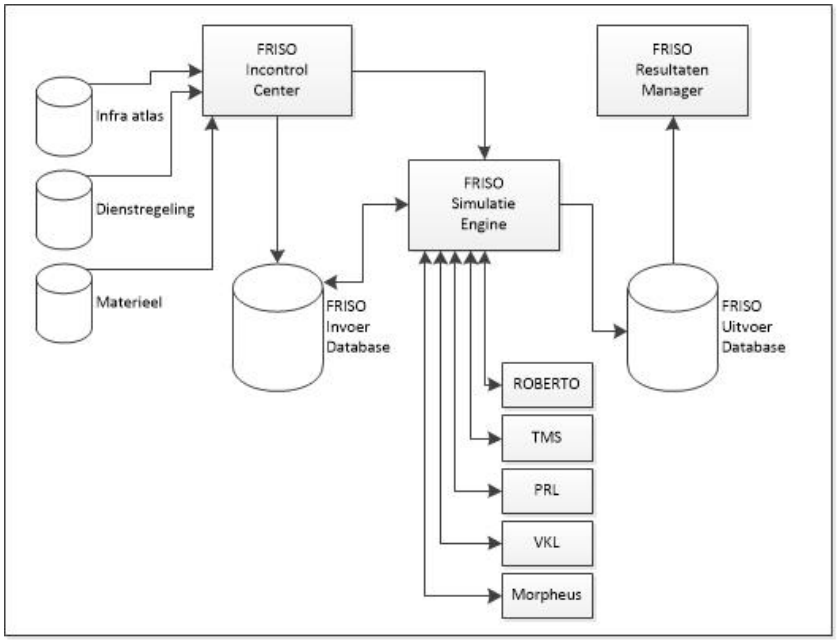


Figure 6.1: Schematic overview of all components in FRISO (source: Steneker and Cunes [88])

The input database consists of all the information and data which are needed to perform the simulation, such as infrastructure (from Infra Atlas), timetables (output of DONNA), rolling stock (from DONS) and simulation settings. Since the data comes from different kind of sources, the tables are first converted in FIC and then send to the input database and simulation engine. The simulation engine performs the actual simulation and can be connected to other models and simulators. The results of the simulation will be logged into the Output database and can be analysed in the FRISO Results Manager. In addition to the simulators, FRISO can generate an output that can be used as input for Roberto. Roberto is a tool to calculate large sets of headway times based on the infrastructure, train data and signalling system.

At the beginning of each simulation all the trains which will operate are generated. Each train gets a list of events – based on the timetable – and its attributes, such a ID, serialnumber, etc. Just before its first event, the train will be activated and placed in the model. Then the train will simulate all the events occurring within the pre-selected simulation time period. These events can be: departure, arrival, passage or short stop (standstill time). During the events the trains will operate under protection of NS ’54 or ERTMS (ETCS Level 2), based on the availability and the settings of the simulation. In this case study the trains will operate ’according plan’, but other possibilities are ’keeping train order’, where disturbances are possible but the train order will not change, or First Come First Serve (FCFS) [88].

In FRISO it is possible to simulate different scenarios by adding disturbances to the model. The possibilities are:

- Entrance disruption: the time a train will be placed in the model will be disrupted;
- Dwell time disruption: the time a train standstill at a stop will be disrupted;
- Acceleration disruption: the acceleration of the train will be disrupted. This can be useful to simulate cases where there is less power available to accelerate;
- Departure disruption: a disruption of the departure in seconds after the departure time;
- Maximum speed disruption: the maximum allowed speed is lower than in undisturbed cases.

The model described in this research uses planned time-distance data, so in this case study there are no disruptions added to the simulation.

6.2. Model setup

6.2.1. Study area

For this case study four corridors are selected, corrisponding to the four colors in figure 6.2:

1. Rotterdam – Schiphol – Arnhem (RoSA corridor)
2. Amsterdam – Maastricht (A2 corridor)
3. Roosendaal – Venlo (Brabant route)
4. Tilburg – 's-Hertogenbosch – Nijmegen – Arnhem

This study area covers 5 of 7 sub-programms of the national Program High frequent train operation (PHS, dutch: Programma Hoogfrequent Spoorvervoer), which are Amsterdam-Eindhoven, Schiphol-Utrecht-Nijmegen, Breda-Eindhoven, Meteren-Boxtel and Rijswijk-Rotterdam. This program aims to improve connections by making changes to the infrastructure and signalling system. These improvements should make it possible to have an intercity train every 10 minutes and more space for cargo trains [70]. Since December 2017 there is an intercity train every 10 minutes on the corridor Amsterdam-Eindhoven and in the timetable of 2022 two other corridors were added: Rotterdam-Schiphol and Schiphol-Nijmegen. [52]

Figure 6.2: Study area

6.2.2. Timetable and rolling stock

In FRISO the PPLG's (Interlocking areas, Dutch: Primair Procesleidingsgebieden) are selected which correspond to the four corridors. In figure 6.3 an example is shown of the PPLG's around The Hague, where the green PPLG's are *active*. All trains who pass an active PPLG were selected to be simulated, but only for the parts with an active PPLG. For example, the sprinters between The Hague CS (Gvc) and Dordrecht (Ddr) are only simulated between The Hague HS (Gv) and Rotterdam (Rtd). The trains will operate according to the timetable of 2021. For the simulation the Roberto function in FRISO has been used, which means that each train will be simulated individually. The advantage of this function is that the trains cannot interact with each other, so the output corresponds to nominal undisturbed conflict-free scheduled train operations.

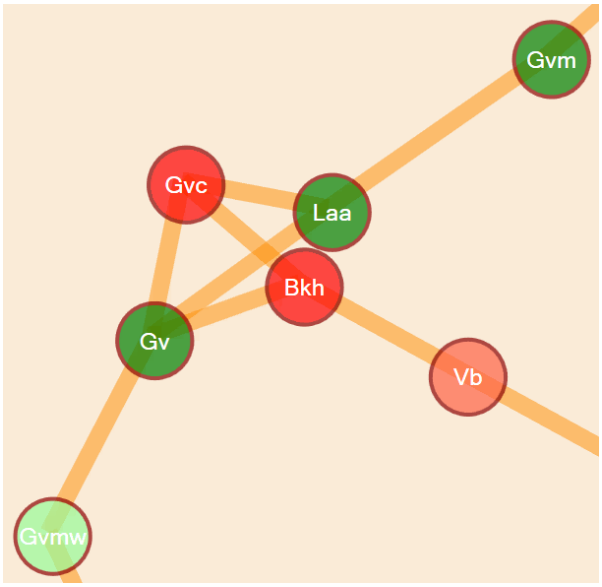


Figure 6.3: Active interlocking areas (PPLG's) in green around The Hague

In total 174 trains were selected for this simulation, whereof each train is simulated once. In appendix E the full list is given, which includes the names, train types, rolling stock type, length and route.

6.2.3. Infrastructure

In appendix F the infrastructure is given on a microscopic level, including the locations of the signals. The maps are made in Inframonitor. The study area can be split up in 28 Engineering line references,

in the infra database called 'Kilometerlint'. They identify different cartesian systems to report the curvilinear progressive representation of the railway tracks pertaining to a given region of the Dutch rail network. On all lines the Dutch fixed-block NS'54/ATB-EG three aspect signalling is in use.

In the model blocks between two signals are not named. Instead of that, it uses track vacancy detection sections as reference to a piece of track. These sections are small parts of infrastructure – varying from 100m up to 1000m – and have a name and ID number. Often signals serve multiple sections, so one could say that a block consists of multiple sections. It can happen that a name of a section occurs multiple times in a network, but all sections have an unique ID. These ID numbers can be used as reference to calculate the blocking- and buffer times later in this study. Later on in this study the sections will be used to compare the blocking times between fixed- and moving block.

6.2.4. Route setup times

For NS'54 the following route setup times, in FRISO called 'omlooptijden', are used:

- 2 seconds as base value to set the route
- 2 seconds as delay between locking the switches and set the signals
- Time to move a switch:
  - Angle > 1:20 : 13 seconds
  - 1:12 < Angle ≤1:20 : 7 seconds
  - Angle ≤1:12 : 3 seconds

  It is assumed that switches are set and locked parallel.

6.3. Data collection and processing

During simulation practitioners have the possibility to save the section occupation times. This dataset has the following information:

- Section ID
- Section name
- Train name
- Entrance time [s]
- Clearance time [s]
- Kilometerlint (Line reference)
- Entrance location on the Kilometerlint [m]
- Exit location on the Kilometerlint [m]

However, all other required data, such at train trajectories and infrastructure data cannot be logged by practitioners. Therefore, the company Incontrol [35] was asked to log this data via the back-end of the simulation software. This resulted in three other datasets:

- **Trajectory data** – for each infrastructure element a train passes, the following information i.a. was logged: Train name, train type, train length, infrastructure element (ID, name and type), line reference, section name, braking distance and for the front and back of the train: passage time, speed and interval time.
- **Train data** – every time the acceleration of the train changes due to e.g. change of slope or a change in power usage, the state of the train will be logged. This contains i.a. the following information: Time, location, speed, acceleration, gradient, power usage, train name and the line reference.
- **Route formation data** – the times a route will be set in an interlocking area, including train name, route name, and of the start and end location of the route: signal type, signal name, signal ID, line reference and location on that line.

During the data processing it was discovered that in the section occupation dataset the Engineering line reference Dvaw-Asb didn't exist, while it is present in the Trajectory dataset. The sections that are located on this line had the (neighbouring) line reference Gpda-Asra. At the moment this thesis was published, this is the only inconsistency in the line references that is known by the author. It was not found out that there are any inconsistencies in the ID numbers of the infrastructure elements, such as signals and sections. Therefore the ID numbers will be used as reference to process the data in the rest of this research, instead of the Engineering line references.

6.3.1. Fixed block blocking times

In FRISO the occupation times of the tracks can be determined in two ways: section occupation and blocking times. In FRISO the section occupation is the time when a train enters a section until it has completely left that section. In other words: the physical occupation time, so the running time plus clearing time of a section. The blocking time is the time starting when a route will be set until it has been released. For this, FRISO makes a distinction between interlocking areas and open lines, which are parts of infrastructure without any switches. In interlocking areas a block will be reserved according to the location of a train and its scheduled route. In some cases a single block will be reserved at a time (Dutch: enkelvoudige route), but there are also cases where multiple successive blocks will be reserved at the same time (Dutch: samengestelde route). In open lines a track will be reserved until the next red signal. A red signal can occur when there is a train in front or at the entrance of a interlocking area. The problem is that in both cases the blocking times are higher then one would expect when each block is being reserved individually. For example, the full length between Abcoude and Breukelen (12km) is marked as interlocking area. Train A120 reserves the blocks over this full length at the same time, which results in a blocking time up to 500s. A similar thing happens in open lines, where multiple successive blocks are reserved until the first red signal. In practise this is not a problem, because in an open line area there are no switches, so it is impossible for any other train to enter that area. However, this results in a longer blocking time than needed from a safety perspective, so this data cannot directly be used to perform a capacity assessment.

To overcome this, the blocking times for fixed block signalling will be recalculated according to the blocking time theory, assuming an automatic block signalling rather than an interlocking area with composite routes. For this a block is defined as the distance between two main signals. During a simulation in FRISO the following information (i.a.) can be logged in the trajectory data:

- Train
- Signal ID, name and type (interlocking/open line)
- Line reference
- Location [m]
- Time when the front (T_f) and back (T_b) passes a signal [s]
- Braking distance (based on the speed the train enters the section) [m]

T_f is the entry time of a block and equal to the start of the running time. The start of the blocking time can be determined by subtracting the approach, reaction and setup time from T_f.

For the approach time, the braking distance is important. When the braking distance (l) at the previous signal ($i-1$) is shorter than the block length ($s_{f,i} - s_{f,i-1}$), then the approach time is the running time through that block ($T_{f,i} - T_{f,i-1}$). If the length of the previous block section is shorter than the absolute braking distance, then the approaching time is considered to be the running time over multiple consecutive preceding block sections for which the sum of their lengths exceeds the absolute braking distance. Thus, the approach time can be determined by the minimum value of:

$$t_{approach} = T_{f,i} - T_{f,i-j} \tag{6.1}$$

subjected to:

$$l_{i-j} \leq s_i - s_{i-j} \tag{6.2}$$

where: i = index of block
j = 1,2,…,i
$T_{f,i}$ = front passage time at block i [s]
s_i = entrance location of block i [m]
l_{i-j} = Braking distance at previous block(s) [m]

Just as in the verification study, for the setup and reaction time two fixed values will be taken. This will be respectively 1 and 9 seconds. The end of the occupation time can be calculated by adding the running, clearing and release time to t_f. The running time is the time it takes for the front of the train to run through the block to the next signal. The clearing time can be determined by taking the differences of the passage time of the front and back of the train at the next signal:

$$t_{running} = T_{f,i+1} - T_{f,i} \tag{6.3}$$

$$t_{clear} = T_{b,i+1} - T_{f,i+1} \tag{6.4}$$

where: i = index of block
$T_{f,i}$ = passage time (front) at block i [s]
$T_{b,i}$ = passage time (back) at block i [s]

The release time will be assumed as a fixed value (1 s.). The blocking times can be calculated with:

$$T_{StartBlockTime} = T_{f,i} - t_{approach} - t_{reaction} - t_{setup} \tag{6.5}$$

$$T_{EndBlockTime} = T_{f,i} + t_{running} + t_{clear} + t_{release} \tag{6.6}$$

$$t_{Blockingtime} = T_{EndBlockTime} - T_{StartBlockTime} \tag{6.7}$$

On this method one exception will be taken into account. For the first block after a stop at a station, the start of the blocking time is the time the route will be set in the route formation dataset. Otherwise this block will already be occupied when the train arrived at the platform, resulting in a higher blocking time.

Once the blocking times for fixed block are calculated at a signal location, the blocking times will be copied to all sections that are served by that signal at that time, except for sections in interlocking areas. In interlocking areas track sectional route release will be used. The end of the occupation time can be determined by the clearing time (t_{clear}) in the dataset 'section occupation' plus adding the fixed value for the release time:

$$T_{EndBlockTime} = T_{clear} + t_{release} \tag{6.8}$$

By having the blocking times for fixed block per section, these blocking times can be compared to the blocking times for moving block, calculated later on in this study. The python script of the complete calculation can be found in appendix D

6.3.2. Planned time-distance data

FRISO is an event-driven simulation model, which means that the software can make a logging when an event occurs. To collect a trajectory dataset, a log has been made every time a train passes an infrastructure element. The elements included are:

- Buffer stop signal
- Level crossing (7 different types)
- Auto signal
- Auto signal P
- Balise
- signal (interlocking)
- Stopping mark
- ETCS Block marker
- Line reference point
- Platform track
- End weld
- ERTMS_IN
- ERTMS_OUT
- Border (Country)
- H_signal
- Repeat
- Height
- Map border
- Offset
- Intersection
- L_signal
- Weld
- Single weld
- Double weld
- R_sign
- Section
- Speed sign A
- Speed sign CA
- Speed sign E
- Speed sign M
- Speed sign O
- Track
- Stop sign
- TF weld
- TF weld double
- Station announcement sign
- Distant signal
- Switch

This resulted in an output of 108.836 datapoints over the whole study area. With these datapoints the trajectories (time, location, speed) of the trains can be constructed. This applies: the smaller the interval between two datapoints, the more precise the trajectory will be. About a fifth of those datapoints are located at the same location as another infrastructure element, resulting in an interval of 0 seconds in the logging of those datapoints. Since the type of infrastructure is not important for the model, but only the time, location, speed and braking possibility at that location, these *double* loggings can be removed. When that is done, 87.793 datapoints are leftover. The time intervals are plotted in a histogram in figure 6.4. The average interval is 4.0s and, whereof the maximum is 620s. Large time intervals are mostly caused by a stop at at station. From approximately one third of the datapoints the interval is smaller than 1 second, half the datapoints are smaller than 1.9s and 95% of the datapoints have a smaller interval than 11.9s. Remember that in EGTRAIN the time interval was always 1 second, so in FRISO this is a bit higher. The plots of the trajectories can be found in appendix G.

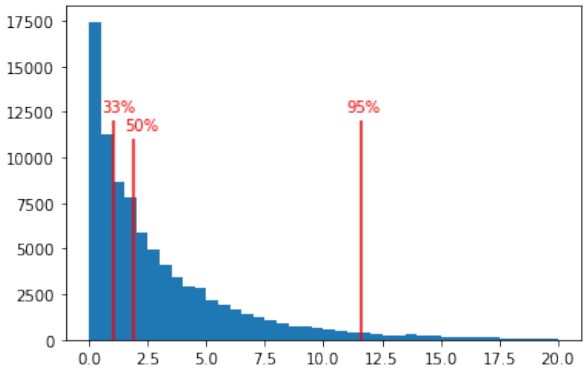


Figure 6.4: Intervals between loggings [s]

6.3.3. Rolling stock parameters

In FRISO the braking rate is an input variable. In practise and in the planning phase a braking rate of 0.5 m/s^2 is used for passenger trains, so this value will be adopted as input parameter in the case study. Besides this, in the defined model the train length was used to calculate the clearing time. However, in FRISO in the passagetime of the front and back of the train is being logged, so the clearing time can be calculated by subtracting these two values. This means that the train length will not be used in this case study.

6.3.4. infrastructure characteristics

The processing of the infrastructure data will be done in the same way as described in 5.3.1. As example, in table 6.1 a sample of the infrastructure data of train B73500 is presented. Based on the location, the gradient has been copied to the corresponding locations in the trajectory data in table 6.2. Then the possible deceleration has been calculated, which is the sum of the deceleration rate of the train and the gravitational effect.

Table 6.1: Sample of the infrastructure data

Train	Line reference	Location [m]	Time[s]	Gradient
B73500-H-1	Asra-Wmd	20126,39749	562,8275601	0,00148718
B73500-H-1	Asra-Wmd	20114,83903	563,9521665	0,0002069
B73500-H-1	Asra-Wmd	20102,06086	565,13169	0,0002069
B73500-H-1	Asra-Wmd	20025	572,0671675	0
B73500-H-1	Asra-Wmd	19976	576,4771675	-0,00115

Table 6.2: Sample of the trajectory data, where column 'Gradient' is added from of table 6.1 and 'Deceleration' is calculated with eq. 4.2

Train	Infra ID	Infra name	Infra type	Line reference	Location [m]	Time [s]	Gradient	Deceleration [m/s^2]
B73500	4397	1183B	Switch	Asra-Wmd	20125	562,96	0,001487	0,514589
B73500	57387	1183BT	Section[1]	Asra-Wmd	20096	565,67	0,000207	0,50203
B73500	57386	1181A-L-300	Offset	Asra-Wmd	20080	567,11	0,000207	0,50203
B73500	57385	1181A-L-200	Weld	Asra-Wmd	20074	567,65	0,000207	0,50203
B73500	57384	1181AT	Section[1]	Asra-Wmd	20049	569,90	0,000207	0,50203
B73500	4394	1181A	Switch	Asra-Wmd	20025	572,06	0	0,5
B73500	57381	1177A-L-100	Weld	Asra-Wmd	20000	574,31	0	0,5
B73500	4391	1177A	Switch	Asra-Wmd	19976	576,47	-0,00115	0,488719
B73500	4390	1175	Switch	Asra-Wmd	19976	576,47	-0,00115	0,488719
B73500	57129	1123B-V-2400	Border	Asra-Wmd	19936	580,07	-0,00115	0,488719

[1]Although a section is not a *point*, but a piece of track, the label of a section is located at a point

6.4. Model application

In this model application the blocking times for moving block will be forecast, using the computed blocking times for fixed block, planned time-distance data, rolling stock parameters and infrastructure characteristics according to the method described in chapter 4. In this chapter the method has been described for a general dataset. However, the output data of FRISO has some specifications, whereof some can make the application easier, but for others small adaptions are needed in the model description. In this section it is described how the model was applied specifically to the FRISO output.

For the route setup-, reaction- and release time the same values are used as for fixed block signalling, which are respectively 1, 9 and 1 second. The calculation of the approach-, running- and clearing time are discussed in the following three subsections:

6.4.1. Approach time

The calculation of the approach time can be split in four steps:

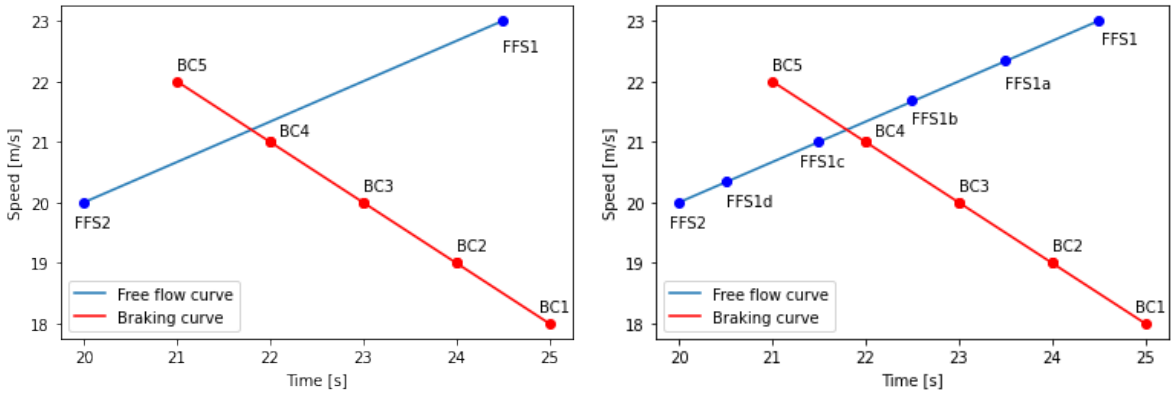
1. **Selection of End of Authority (EoA) locations**
   The blocking time will be calculated for every infrastructural element that is logged. Once the approach time for a specific element will be calculated, it will be assigned as EoA location. The passage time of every location is known with an accuracy of 0.0001 second, which is higher than in the verification study where it was 1 second.

2. **Construction of braking curve**
   An issue with the chosen study area is that a subset of trains enters the model with a passage instead of a departure from a station. For example, in the Dutch timetable train series 2400 operates between Lelystad Centrum and Dordrecht. However, the route between Lelystad and

Duivendrecht is not included in the study area, so the train enters the model while it is moving with a passage at Duivendrecht. Something similar happens with the international train A120. This train enters the model at Zevenaar (also known as the German border nearby Arnhem) while it is moving. The difference is that for the 2400-series it would have been possible to select a larger study area to make sure that this train would enter the model at a standstill position, but for train A120 this was not possible, since that part is in Germany and not included in the FRISO model. On the other hand, in the model the tracks on the German side of the border are long enough to gather enough data to construct a braking curve once it reaches the Dutch-German border. To summarize it, a limitation of the model to calculate the blocking times for MB is that when a train starts in motion, the first (couple of) sections the approach time cannot be calculated, since there is no data of the trains when they approach these sections.

For each EoA location, a braking curve will be constructed according to the method described in chapter 4, using a time step of 1 second.

3. **Finding indication point**
   A disadvantage of an event-driven simulation is that the interval between two datapoints is not fixed (see figure 6.4), both for distance [m] as for time [s]. Small intervals in the trajectory dataset are not a problem for finding the indication point, it can even be favorable because it gives more information about the free flow speed. However, large intervals – let's say 5 seconds and more – are an issue, because during that interval the train could brake or accelerate. How this is an issue, can be seen a fictional example in figure 6.5.
   Here the braking curve (red line) is being constructed (backwards in time) and the final part is shown together with a part of the Free flow speed (blue line). Since BC2 has passed FFS1, the speed of the braking curve will be compared to FFS2. At BC3 – the next point constructed by the braking curve – its speed is higher than FFS2. This means that the train should brake at FFS2, since this is the last known position before it reaches the braking curve. However, there is a large gap between FFS2 and BC3, so braking at FFS2 is a loss of capacity. Besides, braking at BC3 would be too late, because the trains speed is higher than the braking curve at that location.
   To overcome this in the free flow speed curve multiple points are added with an interval equal to Δt (see figure 6.5b). Between FFS1 and FFS2 the acceleration is known, so for the same time interval as the braking curve is constructed, the extra points will be determined (FFS1a-d). For example: between FFS1 and FFS2 the acceleration is 0.67m/s^2, so the speed of FFS1a-d is respectively 22.33, 21.67, 21.00 and 20.33m/s. BC4 has the same speed as FFS1c, so the indication point will be set at FFS1c.
   Finding the intersection point between the braking curve and the Free flow speed without the added intermediate data points and using that as indication point is not possible. This is because of the way the model is programmed. Because the speed at BC3 is higher than at FFS2, the model would break out of its loop, so BC4 and BC5 would not have been constructed. Also,



(a) Without substeps

(b) With substeps

Figure 6.5: Example of the construction of the final part of the braking curve

finding the intersection point between the braking curve and the Free flow speed wíth the added intermediate data point would have been possible, but not necessarily needed. With the previously described approach the intervals between loggings will be equal to a time-driven dataset, so a similar accuracy is reached.

4. **Calculation of approach time**
   The approach is the time it takes of the train to travel from the indication point to the EoA location at free flow speed. So, the approach time can simply be calculated by subtracting the passage time at the indication point from the passage time at the EoA location.

6.4.2. Clearing time

To calculate the clearing time, the front- and back passage time of the train can be used:

$$t_{clear} = T_{b,i} - T_{f,i} \tag{6.9}$$

where i is the index of a section. Because both passage times were logged in the dataset, the length of the train is not needed to calculate the clearing time.

6.4.3. Running time

When a Moving Block systems will be implemented in the Netherlands, it will be ERTMS Level 3 Moving Block. Currently the position report (PR) is updated every 6 seconds. This can be splitted into the following lead times [7]:

- Train integrity (TI) message to the On-Board Unit (OBU): ca 0,5- 1 sec
- OBU generates a PR: ca 0,2 - 1 sec
- OBU sends a moving authority (MA) request, together with the PR, via the GSM-R to the Radio Block Centre RBC: ca 0,5 - 1 sec
- The RBC generates a MA: ca 0,2 - 1 sec
- The RBC sends a MA back to the OBU: ca 0,5 - 1 sec
- The OBU processes the MA: ca 0,5 - 1 sec

However, it is expected that in the future, with Packet Switched and Future Railway Mobile Communication System (FRMCS), the frequency will be reduced to 1 second. Therefore for this case study a running time of 1 second is assumed.

On top of that, ProRail uses the following setup times for switches with ERTMS [88]:

- 2 seconds to evaluate its directions and if necessary to give order to move it
- Time to move the switch:
  - Angle > 1:20 : 16 seconds
  - 1:12 < Angle ≤1:20 : 7 seconds
  - Angle ≤1:12 : 3 seconds
- 5 seconds for the communications to the train

Unfortunately the angle of a switch has not been logged in the dataset. A sample of the switches in the study area is taken in Inframonitor. The most common angle is 1:15, so a setup time of 14 seconds (2+7+5) will be used. It is assumed that switches are set and locked parallel.

For level crossings there is no extra setup time needed, since in the Netherlands level crossings are a fail-safe system. All other places where a train should not come to a standstill (see section 2.1) – such as steep slopes or the overhead catenary design – are not given in the logged datasets, so these will not be taken into account.

6.5. Buffer time calculations

The buffer times were calculated using eq. 4.20. Practically, it was performed by the following steps:

1. **Create a dataset with all (unique) sections**, consisting of column headers: SectionID, section name, Engineering Line Reference (ELR), name of the interlocking area or open line (Dutch: dienstregelpuntcode, DRPcode), type of area (i.e. Station, open line, yard, junction, etc). In table 6.3 an example is given.

2. **Create one dataset of blocking times of all the trains**, consisting of section ID, name of the train, start- and end of the blocking time.

3. **Fit all blocking times in one hour**. Each train in the FRISO model was simulated once. For example, when a train departs at 10 minutes past every hour, it will depart at 600s in the simulation model. When the route is long enough, the start of the blocking time will exceed 3600 seconds, or even 7200s or 10800s. For the buffer time analysis, it will be less complicating to have a clock-face schedule, e.g. all the starting time between 0 and 3600s. If at a certain point the start of a blocking time exceeds 3600 seconds, 3600 will be subtracted from the blocking time. This also applies to 7200, 10800, etc. This way a clock-face schedule can be created. As example, in figure 6.6 this principle has been visualised for the South West Main Line between London Waterloo and Surbiton.

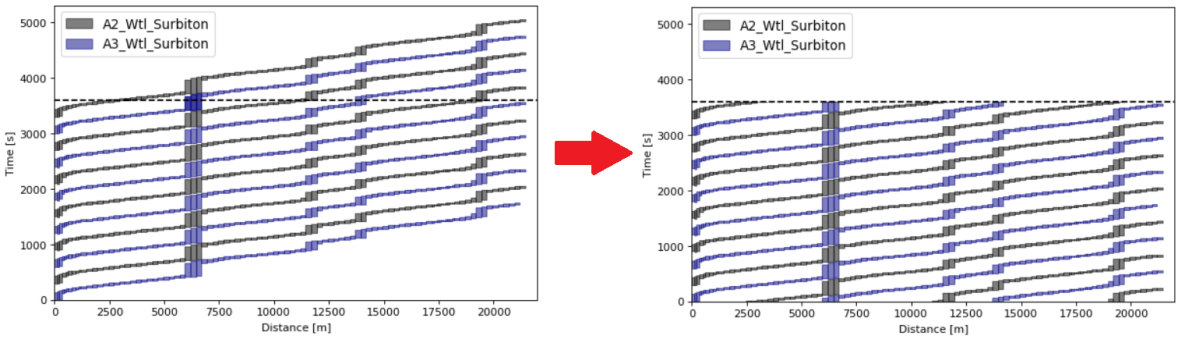


Figure 6.6: Example of fitting all the blocking times in one hour

4. **Sort the dataset of blocking times by the start of the blocking time**, in such a way that sections which are occupied first are at the top and the last occupied section is at the bottom. In table 6.4 an example is given.

5. **Assign the sorted blocking times to dataset of sections**, in such a way that a dataset will be created with all sections and when it is occupied by which train. E.g. tables from step 1 and 4 will be merged. This includes these columns: Section ID, section name, engineering line reference, name of the interlocking area of open line, type of area, Train_1, StartOccTime_1, EndOccTime_1, Buffertime_1, Train_2, StartOccTime_2, EndOccTime, Buffertime_2, Train_3, etc. At this point the columns of the buffer times are still empty.

6. **Repeat the first train of every section**, so the buffer time between the last train of the first hour and the first train of the next hour can also be calculated.

7. **Calculate the buffer times**, which is the start of the block of the following train minus the end of the block of the leading train. An example of the final result can be seen in table 6.5.

In chapter 7 the results of the calculations are analysed en discussed.

Table 6.3: Sample of sections

Section ID	Section Name	ELR	DRP Code	Type
5219	1109A	Asd-Rtd	Ledn	Station
100016	2340AT	Asa-Zvg	Utma	Junction
17121	486AT	Asd-Asa	Asdm	Station

Table 6.4: Sample of blocking times

SectionID	Train	Start [s]	End [s]
5219	B2400	10,0	60,1
100016	B120	11,0	92,5
17121	A3000	12,0	54,7
100016	H3100	249,9	334,8
17121	A3900	545,8	591,0
5219	D2400	1810,0	1860,1

Table 6.5: Sample of the calculated buffer times for moving block signalling

SectionID	…	type	Train_1	Start_1	End_1	Buffer_1	Train_2	Start_2	End_2
5219	…	Station	B2400	10,0	60,1	1800,0	D2400	1810,0	1860,1
100016	…	Junction	B120	11,0	92,5	239,0	H3100	249,9	334,8
17121	…	Station	A3000	12,0	54,7	533,9	A3900	545,8	591,0

7

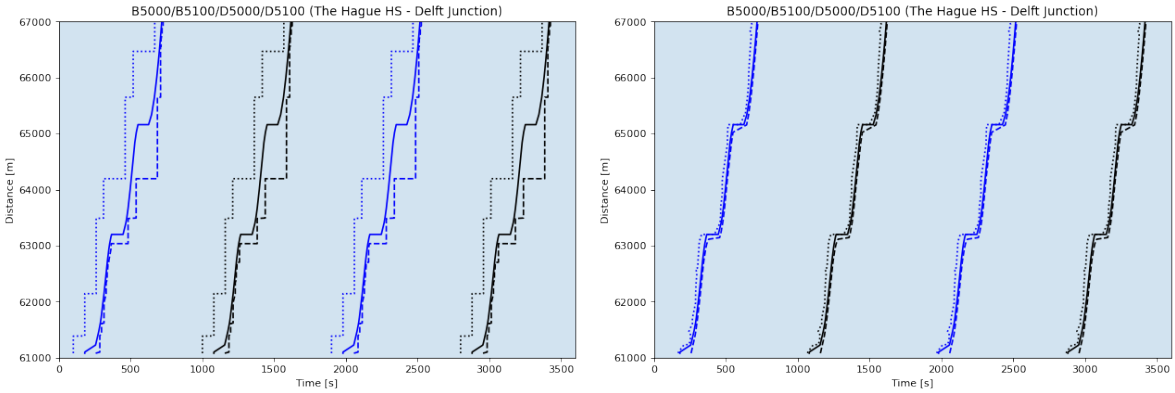
Results

In chapter 6 the setup of the case study was explained and how the model was applied. In this chapter the results are presented. Although the summation of blocking times to identify bottlenecks can only be used in homogeneous traffic situations, in section 7.1 one example will be shown how this method could be applied for the case study. In section 7.2 the minimal buffer time method is used to identify capacity bottlenecks. These will be compared to the results generated with the tool Roberto in section 7.3.

7.1. Summing blocking times

As it has been verified in section 5.4, the summation of blocking times to identify bottlenecks can only be used in homogeneous traffic situations. However, these situations are very limited in the Netherlands. Often a piece of track is used by two or more different types of trains.

In the study area a homogeneous traffic situation can be found between The Hague HS and Delft Junction, located North of Delft station. This line section will be used as example to apply this method. On this line section there are 4 tracks, whereof two are only used by sprinters, each in one direction. In this example a look is taken at the sprinter train from The Hague HS[1] to Delft. For these sprinters the blocking times under fixed- and moving block are calculated accordingly to the method described in sections 6.3.1 and 6.4. The blocking times are plotted in figure 7.1.



(a) Fixed block

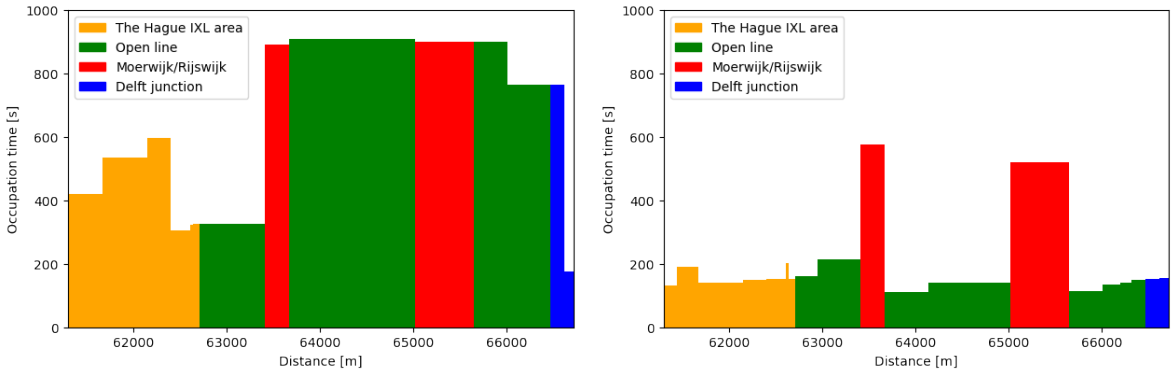
(b) Moving Block

Figure 7.1: Sprinter services from The Hague HS to Delft Junction (dotted line: start blocking time, solid line: trajectory, dashed line: end blocking time)

[1] The sections between The Hague Central Station and HS were not included in the simulation, so the dwelling time at The Hague HS is not known

For the same corridor the summed blocking times are plotted in figure 7.2. The orange bars are located in the interlocking area at The Hague HS, the red bar are respectively The Hague Moerwijk and Rijswijk and the blue bars are the first two sections of Delft Junction (see figure 7.3). In between there are open line sections, colored in green. In the fixed block situation (fig. 7.2a) there isn't a clear bottleneck: both the blocks at the stations as the open lines have a high blocking time. This is different in the moving block situation, where the stations are the main bottlenecks. When the summed blocking times are compared, there is a clear difference between the station areas and open lines. In the open lines there is a reduction up to 87%, while it is at Moerwijk 35% and Rijswijk 42%. The lowest reduction (11%) is at section 466T, just before Delft Junction. This is mainly because in the fixed block situation it has a short approach and running time, because the blocks are short and there is sectional route release at the junction.

Note that in this situation stations Moerwijk and Rijswijk are considered as bottlenecks, but the sprinters interfere with intercity trains at The Hague Central Station and Delft station. It is possible that (one of) these stations are a much larger bottleneck to the capacity. With this method it is not possible to know this, which makes it one of the limitations. To identify capacity bottlenecks in the rest of the study area, it is needed to analyse the buffer times.



(a) Fixed block **(b)** Moving Block

Figure 7.2: Summed blocking times of the sprinter services from The Hague HS to Delft Junction (Sections A304AT until 466T)

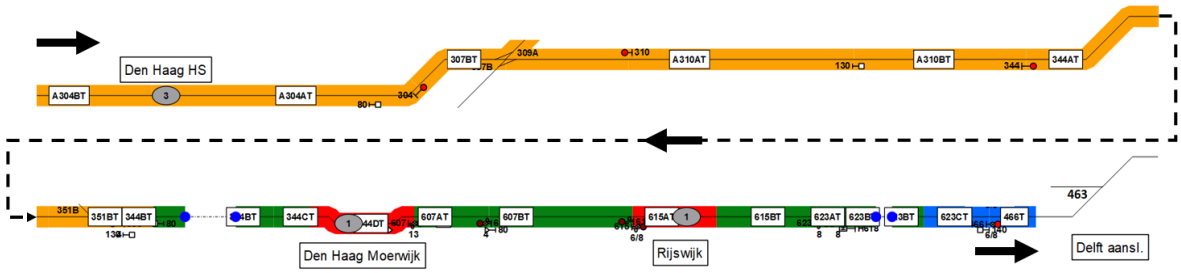
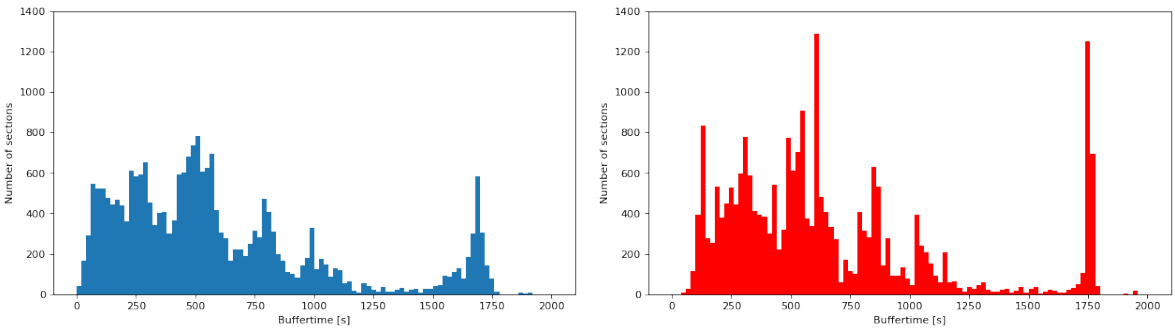


Figure 7.3: Infrastructure between The Hague HS and Delft Junction (not on scale). Colors correspond to figure 7.2

7.2. Buffer times analysis

In figures 7.4 and 7.5 the buffer times of all sections in the whole study area are plotted in a histogram. In both figures there are peaks visible at various buffer times. Since in most situations there is mixed traffic with varying headway times, it is not possible to assign peaks to a certain traffic pattern (i.e. 4 or 6 six trains/h), except for the peak around 1750 seconds, which relates to a pattern of 2 trains/h for that piece of track.

In figure 7.5 the fixed block histogram (fig.7.4a) has been plotted on top of the moving block histogram (fig.7.4b), where the fixed block histogram is slightly transparent to make the moving block histogram



(a) Fixed block **(b)** Moving Block

Figure 7.4: Histogram of the buffer times at sections in the study area

visible. Here a clear effect of moving block can be seen: the peaks are shifted towards larger buffer times, which indicate an overall decrease in the used capacity. Besides this, the peaks are tighter, which indicate less variation in the blocking times. A reason could be that in open lines, where trains are moving with a constant speed, the blocking times with moving block are (more or less) constant. In the fixed block situation there is a large variation in block lengths, due to all kind of reasons. Since the blocking times heavily depends on the length of the blocks, there will also be a large variation in the blocking times.

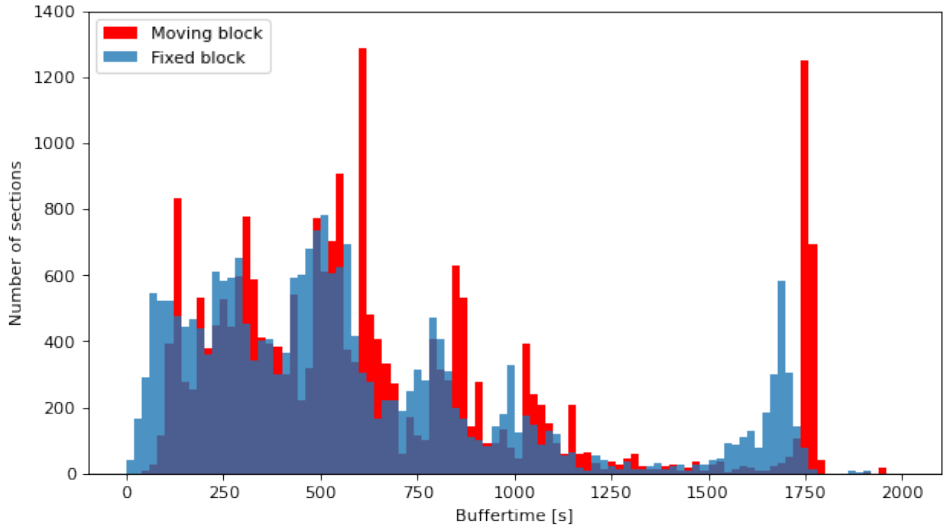
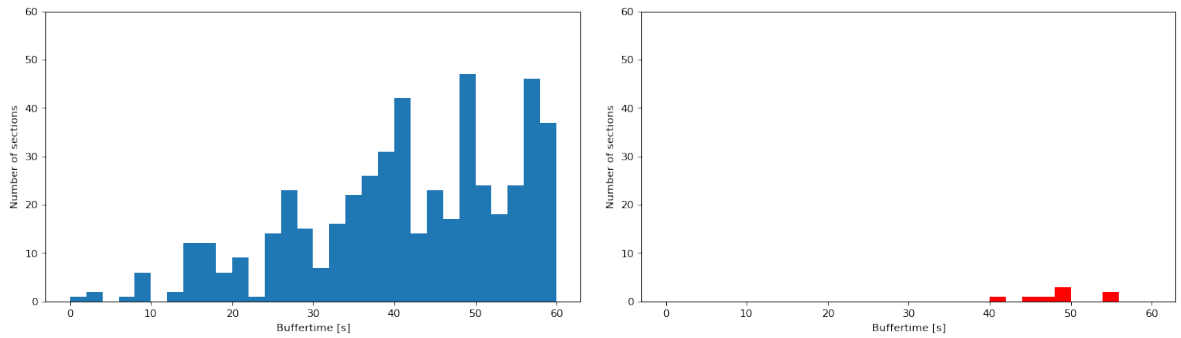


Figure 7.5: Comparison of the buffer times of Fixed and Moving block (Figures 7.4a and 7.4b combined)

In the rest of this chapter the focus is on the sections with the lowest buffer times, since this would indicate a bottleneck. In figure 7.6 the buffer times until 60 seconds for fixed- and moving block are plotted. These graphs show again that the buffer times increase with moving block. In the fixed block situation there are in total 498 blocks with a buffer time lower than 60 seconds, of which 3 are lower than 5 seconds. Probably this is not how it is planned, because in practise the train operation will be stochastic, so this would quickly result in delays. According to the planning norms of ProRail [93] the (minimal) buffer times on the main lines (Hoofdrailnet) are 60 seconds. A reason for the small buffer times in the dataset could be a consequence of the assumptions that are made during the calculations of the buffer times in section 6.3.1.

The train pairs which have the shortest buffer times are D800-B7400, H7400-B3900 and A4400-C3500. These will be analysed in more detail in the next subsections.

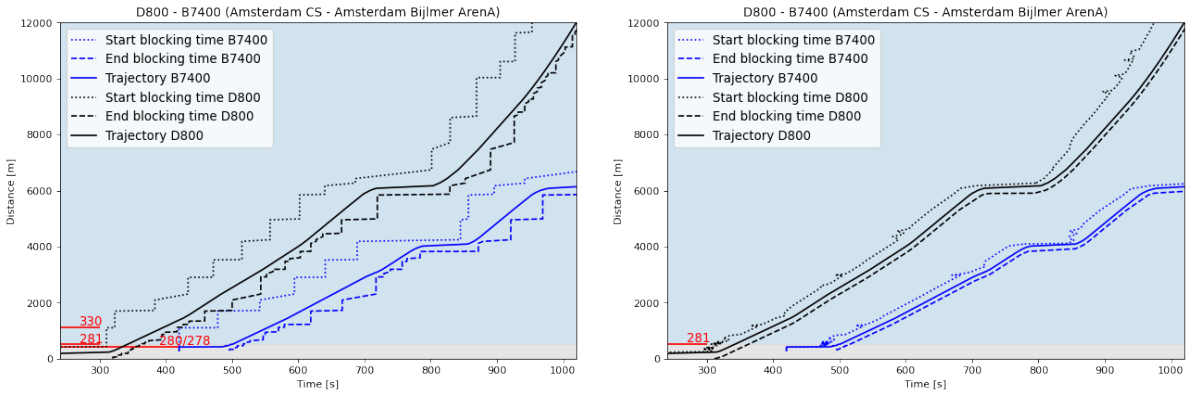


(a) Fixed block **(b)** Moving Block

Figure 7.6: Buffer times at sections lower than 1 minute in the study area

7.2.1. D800-B7400 (Amsterdam CS - Amsterdam Bijlmer ArenA)

In figure 7.7 the blocking time diagrams of trains D800 (IC) and B7400 (Sprinter) can be seen. Both trains depart from Amsterdam Central Station towards Utrecht, but from respectively platforms 4b and 5c. From switch 281 in section 263BT they share the infrastructure (see figure 7.8). After D800 has passed signal 330 the route can be set for B7400 from signal 278. Since sectional route release is applied, section A310T will be released last, so this section is the critical section. When moving block is applied, train B7400 has only to wait for switch 281 to be shifted and locked. This means that there is a possibility to allow train B7400 depart earlier than it is possible with fixed block. In figure 7.7b it can be seen that there is much more buffer time between the two trains compared to fixed block signalling. For this part of the corridor, the blocking time reduces on average 53% with MB for D800 and 63% for B7400. Note that the start of the moving block blocking times show multiple discontinuities, also in figures 7.9b and 7.11b. This caused by switches which need extra setup time.



(a) Fixed block **(b)** Moving Block

Figure 7.7: Time-distance diagrams of train pair D800-B7400 between Amsterdam Amstel and Amsterdam Bijlmer ArenA, with in red the location of the relevant switch and signals

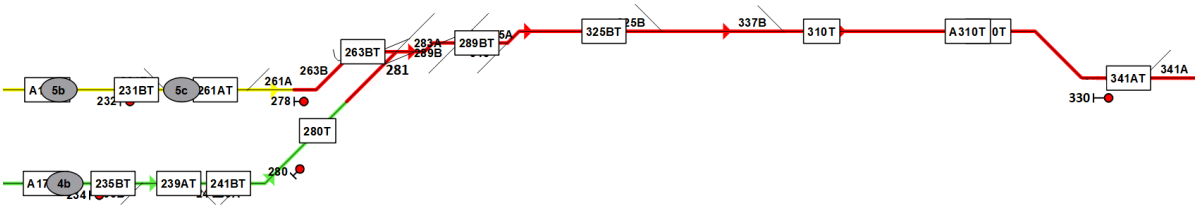
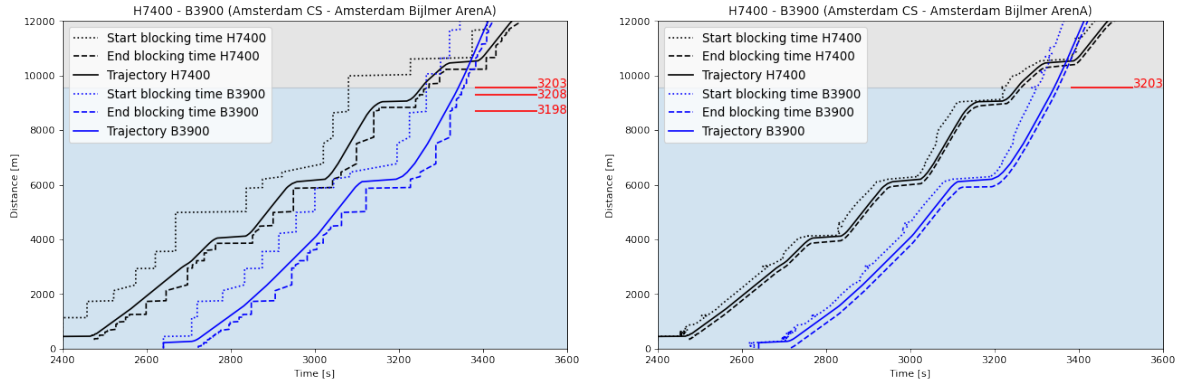


Figure 7.8: Used infrastructure by D800 (green) and B7400 (yellow) at Amsterdam CS, in red the shared infrastructure

7.2.2. H7400-B3900 (Duivendrecht - Amsterdam Bijlmer ArenA)

In figure 7.9 the blocking time diagrams of train pair H7400 (Sprinter) and B3900 (Intercity) can be seen. Both trains depart from Amsterdam CS and make a stop at Amsterdam Amstel. At Amsterdam Bijlmer ArenA the trains diverge and B3900 overtakes H7400. In figure 7.10 the infrastructure between Duivendrecht and Bijlmer ArenA is given in detail. H7400 makes a stop at Bijlmer ArenA at platform 6. After section 3203T has been cleared, the switch can be shifted and locked for B3900. Then B3900 passes Bijlmer ArenA via platform 8 (without a stop) and pass H7400. With fixed block section 3203T is the critical section. With moving block the buffer time will increase. However switch 3203 will still be the critical element, since a virtual block is applied for this location (see figure 7.9b) and time is reserved to shift and lock the switch. For this part of the corridor the blocking time reduces on average 66% for H7400 and 49% for B3900 when moving block signalling is applied. All in all, it can be concluded that the buffer time increases, but the bottleneck location will not change.



(a) Fixed block

(b) Moving Block

Figure 7.9: Time-distance diagrams of train pair H7400 B3900 between Amsterdam CS and Amsterdam Bijlmer ArenA, with in red the location of the relevant switch and signals

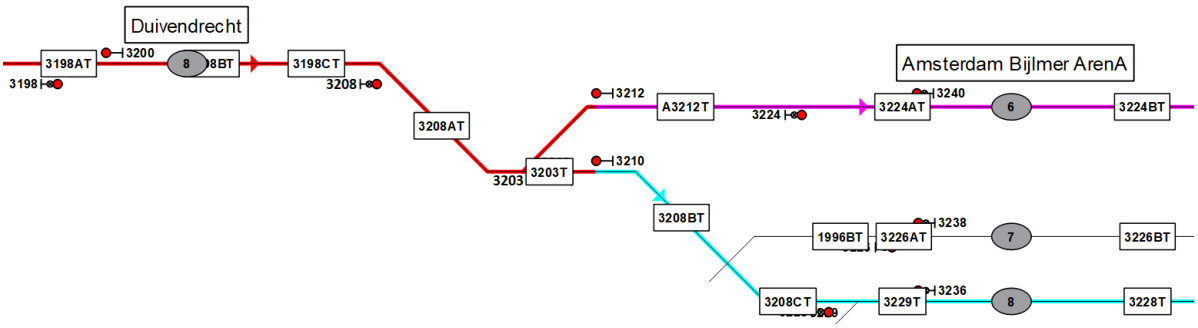


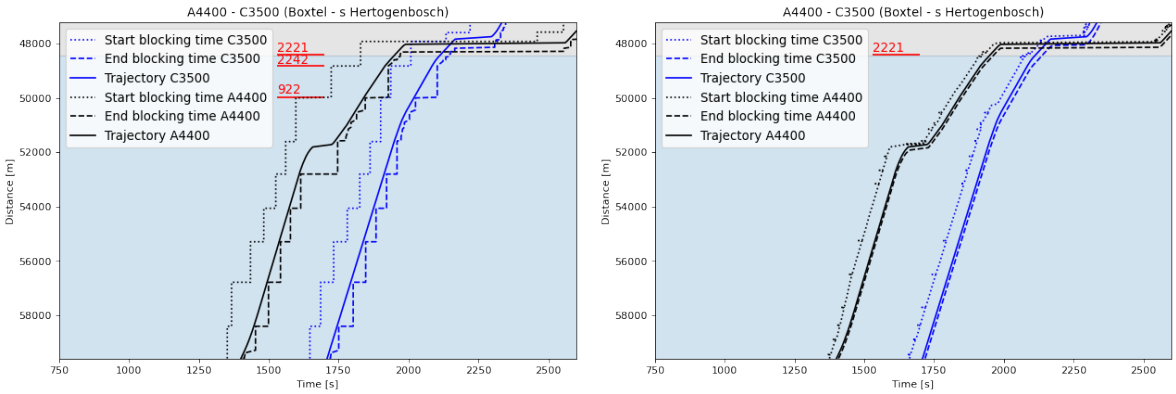
Figure 7.10: Used infrastructure by H7400 (purple) and B3900 (cyan) between Duivendrecht and Amsterdam Bijlmer ArenA, in red the shared infrastructure

7.2.3. A4400-C3500 (Boxtel - 's-Hertogenbosch)

Trains A4400 (Sprinter) and C3500 (Intercity) share tracks between Boxtel (Btl) and 's-Hertogenbosch (Ht). In figure 7.11 the blocking time diagrams of these trains are given. At Ht both trains make a stop – at different platforms – and A4400 continues towards Oss, while C3500 continues in the direction of Utrecht. In figure 7.12 the infrastructure before Ht can be seen in detail, where the trains operate from right to left. With fixed block signalling the critical block is at signal 922, because this block is longer than the distance between signal 2242 and switch 2221 and from signal 2242 sectional route release is applied. With moving block the critical location moves to switch 2221. Here a virtual block is applied and it will take time to shift and lock the switch. In consequence, the blocking time will be longer than on the tracks before the switch, resulting in a bottleneck. For this part of the corridor the blocking time

reduces on average 64% for A4400 and 46% for C3500 when moving block signalling is applied.

In the above three examples the blocking time reduction for sprinters is larger than for intercity trains. When data of the whole study area is analysed, the same pattern is visible. On average the blocking time for sprinter trains reduces with 57%, for intercity trains 49% and high speed lines 45%. This could be explained by the fact that with an (average) higher speed, the approach and running times with fixed block signalling will be lower. This means that the time differences between fixed and moving block will be smaller for faster trains.



(a) Fixed block **(b)** Moving Block

Figure 7.11: Time-distance diagrams of train pair A4400-C3500 between Boxtel and 's-Hertogenbosch, with in red the location of the relevant switch and signals

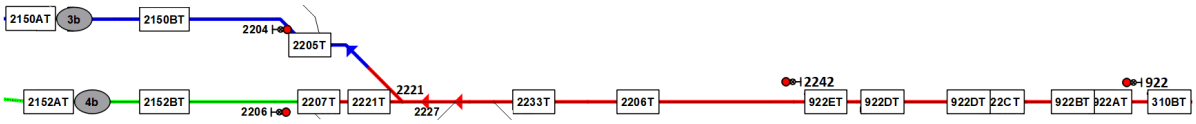


Figure 7.12: Used infrastructure by A4400 (green) and C3500 (blue) before 's-Hertogenbosch (Ht), in red the shared infrastructure. Figure not on scale.

7.2.4. Critical sections

A critical section is a section with the minimal buffer time between two blocking times. For each train pair in the dataset a critical section can be determined. To discover a possible trend in the set of all critical sections, the critical sections can be organized in three different categories: Section ID, name of the area where the sections are located (name of station or junction) and type of area.

In table 7.1 the categories are presented in three tables. In table 7.1a the top 12 stations with most critical sections for fixed block are given. What can be noticed is that the area around Hoofddorp is largely present in this list. Three reasons can be mentioned: first, it is a busy area between Schiphol and Leiden and also the High Speed line between Rotterdam and Schiphol merges here. Second, there are a lot of shunting movements of empty rolling stock to and from the yard, especially at Hfdm and Hfdo. Third, the number of tracks change here from 4 to 2 (and v.v.), so this makes it a physical bottleneck. Besides Hoofddorp, the top 12 is dominated by large stations: Utrecht CS, Arnhem CS, Eindhoven and Amsterdam CS. This is mainly because a lot of trains serve these stations. Note that this case study considers only 4 corridors of the Netherlands, so these are not necessarily the most critical stations in the Netherlands.

When a look is taken at the type of area, it can be seen that the amount of critical locations at stations increases, especially at stops (Dutch: Haltes). The difference between stops and passenger stations is that at passenger stations it is possible to change tracks via switches, while at stops this is not possible.

Table 7.1: Number of critical sections in the study area deviated for different categories

(a) Sorted per area (DRP)

	FB	MB
Utrecht CS, Ut	42	32
Arnhem CS, Ah	41	45
Eindhoven, Ehv	36	32
Hoofddorp m., Hfdm	36	40
Hoofddorp yard, Hfdo	32	35
Amsterdam CS, Asd	30	33
s-Hertogenbosch, Ht	26	26
Geldermalsen, Gdm	20	22
Sittard, Std	20	22
Hoofddorp, Hfd	16	6
Maastricht, Mt	15	13
Dijksgracht west, Dgrw	13	2

(b) Sorted per section (6 train pairs and higher)

Fixed block			Moving block		
Section	DRP[1]	N[2]	Section	DRP[1]	N[2]
1104BT	Hfd	8	1181BT	Hfdo	8
1183BT	Hfdo	8	141AT	Nm	8
141AT	Nm	7	1183BT	Hfdo	8
2135T	Ah	6	1125T	Hfdm	7
1181BT	Hfdo	6	537T	Gdm	6
1177BT	Hfdo	6	115BT	Ehv	6
115BT	Ehv	6	744BT	Asdz	6
A1162T	Hfdm	6	754T	Asdz	6
754T	Asdz	6	1129BT	Hfdm	6
			1523T	Dvaw	6

(c) Sorted per type of area

	Fixed block		Moving block		n[3]	ΔBT [s][4]	ΔBT [%][4]
Station							
Terminal (Ut, Asd)	72	10,6%	65	9,5%	21	65,3	17,0
Passenger station	370	54,3%	383	56,2%	160	67,0	32,0
Stop (NL: Halte)	28	4,1%	50	7,3%	15	91,5	77,9
Border station	2	0,3%	3	0,4%	-		
Junction							
Connection	33	4,8%	55	8,1%	12	26,5	10,3
Crossover	57	8,4%	59	8,7%	21	64,8	49,6
Open line	87	12,8%	25	3,7%	-		
Cargo yard	32	4,7%	39	5,7%	34	63,2	37,4
Bridge	-		2	0,3%	-		
Sum	681		681		263		

[1]Dienstregelpunt, name of area; [2]Number of train pairs; [3]Number of train pairs for which the critical section doesn't change; [4]Absolute/relative average difference between FB- and MB Buffer time of n;

Also the number of critical locations at junctions increases, while there is a large decrease in open line areas. With the examples in subsections 7.2.1 to 7.2.3 in mind, one can conclude with moving block a large part of the critical locations move towards switches, located in station and junction areas. There are also a few exceptions, where the critical block for two trains remain in open lines. Most of these cases concerns a sprinter or stops train departing after an IC. Since a sprinter train accelerates faster than an IC, the buffer time will reduce for the first couple of kilometers. However, at a certain point the

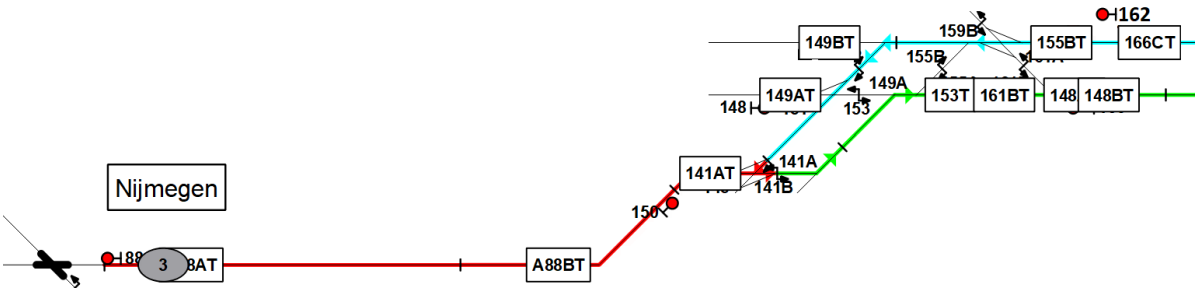


Figure 7.13: Rolling stock turning around at Nijmegen (Nm). In red shared infrastructure

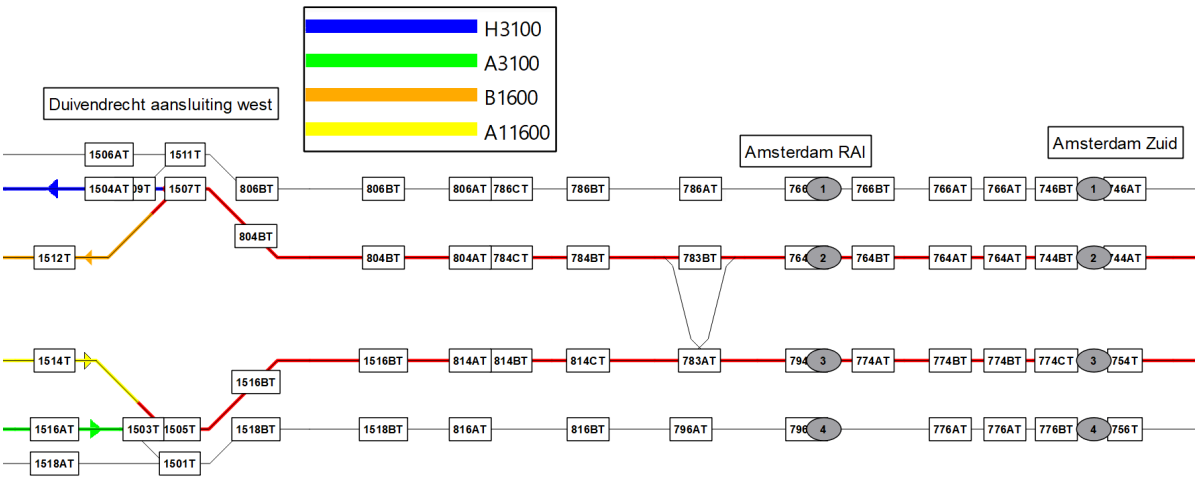


Figure 7.14: Intercity trains merging and diverging between Duivendrecht and Amsterdam South. In red shared infrastructure

sprinter train has to brake for a stop, while the IC continues. This results in a critical location in an open line area.

When the focus is on sections individually, Hoofddorp is again present in the list of sections which are most critical (Table 7.1b). Two sections outside of Hoofddorp, which are both present in the list of fixed- and moving block, will be highlighted. First, section 141AT, which is in extension to platform 3 at Nijmegen (Nm). The infrastructure layout is presented in figure 7.13. For multiple lines this is the end/start station, so actually rolling stock turn around at this station to operate another line. With moving block, the *new* train pair at this section is H3100-O3000. With fixed block the critical section for this train pair was A88BT, which is next to 141AT. Secondly, section 754T, which is located at platform 3 at Amsterdam Zuid (Asdz) (see figure 7.14).This is the critical section for IC trains coming from Amsterdam CS (via Diemen) and Utrecht CS. This also applies in the opposite direction, where 744BT is the critical section (platform 2). However, with fixed block the critical section is 764CT at station Amsterdam RAI.

Considering a reduction of the blocking times and possible shift of bottlenecks, the minimum buffer time – without timetable compression – between two trains increases on average by 75 seconds. This corresponds to an average increase of 60%. Unfortunate it is not possible to make a direct distinction per type of area. Because the critical section of two trains may change between fixed- and moving block, the set of train pairs which have a critical section in a certain type of area are not the same. This means these cannot be compared. Only train pairs for which the critical section won't change can be taken into account for this comparison. In total, this applies for 263 train pairs. In table 7.1a this is divided per type of area. For example, 160 of them have a critical section in a station area, where the average buffer time increases by 67 seconds (31%). It can be concluded that on average the buffer time increases in all type of areas.

7.3. Roberto

Besides the collected data, FRISO generates also an input file for Roberto. Roberto is used to calculate the minimal headway times between two trains and to determine the critical block in undisturbed circumstances. In the current version of Roberto (5.1.0.6) the user can choose between two different *signalling systems*: **NS'54** and Driving On Braking Distance (**ORR**, Dutch: Op Remweg Rijden). In this second option the minimal headway is equal to the clearing time plus the approach time. Compared to moving block, ORR does not include the route setup-, reaction-, release time and extra setup time for switches. These time components are all assumed to be constant and sum up to 25s (1+9+1+14). This means that ORR would be comparable to a moving block signalling system, so the case study results can be compared to the output of Roberto.

In this section the capabilities of Roberto will be shown. Roberto can identify critical blocks by compressing the train path of the following train as close as possible to the leading train. This feature can be used to validate the results in section 7.2. Also, using the calculated blocking times, the minimum headway times can be determined. These can be compared to the minimum headway times of Roberto, both with fixed- as moving block. This way capacity gains of moving block can be determined and at the same time experience with Roberto will be gained, so it is possible to make recommendations how the model to estimate moving block blocking times can be implemented in Roberto and answer the research question how the defined method can support ProRail.

One important thing to note is that Roberto makes use of the Protected Zone Model for NS'54 instead of the Blocking time theory. The basic idea is that behind every train, the signalling system provide a certain zone to protect that train against following trains. In case of NS'54 this zone is a block length between two signals. Figure 7.15 visualises the model. The protected zone can be divided into two parts. First, there is the part of absolute protection (in red). This part consists of all track sections the protected train has exclusive authority for. Second, there is the part in which other trains are forced to slow down (in yellow) to prevent them from running into the part of absolute protection. The protected zone is at its maximum (fig. 7.15b) when the train is going to release the block. When the block has been released, the zone is at its minimum (fig. 7.15c). The main difference between the protected zone model and the blocking time theory is that the approach time is added at the end of the occupation of the block section instead of the beginning. This means that the minimal headway time fully depends on the leading and how fast it clears a block. In the blocking time theory the approach time depends on the speed of the following train. Because of this, one will see time difference between the minimal headway time of Roberto and the model.
Although the protected zone model works well for traditional signalling systems, such as NS'54, it has also its limitations. It cannot handle cab signalling systems in which the signalled distance depends on the actual speed of the train, such as moving block systems [32, p.33-34]. Therefore, the protected zone model is not used for ORR.

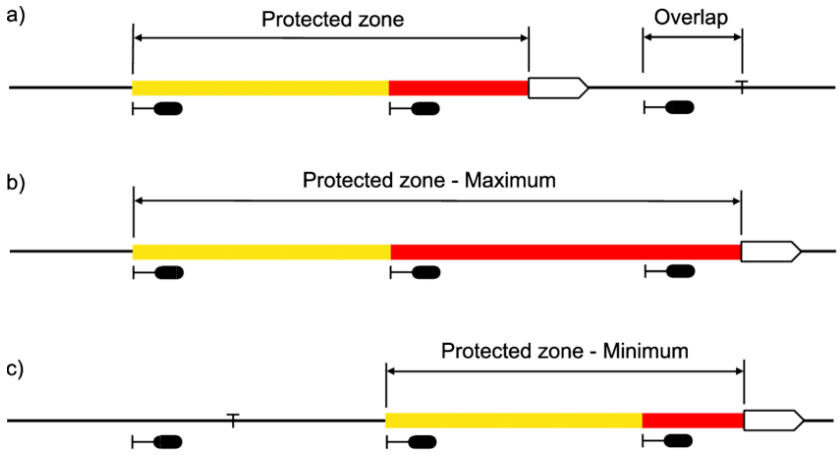


Figure 7.15: The protected zone model (Source: Hansen and Pachl [32], p.34)

In the following three subsections the same train pairs will be analysed as in the previous section. The results of the defined and developed model in this thesis, in the rest of this chapter referred to as *the model*, will be compared to the outcome of Roberto. The layout of the infrastructure of each example can be found in section 7.2. Unfortunate with Roberto it is not possible to determine the minimum headway times for all trains at once. Given the size of the study area, it will be very time consuming to apply Roberto for each train pair. Therefore it is not possible to give any main findings or conclusions between the results of the defined model and Roberto, i.e. the difference in buffer times or minimum headway times.

7.3.1. D800-B7400 (Amsterdam CS - Amsterdam Bijlmer ArenA)

In figure 7.16 the compressed time-distance diagram is given for train pair D800-B7400 operating under NS'54. The trains are presented by three lines: sight-and-reaction-time line, front of the train and back of the train. The colors of the signals are presented by a red, yellow and orange line. When the train has entered a block, the signal turns red. After the block has been released, the signal will turn yellow and the previous signal turns green. The orange lines present a yellow light in combination with a number on the signal. The blue background shows that trains use the same infrastructure on that part of the track. On the left side of the figure names and locations of the signals are given, of which the blue name is the critical signal according to Roberto.

In figure 7.17 a similar figure is given, but in this case ORR is applied. Here the first blue line presents the braking distance (or approach time). As one could see (also in figures 7.19 and 7.21), there are some errors in the calculation of the braking distance, where it sometimes falls back to the last station it has passed. However, in general the braking distance line has a very similar pattern as the start-of-the-blocking-time line with moving block, for example in figure 7.7. One major difference is that in moving block there is time reserved for switches, which is not considered in ORR.

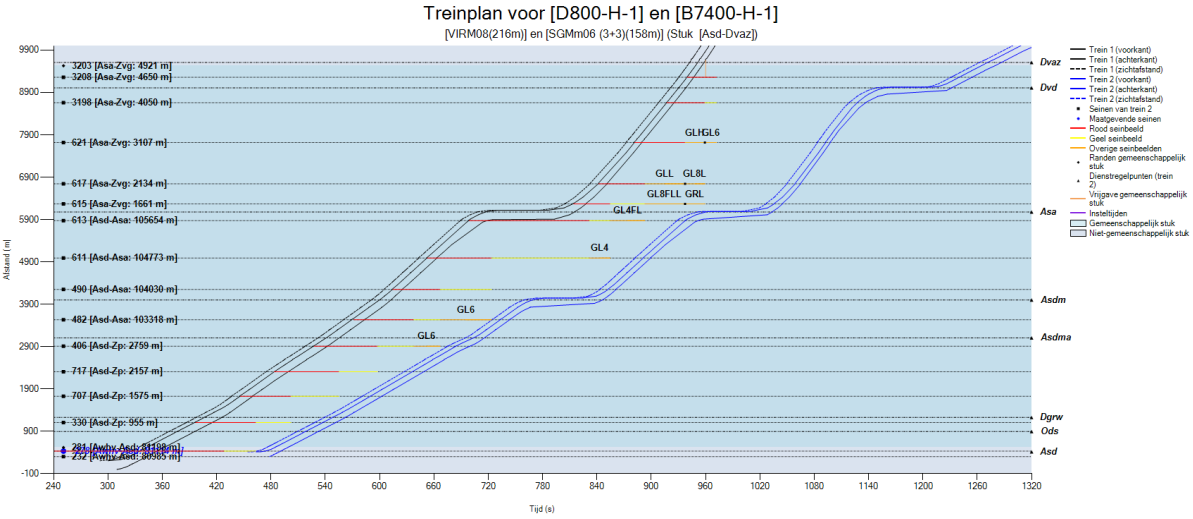


Figure 7.16: Compressed train paths of trains D800 (Sprinter) and B7400 (Intercity) operating under NS'54, made in Roberto

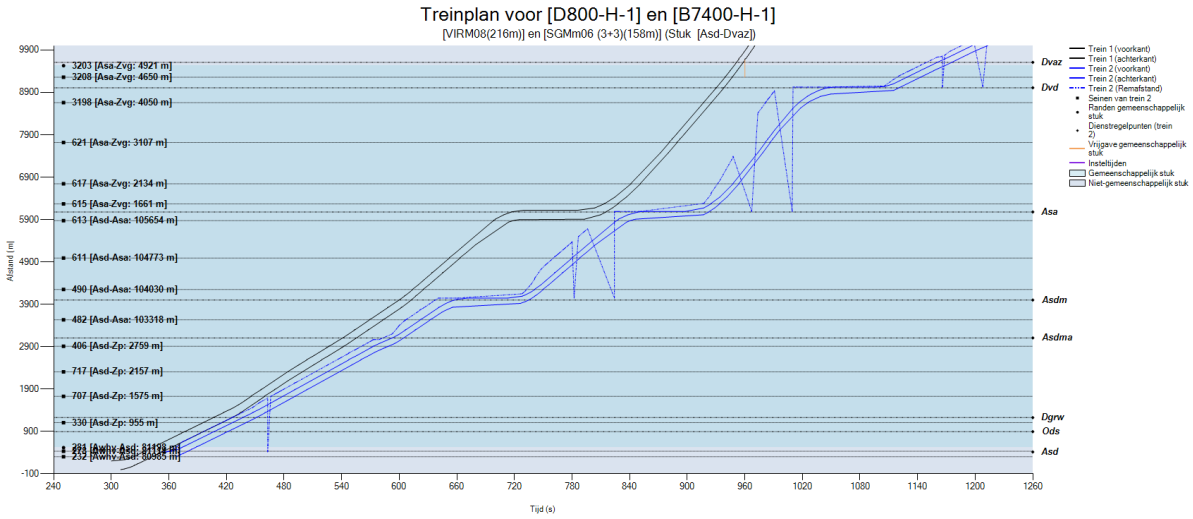


Figure 7.17: Compressed train paths of trains D800 (Sprinter) and B7400 (Intercity) operating on braking distance (ORR), made in Roberto

In table 7.3 the corresponding headway- and buffer times of figures 7.16 and 7.17 are given for all signals that the trains pass, including the first switch. The abbreviations of the column headers and terminology used in Roberto are explained in table 7.2. In Roberto, the local- and global headway time (HWT) is calculated for a compressed situation. This means that the buffer time (BT) at each signal in Roberto can be determined by the difference between the local- and global headway time. Besides this, the shift from an uncompressed, e.g. the timetable, to compressed situation can be calculated by:

$$T_{Shift} = HWT - Global\ HWT \tag{7.1}$$

In the developed model of this research the buffer time for each section is calculated by subtracting the end of the blocking time of the leading train – in this case B800 – from the start of the blocking time of following train, here B7400. It is important to note that this buffer time is for an uncompressed situation, while in Roberto it is in a compressed situation. In other words, when Roberto and the model would produce the same results, then:

$$BT_{model} = T_{Shift;Roberto} + BT_{Roberto} \tag{7.2}$$

When the buffer times of the model has been determined, the minimal headway time can be calculated by the difference of the headway time in the timetable and the minimal buffer time of the whole line section.

According to Roberto with fixed block B7400 could be shifted by 15 seconds towards D800, while the model indicates a possible shift of 3 seconds. According to the data – used as input in the model – B800 releases the critical block at 417s, while B7400 sets the route at 420s. It is likely that there is a difference in the simulation settings between Friso and Roberto, but it couldn't be found what this difference is. When ORR would be applied, B7400 could be shifted by 127 seconds. In case of moving block the shift would be 111,5 seconds. This difference has mainly to do with shifting and locking switch 281. Besides this, there are (small) differences between the calculated approach time in the model and in Roberto (ΔAT). These differences – including the difference in clearing time (ΔCT) – can be seen in table 7.4. In Friso the braking distance (in meters) is calculated with a constant deceleration, so any changes in gradient are not taken into account. This leads to small differences in the results. However, also larger differences can be found in the table, such as the approach time for signal 3198 (15s.). Probably the model has a large error when it calculates the approach time for this signal. Unfortunate Roberto doesn't produce one output file with all approach times of the whole study area, so it will be a very time consuming to gather the buffer times of all train pairs and perform a full comparison study.

Table 7.2: Abbreviations of tables 7.3 to 7.8.

Abbreviation	Definition
T	Type of infrastructure element: Switch, Open line signal, Interlocking signal
HWT	Headway time according to the timetable
Local HWT	Minimal local headway time by considering only that signal, i.e. red and yellow time
Global HWT	Minimal Global Headway time. Term used in Roberto and synonym for minimal headway time. Global HWT = Local HWT + $BT_{Roberto}$
Min. HWT	Minimal headway time. Min HWT = HWT $-min(BT_{Model})$
BT	Buffer time
Approach time	Time to run through the braking distance (Dutch: remtijd)
Clearing time	Calculation of the clearing time in Roberto: Clearing time = HWT – Approach time – BT. In the model it is calculated accordingly to chapter 4.
ΔCT	$= Clearing\ time_{Model} - Clearing\ time_{Roberto}$
ΔAT	$= Approach\ time_{Model} - Approach\ time_{Roberto}$

Table 7.3: Headway- and buffer times for train pair D800-B7400 between Amsterdam CS and Amsterdam Bijlmer ArenA

			Fixed Block					ORR			Moving Block	
			Roberto			Model		Roberto			Model	
Signal Name	T	HWT	Local HWT	BT	Global HWT	BT	Min. HWT	Local HWT	BT	Global HWT	BT	Min. HWT
281	S	158,3	141	2	143	3,0	155,3	141	-109	32	111,5	46,8
330	I	158,3	116	27	143	19,7	155,3	116	-85	31	114,9	46,8
707	O	160,7	119	26	145	41,7	157,7	119	-85	34	121,9	49,2
717	O	169,1	124	30	154	50,8	166,1	124	-81	43	129,1	57,6
406	I	174,1	120	38	158	53,8	171,1	120	-73	47	133,4	62,6
482	I	180,7	107	58	165	61,9	177,7	107	-53	54	138,0	69,3
490	I	255,0	119	120	239	179,5	252,0	119	9	128	228,0	143,5
611	O	255,6	212	28	240	137,5	252,6	212	-83	129	212,6	144,1
613	O	256,0	204	36	240	66,8	253,0	204	-75	129	209,9	144,5
615	O	239,1	133	90	223	89,9	236,1	133	-21	112	208,8	127,7
617	O	234,1	127	91	218	150,2	231,1	127	-20	107	204,3	122,7
621	O	227,9	100	112	212	137,4	224,9	100	1	101	186,8	116,4
3198	I	227,3	65	147	212	149,1	224,3	65	36	101	187,1	115,8
3208	I	317,3	42	259	301	124,1	314,3	42	148	190	287,9	205,8

Table 7.4: Comparison of approach- and clearing times of train pair D800-B7400 between Roberto and the model

		Roberto (ORR)				Model (MB)			
Signal Name	Type	Global HWT	Approach time	BT	Clearing time	Clearing time (D800)	Approach time (B7400)	ΔCT	ΔAT
330	I	31	12	1	18	19,4	12,0	1,4	0,0
707	O	33	13	7	13	14,1	12,6	1,1	-0,4
717	O	42	13	14	15	15,4	12,6	0,4	-0,4
406	I	47	13	19	15	15,5	13,2	0,5	0,2
482	I	53	16	24	13	14,1	16,6	1,1	0,6
490	I	128	9	108	11	11,1	3,8	0,1	-5,2
611	O	128	20	98	10	11,1	19,9	1,1	-0,1
613	O	129	18	91	20	20,2	13,9	0,2	-4,1
615	O	112	12	86	14	14,5	3,8	0,5	-8,2
617	O	107	17	81	9	9,9	8,0	0,9	-9,0
621	O	101	26	67	8	8,2	20,9	0,2	-5,1
3198	I	100	35	57	8	8,2	20,0	0,2	-15,0
3208	I	190	11	172	7	7,5	9,9	0,5	-1,1

With such a file it would become easier to find any pattern in the performance of the model compared to Roberto. Despite these errors, in all cases the critical location is at switch 281.

7.3.2. H7400 - B3900 (Duivendrecht - Amsterdam Bijlmer ArenA)

In figures 7.18 and 7.19 the time distance diagrams of train pair H7400-B3900 are shown for respectively NS'54 and ORR. This train pair is a good example how NS'54 limits the capacity and how a moving block system could improve this. The critical block is at signal 617, located between two stations where trains operate at full speed. The length of the block prevents a possible shorter headway time. In case of ORR the critical location moves to the end of the line section at Duivendrecht (Dvd), although at station Amsterdam Amstel (Asa) the buffer time is also very short. Note that there is no signal at the end of this line section, so Roberto doesn't give the buffer- and headway time in the output. In consequence, the buffer- and headway at that location is not given in table 7.5, but in figure 7.19 it can be seen that the critical location is at switch 3203.

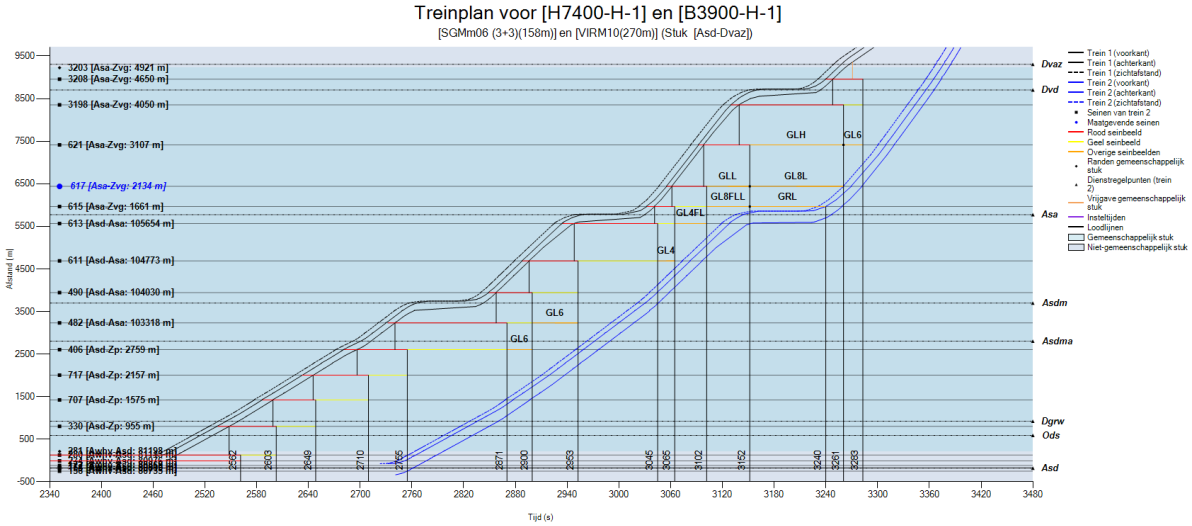


Figure 7.18: Compressed train paths of trains H7400 (Sprinter) and B3900 (Intercity) operating under NS'54, made in Roberto

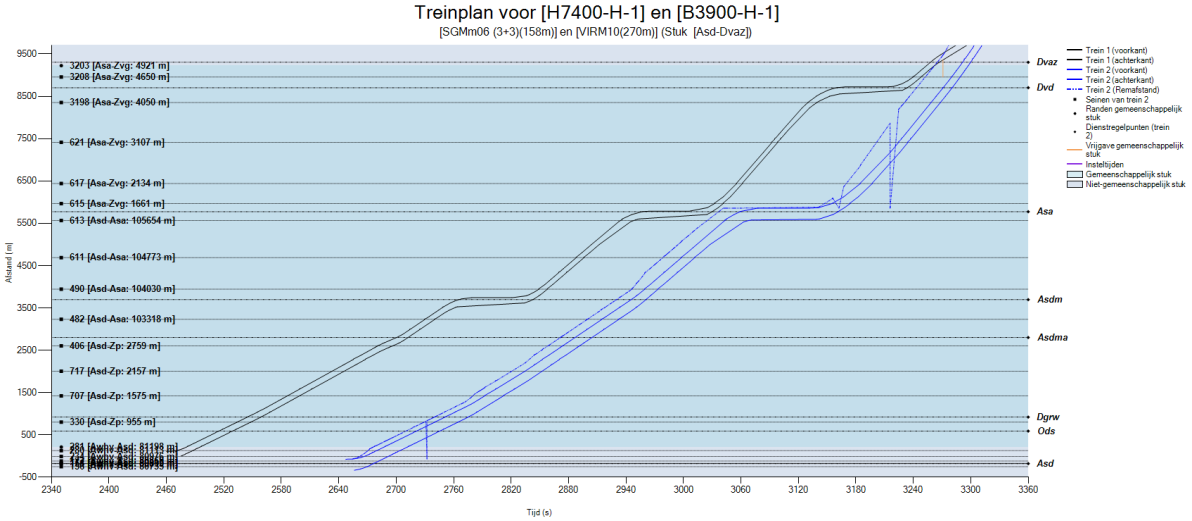


Figure 7.19: Compressed train paths of trains H7400 (Sprinter) and B3900 (Intercity) operating on braking distance (ORR), made in Roberto

When the results of the model are compared to Roberto, first it can be noticed that the critical block is different. It is likely that this is because NS'54 makes use of signals with speed limits, which was not taken into account in the calculation of the fixed block blocking times. Secondly, with fixed block the minimal (global) headway is in Roberto on average 36s longer than in the model, which could be a result of the different determinations of the critical block. Third, both with ORR as moving block the critical location moves to the end of the line section at switch 3203. On average there is a difference of 27 seconds between the minimal (global) headway of ORR and moving block, which is close to the expected difference of 25 seconds. However, this difference depends on the minimal headway time at the critical block, which depends on the performance of the model. In other words: when the model has a (large) error at the critical block, it will immediately have effect on the minimal headway time on the rest of the line section.

Table 7.5: Headway- and buffer times for train pair H7400-B3900 between Amsterdam CS and Amsterdam Bijlmer ArenA

			Fixed Block					ORR			Moving Block	
			Roberto			Model		Roberto			Model	
Signal Name	T	HWT	Local HWT	BT	Global HWT	BT	Min. HWT	Local HWT	BT	Global HWT	BT	Min. HWT
281	S	259,2	139	153	292	157,2	256,6	139	68	207	208,0	234,1
330	I	259,2	124	168	292	121,5	256,6	124	83	207	220,9	234,1
707	O	259,0	133	159	292	134,5	256,3	133	74	207	222,9	233,9
717	O	254,3	131	156	287	135,8	251,7	131	71	202	215,6	229,2
406	I	245,7	199	80	279	131,1	243,1	199	-5	194	201,8	220,6
482	I	237,6	232	39	271	55,6	235,0	232	-46	186	199,4	212,5
490	I	163,4	113	83	196	91,7	160,7	113	-2	111	128,0	138,3
611	O	162,7	187	9	196	51,1	160,1	187	-76	111	122,7	137,6
613	O	161,9	140	55	195	2,6	159,3	140	-30	110	119,0	136,8
615	O	175,4	79	130	209	21,7	172,8	79	45	124	150,0	150,3
617	O	182,0	215	0	215	96,5	179,4	215	-84	131	155,3	156,9
621	O	185,1	200	18	218	84,9	182,4	200	-67	133	146,2	160,0
3198	I	181,3	162	52	214	16,4	178,7	162	-33	129	132,4	156,2
3208	I	93,1	53	73	126	16,4	90,4	53	-12	41	43,5	68,0
3203	S	87,9									25,1	62,8

Table 7.6: Comparison of approach- and clearing times of train pair H7400-B3900 between Roberto and the model

		Roberto (ORR)				Model (MB)			
Signal Name	Type	Global HWT	Approach time	BT	Clearing time	Clearing time (D800)	Approach time (B7400)	ΔCT	ΔAT
330	I	207	12	182	13	14,2	12,0	1,2	0,0
707	O	207	13	181	13	12,6	11,4	-0,4	-1,6
717	O	202	14	175	13	12,6	14,0	-0,4	0,0
406	I	193	16	163	14	15,5	16,4	1,5	0,4
482	I	185	16	160	9	10,3	15,8	1,3	-0,2
490	I	111	14	89	8	8,8	14,6	0,8	0,6
611	O	110	20	83	7	8,1	19,9	1,1	-0,1
613	O	110	19	77	14	14,0	16,9	0,0	-2,1
615	O	123	8	106	9	9,0	4,4	0,0	-3,6
617	O	130	15	109	6	6,1	8,6	0,1	-6,4
621	O	133	23	104	6	6,0	20,9	0,0	-2,1
3198	I	129	28	92	9	9,4	27,5	0,4	-0,5
3208	I	41	28	5	8	8,3	29,3	0,3	1,3

7.3.3. A4400-C3500 (Boxtel - 's-Hertogenbosch)

In figures 7.20 and 7.21 the compressed time distance diagrams of train pair A4400-C3500 between Boxtel (Btl) and 's-Hertogenbosch (Ht) are shown for respectively NS'54 and ORR.

According to Roberto the critical block is at signal 310. The main reason is that the blocks between signals 310 and 922 and between 922 and 2242 are relatively long, resulting in a long blocking time. Besides this, when A4400 has passed switch 2221 (at 5582s), it will be shifted and locked. This means that signals 2242 and 922 can turn green at the same time. Consequently, the blocks after signals 922 and 2242 are relatively short occupied, which makes signal 310 the critical block. However, this works only for this specific train pair and because Roberto uses the protected zone model. At Ht both trains use different tracks, so they can have a short headway time before that station. In comparison, in the blocking time theory the blocking times will be calculated independent of other trains on the tracks.

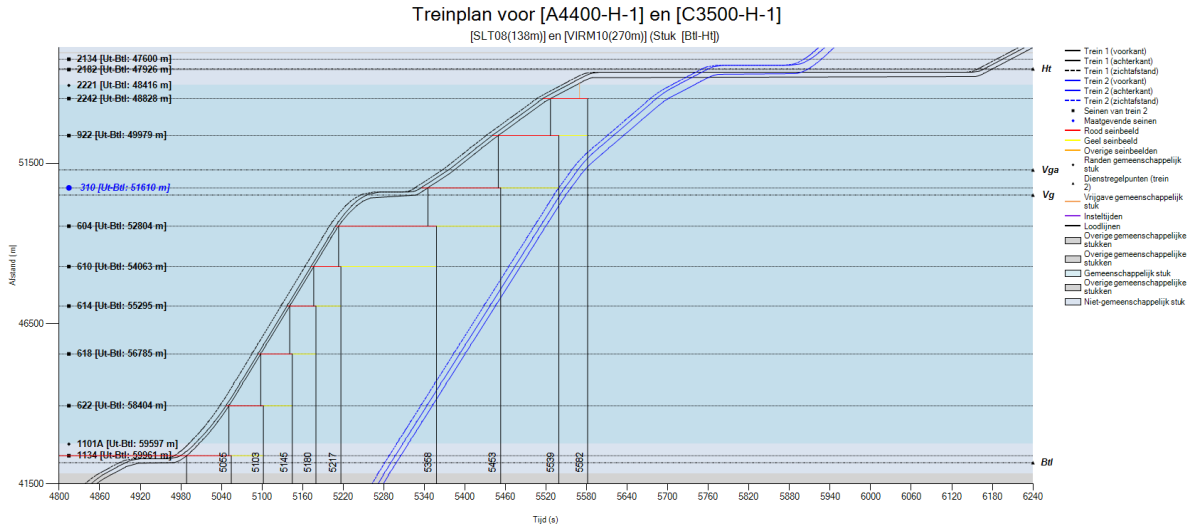


Figure 7.20: Compressed train paths of trains A4400 (Sprinter) and C3500 (Intercity) between Boxtel and 's-Hertogenbosch operating under NS'54, made in Roberto

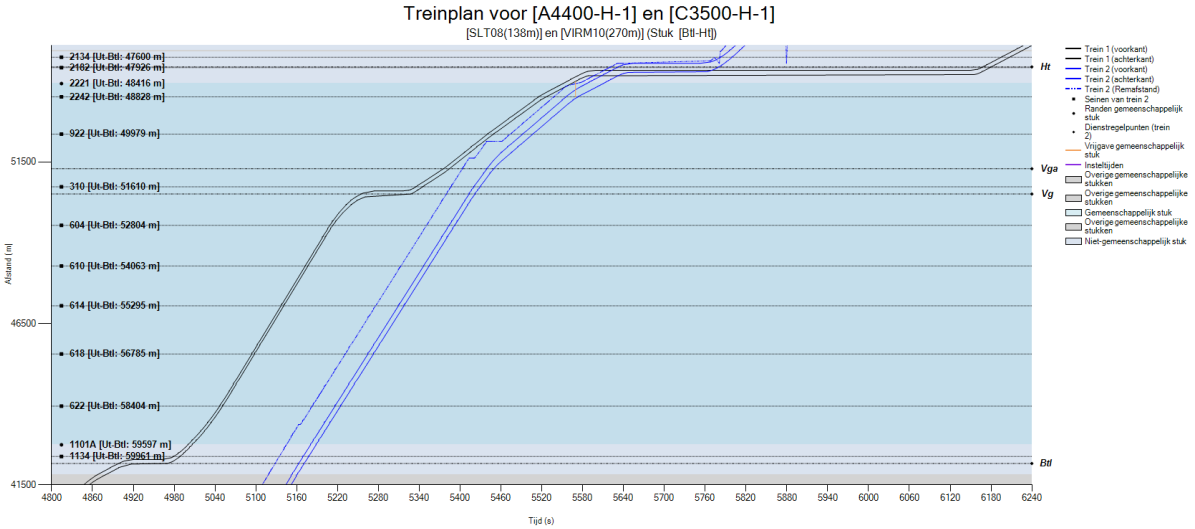


Figure 7.21: Compressed train paths of trains A4400 (Sprinter) and C3500 (Intercity) between Boxtel and 's-Hertogenbosch operating on braking distance, made in Roberto

According to this theory signal 922 is the critical block (see table 7.7 and figure 7.11). On the other hand, one could also argue that signal 310 is yellow because signal 922 is red, so signal 922 would be the bottleneck.

Both with ORR as moving block the critical location moves to switch 2221, which is the end of the line section. Note that Roberto only calculates the buffer- and headway times for the first switch of the line sections and all signals, so the numbers of Roberto for switch 2221 are not included in table 7.7. Between the minimal (global) headway time of ORR and moving block is an average difference of 17s, instead of the expected 25 seconds. For this train pair it is harder to explain this difference. The critical location is at switch 2221, but Roberto gives only the approach time for signals. Looking at tables 7.4 and 7.6 an error of 7 seconds could happen, however in table 7.8 the time differences are all within 1 second. Unfortunate due to missing data it is not possible to explain the reason with certainty of the minimal headway time difference.

Table 7.7: Headway- and buffer times for train pair A4400-C3500 between Boxtel and 's-Hertogenbosch

			Fixed Block					ORR			Moving Block	
			Roberto			Model		Roberto			Model	
Signal Name	T	HWT	Local HWT	BT	Global HWT	BT	Min. HWT	Local HWT	BT	Global HWT	BT	Min. HWT
1101A	S	308,3	122	186	308	222,1	299,3	122	59	181	241,4	198,5
622	I	297,6	108	190	298	186,5	288,6	108	63	171	247,4	187,8
618	O	298,7	95	204	299	190,2	289,7	95	76	171	248,4	188,9
614	O	300,5	89	212	301	203,3	291,5	89	84	173	251,1	190,7
610	I	302,0	195	107	302	211,3	293,0	195	-20	175	252,3	192,2
604	I	303,4	254	50	304	116,2	294,4	254	-77	177	253,3	193,6
310	I	213,1	213	0	213	74,2	204,1	213	-127	86	157,1	103,3
922	O	179,7	151	30	181	9,0	170,8	151	-98	53	139,9	70,0
2242	O	168,3	75	93	168	80,1	159,3	75	-34	41	126,6	58,5
2221	S	162,8									109,8	53,1

Table 7.8: Comparison of clearing- and buffer times of train pair A4400-C3500 between Roberto and the model

		Roberto (ORR)				Model (MB)			
Signal Name	Type	Global HWT	Approach time	BT	Clearing time	Clearing time (D800)	Approach time (B7400)	ΔCT	ΔAT
622	I	170	34	132	4	4,4	33,9	0,4	-0,1
618	O	171	34	134	3	4,0	34,4	1,0	0,4
614	O	173	34	136	3	4,0	33,5	1,0	-0,5
610	I	174	34	137	3	4,0	33,7	1,0	-0,3
604	I	176	34	138	4	4,4	33,7	0,4	-0,3
310	I	85	34	42	9	9,9	34,0	0,9	0,0
2242	O	41	19	12	10	10,9	18,8	0,9	-0,2

8

Discussion

In this chapter three topics are discussed. First, the limitations of the model to estimate the blocking times for moving block. Second, another way to calculate the approach time is discussed. Last but not least, it will be discussed if realised data can be used as input data for the model.

8.1. Limitations of the model

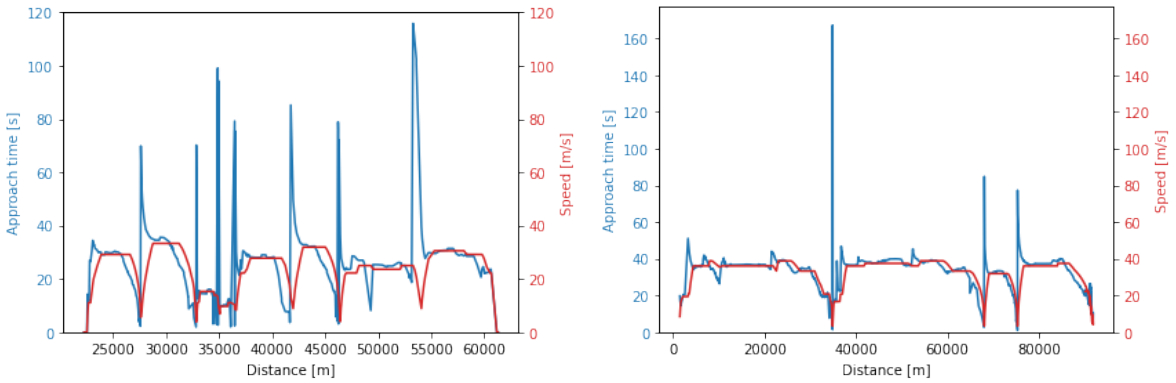
The model as described in chapter 4 has a couple of limitations:

- **Disturbances and delays**
  Since the minimal headway time is different between fixed- and moving block signalling, with both systems trains will operate differently in case of disturbances and/or delays. With moving block the blocking time is shorter, so tracks can be released earlier. This means that a following train can also continue earlier than with fixed block, resulting in a complete different operations [84]. However, this model only calculates the buffer times, so it will not change the trajectory of trains. Therefore this model is suited for capacity studies and planning support, but should not be applied for disturbed situations.
- **Quality of dataset**
  In chapter 5 the model was verified by a dataset produced with EGTRAIN, a time-driven simulation model. Then, the model was applied to a dataset produced with FRISO, an event-driven simulation model. This difference is not important for the model, since both datasets contains all the needed information. However, the interval of the data is more important. If the intervals are smaller, the trajectory and course of the gradient can be reconstructed more precise, resulting in a much more precise estimation of the blocking time.
- **Approach time calculation at stops**
  When the model would be simplified by ignoring gradients and assuming constant braking, the approach time could be calculated by:

$$t_{Approach} = \frac{V}{2a} \tag{8.1}$$

  where: V = Speed [m/s]
  a = Deceleration (braking rate) [m/s^2]

  When a constant braking of $0.5 m/s^2$ is assumed, which was also the braking rate of trains in the case study, one could say that $t \sim V$. This means that when the approach time and speed would be plotted in the same graph, both lines would overlap. This has been done in figure 8.1 for a sprinter and intercity train.
  In the plots there are peaks visible when a train brakes for a stop at a station. In the figures it seems that the trains do not brake to full standstill, but actually they do. FRISO makes a logging when it passes an infrastructural element, which means the train is moving. So all moments when



(a) A7300 (SPR) from De Haar Junction to Breukelen

(b) O3000 (IC) from Arnhem to Amsterdam Amstel

Figure 8.1: Approach time and speed diagrams for two trains

a train stand still are not captured in this trajectory dataset. When a train stops, the passage interval time between two elements increases. The interval time is directly used in the calculation of the approach time, resulting in errors such as the peaks in figure 8.1.
Although the standstill moments are not present in the trajectory dataset, the arrival, departure and passage times at every station were captured in a separate dataset. This could have been used to overcome this problem. According to author's observation it hadn't an impact on the calculation of the blocking- and buffer times. Also in the blocking time diagrams, such as figures 7.7b, 7.9b and 7.11b, this effect was not directly noticeable.
Remember that the dwelling time at stations should have been included in the running time (see section 4.1). Because the arrival and departure times were not included, the dwelling time at stations (for the running time) was not included in the calculations. Taking this into account, the long approach times were canceled out by missing the dwelling times. To conclude, it could be that there is an error of a couple of seconds of the blocking times at stations, but this hadn't an affect on the identification of bottlenecks.

8.2. Calculation of approach time

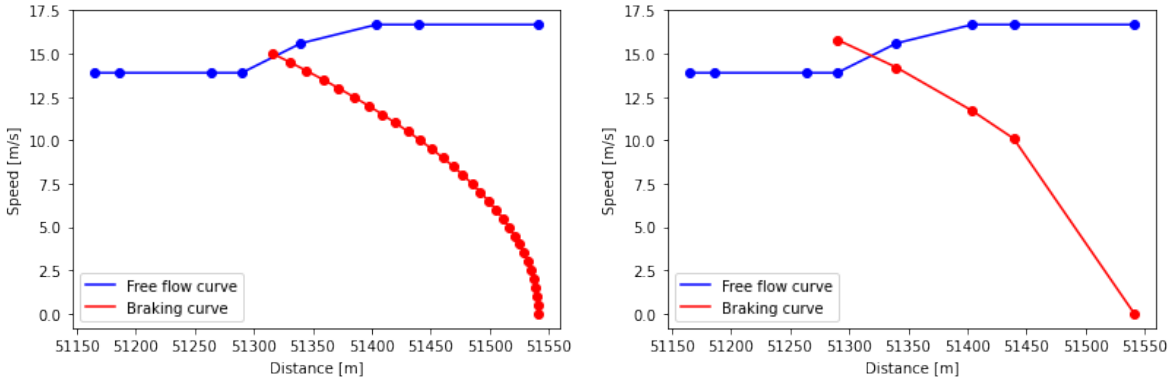
Remember that the approach time was calculated based on this formula:

$$s(t) = \frac{1}{2}at^2 + v_0 t \tag{8.2}$$

where: s = Distance [m]
t = Time period [s]
a = deceleration during t (see eq. 2.3) [m/s^2]
v_0 = Speed before the train brakes [m/s]

In chapter 4 it was explained that the braking curve (BC) was constructed in time steps. Based on the gradient at the previous datapoint, a piece of the BC was constructed. In the case study time steps of 1 second were used. However, the consequence is that these pieces are between two datapoints of the trajectory dataset. This has been made visible in figure 8.2a, were each red dot is a new piece of the BC. With this approach the gradient for each new piece is not exactly known at that location, but the gradient of the previous datapoint in the trajectory dataset is used.

Now, the approach time could also be calculated in a different way, which has been made visible in figure 8.2b. Instead of constructing the BC in fixed time steps, it could also be calculated by taking the distance between two datapoints in the trajectory dataset. This means that s is given and t should be found by solving eq. 8.2. In this way each red dot is at the same position as the blue dot, so the gradient is exactly known. Then the speed v can be found with $v = s \cdot t$. Once the indication point has been found, the approach time could be calculated can be calculated in the same way as described in chapter 4.



(a) Construction per time step

(b) Construction per distance step

Figure 8.2: Construction of the braking curve (in red) for a certain location of the sprinter A4400 (Blue)

At first sight it can be noted that the BC per time step overall is much more precise. However, the goal of the construction is to find the indication point. Based on the figures it seems that both methods have the indication point at the same location. It is likely that the result is different, but it seems more a difference of tenths of a second, than a couple of seconds.

8.3. Model application with realised data

In this thesis it was shown how the model to estimate the occupancy time for moving block can be used with simulated data. An unanswered question is if the model can also be used for realised data from the outside world.

In trains there are multiple tracking systems present. Most of those systems use GPS to locate the train. First, there is MTPS, a rolling stock position service device. It logs the time, position and speed of the train once every ten seconds. This large time interval is a disadvantage of the system. Since this system is not used for the safety of the train, so it is not very precise. Also with GPS it cannot be determined which track a train uses – in case of multi-track lines – and the exact time a section or block is released. For this, other systems are needed, such as TROTS (Train detection & Tracking system)[91]. An advantage of MTPS is that most rolling stock in the Netherlands are equipped with this system, so this would cover the whole network.

A similar system to MTPS is Realtime Train Monitoring (RTM) system. This system logs the train behaviour more frequently, however only a part of the rolling stock are equipped with it [18].

A third system is the on-board safety system ORBIT. This system was installed to reduce the number of red-light passages by warning the train driver [61]. It calculates the braking curve based on the location, speed and weight of the train[103]. Because it is a safety system, in general the logged data is very accurate. However, it is not a fail-safe system, and because it uses a GPS-connection, inaccuracies can occur in tunnels. A big disadvantage is that it only logs data when a train is approaching a red signal, so when a train should brake. This makes this data not usable for the model, since data is needed of the complete trajectory.

A fourth logging system is QATS. QATS is developed for the monitoring and troubleshooting of ERTMS L2, L3 and interlocking [26]. This system logs the ETCS Trains Position and status reports, so it is much more precise than GPS data. However, it is only available on tracks where ERTMS is equipped, so in the Netherlands it is only available for a few lines.

To conclude, there are multiple different data sources that can be used. However, all mentioned disadvantages can be covered by using data out of a simulation model. Besides, the model is suited for planned operations, so one should scan the realised data for disturbances. So when the simulation model can describe real operations accurately and precise, data out of a simulation model would be preferred.

9

Conclusion & recommendations

This chapter provides the conclusions that can be drawn from this report. The research questions that were asked in the introduction will be answers in section 9.1. In section 9.2 recommendations on future research and improvements to FRISO will be given.

9.1. Conclusion

Timetable compression method

What are the shortcomings in the timetable compression method and which solutions do exist?

The timetable compression method is arguably the most widely used method to assess rail capacity. Since it was published in 2013, multiple shortcomings and limitations are addressed in literature, but also solutions are proposed.

Based on a literature review, Bešinović and Goverde 2018 concluded that there are four open challenges. For two of these challenges they propose a solution. First, there is shortcoming the capacity assessment method for nodes. UIC Code 406 proposes to decompose a node in a switch and platform area and evaluate each segment independently. Rotoli et al. gave a simplified approach, using this decomposition and by assuming a general node layout. However, by decomposing the nodes, route dependencies will not be considered, leading to underestimation of capacity occupation. To overcome this, Bešinović and Goverde introduced an analytical max-plus automata model.

A second limitation is the length of the decomposed line sections, which affect the resulting capacity occupation. As solution, Bešinović and Goverde proposed a network model for capacity assessment that preserves microscopic detail of the infrastructure and all train dependencies using max-plus algebra.

One of the remaining limitations is due to the network decomposition to line sections. Just as with nodes, train dependencies will be neglected, resulting in an underestimated capacity occupation. The other remaining limitation is the interpretation of Code 406. Where lines should be decomposed and the occupancy- and additional time rates are guidelines. Infrastructure managers should keep their own experience in mind when they apply the method. This means that the occupancy time rates of lines can exceed the proposed time rates in the leaflet when it still leads to reliable train operations.

The original purpose of the UIC 406 capacity method was to measure capacity occupation of a given timetable [10]. This is not the purpose of the model presented in this research, so it does not intend to substitute the UIC 406 leaflet. However, this research still provides a method to quickly come to a capacity assessment while overcoming some of the limitations of the UIC 406. For example, for the model it is not needed to decompose the network. Also the occupancy- and additional time rates are not being used, so the model bypasses this remaining limitation.

Data-driven capacity assessment methods
Which potential data-driven methods to assess capacity do exist?

In the search for potential data-driven capacity assessment methods for railways, a literature review is performed focusing on aviation, road traffic and maritime transport.

In aviation a small error can lead to a fatal accident, so a large number of sensors are embedded to increase safety. These sensors produce large amount of data (”Big Data”), which results also in a lot of different sources that can be used for research. Considering that runway capacity is the most stringent constraint on terminal capacity, mostly trajectory data is used. For this, the Flight Data Recorder (FDR) and Quick Access Recorder (QAR) – both mandatory in all aircraft – and the ADS-B recorder on the ground are typical sources for trajectory data. Out of this data, researchers gathered headway times and runway occupation times to perform capacity analyses. Besides trajectory data, also weather circumstances and runway design were used as input variables. Herrema et al. showed how all these variables can be used as input for a machine learning model to predict runway occupation. Also in railways all these types of data, especially weather data, can be used to improve daily operations, resulting in more capacity.

Compared to aviation and railways, in road traffic a significantly higher part of the vehicles are privately owned. This means that ”in-situ” technologies are more common to gather trajectory data, such as road tubes or cameras. However, (commercial) companies track also vehicles via navigation software and mobile phones, resulting in Float Car Data. This is different in railways, where the traffic size is planned, so these types of technology are less suitable to use. A popular tool to analyse road traffic data is the fundamental diagram. Song et al. and Diaz de Rivera et al. show how the fundamental diagram can be a potential capacity assessment method in railways, but there are still a lot of unanswered questions on this topic. In maritime transport there is also trajectory data available, but it is less used for capacity analysis.

The model presented in this thesis uses trajectory data of trains operating under fixed block signalling to assess moving block capacity. The model is not a pure data-driven model, but it is an analytical approach using different datasets as input.

Modelling moving block
How can the blocking times under moving block signalling be estimated?

Just as for fixed block signalling, the blocking time for moving block can be split into six time components: route setup-, sight and reaction-, approach-, running-, clearing- and release time. The length of the route setup time and release time depend on system characteristics and safety margins. In the model presented in this research these are fixed values. The sight and reaction time mainly depend on the reaction time and the level of attention of the train driver. Also this time component is modelled as a fixed value.

The approach time is defined as the time the train runs through the absolute braking distance. The approach time can be estimated with the braking characteristics of the train, gradient of the tracks and trajectory data of the train (distance, time, speed). With the braking characteristics and the gradient a braking curve can be constructed per timestep Δt. The braking curve will be constructed from the target location backwards in time and distance until it reaches the indication point, where it has the same speed as the free flow speed of the trajectory data. The distance between the indication point and the EoA is the braking distance which the train runs through.

For the running time there are three different situations. First, on a normal trackline, the running time is equal to the time period between two requests for MA. Second, when a virtual block is applied, it is the time it takes to run through that block. Third, for a stop at a station, the running time is equal to the dwelling time.

The clearing time is the time it takes a train has fully passed a location. Then, the total length of the blocking time is the sum of all six time components. The start time is the passage time minus the approach, reaction and setup time. The end time is the passage time plus the running, clearing and release time. It has been verified that in 95% of the cases the length of the blocking time is correctly estimated within a range (e.g. error margin) of 3 seconds. In the other 5% the error is larger up to 15 seconds.

To conclude, this analytical approach can estimate the blocking times under moving block signalling

can be estimated with data of fixed block operations with a 95% reliability of having deviations below 3 seconds.

Bottleneck identification
How can bottlenecks be identified without splitting lines?

Two methods are described to identify bottlenecks in a network. A first approach is to sum the blocking times of all trains that pass a block during a time period. The block with the highest summed blocking time indicates a bottleneck. During the verification it was made clear that this method only works for lines with homogeneous traffic. Since most railway networks also have heterogeneous lines, it is needed to split lines. Therefore this method is less suitable.

A second approach is to analyse the buffer time between two trains. Bottlenecks can be identified using the following steps: first, determine all the start- and end of the blocking time of each section. Second, calculate the buffer times of each section. Third, create a dataset of the calculated buffer times, including columns: section ID and -name, leading- and following train, name- and type of area and the engineering line reference. Fourth, sort the dataset by buffer time, where the shortest buffer time is first mentioned. Fifth, reduce the dataset to only the bottlenecks, by keeping the buffer time that appears first in the list for each train pair combination and remove all the others. The sections that are kept can be identified as bottlenecks. Each unique train pair combination has one bottleneck. It was verified that this method works for both homogeneous as heterogeneous traffic. For this method it is not needed to split lines, so train dependencies are being kept.

Effects on bottlenecks
How does Moving Block affect bottlenecks?

The model to estimate blocking times for moving block has been applied to a case study in the Netherlands. Using the buffer times between trains, bottlenecks for fixed- and moving block are identified and compared. With moving block, more than 2/3 of the critical sections in open line areas move towards stations or junctions. This is mainly because switches will become the critical location in the network, instead of the long fixed blocks in NS'54. On average the blocking times reduces by 45% to 60%. By keeping the same timetable, the buffer time between two trains increases on average by 75 seconds (60%). This means that rolling out a Moving Block signalling system has a significantly effect on capacity bottlenecks. In this research the (overall) infrastructure occupation rate has not been calculated, so it is not possible to draw any conclusions on this.

In all (heterogeneous) cases that were discussed in detail in this research it was seen that bottlenecks with moving block moved towards switches. The first reason is that switches need a longer setup time than regular tracks, because they need to shift and lock. Second, when an intercity is following a sprinter, the trains will converge in the a time-distance diagram. In other words: the headway time will decrease over time. The last infrastructure element they share will always be switch, so this will be the critical location. The same applies vice versa: when a sprinter follows an intercity, they will diverge. This means the first element they share is critical, which is a switch.

Support for the railway industry
How can the defined method support ProRail and the rail industry in assessing capacity impacts of Moving Block signalling?

The models presented in this research can estimate the blocking- and buffer times for moving block signalling. To do this, it is not needed to build a simulation model which includes moving block signalling, which is an advantage of this model. However, one should consider that the output will have an error of a couple of seconds, which can be in some exceptions ten to twenty seconds. It seems that the model has a larger deviation when the approach time is shorter or longer than average, but a reason for this was not found by the author. Trajectory data of trains operating under another signalling system can be used as input variable. Although simulation data has its own limitations, it is preferred to use this type of data over realised data, because intervals between loggings will be smaller and in realised data inaccuracies occur in the determination of the location by GPS.

With the output of the model, it can be estimated where bottlenecks will be located when a moving

block signalling system will be rolled out. For this, it is not needed to split corridors into lines or line sections.

It can be imagined that (simulation) software used by companies in the railway industry already have some parts of the model presented in this thesis, for example a way to calculate one or more time components of the blocking time. In that case it will be easier to implement the presented model in their software by adding the calculations of the remaining time components. In section 9.2.2 detailed improvements will be given how the model could be implemented in FRISO and Roberto.

Main research question
How can capacity gains of Moving Block be assessed with data for both railway corridors and complex nodes?

Moving block signalling promises a significant reduction of the infrastructure occupation compared to a fixed block system, such as NS'54/ATB. This is mainly caused by a strong reduction of the approach- and running time.

With infrastructure data, rolling stock parameters and planned time-distance data, blocking times for a moving block signalling system can be estimated. The model presented in this thesis has an average error of 0.87s to the blocking time. In 95% of the cases the error is within a range of (-3,3) seconds. With the blocking times of all trains, bottlenecks in both railway corridors and complex nodes can be identified. One could sum all blocking times per block and consider blocks with the highest summed blocking times as bottleneck. However, this can only be applied for homogeneous traffic situations. Another approach is identifying bottlenecks by the shortest buffer time between two trains, also called a critical block. This can be applied for both homogeneous as heterogeneous traffic situations. An advantage is that it is not needed to split corridors into line sections. One could analyse a whole network at once and identify bottleneck at a microscopic level.

9.2. Recommendations

9.2.1. Future work and research

Several directions for future research can be recommended. First, a new verification study can be made to determine the precision and accuracy of the model. In the performed verification study 12 trains were simulated, while there were actually only two different services. So 6 trains of each service operate on the same infrastructure with the same timetable and driving behaviour. Besides this, the two services are very similar and mostly share the same infrastructure. All in all, this results in a large dataset, but the number of unique values are limited. Also, the size of the dataset has directly an influence on the correlation. Therefore, it is recommended to perform an extra verification study, with more unique train operations, so the accuracy and precision of the model can be better determined.

In section 6.5 a list of critical sections were analysed and marked as bottlenecks. However, multiple different criteria can be made which section is more critical than another. One could look at the critical section which as the shortest buffer time. Another possibility is to look how often a section is critical for multiple train pairs, such as table 7.1b. A third possibility is to compare the list of critical sections to the bottlenecks in the outside world. One could make a list of all places were secondary delays occurred during real operations and see if they match with the critical sections found in simulation.

When the research plan for this thesis was written, one of the possible ways was defining a data-driven approach to identify capacity bottlenecks. During the research it became an analytical method, but it is recommended to investigate if it possible to describe a data-driven approach which can find capacity bottlenecks. Possible input variables could be infrastructure properties, such as block lengths or (type of) switches, traffic size or headway times.

As mentioned in the introduction of this research, ERTMS L3 hasn't been fully developed yet. Within L3, there are four different types: Overlay, Hybrid, Virtual block and Moving block. Currently, L3 Hybrid is the most advanced (in terms of development) of the four different types and seen as the low risk solution for application on the rail networks [27]. However, the question is how long the virtual blocks

should be to reach the same capacity gains as moving block. For this an optimisation study can be set up, or a comparison study where different lengths of blocks will be compared to moving block. Given that the model presented in this thesis calculates the blocking times for moving block, it is expected that it can be applied in the timetable compression method and calculate the infrastructure occupation. For a comparison study, also the blocking times of the ERTMS L3 Hybrid should be calculated or estimated. Because this system is not implemented in FRISO and it uses virtual blocks, so the model presented in this thesis cannot be used, a new model should be described to estimate the blocking times of ERTMS L3 Hybrid.

In the case study corridors with mixed traffic were simulated and analysed. The traffic consisted of passenger trains such as sprinters, intercity and high speed trains. Unfortunate cargo trains were not included. During the selection of corridors the idea was to have corridors which had train paths for cargo trains. After the simulation and data collection it was found out that these were not present in the timetable. An important difference is that cargo trains accelerate much slower than passengers trains. Also, the difference between the braking distance and the block length for cargo trains is lower than for passenger trains. This means less capacity gain for cargo trains. It would be interesting to see what the effect will be on bottlenecks in combination with passenger trains.

9.2.2. Practical improvements to FRISO and Roberto

With regard to experiences with FRISO and Roberto and the application of this model, the following is recommended:

1. **Logging data**
   FRISO is a user friendly microsimulation model and is rich of tools to analyse track occupation, train behaviour, etc. It also logs all kind of different data, for which only a part was used in this study. However, the user himself can export only a small part of all data. Currently it is only possible to export section occupation data. It is advised to create an option where the user can choose which data he want to export, for example by ticking boxes before a simulation run. Then the user can also perform his own analysis and is not limited to the tools given in FRISO. This will makes FRISO also more attractive to use for research purposes.
2. **Implement the blocking time theory**
   This recommendation covers two topics. First, currently the blocking times in FRISO are not in line with UIC code 406. In the current state the section occupation times only give the time a train is physically in a block and the blocking time is too long (see also section 6.3). Second, when ETCS Level 2 will be implemented in the Netherlands, the protected zone model cannot be used, since ETCS Level 2 uses in-cab signalling. So to cover this, the blocking times should be calculated according to the blocking time theory and not the protected zone model.
3. **Gathering output of Roberto**
   Just as for FRISO, the possibilities to export data is very limited. Currently it is possible to export a .csv file with only the minimal local headway time and buffer time for one train pair at a time. However, Roberto has also other information in the back-end, but this is not given in the output file. One could think of the red- and yellow times of the signals. But also the headway times when Driving On Braking distance (Dutch: Op Remweg Rijden, ORR) is applied. Currently these are only given at signals, but Roberto should have them also at other infrastructural elements. This can be seen in the graphs for two reasons. First, ORR shows kinks at other places than signals, meaning that it has information at these places. Secondly, when Roberto compresses the train paths, the ORR line touches the back of the leading train at not-signals locations, meaning that Roberto has the passage time of the train at these locations.
4. **Implementation of the model in FRISO/Roberto**
   With ORR Roberto has already a strong base to implement the model to estimate the blocking time for moving block into its simulation tool. However, also a few improvements can be made. First of all, the graphs show sometimes an error of the ORR line, which probably is also present in the back-end dataset. These could be fixed. Secondly, virtual blocks could be implemented, including extra occupation time for switches. Third, other fixed parameters can be added for the reaction, route setup and release time.

Bibliography

[1] M. Abril, M. A. Salido, F. Barber, L. Ingolotti, A. Lova, and P. Tormos. A heuristic technique for the capacity assessment of periodic trains. In *Frontiers in Artificial Intelligence and Applications*, volume 131, pages 339–346, 2005. ISBN 1586035606.

[2] Alfredo Alessandrini, Diego Guizzardi, Greet Janssens-Maenhout, Enrico Pisoni, Marco Trombetti, and Michele Vespe. Estimation of shipping emissions using vessel long range identification and tracking data. *Journal of Maps*, 13(2):946–954, 11 2017. ISSN 17445647. doi: 10.1080/17445647.2017.1411842. URL https://www.tandfonline.com/doi/full/10.1080/17445647.2017.1411842.

[3] Alfredo Alessandrini, Fabio Mazzarella, and Michele Vespe. Estimated Time of Arrival Using Historical Vessel Tracking Data. *IEEE Transactions on Intelligent Transportation Systems*, 20(1): 7–15, 1 2019. ISSN 15249050. doi: 10.1109/TITS.2017.2789279.

[4] Joelle Aoun, Egidio Quaglietta, Rob M.P. Goverde, Martin Scheidt, Marcelo Blumenfeld, Anson Jack, and Bill Redfern. A hybrid Delphi-AHP multi-criteria analysis of Moving Block and Virtual Coupling railway signalling. *Transportation Research Part C: Emerging Technologies*, 129, 8 2021. ISSN 0968090X. doi: 10.1016/j.trc.2021.103250.

[5] John Armstrong and John Preston. Capacity utilisation and performance at railway stations. *Journal of Rail Transport Planning and Management*, 7(3):187–205, 12 2017. ISSN 22109706. doi: 10.1016/j.jrtpm.2017.08.003.

[6] Jacob Avery and Hamsa Balakrishnan. Data-driven modeling and prediction of the process for selecting runway configurations. *Transportation Research Record*, 2600:1–11, 1 2016. ISSN 03611981. doi: 10.3141/2600-01. URL https://journals.sagepub.com/doi/abs/10.3141/2600-01.

[7] Maarten Bartholomeus (ProRail). Information retrieved by e-mail, 6 2021.

[8] Baris Baspinar, N. Kemal Ure, Emre Koyuncu, and Gokhan Inalhan. Analysis of Delay Characteristics of European Air Traffic through a Data-Driven Airport-Centric Queuing Network Model. *IFAC-PapersOnLine*, 49(3):359–364, 1 2016. ISSN 24058963. doi: 10.1016/j.ifacol.2016.07.060.

[9] Roberto Bellasio. Analysis of wind data for airport runway design. *Journal of Airline and Airport Management*, 4(2):2014–2018, 9 2014. ISSN 2014-4865. doi: 10.3926/jairm.26. URL http://dx.doi.org/10.3926/jairm.26.

[10] Nikola Bešinović and Rob M.P. Goverde. Capacity Assessment in Railway Networks. *International Series in Operations Research and Management Science*, 268:25–45, 2018. ISSN 08848289. doi: 10.1007/978-3-319-72153-8{_}2.

[11] Thorsten Büker, Thomas Graffagnino, Eike Hennig, and Alexander Kuckelberg. Enhancement of Blocking-time Theory to Represent Future Interlocking Architectures. *8th International Conference on Railway Operations Modelling and Analysis (ICROMA)*, pages 219–240, 2019. URL https://easychair.org/publications/preprint/9v95.

[12] Gerrit Burmester, Hui Ma, Dietrich Steinmetz, and Sven Hartmannn. Big Data and Data Analytics in Aviation. In *Advances in Aeronautical Informatics: Technologies Towards Flight 4.0*, chapter 5, pages 55–65. Springer International Publishing, 5 2018. ISBN 9783319750583. doi: 10.1007/978-3-319-75058-3{_}5. URL https://doi.org/10.1007/978-3-319-75058-3_5.

[13] Igor Bychkov, Alexander Kazakov, Anna Lempert, and Maxim Zharkov. Modeling of Railway Stations Based on Queuing Networks. *Applied Sciences*, 11(5):2425, 3 2021. ISSN 2076-3417. doi: 10.3390/app11052425. URL https://www.mdpi.com/2076-3417/11/5/2425.

[14] Yashovardhan S Chati and Hamsa Balakrishnan. Analysis of Aircraft Fuel Burn and Emissions in the Landing and Take Off Cycle using Operational Data. In *6th International Conference on Research in Air Transportation*, 2014.

[15] Tsan-Ming Choi, Stein W. Wallace, and Yulan Wang. Big Data Analytics in Operations Management. *Production and Operations Management*, 27(10):1868–1883, 10 2018. ISSN 10591478. doi: 10.1111/poms.12838. URL http://doi.wiley.com/10.1111/poms.12838.

[16] Sai Ho Chung, Hoi Lam Ma, Mark Hansen, and Tsan Ming Choi. Data science and analytics in aviation, 2 2020. ISSN 13665545.

[17] Francesco Corman, Jonas Henken, and Mehdi Keyvan-Ekbatani. Macroscopic fundamental diagrams for train operations-are we there yet? *MT-ITS 2019 - 6th International Conference on Models and Technologies for Intelligent Transportation Systems*, 2019. doi: 10.1109/MTITS.2019.8883374.

[18] Data Science Lab. ProRail, 1 2022.

[19] Adrian Diaz de Rivera and C. Tyler Dick. Illustrating the implications of moving blocks on railway traffic flow behavior with fundamental diagrams. *Transportation Research Part C: Emerging Technologies*, 123:102982, 2 2021. ISSN 0968090X. doi: 10.1016/j.trc.2021.102982.

[20] Adrian Diaz de Rivera, C. Tyler Dick, and Leonel E. Evans. Improving Railway Operational Efficiency with Moving Blocks, Train Fleeting, and Alternative Single-Track Configurations. *Transportation Research Record*, 2674(2):146–157, 2020. ISSN 21694052. doi: 10.1177/0361198120905842.

[21] Michal Dorda and Dušan Teichmann. Modelling of freight trains classification using queueing system subject to breakdowns. *Mathematical Problems in Engineering*, 2013. ISSN 1024123X. doi: 10.1155/2013/307652.

[22] Lynnette Dray. An empirical analysis of airport capacity expansion. *Journal of Air Transport Management*, 87:101850, 8 2020. ISSN 09696997. doi: 10.1016/j.jairtraman.2020.101850.

[23] L.C. Edie. Discussion of traffic stream measurements and definitions. In *Proceedings of the Second International Symposium on the Theory of Traffic Flow*, OECD, Paris, France, 1965.

[24] European Railway Agency. ERTMS/ETCS System Requirements Specification Chapter 3 Principles. Technical Report 3.6.0, ERA, 2016.

[25] European Railway Agency. Introduction To ETCS Braking Curves. Technical Report 1.5, ERA, 2020. URL ERA_ERTMS_040026.

[26] Expandium. QATS signalling, 1 2022.

[27] Nicola Furness, Henri van Houten, Laura Arenas, and Maarten Bartholomeus. ERTMS Level 3: the Game-Changer. *Institution of Railway Signal Engineers*, (232):1–9, 2017. URL https://www.irse.nl/resources/170314-ERTMS-L3-The-gamechanger-from-IRSE-News-Issue-232.pdf.

[28] Babak Ghalebsaz-Jeddi, George L Donohue, and John F Shortle. A Statistical Analysis of the Aircraft Landing Process. Technical Report 3, George Mason University, 1 2009. URL https://www.sid.ir/en/journal/ViewPaper.aspx?id=243131.

[29] Rob M.P. Goverde and Ingo A. Hansen. Performance indicators for railway timetables. *IEEE ICIRT 2013 - Proceedings: IEEE International Conference on Intelligent Rail Transportation*, pages 301–306, 2013. doi: 10.1109/ICIRT.2013.6696312.

[30] Rob M.P. Goverde, Francesco Corman, and Andrea D'Ariano. Railway line capacity consumption of different railway signalling systems under scheduled and disturbed conditions. *Journal of Rail Transport Planning and Management*, 3(3):78–94, 8 2013. ISSN 22109706. doi: 10.1016/j.jrtpm.2013.12.001.

[31] B. D. Greenshields. A Study of Traffic Capacity. *Proceedings Highway Research Board*, (14): 448–477, 1934.

[32] Ingo Arne Hansen and Jorn Pachl. *Railway Timetabling & Operations*. DVV Media Group GmbH | Eurailpress, 2nd edition, 2014. ISBN 978-3-7771-0462-1.

[33] Floris Herrema, Ricky Curran, Sander Hartjes, Mohamed Ellejmi, Steven Bancroft, and Michael Schultz. A machine learning model to predict runway exit at Vienna airport. *Transportation Research Part E: Logistics and Transportation Review*, 131:329–342, 11 2019. ISSN 1366-5545. doi: 10.1016/J.TRE.2019.10.002.

[34] International Association of Lighthouse Authorities (IALA). IALA GUIDELINES ON THE UNIVERSAL AUTOMATIC IDENTIFICATION SYSTEM. *IALA Guidelines*, Volume 1(Part II - Technical Issues Edition 1.1), 2002. URL www.iala-aism.org.

[35] Incontrol Simulation Software. Simulation of Railway Networks. URL https://www.incontrolsim.com/application-areas/railway-simulation/.

[36] International Union of Railways (UIC). Code 405 OR - Links between Railway Infrastructure Capacity and the Quality of Operations. Technical report, UIC, 1996.

[37] International Union of Railways (UIC). UIC Code 406. Technical Report 2nd edition, UIC, 2013.

[38] Philip T.G. Jackson, Carl J. Nelson, Jens Schiefele, and Boguslaw Obara. Runway detection in High Resolution remote sensing data. In *9th International Symposium on Image and Signal Processing and Analysis, ISPA 2015*, pages 170–175, Zagreb, Croatia, 2015. Institute of Electrical and Electronics Engineers Inc. ISBN 9781467380324. doi: 10.1109/ISPA.2015.7306053.

[39] A. Jamili. Computation of practical capacity in single-track railway lines based on computing the minimum buffer times. *Journal of Rail Transport Planning and Management*, 8(2):91–102, 9 2018. ISSN 22109706. doi: 10.1016/j.jrtpm.2018.03.002.

[40] Lars Wittrup Jensen, Alex Landex, Otto Anker Nielsen, Leo G. Kroon, and Marie Schmidt. Strategic assessment of capacity consumption in railway networks: Framework and model. *Transportation Research Part C: Emerging Technologies*, 74:126–149, 1 2017. ISSN 0968090X. doi: 10.1016/j.trc.2016.10.013.

[41] Lars Wittrup Jensen, Marie Schmidt, and Otto Anker Nielsen. Determination of infrastructure capacity in railway networks without the need for a fixed timetable. *Transportation Research Part C: Emerging Technologies*, 119:102751, 10 2020. ISSN 0968090X. doi: 10.1016/j.trc.2020.102751.

[42] Tobias Jeske. Floating Car Data from Smartphones: What Google and Waze Know About You and How Hackers Can Control Traffic. In *Proceedings of the BlackHat Europe*, pages 1–12, 2013. URL https://www.google.com/loc/m/api.

[43] Predrag Jovanović, Norbert Pavlović, Ivan Belošević, and Sanjin Milinković. Graph coloring-based approach for railway station design analysis and capacity determination. *European Journal of Operational Research*, 287(1):348–360, 11 2020. ISSN 03772217. doi: 10.1016/j.ejor.2020.04.057.

[44] Stefan Kern and Michael Schultz. Evaluation of a standardized single runway airport model with respect to runway capacity. In *16th AIAA Aviation Technology, Integration, and Operations Conference*, pages 1–13, 2016. ISBN 9781624104404. doi: 10.2514/6.2016-4071. URL http://arc.aiaa.org.

[45] J. W.; Kerssies. NS: Het spoor is over acht jaar vol - OV-Magazine, 2019. URL https://www.ovmagazine.nl/2019/08/ns-het-spoor-is-over-acht-jaar-vol-1541/.

[46] KiM. Kerncijfers Mobiliteit 2020. Technical report, KiM, 2020. URL https://www.bovag.nl/BovagWebsite/media/BovagMediaFiles/Cijfers/2020/Kerncijfers-Auto-2020-DEF.pdf?ext=.pdf.

[47] Victor L. Knoop and Winnie Daamen. Automatic fitting procedure for the fundamental diagram. *Transportmetrica B: Transport Dynamics*, 5(2):129–144, 4 2017. ISSN 2168-0566. doi: 10.1080/21680566.2016.1256239. URL https://www.tandfonline.com/doi/full/10.1080/21680566.2016.1256239.

[48] Victor L. Knoop, Andreas Hegyi, Maria Salomons, Hans Van Lint, Yufei Yuan, and Ramon Landman. *CIE4825 and CIE5821 Lecture Notes; Traffic Flow Modelling & Control*. TU Delft, Transport & Planning, 6th editio edition, 2019.

[49] Vivek Kumar, Lance Sherry, and Rafal Kicinger. Runway Occupancy Time Extraction and Analysis Using Surface Track Data. Technical report, Center for Air Transportation Systems Research; Department of Systems Engineering and Operations Research; George Mason University, 2009.

[50] Alex Landex. *Methods to estimate railway capacity and passenger delays*. PhD thesis, Technical University of Denmark, 2008. URL http://wwwdtu2.sitecore.dtu.dk/upload/institutter/dtutransport/rapporter/rap5_2008_phd-thesis_al_hjemmeside.pdf.

[51] Anna-Liese S. Lapinski. LRIT and AIS; An analysis of October 2010 data. Technical Report Technical Memorandum, Defence R&D Cananda - Atlantic, 2014.

[52] Arno Leblanc. Derde tienminutentrein tussen Rotterdam, Den haag, Leiden en Schiphol, 10 2021.

[53] Guillaume Leduc. Road Traffic Data : Collection Methods and Applications. *EUR Number: Technical Note: JRC 47967*, JRC 47967(May):55, 2008. URL http://ftp.jrc.es/EURdoc/EURdoc/JRC47967.TN.pdf.

[54] Anna Lempert, Alexander Kazakov, and Maxim Zharkov. A Stochastic Model of a Transport Hub and Multi-phase Queueing Systems. In *Vth International workshop "Critical Infrastructures: Contingency Management, Intelligent, Agent-based, Cloud Computing and Cyber Security" (IWCI 2018)*, pages 117–123. Atlantis Press, 8 2018. doi: 10.2991/iwci-18.2018.21. URL https://www.atlantis-press.com/proceedings/iwci-18/25899812.

[55] Wiebke Lenze and Nils Nießen. Modelling the Prohibition of Train Crossings in Tunnels with Blocking Time Theory. In *8th International Conference on Railway Operations Modelling and Analysis (ICROMA)*, pages 623–649, Norrköping, Sweden, 2019. Linköping University Electronic Press.

[56] B. Levy, J. Legge, and M. Romano. Opportunities for improvements in simple models for estimating runway capacity. *The 23rd Digital Avionics Systems Conference (IEEE Cat. No.04CH37576)*, 1:5–21, 2004. doi: 10.1109/DASC.2004.1391282. URL http://ieeexplore.ieee.org/document/1391282/.

[57] Zhengwen Liao, Haiying Li, Jianrui Miao, and Francesco Corman. Railway capacity estimation considering vehicle circulation: Integrated timetable and vehicles scheduling on hybrid time-space networks. *Transportation Research Part C: Emerging Technologies*, 124:33, 3 2021. ISSN 0968090X. doi: 10.1016/j.trc.2020.102961.

[58] Anders Lindfeldt. Validation of a simulation model for capacity evaluation of double-track railway lines. In *Proceedings of the 6th International Seminar on Railway Operations Modelling and Analysis (RailTokyo2015)*, Tokyo, Japan, 2015.

[59] Tobias Lindner. Applicability of the analytical UIC Code 406 compression method for evaluating line and station capacity. *Journal of Rail Transport Planning and Management*, 1(1):49–57, 11 2011. ISSN 22109706. doi: 10.1016/j.jrtpm.2011.09.002.

[60] Cong Liu, Jingxian Liu, Xun Zhou, Zhen Zhao, Chengpeng Wan, and Zhao Liu. AIS data-driven approach to estimate navigable capacity of busy waterways focusing on ships entering and leaving port. *Ocean Engineering*, 218:108215, 12 2020. ISSN 00298018. doi: 10.1016/j.oceaneng.2020.108215.

[61] W.J. Mansveld. Veiligheid van het railvervoer, 6 2014.

[62] Julio Mar-Ortiz, Norberto Castillo-García, and María D. Gracia. A decision support system for a capacity management problem at a container terminal. *International Journal of Production Economics*, 222:107502, 4 2020. ISSN 0925-5273. doi: 10.1016/J.IJPE.2019.09.023.

[63] Karen B. Marais and Matthew R. Robichaud. Analysis of trends in aviation maintenance risk: An empirical approach. *Reliability Engineering and System Safety*, 106:104–118, 10 2012. ISSN 09518320. doi: 10.1016/j.ress.2012.06.003.

[64] A. D. Middelkoop and L. Loeve. Simulation of traffic management with FRISO. *WIT Transactions on the Built Environment*, 88:501–509, 2006. ISSN 17433509. doi: 10.2495/CR060501.

[65] Dick Middelkoop, Joris Steneker, Sebastiaan Meijer, Emdzad Sehic, and Maura Mazzarello. Simulation backbone for gaming simulation in railways: A case study. In *Proceedings - Winter Simulation Conference*, 2012. ISBN 9781467347792. doi: 10.1109/WSC.2012.6465195.

[66] Dick Middelkoop, Maura Mazzarello, and Douwe De Vries. Optimizing Train Traffic: Demonstrating Benefits in a Case Study Optimizing Train Traffic: Demonstrating Benefits in a Case Study Douwe de Vries. 2013. URL https://www.researchgate.net/publication/316601277.

[67] Ministry of I&W. Dossier Programmabeslissing. Technical report, 2019. URL https://www.rijksoverheid.nl/documenten/rapporten/2019/05/17/railmap-ertms-4-0.

[68] Fabrizio Natale, Maurizio Gibin, Alfredo Alessandrini, Michele Vespe, and Anton Paulrud. Mapping Fishing Effort through AIS Data. *PLOS ONE*, 10(6):e0130746, 6 2015. ISSN 1932-6203. doi: 10.1371/journal.pone.0130746. URL https://dx.plos.org/10.1371/journal.pone.0130746.

[69] Zohreh Nazeri and Jianping Zhang. Mining aviation data to understand impacts of severe weather on airspace system performance. In *Proceedings - International Conference on Information Technology: Coding and Computing, ITCC 2002*, pages 518–523. Institute of Electrical and Electronics Engineers Inc., 2002. ISBN 0769515061. doi: 10.1109/ITCC.2002.1000441.

[70] ProRail. Programma Hoogfrequent Spoorvervoer, 10 2021.

[71] Egidio Quaglietta. *A microscopic simulation model for supporting the design of railway systems: development and applications*. PhD thesis, University of Napoli, 2011. URL http://www.fedoa.unina.it/8599/1/Quaglietta_Egidio_24.pdf.

[72] Egidio Quaglietta. A simulation-based approach for the optimal design of signalling block layout in railway networks. *Simulation Modelling Practice and Theory*, 46:4–24, 8 2014. ISSN 1569190X. doi: 10.1016/j.simpat.2013.11.006.

[73] Egidio Quaglietta and Rob M. P. Goverde. Exploring Virtual Coupling : Principles and Analysis Operational. In *Proc. of the 10th ASPECT Conference of the Institution of Railway Signalling Engineers*, number 3, pages 1–13, 2019.

[74] Varun Ramanujam and Hamsa Balakrishnan. Data-Driven Modeling of the Airport Configuration Selection Process. *IEEE Transactions on Human-Machine Systems*, 45(4):490–499, 2015. ISSN 21682291. doi: 10.1109/THMS.2015.2411743.

[75] Cyril Ray, Romain Gallen, Clement Iphar, Aldo Napoli, and Alain Bouju. DeAIS project: Detection of AIS spoofing and resulting risks. In *MTS/IEEE OCEANS 2015 - Genova: Discovering Sustainable Ocean Energy for a New World*. Institute of Electrical and Electronics Engineers Inc., 9 2015. ISBN 9781479987368. doi: 10.1109/OCEANS-Genova.2015.7271729.

[76] RMCon. RailSys Software, 2021. URL https://www.rmcon-int.de/railsys-en/railsys-suite/.

[77] Francesco Rotoli, Elena Navajas Cawood, and Antonio Soria. Capacity assessment of railway infrastructure: Tools, methodologies and policy relevance in the EU context. *JRC Working Papers*, 2016. URL https://ideas.repec.org/p/ipt/iptwpa/jrc100509.html.

[78] Francesco Rotoli, Elena Navajas Cawood, and Soria Antonio Ramirez. JRC Publications Repository - Capacity assessment of railway infrastructure: Tools, methodologies and policy relevance in the EU context. Technical report, European Union (EU), Sevilla, Spain, 2016. URL https://publications.jrc.ec.europa.eu/repository/handle/JRC100509.

[79] Christoph Schmitz, Norman Weik, Stephan Zieger, Nils Nießen, and Anke Schmeink. Markov Models for the Performance Analysis of Railway Networks. Technical report, RWTH Aachen University, 2017. URL http://www.via.rwth-aachen.de/downloads/RailLille_Schmitz_Weik_Zieger_Niessen_Schmeink.pdf.

[80] Wulf Schwanhäußer. *Die Bemessung der Pufferzeiten im Fahrplangefüge der Eisenbahn T E X T*. PhD thesis, Rheinisch-Westfälischen Technischen Hochschule Aachen, 1974.

[81] Wulf Schwanhäußer. The status of German railway operations management in research and practice. *Transportation Research Part A*, 28(6):495–500, 11 1994. ISSN 09658564. doi: 10.1016/0965-8564(94)90047-7.

[82] Katsuhiro Sekine, Furuto Kato, Kota Kageyama, and Eri Itoh. Data-driven simulation for evaluating the impact of lower arrival aircraft separation on available airspace and runway capacity at tokyo international airport. *Aerospace*, 8(6), 2021. ISSN 22264310. doi: 10.3390/aerospace8060165.

[83] Simcon. Villon, software simulation tool, 2021. URL https://www.simcon.sk/en/tools/villon.

[84] Rogier Jakob Simons. *The influence of railway signalling characteristics on resilience (MSc thesis)*. 2019.

[85] Samuel L Sogin. *Simulations of mixed use rail corridors: how infrastructure affects interactions among train types*. PhD thesis, University of Illinois at Urbana-Champaign, 2013. URL https://www.ideals.illinois.edu/handle/2142/46672.

[86] I. Song, I. Cho, T. Tessitore, T. Gurcsik, and H. Ceylan. Data-Driven Prediction of Runway Incursions with Uncertainty Quantification. *Journal of Computing in Civil Engineering*, 32(2):04018004, 3 2018. ISSN 0887-3801. doi: 10.1061/(ASCE)CP.1943-5487.0000733. URL http://ascelibrary.org/doi/10.1061/%28ASCE%29CP.1943-5487.0000733.

[87] Tai Jin Song, Billy M. Williams, and Nagui M. Rouphail. Data-driven approach for identifying spatiotemporally recurrent bottlenecks. *IET Intelligent Transport Systems*, 12(8):756–764, 10 2018. ISSN 1751956X. doi: 10.1049/iet-its.2017.0284.

[88] Joris Steneker and B.D. Cunes. FRISO Conceptueelmodel. Technical Report december, INCONTROL Simulation Solutions, Utrecht, 2019.

[89] Apa Sun, J Ellerbroek, and J Hoekstra. Modeling and Inferring Aircraft Takeoff Mass from Runway ADS-B Data. In *7th International Conference on Research in Air Transportation*, Philadelphia, USA, 2016.

[90] Chao Tong, Xiang Yin, Shili Wang, and Zhigao Zheng. A novel deep learning method for aircraft landing speed prediction based on cloud-based sensor data. *Future Generation Computer Systems*, 88:552–558, 11 2018. ISSN 0167739X. doi: 10.1016/j.future.2018.06.023.

[91] Marlies Van der Goot. Waar is die trein?, 2 2018.

[92] Marieke Van Gompel. ProRail verdeelt capaciteit spoor voortaan per zes seconden, 2019. URL https://www.spoorpro.nl/goederenvervoer/2019/08/19/prorail-kan-capaciteit-spoor-voortaan-per-tien-seconden-verdelen/?gdpr=accept.

[93] R. Vergroesen. *ERTMS / ETCS Hybrid Level 3 and ATO (MSc thesis)*. 2020.

[94] Petr Veselý and Michael Bazant. Zobrazit SWITCH AREA CAPACITY ASSESSMENT USING UIC 406 – THE PROCEDURE AND SOFTWARE TOOL. *Perner's Contacts*, X(4):98–106, 2015. URL https://pernerscontacts.upce.cz/index.php/perner/article/view/991/826.

[95] Michele Vespe, Harm Greidanus, and Marlene Alvarez Alvarez. The declining impact of piracy on maritime transport in the Indian Ocean: Statistical analysis of 5-year vessel tracking data. *Marine Policy*, 59:9–15, 9 2015. ISSN 0308597X. doi: 10.1016/j.marpol.2015.04.018.

[96] VIA Consulting & Development. LUKS | VIA Consulting & Development GmbH, 2021. URL https://www.via-con.de/en/development/luks/.

[97] Anderson P. Vieira, Luciano M. Christofoletti, and Plínio R.S. Vilela. Analyzing railway capacity using a planning tool. In *2018 Joint Rail Conference, JRC 2018*. American Society of Mechanical Engineers (ASME), 6 2018. ISBN 9780791850978. doi: 10.1115/JRC2018-6160.

[98] Xiaoping Wang, Yunliang Zhao, Peng Sun, and Xiaobin Wang. An analysis on convergence of data-driven approach to ship lock scheduling. *Mathematics and Computers in Simulation*, 88: 31–38, 2 2013. ISSN 03784754. doi: 10.1016/j.matcom.2013.03.005.

[99] Norman Weik and Nils Nie. Journal of Rail Transport Planning & Management Quantifying the effects of running time variability on the capacity of rail corridors. *Journal of Rail Transport Planning and Management*, 15(August 2019):1–12, 2020. doi: 10.1016/j.jrtpm.2020.100203.

[100] Norman Weik, Jennifer Warg, Ingrid Johansson, Markus Bohlin, and Nils Nießen. Extending UIC 406-based capacity analysis – New approaches for railway nodes and network effects. *Journal of Rail Transport Planning and Management*, 15:100199, 9 2020. ISSN 22109706. doi: 10.1016/j.jrtpm.2020.100199.

[101] H. Widyastuti and W. S. Budhi. Railway capacity analysis using Indonesian method and UIC code 405 method. In *IOP Conference Series: Materials Science and Engineering*, volume 930. IOP Publishing Ltd, 11 2020. doi: 10.1088/1757-899X/930/1/012059.

[102] Windward. AIS Data on the High Seas: An Analysis of the Magnitude and Implications of Growing Data Manipulation at Sea - News, 2014. URL http://maritime-connector.com/news/general/ais-data-on-the-high-seas-an-analysis-of-the-magnitude-and-implications-of-growing-d

[103] Edwin Winterkamp. Veiliger spoor door vermindering rood sein-passages (ORBIT), 1 2022.

[104] Xuri Xin, Kezhong Liu, Xing Yang, Zhitao Yuan, and Jinfen Zhang. A simulation model for ship navigation in the "Xiazhimen" waterway based on statistical analysis of AIS data. *Ocean Engineering*, 180:279–289, 5 2019. ISSN 00298018. doi: 10.1016/j.oceaneng.2019.03.052.

[105] Li Da Xu, Eric L. Xu, and Ling Li. Industry 4.0: State of the art and future trends. *International Journal of Production Research*, 56(8):2941–2962, 2018. ISSN 1366588X. doi: 10.1080/00207543.2018.1444806. URL https://www.tandfonline.com/action/journalInformation?journalCode=tprs20.

[106] Nale Zhao, Lei Yu, Hui Zhao, Jifu Guo, and Huimin Wen. Analysis of Traffic Flow Characteristics on Ring Road Expressways in Beijing. *Transportation Research Record: Journal of the Transportation Research Board*, 2124(1):178–185, 1 2009. ISSN 0361-1981. doi: 10.3141/2124-17. URL http://journals.sagepub.com/doi/10.3141/2124-17.

[107] Marina Zhuravskaya, Anna Lempert, Nataliia Anashkina, and Maksim Zharkov. Issues of Sustainable Urban Mobility Simulation. In *2018: Proceedings of The 18th International Scientific Conference Business Logistics in Modern Management*, volume 18, pages 439–452, 2018.

[108] Stephan Zieger, Norman Weik, and Nils Nießen. Der Einfluss von Pufferzeitverteilungen im Fahrplan auf die Modellierung der Folgeverspätungen im Eisenbahnwesen. *26. Verkehrswissenschaftliche Tage 2018 an der Technischen Universität Dresden*, pages 731–745, 2018. URL https://publications.rwth-aachen.de/record/721529.

A

Different shapes of the fundamental diagram

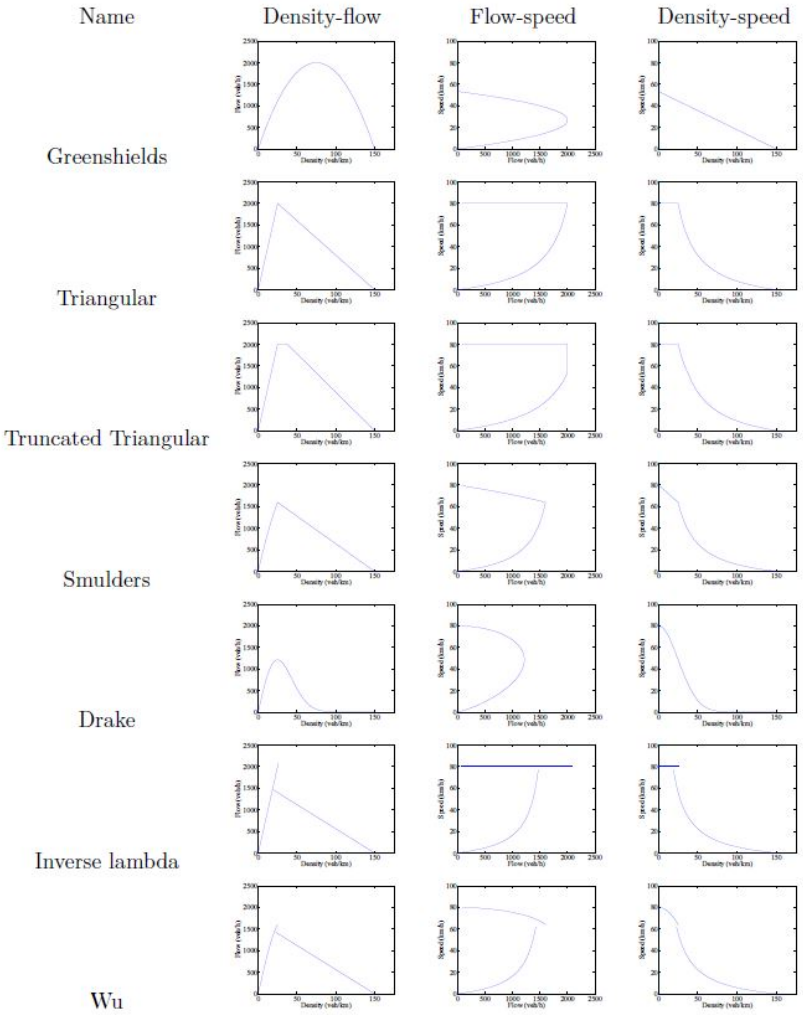


Figure A.1: Different shapes of the fundamental diagram [48]

B

Added code lines in EGTRAIN

Blocking time data

```
//Function to Print out the blocking times of all the Trains
void PrintTrainBlockingTimes(string MainFolder) {
  string FileName;
  FileName = FileName + MainFolder + "/BlockingTimes.txt"; //Name of output file
  ofstream OutputFile;
  OutputFile.open((char*)FileName.c_str(), ios::binary);

  for (int i = 0; i<N_Reg; i++) {
    OutputFile << "TrainDescription ";  //Print text "TrainDescription"
    //Print the train description
    OutputFile << T[i].TrainDescription << "\n";

    //Print ID of the block
    OutputFile << "BlockID ";

    for (int j = 0; j<T[i].N_BlockTimeComplete; j++) {
      OutputFile << T[i].BlockTime[j].BlockID << " ";
    }
    OutputFile << "\n";

    //Print the start location of the block
    OutputFile << "GeoPosStart ";

    for (int j = 0; j<T[i].N_BlockTimeComplete; j++) {
      OutputFile << T[i].BlockTime[j].GeoPosStart << " ";

    }
    OutputFile << "\n";

    //Print the end location of the block
    OutputFile << "GeoPosEnd ";

    for (int j = 0; j<T[i].N_BlockTimeComplete; j++) {
      OutputFile << T[i].BlockTime[j].GeoPosEnd << " ";
    }

    OutputFile << "\n";

    //Print the time when the occupation time starts
    OutputFile << "StartOccTime ";

    for (int j = 0; j<T[i].N_BlockTimeComplete; j++) {
      OutputFile << T[i].BlockTime[j].StartOccTime << " ";
    }
    OutputFile << "\n";

    //Print the time when the occupation time ends
    OutputFile << "EndOccTime ";
```

```
49
50      for (int j = 0; j<T[i].N_BlockTimeComplete; j++) {
51        OutputFile << T[i].BlockTime[j].EndOccTime << " ";
52      }
53      OutputFile << "\n";
54
55      //Print the length of the setup time
56      OutputFile << "setupTime ";
57
58      for (int j = 0; j < T[i].N_BlockTimeComplete; j++) {
59        OutputFile << T[i].BlockTime[j].setupTime << " ";
60      }
61      OutputFile << "\n";
62
63      //Print the length of the sight and reaction time
64      OutputFile << "sightReacTime ";
65
66      for (int j = 0; j < T[i].N_BlockTimeComplete; j++) {
67        OutputFile << T[i].BlockTime[j].sightReacTime << " ";
68      }
69      OutputFile << "\n";
70
71      //Print the length of the approach time
72      OutputFile << "ApproachTime ";
73
74      for (int j = 0; j < T[i].N_BlockTimeComplete; j++) {
75        OutputFile << T[i].BlockTime[j].ApproachTime << " ";
76      }
77      OutputFile << "\n";
78
79      //Print the length of the running time
80      OutputFile << "RunTime ";
81
82      for (int j = 0; j < T[i].N_BlockTimeComplete; j++) {
83        OutputFile << T[i].BlockTime[j].RunTime << " ";
84      }
85      OutputFile << "\n";
86
87      //Print the length of the clearing time
88      OutputFile << "clearingTime ";
89
90      for (int j = 0; j < T[i].N_BlockTimeComplete; j++) {
91        OutputFile << T[i].BlockTime[j].clearingTime << " ";
92      }
93      OutputFile << "\n";
94
95          //Print the length of the release time
96      OutputFile << "ReleaseTime ";
97
98      for (int j = 0; j < T[i].N_BlockTimeComplete; j++) {
99        OutputFile << T[i].BlockTime[j].ReleaseTime << " ";
100     }
101     OutputFile << "\n";
102
103
104     //Print the length of the block in meters
105     OutputFile << "BlockLength ";
106
107     for (int j = 0; j < T[i].N_BlockTimeComplete; j++) {
108       OutputFile << T[i].BlockTime[j].length << " ";
109     }
110     OutputFile << "\n";
111
112         //Print the speed when the train enters the block
113     OutputFile << "V_run ";
114
115     for (int j = 0; j < T[i].N_BlockTimeComplete; j++) {
116       int srt = (int) T[i].BlockTime[j].StartRunTime;
117       OutputFile << T[i].V[srt] << " ";
118     }
119     OutputFile << "\n";
```

```
120
121     OutputFile << "V_Approach ";
122
123         //Print the speed when the train is at the approach distance
124     for (int j = 0; j < T[i].N_BlockTimeComplete; j++) {
125       int srt = (int)T[i].BlockTime[j].StartApproachTime;
126       OutputFile << T[i].V[srt] << " ";
127     }
128     OutputFile << "\n";
129
130     OutputFile << "V_Clear ";
131
132         //Print the speed when the train leaves the block
133     for (int j = 0; j < T[i].N_BlockTimeComplete; j++) {
134       int srt = (int)T[i].BlockTime[j].StartClearTime;
135       OutputFile << T[i].V[srt] << " ";
136     }
137     OutputFile << "\n";
138   }
139 }
```

Speed-distance data

```
1  void PrintSpeedDiagram(Train *Train, string FolderName) {
2
3     //Create output file
4   string FileName;
5   FileName = FolderName + "/SpeedDiagram.txt";
6   ofstream OutputFile;
7   OutputFile.open((char*)FileName.c_str(), ios::binary);
8
9
10    //Print timesteps
11  OutputFile << "Time ";
12
13  for (int t = 0; t <= 9500; t++) {
14    OutputFile << t * timestep << " ";
15  }
16
17  OutputFile << "\n";
18
19
20  for (int i = 0; i < N_Reg; i++) {
21
22    //Print the speed at every timestep
23    OutputFile << T[i].TrainDescription << "_Speed ";
24    for (int t = 0; t <= 9500; t++) {
25        OutputFile << Train[i].V[t] << " ";
26    }
27    OutputFile << "\n";
28
29        //Print the position at every timestep
30    OutputFile << T[i].TrainDescription << "_Position ";
31    for (int t = 0; t <= 9500; t++) {
32      OutputFile << Train[i].S[t] << " ";
33    }
34    OutputFile << "\n";
35  }
36 }
```

C

Python scripts verification study

```
import pandas as pd
import numpy as np
pd.options.mode.chained_assignment = None
import matplotlib.pyplot as plt
import math

from matplotlib.collections import PatchCollection
from matplotlib.patches import Rectangle

import seaborn as sns

import matplotlib.patches as mpatches
from matplotlib.pyplot import figure
```

Merge Link and node data

```
#Load link and node data
Link = pd.read_excel('link database.xlsx')
node = pd.read_excel('Node database.xlsx')

#Merging
Link['Length'] = 0.0
Link['LengthCum'] = 0.0

for i in range(0,len(Link)):
    for j in range(0,len(node)-1):
        if Link.StartNodeID[i] == node.NodeID[j]:
            Link.Length[i] = node.X[j+1] - node.X[j]
            if i == 0:
                Link.LengthCum[i] = Link.Length[i]
            else:
                Link.LengthCum[i] = Link.LengthCum[i-1] + Link.Length[i]
            i += 1

Link['Distance'] = Link.LengthCum * 1000.0


Link['A_Gradient'] = 0.0
Link['Acceleration'] = 0.0
A = 0.6 #braking deceleration of both trains in the verification study is 0.6m/s^2

for i in range(len(Link)):
    Link.A_Gradient[i] = 9.81 * math.sin(math.atan(Link.Gradient[i]))
    Link.Acceleration[i] = Link.A_Gradient[i] + A

Link.to_excel('Link_data.xlsx')
```

Approach time model

```
#Load link data
Link = pd.read_excel('Link_data.xlsx')

#Load fixed block occupation data
A2_1 = pd.read_excel('FB_A2_1.xlsx')
A2_2 = pd.read_excel('FB_A2_2.xlsx')
A2_3 = pd.read_excel('FB_A2_3.xlsx')
A2_4 = pd.read_excel('FB_A2_4.xlsx')
A2_5 = pd.read_excel('FB_A2_5.xlsx')
A2_6 = pd.read_excel('FB_A2_6.xlsx')

A3_1 = pd.read_excel('FB_A3_1.xlsx')
A3_2 = pd.read_excel('FB_A3_2.xlsx')
A3_3 = pd.read_excel('FB_A3_3.xlsx')
A3_4 = pd.read_excel('FB_A3_4.xlsx')
A3_5 = pd.read_excel('FB_A3_5.xlsx')
A3_6 = pd.read_excel('FB_A3_6.xlsx')

#Load trajectory data
A2_1_TSD = pd.read_excel('A2_1_TSD2.xlsx')
A2_2_TSD = pd.read_excel('A2_2_TSD2.xlsx')
A2_3_TSD = pd.read_excel('A2_3_TSD2.xlsx')
A2_4_TSD = pd.read_excel('A2_4_TSD2.xlsx')
A2_5_TSD = pd.read_excel('A2_5_TSD2.xlsx')
A2_6_TSD = pd.read_excel('A2_6_TSD2.xlsx')

A3_1_TSD = pd.read_excel('A3_1_TSD2.xlsx')
A3_2_TSD = pd.read_excel('A3_2_TSD2.xlsx')
A3_3_TSD = pd.read_excel('A3_3_TSD2.xlsx')
A3_4_TSD = pd.read_excel('A3_4_TSD2.xlsx')
A3_5_TSD = pd.read_excel('A3_5_TSD2.xlsx')
A3_6_TSD = pd.read_excel('A3_6_TSD2.xlsx')

#Calculate performable deceleration at every timestep
def Acc(Speed):
    Speed['Acceleration'] = 0.0
    Link = pd.read_excel('Link_data.xlsx')
    for i in range(len(Speed)):
        for j in range(1,len(Link)):
            if Speed.Distance[i] <  Link.Distance[j] and Speed.Distance[i] >= Link.Distance[j
    -1]:
                Speed.Acceleration[i] = Link.Acceleration [j]
                if i < len(Speed)-1:
                    i+=1

    return(Speed)

#Approach time estimation model
def Gradient(FB, Speed):

    FB['Est_App_Time2'] = 0.0
    for j in range(1,(len(FB))):
        V = 0.0
        s = 0.0
        v_null = 0.0
        for i in range(len(Speed)):
            if Speed.Distance[i] >= FB.GeoPosStart[j]:
                if Speed.Speed[i-1] == 0.0: #als de snelheid al nul is, dan approach time ook
                    FB.Est_App_Time[j] = 0.0
                else: #Als de trein rijdt, dan approach time uitrekenen
                    brake_loc = Speed.Distance[i-1]
                    for p in range(1,200):
                        for m in range(100):
                            if V > Speed.Speed[i-p-1]:
                                break
                            elif brake_loc >= Speed.Distance[i-p-1]:
                                s += 0.5 * Speed.Acceleration[i-p-1] + V
                                V += Speed.Acceleration[i-p-1]
                                brake_loc = Speed.Distance[i-1] - s
```

```
                            elif brake_loc < Speed.Distance[i-p-1]:
                                break
                        if V > Speed.Speed[i-p-1]:
                            break
                        else:
                            continue

                        print('Warning: hier hadden we niet moeten komen')

                    FB.Est_App_Time2[j] = Speed.Time[i-1] - Speed.Time[i-p-1]
                    print ('Est App Time: ', FB.Est_App_Time2[j])
                    break

#execute the models
print('Start dataset 1')
Acc(A2_1_TSD)
print('Start dataset 2')
Acc(A2_2_TSD)
print('Start dataset 3')
Acc(A2_3_TSD)
print('Start dataset 4')
Acc(A2_4_TSD)
print('Start dataset 5')
Acc(A2_5_TSD)
print('Start dataset 6')
Acc(A2_6_TSD)

print('Start dataset 1')
Acc(A3_1_TSD)
print('Start dataset 2')
Acc(A3_2_TSD)
print('Start dataset 3')
Acc(A3_3_TSD)
print('Start dataset 4')
Acc(A3_4_TSD)
print('Start dataset 5')
Acc(A3_5_TSD)
print('Start dataset 6')
Acc(A3_6_TSD)

print('Start dataset 1')
Gradient(A2_1, A2_1_TSD)
print('Start dataset 2')
Gradient(A2_2, A2_2_TSD)
print('Start dataset 3')
Gradient(A2_3, A2_3_TSD)
print('Start dataset 4')
Gradient(A2_4, A2_4_TSD)
print('Start dataset 5')
Gradient(A2_5, A2_5_TSD)
print('Start dataset 6')
Gradient(A2_6, A2_6_TSD)

print('Start dataset 1')
Gradient(A3_1, A3_1_TSD)
print('Start dataset 2')
Gradient(A3_2, A3_2_TSD)
print('Start dataset 3')
Gradient(A3_3, A3_3_TSD)
print('Start dataset 4')
Gradient(A3_4, A3_4_TSD)
print('Start dataset 5')
Gradient(A3_5, A3_5_TSD)
print('Start dataset 6')
Gradient(A3_6, A3_6_TSD)
```

Clearing time model

```
#Load data
A2_1 = pd.read_excel('FB_A2_1.xlsx')
A2_2 = pd.read_excel('FB_A2_2.xlsx')
A2_3 = pd.read_excel('FB_A2_3.xlsx')
A2_4 = pd.read_excel('FB_A2_4.xlsx')
A2_5 = pd.read_excel('FB_A2_5.xlsx')
A2_6 = pd.read_excel('FB_A2_6.xlsx')

A3_1 = pd.read_excel('FB_A3_1.xlsx')
A3_2 = pd.read_excel('FB_A3_2.xlsx')
A3_3 = pd.read_excel('FB_A3_3.xlsx')
A3_4 = pd.read_excel('FB_A3_4.xlsx')
A3_5 = pd.read_excel('FB_A3_5.xlsx')
A3_6 = pd.read_excel('FB_A3_6.xlsx')

A2_1_TSD = pd.read_excel('A2_1_TSD2.xlsx')
A2_2_TSD = pd.read_excel('A2_2_TSD2.xlsx')
A2_3_TSD = pd.read_excel('A2_3_TSD2.xlsx')
A2_4_TSD = pd.read_excel('A2_4_TSD2.xlsx')
A2_5_TSD = pd.read_excel('A2_5_TSD2.xlsx')
A2_6_TSD = pd.read_excel('A2_6_TSD2.xlsx')

A3_1_TSD = pd.read_excel('A3_1_TSD2.xlsx')
A3_2_TSD = pd.read_excel('A3_2_TSD2.xlsx')
A3_3_TSD = pd.read_excel('A3_3_TSD2.xlsx')
A3_4_TSD = pd.read_excel('A3_4_TSD2.xlsx')
A3_5_TSD = pd.read_excel('A3_5_TSD2.xlsx')
A3_6_TSD = pd.read_excel('A3_6_TSD2.xlsx')

#Model
def Clearing(df, tsd):
    L = 161.84 #Length of the trains
    df['Est_ClearingTime'] = 9999.0
    for i in range(len(df)):
        for j in range(len(tsd)):
            if df.StartRunTime[i] == tsd.Time[j]:
                for k in range(600):
                    if tsd.Distance[j+k] > tsd.Distance[j] + L:
                        df.Est_ClearingTime[i] = k
                        break
                    else:
                        continue
            else:
                continue
            break

#Execute the model of each train separately
Clearing(A2_1, A2_1_TSD)
Clearing(A2_2, A2_2_TSD)
Clearing(A2_3, A2_3_TSD)
Clearing(A2_4, A2_4_TSD)
Clearing(A2_5, A2_5_TSD)
Clearing(A2_6, A2_6_TSD)

Clearing(A3_1, A3_1_TSD)
Clearing(A3_2, A3_2_TSD)
Clearing(A3_3, A3_3_TSD)
Clearing(A3_4, A3_4_TSD)
Clearing(A3_5, A3_5_TSD)
Clearing(A3_6, A3_6_TSD)

#merge all trains in one dataset
df = pd.concat([A2_1, A2_2, A2_3, A2_4, A2_5, A2_6,
                A3_1, A3_2, A3_3, A3_4, A3_5, A3_6], sort=False)

#Calculate correlation
df.corr()
```

D

Python scripts case study

```
import pandas as pd
import numpy as np
pd.options.mode.chained_assignment = None
import matplotlib.pyplot as plt
import math

from matplotlib.collections import PatchCollection
from matplotlib.patches import Rectangle

import seaborn as sns

import matplotlib.patches as mpatches
from matplotlib.pyplot import figure
```

Calculation of deceleration rates

```
blok = pd.read_excel('blokbezetting friso.xlsx')

#Creating list of trains
trains = []

for i in range(len(blok)):
    if blok.TreinNaam[i] not in trains:
        trains.append(blok.TreinNaam[i])
    else:
        continue

#Creating 'blokbezetting' and 'gradient' files per train
for i in range(len(trains)):
    df = blok[blok.TreinNaam == trains[i]]
    df.to_excel('blokbezetting_'+trains[i]+'.xlsx')

for i in range(len(trains)):
    df = rosa[rosa.Trein == trains[i]]
    df.to_excel('gradient_'+trains[i]+'.xlsx')

#Define function to link datasets
def Remweg(q):
    print('Start met: ', trains[q])
    blok = pd.read_excel('blokbezetting_'+trains[q]+'.xlsx')
    rosa = pd.read_excel(trains[q]+'_rosagradient.xlsx')

    blok['remweg'] = np.nan


    for i in range(len(blok)):
        for j in range(len(rosa)):
```

```
            if rosa.id2[j] == blok.SeinID[i] and rosa.kilometrering[j] == blok.SeinKilometrering[i]:
                blok.remweg[i] = rosa.remAfstandM[j]
                if i < len(blok)-1:
                    i += 1
                    continue
                else:
                    break

    blok.to_excel('blok_remweg_'+trains[q]+'.xlsx')
    print('klaar met ', trains[q])

#Run function
for q in range(len(trains)):
    Remweg(q)

for i in range(len(trains)):
    df = dis[dis.naam == trains[i]]
    df.to_excel('rosa_data_'+trains[i]+'.xlsx')

#Define function to link the gradient to the rosa data and calculate acceleration rates
def grad(q):
    print('Start met: ', trains[q])
    rosa = pd.read_excel('rosa_data_'+trains[q]+'.xlsx')
    gradient = pd.read_excel('gradient_'+trains[q]+'.xlsx')

    print('gradient is nul')
    rosa['gradient'] = 0.0

    for i in range(0,len(rosa)):
        for j in range(1,len(gradient)):
            if rosa.Distance[i] == gradient.Distance[j]:
                rosa.gradient[i] = gradient.Gradient[j-1]
                break
            elif rosa.Distance[i] < gradient.Distance[j]:
                rosa.gradient[i] = gradient.Gradient[j-1]
                break
            else:
                rosa.gradient[i] = 0.0
        continue

    print('Acceleratie uitrekenen')
    rosa['Acceleration'] = 0.0
    for i in range(len(rosa)):
        grav = 9.81 * math.sin(math.atan(rosa.gradient[i]))
        if rosa.treinType[i] == 'IC' or rosa.treinType[i] == 'SPR' or rosa.treinType[i] == 'HS' or rosa.treinType[i] == 'IR' or rosa.treinType[i] == 'R' or rosa.treinType[i] == 'LM':
            rosa.Acceleration[i] = 0.5 + grav
        elif rosa.treinType[i] == 'G':
            rosa.Acceleration[i] = 0.2 + grav
        else:
            print(rosa.treinType[i], 'NOT FOUND!')

    rosa.to_excel(trains[q]+'_rosagradient.xlsx')
    print('Klaar met: ', trains[q])
```

Calculation fixed block occupation times

```
blok = pd.read_excel('blokbezetting friso.xlsx')

for i in range(len(trains)):
    df = blok[blok.TreinNaam == trains[i]]
    df.to_excel('blokbezetting_'+trains[i]+'.xlsx')

def Remweg(q):
    print('Start met: ', trains[q])
    blok = pd.read_excel('blokbezetting_'+trains[q]+'.xlsx')
    rosa = pd.read_excel(trains[q]+'_rosagradient.xlsx')
```

```
11
12      blok['remweg'] = np.nan
13
14      for i in range(len(blok)):
15          for j in range(len(rosa)):
16              if rosa.id2[j] == blok.SeinID[i] and rosa.kilometrering[j] == blok.
    SeinKilometrering[i]:
17                  blok.remweg[i] = rosa.remAfstandM[j]
18                  if i < len(blok)-1:
19                      i += 1
20                      continue
21                  else:
22                      break
23
24      blok.to_excel('blok_remweg_'+trains[q]+'.xlsx')
25      print('klaar met ', trains[q])
26
27  for q in range(len(trains)):
28      Remweg(q)
29
30  def block(trains):
31      for i in range(len(trains)):
32          print('Start met: ', i)
33          df = pd.read_excel('blok_remweg_'+trains[i]+'.xlsx')
34          df['setup'] = 3.0
35          df['reaction'] = 9.0
36          df['approach'] = 0.0
37          df['running'] = 0.0
38          df['clearing'] = 0.0
39          df['release'] = 1.0
40          df['StartOccTime'] = 0.0
41          df['EndOccTime'] = 0.0
42          df['BlokLengte'] = 0.0
43
44          for k in range(1,len(df)):
45              for l in range(10):
46                  if k >= l:
47                      if df.remweg[k-l] <= df.AfgelegdeAfstandTrein[k] - df.
    AfgelegdeAfstandTrein[k-l]:
48                          df.approach[k] = df.TijdstipVoorkantPassage[k] - df.
    TijdstipVoorkantPassage[k-l]
49                          break
50                  elif (k-l) == 0:
51                      if df.remweg[k-l] <= df.AfgelegdeAfstandTrein[k] - df.
    AfgelegdeAfstandTrein[k-l]:
52                          df.approach[k] = df.TijdstipVoorkantPassage[k] - df.
    TijdstipVoorkantPassage[0]
53                          break
54                  else:
55                      continue
56
57          df.approach[0] = 0
58
59          for j in range(0,len(df)-1):
60              df.running[j] = df.TijdstipVoorkantPassage[j+1] - df.TijdstipVoorkantPassage[j]
61              df.BlokLengte[j] = df.AfgelegdeAfstandTrein[j+1] - df.AfgelegdeAfstandTrein[j]
62
63              if df.TijdstipAchterkantPassage[j+1] == 0.0:
64                  df.clearing[j] = 0.0
65              else:
66                  df.clearing[j] = df.TijdstipAchterkantPassage[j+1] - df.
    TijdstipVoorkantPassage[j+1]
67
68          df.clearing[len(df)] = 0.0
69          df.running[len(df)] = 0.0
70          df.BlokLengte[len(df)] = 0.0
71
72          for h in range(len(df)):
73              df.StartOccTime[h] = df.TijdstipVoorkantPassage[h] - df.approach[h] - df.reaction
    [h] - df.setup[h]
74              df.EndOccTime[h] = df.TijdstipVoorkantPassage[h] + df.running[h] + df.clearing[h]
```

```
     + df.release[h]

        df.to_excel('FB_blockingtimes_'+trains[i]+'.xlsx')
        print('klaar met: ', i)

block(trains)
```

Moving block model

```
def AppTime(pos):
    for r in range(0,len(pos)):
        print('Start met ', r, ' -- trein: ', pos[r])
        dfrosa = pd.read_excel(pos[r]+'_rosagradient.xlsx')
        dfsec = dfrosa
        dfsec['Section_found'] = 9999.0
        dfsec['MB_Route_form'] = 1.0
        dfsec['MB_ReacTime'] = 9.0
        dfsec['MB_ApproachTime'] = 9999.0
        dfsec['MB_ReleaseTime'] = 1.0
        dfsec['MB_RunningTime'] = 0.0
        dfsec['MB_ClearingTime'] = 0.0

        for j in range(0, len(dfsec)):
            V = 0.0
            s = 0.0
            v_null = 0.0
            brake_loc = 0.0
            PassTime = 0.0
            for i in range(0,(len(dfrosa))):

                if dfsec.id2[j] == dfrosa.id2[i] and dfrosa.voorkantSnelheidMS[i] == 0:
                    dfsec.Section_found[j] = 1.0
                    dfsec.MB_ApproachTime[j] = 0.0
                    break

                elif dfsec.id2[j] == dfrosa.id2[i]:
                    dfsec.Section_found[j] = 2.0
                    PassTime = dfrosa.voorkantPassageTijdstipS[i]

                    brake_loc = dfrosa.Distance[i]
                    for p in range(0,200):

                        for m in range(0,100):
                            V_free_flow = dfrosa.voorkantSnelheidMS[i-p] - (dfrosa.
    voorkantVersnellingMS2[i-p-1] * m)
                            if V >= V_free_flow or brake_loc < 0:
                                break
                            elif brake_loc > dfrosa.Distance[i-p-1]:
                                s += 0.5 * dfrosa.Acceleration[i-p-1] + V
                                V += dfrosa.Acceleration[i-p-1]
                                brake_loc = dfrosa.Distance[i] - s
                            elif brake_loc <= dfrosa.Distance[i-p-1]:
                                break

                        if V >= V_free_flow or brake_loc < 0:
                            break
                        else:
                            continue

                        print('WARNING! hier hadden we niet moeten komen')

                    for h in range(0,10):
                        if dfrosa.voorkantIntervalS[i-p-h] != 0:
                            T = (((dfrosa.Distance[i-p] - brake_loc) / (dfrosa.Distance[i-p]
    - dfrosa.Distance[i-p-1-h])) *
                                        dfrosa.voorkantIntervalS[i-p-h])
                            break
                        else:
                            continue
```

```
                    dfsec.MB_ApproachTime[j] = (PassTime - dfrosa.voorkantPassageTijdstipS[i-p]) + T
                    break
                else:
                    dfsec.Section_found[j] = 0.0


        for b in range(0,len(dfsec)-1):
            dfsec.MB_ClearingTime[b] = dfsec.achterkantPassageTijdstipS[b] - dfsec.voorkantPassageTijdstipS[b]

        dfsec.to_excel('MB_OccTimes_'+pos[r]+'.xlsx')
        print('Klaar met ', r)
```

Coupling Fixed Block and Moving Block datasets

```
df = pd.read_excel('rijweginstelling_14122021.xlsx')

for j in range(len(trains)):
    data = df[df.TreinNaam == trains[j]]
    data.to_excel(trains[j]+'_rijweginstelling.xlsx')

def koppel(j):
    print('Start met: ',j, " - ", trains[j])
    rijweg = pd.read_excel(trains[j]+'_rijweginstelling.xlsx')
    sectie = pd.read_excel(trains[j]+'_Sectiebezetting.xlsx')
    rosa = pd.read_excel('MB_OccTimes_'+trains[j]+'.xlsx')
    FB = pd.read_excel('FB_blockingtimes_'+trains[j]+'.xlsx')

    rosa = rosa[rosa.Distance >= 0.0]
    rosa = rosa.rename(columns={"key": 'TypeRijweg', "value":"Station"})
    rosa['MB_StartOccTime'] = 0.0
    rosa['MB_EndOccTime'] = 0.0
    rosa['MB_OccTime'] = 0.0


    #Calculation of start- and end of the blocking time for moving block
    for k in range(len(rosa)):
        if rosa.MB_ApproachTime[k] >= 0.0:
            rosa.MB_StartOccTime[k] = rosa.voorkantPassageTijdstipS[k] - rosa.MB_ApproachTime[k] - rosa.MB_ReacTime[k] - rosa.MB_Route_form[k]
        if rosa.MB_ApproachTime[k] < 0.0:
            rosa.MB_StartOccTime[k] = rosa.voorkantPassageTijdstipS[k] - rosa.MB_ReacTime[k] - rosa.MB_Route_form[k]

        rosa.MB_EndOccTime[k] = rosa.voorkantPassageTijdstipS[k] + rosa.MB_RunningTime[k] + rosa.MB_ClearingTime[k] + rosa.MB_ReleaseTime[k]


        if rosa.type[k] == 'WISSEL' or rosa.type[k] == 'AHOB' or rosa.type[k] == 'AOB' or rosa.type[k] == 'OVW_ONBEWAAKT' or rosa.type[k] == 'HAHOB' or rosa.type[k] == 'AHOB_FIETS' or rosa.type[k] == 'EBO_HAND' or rosa.type[k] == 'EBO_CENTRAAL':
            if rosa.MB_StartOccTime[k] >= 14.0:
                rosa.MB_StartOccTime[k] = rosa.MB_StartOccTime[k] - 14.0
            else:
                rosa.MB_StartOccTime[k] = 0.0
        else:
            rosa.MB_StartOccTime[k] = rosa.MB_StartOccTime[k] - 1.0

        rosa.MB_OccTime[k] = rosa.MB_EndOccTime[k] - rosa.MB_StartOccTime[k]

    #Adding columns for FB data
    rosa['FB_Route_form'] = 1.0
    rosa['FB_ReacTime'] = 9.0
    rosa['FB_ApproachTime'] = 0.0
    rosa['FB_ReleaseTime'] = 1.0
    rosa['FB_RunningTime'] = 0.0
    rosa['FB_ClearingTime'] = 0.0
```

```
    rosa['FB_StartOccTime'] = 0.0
    rosa['FB_EndOccTime'] = 0.0
    rosa['FB_OccTime'] = 0.0

    FB['SeinNaam'] = FB['SeinNaam'].apply(str)
    FB['SeinType'] = FB['SeinType'].apply(str)
    rosa['naam3'] = rosa['naam3'].apply(str)
    rosa['kilometerLint'] = rosa['kilometerLint'].apply(str)
    rosa['sectieNaam1'] = rosa['sectieNaam1'].apply(str)
    rosa['TypeRijweg'] = rosa['TypeRijweg'].apply(str)
    rijweg['VanSeinNaam'] = rijweg['VanSeinNaam'].apply(str)
    rijweg['VanSeinKilometerling'] = rijweg['VanSeinKilometerling'].apply(str)
    sectie['Kilometerlint'] = sectie['Kilometerlint'].apply(str)

    rosa.TypeRijweg[0] = 'BED'

    sectie['DRPnaam'].replace({'-':np.nan})

    #Finding stations/stops on the route
    for w in range(len(sectie)-1):
        if sectie.AfrijTijd[w] - sectie.OprijTijd[w] >= 65.0:
            for z in range(len(rosa)):
                if sectie.Sectie[w] == rosa.sectieNaam1[z] and sectie.DRPnaam[w] == rosa.
    dienstregelpuntCode[z]:
                    rosa.Station[z] = 'Ja'
                elif sectie.Sectie[w+1] == rosa.sectieNaam1[z] and sectie.DRPnaam[w+1] ==
    rosa.dienstregelpuntCode[z] and sectie.OprijTijd[w+1]*0.97 < rosa.
    voorkantPassageTijdstipS[z] < sectie.OprijTijd[w+1]*1.03:
                    rosa.Station[z] = 'Na'

    #copying fixed block blocking times to all signals
    for i in range(len(rosa)):
        for k in range(len(FB)):
            if rosa.id2[i] == FB.SeinID[k]:
                rosa.TypeRijweg[i] = FB.SeinType[k]
                rosa.FB_ApproachTime[i] = FB.approach[k]
                rosa.FB_RunningTime[i] = FB.running[k]
                rosa.FB_ClearingTime[i] = FB.clearing[k]
                rosa.FB_StartOccTime[i] = FB.StartOccTime[k]

                if rosa.TypeRijweg[i] == 'ONB': #ONB = Onbediend (Open lines)
                    rosa.FB_EndOccTime[i] = FB.EndOccTime[k]
                break

    #Copying approach, running and clearing times from the signal to the other infra elements
     in the block
    for p in range(1, len(rosa)):
        if rosa.TypeRijweg[p] != 'BED' and rosa.TypeRijweg[p] != 'ONB':
            rosa.TypeRijweg[p] = rosa.TypeRijweg[p-1]
            rosa.FB_ApproachTime[p] = rosa.FB_ApproachTime[p-1]
            rosa.FB_RunningTime[p] = rosa.FB_RunningTime[p-1]
            rosa.FB_ClearingTime[p] = rosa.FB_ClearingTime[p-1]

    #Applying sectional route release in interlocking areas
    for m in range(len(rosa)):
        for n in range(len(sectie)):
            if (rosa.sectieNaam1[m] == sectie.Sectie[n] and rosa.TypeRijweg[m] == 'BED'
                and rosa.dienstregelpuntCode[m] == sectie.DRPnaam[n]
                and (sectie.OprijTijd[n]-60.0) < rosa.voorkantPassageTijdstipS[m] < (sectie.
    OprijTijd[n]+60.0)):
                rosa.FB_EndOccTime[m] = sectie.AfrijTijd[n] + 1.0

    #Applying route setup times for the block at and after a stop
    for u in range(len(rosa)):
        if rosa.Station[u] == 'Na' or rosa.Station[u] == 'Ja':
            for g in range(len(rijweg)):
                if (rosa.naam3[u] == rijweg.VanSeinNaam[g]
                    and rosa.id2[u] == rijweg.VanSeinID[g]
                    and rosa.richting[u] == 'M'):
                    rosa.FB_StartOccTime[u] = rijweg.TijdstipBeginRijwegInstelling[g]
            if rosa.Station[u] == 'Na':
```

```
                for t in range(1,20):
                    if rosa.Station[u-t] == 'Na':
                        rosa.FB_StartOccTime[u-t] = rosa.FB_StartOccTime[u]
                        continue
                    else:
                        break

    #Define end times for elements at the start of the simulations
    for u in range(len(rosa)):
        if rosa.FB_EndOccTime[u] == 0.0:
            continue
        else:
            rosa.FB_EndOccTime[0] = rosa.FB_EndOccTime[u]
            break

    #Filling gaps
    for p in range(1, len(rosa)):
        if rosa.FB_StartOccTime[p] == 0.0:
            rosa.FB_StartOccTime[p] = rosa.FB_StartOccTime[p-1]
        if rosa.FB_EndOccTime[p] == 0.0:
            rosa.FB_EndOccTime[p] = rosa.FB_EndOccTime[p-1]

    for y in range(1,len(rosa)):
        if rosa.FB_StartOccTime[y] < rosa.FB_StartOccTime[y-1] and rosa.achterkantSnelheidMS[
    y] > 0.0:
            for p in range(1,50):
                if rosa.FB_StartOccTime[y] == rosa.FB_StartOccTime[y+p] and (y+p+1) < len(
    rosa):
                    continue
                else:
                    for u in range(0,p):
                        rosa.FB_StartOccTime[y+u] = rosa.voorkantPassageTijdstipS[y] - 24.0
                    break

        rosa.FB_OccTime[y] = rosa.FB_EndOccTime[y] - rosa.FB_StartOccTime[y]

    #Finish
    rosa.to_excel(trains[j]+'_FB_MB_OccTimes_v14.xlsx')
    print('Klaar met: ', trains[j])

#Run function
for j in range(20, len(trains)):
    koppel(j)

#Adding all datasets to one dataset
block_list = []

for i in range(len(trains)):
    df = pd.read_excel(trains[i]+'_FB_MB_OccTimes_v14.xlsx')
    block_list.append(df)
    df2 = pd.concat(block_list)

df2 = df2.sort_values(by='FB_StartOccTime')
df2.to_excel('Alles_OccupationTimes_FB_MB_v3.xlsx')
```

Buffer time calculation and ananlysis

```
#Creating basic hour pattern
data = pd.read_excel('Alles_OccupationTimes_FB_MB_v3.xlsx')

for i in range(len(data)):
    if data.FB_StartOccTime[i] > 3600.0 and data.FB_StartOccTime[i] < 7200.0:
        data.FB_StartOccTime[i] = data.FB_StartOccTime[i] - 3600.0
        data.FB_EndOccTime[i] = data.FB_EndOccTime[i] - 3600.0

    elif data.FB_StartOccTime[i] > 7200.0 and data.FB_StartOccTime[i] < 10800.0:
        data.FB_StartOccTime[i] = data.FB_StartOccTime[i] - 7200.0
        data.FB_EndOccTime[i] = data.FB_EndOccTime[i] - 7200.0

```

```
    elif data.FB_StartOccTime[i] > 10800.0 and data.FB_StartOccTime[i] < 14400.0:
        data.FB_StartOccTime[i] = data.FB_StartOccTime[i] - 10800.0
        data.FB_EndOccTime[i] = data.FB_EndOccTime[i] - 10800.0

for j in range(len(data)):
    if data.MB_StartOccTime[j] > 3600.0 and data.MB_StartOccTime[j] < 7200.0:
        data.MB_StartOccTime[j] = data.MB_StartOccTime[j] - 3600.0
        data.MB_EndOccTime[j] = data.MB_EndOccTime[j] - 3600.0

    elif data.MB_StartOccTime[j] > 7200.0 and data.MB_StartOccTime[j] < 10800.0:
        data.MB_StartOccTime[j] = data.MB_StartOccTime[j] - 7200.0
        data.MB_EndOccTime[j] = data.MB_EndOccTime[j] - 7200.0

    elif data.MB_StartOccTime[j] > 10800.0 and data.MB_StartOccTime[j] < 14400.0:
        data.MB_StartOccTime[j] = data.MB_StartOccTime[j] - 10800.0
        data.MB_EndOccTime[j] = data.MB_EndOccTime[j] - 10800.0

data = data.sort_values(by='FB_StartOccTime')
data.to_excel('Alles_OccupationTimes_FB_MB_v4.xlsx')

#Calculation fixed block buffer times
data = pd.read_excel('Alles_OccupationTimes_FB_MB_v4.xlsx')
sec_list = []

for i in range(len(data)):
    if data.type[i] == 'SECTIE' and data.id2[i] not in sec_list:
        sec_list.append(data.id2[i])

df = pd.DataFrame(sec_list, columns=['SectieID'])

df['SectieNaam'] = np.nan
df['kilometerLint'] = np.nan
df['dienstregelpuntCode'] = np.nan
df['kilometrering'] = np.nan
df['Train_count'] = 0.0

for i in range(1,14):
    df['TrainSeries_'+str(i)] = np.nan
    df['FB_Route_form_'+str(i)] = np.nan
    df['FB_ReacTime_'+str(i)] = np.nan
    df['FB_ApproachTime_'+str(i)] = np.nan
    df['FB_RunningTime_'+str(i)] = np.nan
    df['FB_ClearingTime_'+str(i)] = np.nan
    df['FB_ReleaseTime_'+str(i)] = np.nan
    df['FB_StartOccTime_'+str(i)] = np.nan
    df['FB_EndOccTime_'+str(i)] = np.nan
    df['FB_OccTime_'+str(i)] = np.nan
    df['FB_BufferTime_'+str(i)] = np.nan

for p in range(len(data)):
    if data.type[p] == 'SECTIE' and data.FB_StartOccTime[p] != 0.0 and data.MB_ApproachTime[p
    ] != 9999.0 and data.MB_OccTime[p] > 0.0:
        for h in range(len(df)):
            if data.id2[p] == df.SectieID[h]:

                df.Train_count[h] += 1.0

                df.SectieNaam[h] = data.naam3[p]
                df.kilometerLint[h] = data.kilometerLint[p]
                df.dienstregelpuntCode[h] = data.dienstregelpuntCode[p]
                df.kilometrering[h] = data.kilometrering[p]

                q = int(df.Train_count[h])

                df['FB_Route_form_'+str(q)].iloc[h] = data.FB_Route_form[p]
                df['FB_ReacTime_'+str(q)].iloc[h] = data.FB_ReacTime[p]
                df['FB_ApproachTime_'+str(q)].iloc[h] = data.FB_ApproachTime[p]
                df['FB_RunningTime_'+str(q)].iloc[h] = data.FB_RunningTime[p]
                df['FB_ClearingTime_'+str(q)].iloc[h] = data.FB_ClearingTime[p]
                df['FB_ReleaseTime_'+str(q)].iloc[h] = data.FB_ReleaseTime[p]
                df['FB_OccTime_'+str(q)].iloc[h] = data.FB_OccTime[p]
```

```
                df['TrainSeries_'+str(q)].iloc[h] = data.naam[p]
                df['FB_StartOccTime_'+str(q)].iloc[h] = data.FB_StartOccTime[p]
                df['FB_EndOccTime_'+str(q)].iloc[h] = data.FB_EndOccTime[p]

#Repeating the first train
for h in range(len(df)):
    q = int(df.Train_count[h]+1.0)

    df['FB_Route_form_'+str(q)].iloc[h] = df['FB_Route_form_'+str(1)].iloc[h]
    df['FB_ReacTime_'+str(q)].iloc[h] = df['FB_ReacTime_'+str(1)].iloc[h]
    df['FB_ApproachTime_'+str(q)].iloc[h] = df['FB_ApproachTime_'+str(1)].iloc[h]
    df['FB_RunningTime_'+str(q)].iloc[h] = df['FB_RunningTime_'+str(1)].iloc[h]
    df['FB_ClearingTime_'+str(q)].iloc[h] = df['FB_ClearingTime_'+str(1)].iloc[h]
    df['FB_ReleaseTime_'+str(q)].iloc[h] = df['FB_ReleaseTime_'+str(1)].iloc[h]
    df['FB_OccTime_'+str(q)].iloc[h] = df['FB_OccTime_'+str(1)].iloc[h]
    df['TrainSeries_'+str(q)].iloc[h] = df['TrainSeries_'+str(1)].iloc[h]
    df['FB_StartOccTime_'+str(q)].iloc[h] = df['FB_StartOccTime_'+str(1)].iloc[h] +3600.0
    df['FB_EndOccTime_'+str(q)].iloc[h] = df['FB_EndOccTime_'+str(1)].iloc[h] +3600.0

df = df[df.Train_count != 0.0]
df.to_excel('Alles_Sectiebezetting_volg_FB_b.xlsx')

#Calculation moving block buffer times
MB = pd.read_excel('Alles_OccupationTimes_FB_MB_v4.xlsx')
MB2 = MB.sort_values(by='MB_StartOccTime')
MB2.to_excel('OccTimes_volg_MB_4b.xlsx')

data = pd.read_excel('OccTimes_volg_MB_4b.xlsx')

sec_list = []

for i in range(len(data)):
    if data.type[i] == 'SECTIE' and data.id2[i] not in sec_list:
        sec_list.append(data.id2[i])

df = pd.DataFrame(sec_list, columns=['SectieID'])

df['SectieNaam'] = np.nan
df['kilometerLint'] = np.nan
df['dienstregelpuntCode'] = np.nan
df['kilometrering'] = np.nan
df['Train_count'] = 0.0

for i in range(1,14):
    df['TrainSeries_'+str(i)] = np.nan
    df['MB_Route_form_'+str(i)] = np.nan
    df['MB_ReacTime_'+str(i)] = np.nan
    df['MB_ApproachTime_'+str(i)] = np.nan
    df['MB_RunningTime_'+str(i)] = np.nan
    df['MB_ClearingTime_'+str(i)] = np.nan
    df['MB_ReleaseTime_'+str(i)] = np.nan
    df['MB_StartOccTime_'+str(i)] = np.nan
    df['MB_EndOccTime_'+str(i)] = np.nan
    df['MB_OccTime_'+str(i)] = np.nan
    df['MB_BufferTime_'+str(i)] = np.nan


for p in range(len(data)):
    if data.type[p] == 'SECTIE' and data.FB_StartOccTime[p] != 0.0 and data.MB_EndOccTime[p]
    > 0.0 and data.MB_ApproachTime[p] != 9999.0 and data.MB_OccTime[p] > 0.0:
        for h in range(len(df)):
            if data.id2[p] == df.SectieID[h]:

                df.Train_count[h] += 1.0

                df.SectieNaam[h] = data.naam3[p]
                df.kilometerLint[h] = data.kilometerLint[p]
                df.dienstregelpuntCode[h] = data.dienstregelpuntCode[p]
                df.kilometrering[h] = data.kilometrering[p]

                q = int(df.Train_count[h])
```

```
153
154                 df['MB_Route_form_'+str(q)].iloc[h] = data.MB_Route_form[p]
155                 df['MB_ReacTime_'+str(q)].iloc[h] = data.MB_ReacTime[p]
156                 df['MB_ApproachTime_'+str(q)].iloc[h] = data.MB_ApproachTime[p]
157                 df['MB_RunningTime_'+str(q)].iloc[h] = data.MB_RunningTime[p]
158                 df['MB_ClearingTime_'+str(q)].iloc[h] = data.MB_ClearingTime[p]
159                 df['MB_ReleaseTime_'+str(q)].iloc[h] = data.MB_ReleaseTime[p]
160                 df['MB_StartOccTime_'+str(q)].iloc[h] = data.MB_StartOccTime[p]
161                 df['MB_EndOccTime_'+str(q)].iloc[h] = data.MB_EndOccTime[p]
162                 df['MB_OccTime_'+str(q)].iloc[h] = data.MB_OccTime[p]
163                 df['TrainSeries_'+str(q)].iloc[h] = data.naam[p]
164
165 #Repeating the first train for each section
166 for h in range(len(df)):
167     q = int(df.Train_count[h]+1.0)
168
169     df['MB_Route_form_'+str(q)].iloc[h] = df['MB_Route_form_'+str(1)].iloc[h]
170     df['MB_ReacTime_'+str(q)].iloc[h] = df['MB_ReacTime_'+str(1)].iloc[h]
171     df['MB_ApproachTime_'+str(q)].iloc[h] = df['MB_ApproachTime_'+str(1)].iloc[h]
172     df['MB_RunningTime_'+str(q)].iloc[h] = df['MB_RunningTime_'+str(1)].iloc[h]
173     df['MB_ClearingTime_'+str(q)].iloc[h] = df['MB_ClearingTime_'+str(1)].iloc[h]
174     df['MB_ReleaseTime_'+str(q)].iloc[h] = df['MB_ReleaseTime_'+str(1)].iloc[h]
175     df['MB_OccTime_'+str(q)].iloc[h] = df['MB_OccTime_'+str(1)].iloc[h]
176     df['TrainSeries_'+str(q)].iloc[h] = df['TrainSeries_'+str(1)].iloc[h]
177     df['MB_StartOccTime_'+str(q)].iloc[h] = df['MB_StartOccTime_'+str(1)].iloc[h] +3600.0
178     df['MB_EndOccTime_'+str(q)].iloc[h] = df['MB_EndOccTime_'+str(1)].iloc[h] +3600.0
179
180
181 df = df[df.Train_count != 0.0]
182 df.to_excel('Sectiebezettingen_volg_MB_v2.xlsx')
183
184 #Adding Interlocking area types
185 sectie = pd.read_excel('lijst secties.xlsx')
186
187 FB = pd.read_excel('Alles_Sectiebezetting_volg_FB_b.xlsx')
188
189 for i in range(len(FB)):
190     for j in range(len(sectie)):
191         if FB.SectieNaam[i] == sectie.naam[j] and FB.dienstregelpuntCode[i] == sectie.DRP[j]:
192             FB.DRPtype[i] = sectie.DRPtype[j]
193             break
194
195 FB.to_excel('Alles_Sectiebezetting_volg_FB_v2.xlsx')
196
197 df = pd.read_excel('Sectiebezettingen_volg_MB_v2.xlsx')
198 for i in range(len(df)):
199     for j in range(len(sectie)):
200         if df.SectieNaam[i] == sectie.naam[j] and df.dienstregelpuntCode[i] == sectie.DRP[j]:
201             df.DRPtype[i] = sectie.DRPtype[j]
202             break
203
204 df.to_excel('Alles_Sectiebezetting_volg_MB_v2b.xlsx')
205
206 #Calculation of buffer times
207 data = pd.read_excel('Alles_Sectiebezetting_volg_FB_v2.xlsx')
208 for i in range(len(data)):
209     for j in range(2,13):
210         if data['FB_Route_form_'+str(j)].iloc[i] == 1.0:
211             data['FB_BufferTime_'+str(j-1)].iloc[i] = data['FB_StartOccTime_'+str(j)].iloc[i]
      - data['FB_EndOccTime_'+str(j-1)].iloc[i]
212         else:
213             break
214
215 data.to_excel('Alles_Sectiebezetting_volg_FB_v3.xlsx')
216
217 data = pd.read_excel('Alles_Sectiebezetting_volg_MB_v2b.xlsx')
218
219 for i in range(len(data)):
220     for j in range(2,13):
221         if data['MB_Route_form_'+str(j)].iloc[i] == 1.0:
222             data['MB_BufferTime_'+str(j-1)].iloc[i] = data['MB_StartOccTime_'+str(j)].iloc[i]
```

```
     - data['MB_EndOccTime_'+str(j-1)].iloc[i]
        else:
            break

data.to_excel('Alles_Sectiebezetting_volg_MB_v3b.xlsx')

#Creating a dataset with all buffer times in one column, including the details of the first and second train
def FBbufferdataset(FB):
    df = pd.DataFrame()
    df['SectieID'] = np.nan
    df['SectieNaam'] = np.nan
    df['KilometerLint'] = np.nan
    df['dienstregelpuntCode'] = np.nan
    df['kilometrering'] = np.nan
    df['DRPtype'] = np.nan
    df['TrainSeries_1'] = np.nan
    df['Route_form_1'] = np.nan
    df['ReacTime_1'] = np.nan
    df['ApproachTime_1'] = np.nan
    df['RunningTime_1'] = np.nan
    df['ClearingTime_1'] = np.nan
    df['ReleaseTime_1'] = np.nan
    df['StartOccTime_1'] = np.nan
    df['EndOccTime_1'] = np.nan
    df['OccTime_1'] = np.nan
    df['BufferTime'] = np.nan
    df['TrainSeries_2'] = np.nan
    df['Route_form_2'] = np.nan
    df['ReacTime_2'] = np.nan
    df['ApproachTime_2'] = np.nan
    df['RunningTime_2'] = np.nan
    df['ClearingTime_2'] = np.nan
    df['ReleaseTime_2'] = np.nan
    df['StartOccTime_2'] = np.nan
    df['EndOccTime_2'] = np.nan
    df['OccTime_2'] = np.nan

    for i in range(len(FB)):
        for j in range(1,12):

            if FB['FB_BufferTime_'+str(j)].iloc[i] > 0.0:
                df = df.append({'SectieID':FB.SectieID[i]}, ignore_index=True)
                df.SectieNaam[len(df)-1] = FB.SectieNaam[i]
                df.KilometerLint[len(df)-1] = FB.kilometerLint[i]
                df.dienstregelpuntCode[len(df)-1] = FB.dienstregelpuntCode[i]
                df.kilometrering[len(df)-1] = FB.kilometrering[i]
                df.DRPtype[len(df)-1] = FB.DRPtype[i]

                df.TrainSeries_1[len(df)-1] = FB['TrainSeries_'+str(j)].iloc[i]
                df.Route_form_1[len(df)-1] = FB['FB_Route_form_'+str(j)].iloc[i]
                df.ReacTime_1[len(df)-1] = FB['FB_ReacTime_'+str(j)].iloc[i]
                df.ApproachTime_1[len(df)-1] = FB['FB_ApproachTime_'+str(j)].iloc[i]
                df.RunningTime_1[len(df)-1] = FB['FB_RunningTime_'+str(j)].iloc[i]
                df.ClearingTime_1[len(df)-1] = FB['FB_ClearingTime_'+str(j)].iloc[i]
                df.ReleaseTime_1[len(df)-1] = FB['FB_ReleaseTime_'+str(j)].iloc[i]
                df.StartOccTime_1[len(df)-1] = FB['FB_StartOccTime_'+str(j)].iloc[i]
                df.EndOccTime_1[len(df)-1] = FB['FB_EndOccTime_'+str(j)].iloc[i]
                df.OccTime_1[len(df)-1] = FB['FB_OccTime_'+str(j)].iloc[i]
                df.BufferTime[len(df)-1] = FB['FB_BufferTime_'+str(j)].iloc[i]

                df.TrainSeries_2[len(df)-1] = FB['TrainSeries_'+str(j+1)].iloc[i]
                df.Route_form_2[len(df)-1] = FB['FB_Route_form_'+str(j+1)].iloc[i]
                df.ReacTime_2[len(df)-1] = FB['FB_ReacTime_'+str(j+1)].iloc[i]
                df.ApproachTime_2[len(df)-1] = FB['FB_ApproachTime_'+str(j+1)].iloc[i]
                df.RunningTime_2[len(df)-1] = FB['FB_RunningTime_'+str(j+1)].iloc[i]
                df.ClearingTime_2[len(df)-1] = FB['FB_ClearingTime_'+str(j+1)].iloc[i]
                df.ReleaseTime_2[len(df)-1] = FB['FB_ReleaseTime_'+str(j+1)].iloc[i]
                df.StartOccTime_2[len(df)-1] = FB['FB_StartOccTime_'+str(j+1)].iloc[i]
                df.EndOccTime_2[len(df)-1] = FB['FB_EndOccTime_'+str(j+1)].iloc[i]
                df.OccTime_2[len(df)-1] = FB['FB_OccTime_'+str(j+1)].iloc[i]
```

```
            elif FB['FB_BufferTime_'+str(j)].iloc[i] <= 0.0:
                continue

            else:
                break
    #print(df)
    df.to_excel('FB_Buffertijden.xlsx')

FB = pd.read_excel('Alles_Sectiebezetting_volg_FB_v3.xlsx')
FBbufferdataset(FB)

def bufferdataset(FB):
    df = pd.DataFrame()
    df['SectieID'] = np.nan
    df['SectieNaam'] = np.nan
    df['KilometerLint'] = np.nan
    df['dienstregelpuntCode'] = np.nan
    df['kilometrering'] = np.nan
    df['DRPtype'] = np.nan
    df['TrainSeries_1'] = np.nan
    df['Route_form_1'] = np.nan
    df['ReacTime_1'] = np.nan
    df['ApproachTime_1'] = np.nan
    df['RunningTime_1'] = np.nan
    df['ClearingTime_1'] = np.nan
    df['ReleaseTime_1'] = np.nan
    df['StartOccTime_1'] = np.nan
    df['EndOccTime_1'] = np.nan
    df['OccTime_1'] = np.nan
    df['BufferTime'] = np.nan
    df['TrainSeries_2'] = np.nan
    df['Route_form_2'] = np.nan
    df['ReacTime_2'] = np.nan
    df['ApproachTime_2'] = np.nan
    df['RunningTime_2'] = np.nan
    df['ClearingTime_2'] = np.nan
    df['ReleaseTime_2'] = np.nan
    df['StartOccTime_2'] = np.nan
    df['EndOccTime_2'] = np.nan
    df['OccTime_2'] = np.nan

    for i in range(len(FB)):
        for j in range(1,12):

            if FB['MB_BufferTime_'+str(j)].iloc[i] > 0.0:
                df = df.append({'SectieID':FB.SectieID[i]}, ignore_index=True)
                df.SectieNaam[len(df)-1] = FB.SectieNaam[i]
                df.KilometerLint[len(df)-1] = FB.kilometerLint[i]
                df.dienstregelpuntCode[len(df)-1] = FB.dienstregelpuntCode[i]
                df.kilometrering[len(df)-1] = FB.kilometrering[i]
                df.DRPtype[len(df)-1] = FB.DRPtype[i]

                df.TrainSeries_1[len(df)-1] = FB['TrainSeries_'+str(j)].iloc[i]
                df.Route_form_1[len(df)-1] = FB['MB_Route_form_'+str(j)].iloc[i]
                df.ReacTime_1[len(df)-1] = FB['MB_ReacTime_'+str(j)].iloc[i]
                df.ApproachTime_1[len(df)-1] = FB['MB_ApproachTime_'+str(j)].iloc[i]
                df.RunningTime_1[len(df)-1] = FB['MB_RunningTime_'+str(j)].iloc[i]
                df.ClearingTime_1[len(df)-1] = FB['MB_ClearingTime_'+str(j)].iloc[i]
                df.ReleaseTime_1[len(df)-1] = FB['MB_ReleaseTime_'+str(j)].iloc[i]
                df.StartOccTime_1[len(df)-1] = FB['MB_StartOccTime_'+str(j)].iloc[i]
                df.EndOccTime_1[len(df)-1] = FB['MB_EndOccTime_'+str(j)].iloc[i]
                df.OccTime_1[len(df)-1] = FB['MB_OccTime_'+str(j)].iloc[i]
                df.BufferTime[len(df)-1] = FB['MB_BufferTime_'+str(j)].iloc[i]

                df.TrainSeries_2[len(df)-1] = FB['TrainSeries_'+str(j+1)].iloc[i]
                df.Route_form_2[len(df)-1] = FB['MB_Route_form_'+str(j+1)].iloc[i]
                df.ReacTime_2[len(df)-1] = FB['MB_ReacTime_'+str(j+1)].iloc[i]
                df.ApproachTime_2[len(df)-1] = FB['MB_ApproachTime_'+str(j+1)].iloc[i]
                df.RunningTime_2[len(df)-1] = FB['MB_RunningTime_'+str(j+1)].iloc[i]
                df.ClearingTime_2[len(df)-1] = FB['MB_ClearingTime_'+str(j+1)].iloc[i]
                df.ReleaseTime_2[len(df)-1] = FB['MB_ReleaseTime_'+str(j+1)].iloc[i]
```

```
                df.StartOccTime_2[len(df)-1] = FB['MB_StartOccTime_'+str(j+1)].iloc[i]
                df.EndOccTime_2[len(df)-1] = FB['MB_EndOccTime_'+str(j+1)].iloc[i]
                df.OccTime_2[len(df)-1] = FB['MB_OccTime_'+str(j+1)].iloc[i]
            elif FB['MB_BufferTime_'+str(j)].iloc[i] <= 0.0:
                continue
            else:
                break
    df.to_excel('MB_Buffertijden.xlsx')

MB = pd.read_excel('Alles_Sectiebezetting_volg_MB_v3b.xlsx')
bufferdataset(MB)

MB = pd.read_excel('MB_Buffertijden.xlsx')
FB = pd.read_excel('FB_Buffertijden.xlsx')

#Due to an mistake in the timetable, train pair B120-D30900 had to be removed
FB = FB[(FB.train_merge != 'B120-H-1-D30900-H-1') &  (FB.train_merge != 'D30900-H-1-B120-H-1')]
MB = MB[(MB.train_merge != 'B120-H-1-D30900-H-1') & (MB.train_merge != 'D30900-H-1-B120-H-1')]

#Analysis and creating graphs
#Histogram FB buffer times
figure(figsize=(9, 5), dpi=80)
plt.hist(FB.BufferTime, bins=100, range=(0,2000))
plt.xlabel('Buffertime [s]')
plt.ylabel('Number of sections')
plt.ylim(0,1400)

#Histogram MB Buffer times
figure(figsize=(9, 5), dpi=80)
plt.hist(MB.BufferTime, bins=100, range=(0,2000), color='r')
plt.xlabel('Buffertime [s]')
plt.ylabel('Number of sections')
plt.ylim(0,1400)

#Histogram both FB and MB buffer times
figure(figsize=(9, 5), dpi=80)
plt.hist(MB.BufferTime, bins=100, range=(0,2000), color='r', label='Moving block')
plt.hist(FB.BufferTime, bins=100, range=(0,2000), label='Fixed block', alpha=0.8)
plt.xlabel('Buffertime [s]')
plt.ylabel('Number of sections')
plt.legend()
plt.ylim(0,1400)

#Histogram FB buffer times < 60s
figure(figsize=(9, 5), dpi=80)
plt.hist(FB.BufferTime, bins=30, range=(0,60))
plt.xlabel('Buffertime [s]')
plt.ylabel('Number of sections')
plt.ylim(0,60)

#Histogram MB buffer times < 60s
figure(figsize=(9, 5), dpi=80)
plt.hist(MB.BufferTime, bins=30, range=(0,60), color='r')
plt.xlabel('Buffertime [s]')
plt.ylabel('Number of sections')
plt.ylim(0,60)

#Finding the critical section for all train pairs
df3 = pd.read_excel('FB_Buffertijden.xlsx')

df3['Combi'] = np.nan

for i in range(len(df3)):
    df3.Combi[i] = df3.TrainSeries_1[i]+'-'+df3.TrainSeries_2[i]

combi_list = []

for i in range(len(df3)):
    if df3.Combi[i] not in combi_list:
```

```
        combi_list.append(df3.Combi[i])
    else:
        continue

data3=pd.DataFrame(combi_list, columns=['Combi'])

data3['SectieID'] = np.nan
data3['ObjectType'] = 'SECTIE'
data3['ObjectNaam'] = np.nan
data3['Kilometerlint'] = np.nan
data3['kilometrering'] = np.nan
data3['Dienstregelpunt'] = np.nan
data3['DRPtype'] = np.nan
data3['Buffertijd'] = 0.0

for i in range(len(data3)):
    for j in range(len(df3)):
        if df3.Combi[j] == data3.Combi[i]:
            data3.SectieID[i] = df3.SectieID[j]
            data3.ObjectNaam[i] = df3.SectieNaam[j]
            data3.Kilometerlint[i] = df3.KilometerLint[j]
            data3.kilometrering[i] = df3.kilometrering[j]
            data3.Dienstregelpunt[i] = df3.dienstregelpuntCode[j]
            data3.DRPtype[i] = df3.DRPtype[j]
            data3.Buffertijd[i] = df3.BufferTime[j]
            break
        else:
            continue

data3.to_excel('FB_CriticalSections_a.xlsx')
df2 = pd.read_excel('MB_Buffertijden.xlsx')
df2['Combi'] = np.nan

for i in range(len(df2)):
    df2.Combi[i] = df2.TrainSeries_1[i]+'-'+df2.TrainSeries_2[i]

combi_list = []
for i in range(len(df2)):
    if df2.Combi[i] not in combi_list:
        combi_list.append(df2.Combi[i])
    else:
        continue
data2=pd.DataFrame(combi_list, columns=['Combi'])

data2['SectieID'] = np.nan
data2['ObjectType'] = 'SECTIE'
data2['ObjectNaam'] = np.nan
data2['Kilometerlint'] = np.nan
data2['kilometrering'] = np.nan
data2['Dienstregelpunt'] = np.nan
data2['DRPtype'] = np.nan
data2['Buffertijd'] = 0.0

for i in range(len(data2)):
    for j in range(len(df2)):
        if df2.Combi[j] == data2.Combi[i]:
            data2.SectieID[i] = df2.SectieID[j]
            data2.ObjectNaam[i] = df2.SectieNaam[j]
            data2.Kilometerlint[i] = df2.KilometerLint[j]
            data2.kilometrering[i] = df2.kilometrering[j]
            data2.Dienstregelpunt[i] = df2.dienstregelpuntCode[j]
            data2.DRPtype[i] = df2.DRPtype[j]
            data2.Buffertijd[i] = df2.BufferTime[j]
            break
        else:
            continue
data2.to_excel('MB_CriticalSections_a.xlsx')
```

E

Case study: Rolling stock

Train count plus rolling stock ID:

- 2x High Speed (HS)
  - 2x 1427
- 44x Intercity (IC)
  - 4x1379, 10x 1433, 6x 1434, 4x1490, 20x1491
- 72x Sprinter (SPR)
  - 4x 1387, 8x 1395, 6x 1460, 4x1462, 16x 1464, 8x1469, 4x1470, 16x1471, 4x1479, 2x 1482
- 26x Regional
  - 2x1387, 4x1390, 2x1391, 2x1403, 8x1404, 4x1415, 2x1416, 2x1448
- 2x Interregional
  - 2x1399
- 28x Empty rolling stock (LM)
  - 2x 1367, 4x1434, 4x1469, 8x1471, 2x1485, 8x1491

	Train	Train Type	Rolling stock	Length [m]	Route
1	A120-H-1	HSL	ICE 08 bakken	200	Em-Ah-Ut-Asd
2	A700-H-1	Intercity	DDZ10	254	Dvaw-Asdz-Shl-Ledn
3	A800-H-1	Intercity	VIRM08	216	Mt-Std-Rm-Wt-Ehv-Ht-Ut-Asa-Asd
4	A1500-H-1	Intercity	ICMm08	214	Asdm-Asd
5	A1600-H-1	Intercity	ICMm09	243	Dvaw-Asdz-Shl
6	A1700-H-1	Intercity	ICMm09	243	Ut
7	A1800-H-1	Intercity	DDZ10	254	Dvaw-Asdz-Shl-Ledn
8	A2400-H-1	Intercity	VIRM10	270	Rtd-Sdm-Dt-Gv-Laa-Ledn-Shl-Asdz-Dvd
9	A2600-H-1	Intercity	ICMm08	214	Asdm-Asd
10	A3000-H-1	Intercity	VIRM10	270	Nm-Ah-Ed-Klp-Ut-Asa-Asd
11	A3100-H-1	Intercity	VIRM10	270	Nm-Ah-Ed-Db-Ut-Asb-Asdz-Shl
12	A3300-H-1	Sprinter	SNG10	195	Ledn-Ssh-Nvp-Hfd-Shl-Asdl
13	A3500-H-1	Intercity	VIRM10	270	Vl-Br-Hrt-Dn-Hm-Ehv-Ht-Ut-Asb-Asdz-Shl
14	A3900-H-1	Intercity	VIRM10	270	Hrl-Std-Rm-Wt-Ehv-Ht-Ut-Asa-Asd
15	A4300-H-1	Sprinter	SLT06	101	Dvaw-Rai-Asdz-Shl-Hfd
16	A4400-H-1	Sprinter	SLT08	138	Dn-Hmbh-Hm-Hmh-Hmbv-Ehv-Ehs-Bet-Btl-Vg-Ht-O
17	A5000-H-1	Sprinter	SGMm06 (3+3)	158	Rtd-Sdm-Dtcp-Dt-Rsw-Gvmw-Gv
18	A5100-H-1	Sprinter	SGMm06 (3+3)	158	Rtd-Sdm-Dtcp-Dt-Rsw-Gvmw-Gv
19	A5200-H-1	Sprinter	Flirt07 FFF	144	Ehv-Ehs-Bet-Btl-Ot-Tb-Tbu
20	A6300-H-1	Sprinter	SNG07	135	Laa-Gvm-Vst-Dvnk-Ledn-Vh
21	A6400-H-1	Sprinter	Flirt07 FFF	144	Wt-Mz-Hze-Gp-Ehv
22	A6500-H-1	Sprinter	SLT06	101	Htnc-Htn-Utln-Utvr-Ut
23	A6900-H-1	Sprinter	SLT10	170	Gdm-Cl-Htnc-Htn-Utln-Utvr-Ut
24	A7200-H-1	Stop train	GWE03	56	Gdm
25	A7300-H-1	Sprinter	SGMm06 (3+3)	158	Har-Mrn-Db-Bnk-Utvr-Ut-Utzl-Mas-Bkl
26	A7400-H-1	Sprinter	SGMm06 (3+3)	158	Har-Mrn-Db-Bnk-Utvr-Ut-Utzl-Mas-Bkl-Ac-Ashd-Asb-Dvd-Asa-Asdm-Asd
27	A7500-H-1	Sprinter	Flirt03 FFF	63	Ah-Otb-Wf-Ed
28	A7600-H-1	Sprinter	SGMm05	131	Wc-Nmd-Nmgo-Nm-Nml-Est-Ahz-Ah-Ahp-Ahpr-Vp-Rh
29	A7700-H-1	Sprinter	SGMm03	79	Wspl-Rtd
30	A11600-H-1	Intercity	ICMm09	243	Dvaw-Asdz-Shl
31	A11700-H-1	Intercity	ICMm09	243	Ut
32	A14600-H-1	Sprinter	SLT10	170	Asd-Asdm-Assp
33	A15800-H-1	Sprinter	SLT10	170	Asdm-Asd
34	A20000-H-1	Stop train	Flirt05 AB	91	Em-El-Zv-Ah
35	A30700-H-1	Stop train	GTW03	56	Zv-Dvn-Wtv-Ahp-Ah
36	A30900-H-1	Stop train	GTW03	56	Zv-Dvn-Wtv-Ahp-Ah
37	A31100-H-1	Stop train	GTW02	41	Ah-Est-Za-Hmn-Op
38	A31300-H-1	Stop train	Flirt03 FFF	63	Ed-Edc
39	A32400-H-1	Stop train	Flirt04 TAG	91	Rm-Ec-Srn-Std-Lut-Bk-Bde-Mt
40	A32500-H-1	Stop train	GWE02	41	Std-Gln-Sbk-Sn-Nh-Hb-Hrl
41	A71600-H-1	Empty rolling stock	ICMm09	243	Shl-Hfdo

	Train	Train Type	Rolling stock	Length [m]	Route
42	A73100-H-1	Empty rolling stock	VIRM10	270	Shl-Hfdo
43	A73500-H-1	Empty rolling stock	VIRM10	270	Shl-Hfdo
44	A74300-H-1	Empty rolling stock	SLT06	101	Hfd-Hfdo
45	A75700-H-1	Empty rolling stock	SLT10	170	Hfd-Hfdm
46	A75800-H-1	Empty rolling stock	SLT10	170	Hfd-Hfdm
47	A79200-H-1	Empty rolling stock	BR186 2x + 7 HST prio	223	Asd
48	A79300-H-1	Empty rolling stock	Thalys 10 bakken	200	Asd
49	A91400-H-1	Fast train	Flirt10 AB	182	Vl-Kn
50	B120-H-1	HSL	ICE 08 bakken	200	Asd-Ut-Ah-Em
51	B700-H-1	Intercity	DDZ10	254	Laa-Ledn-Shl-Asdz
52	B1500-H-1	Intercity	ICMm08	214	Asd
53	B1600-H-1	Intercity	ICMm09	243	Shl-Asdz-Dvd
54	B1800-H-1	Intercity	DDZ10	254	Laa-Ledn-Shl-Asdz
55	B2400-H-1	Intercity	VIRM10	270	Dvaw-Asdz-Shl-Ledn-Laa-Gv-Dt-Sdm-Rtd
56	B2600-H-1	Intercity	ICMm08	214	Asd
57	B3000-H-1	Intercity	VIRM10	270	Asd-Asa-Ut-Klp-Ed-Ah-Nm
58	B3100-H-1	Intercity	VIRM10	270	Shl-Asdz-Asb-Ut-Db-Ed-Ah-Nm
59	B3500-H-1	Intercity	VIRM10	270	Shl-Asdz-Asb-Ut-Ht-Ehv-Hm-Dn-Hrt-Br-Vl
60	B3900-H-1	Intercity	VIRM10	270	Asd-Asa-Ut-Ht-Ehv-Wt-Rm-Std-Hrl
61	B4300-H-1	Sprinter	SLT06	101	Hfd-Shl-Asdz-Rai-Dvd
62	B4400-H-1	Sprinter	SLT08	138	O-Ow-Rs-Hto-Ht-Vg-Btl-Bet-Ehs-Ehv-Hmbv-Hmh-Hm-Hmbh-Dn
63	B5000-H-1	Sprinter	SGMm06 (3+3)	158	Gv-Gvmw-Rsw-Dt-Dtcp-Sdm-Rtd
64	B5100-H-1	Sprinter	SGMm06 (3+3)	158	Gv-Gvmw-Rsw-Dt-Dtcp-Sdm-Rtd
65	B5200-H-1	Sprinter	Flirt07 FFF	144	Tbu-Tb-Ot-Btl-Bet-Ehs-Ehv
66	B6000-H-1	Sprinter	SLT10	170	Ut-Utvr-Utln-Htn-Htnc-Cl-Gdm-Zbm-Ht
67	B6300-H-1	Sprinter	SNG07	135	Ledn-Dvnk-Vst-Gvm-Laa
68	B6400-H-1	Sprinter	Flirt07 FFF	144	Ehv-Gp-Hze-Mz-Wt
69	B6900-H-1	Sprinter	SLT10	170	Ut-Utvr-Utln-Htn-Htnc-Cl-Gdm
70	B7300-H-1	Sprinter	SGMm06 (3+3)	158	Bkl-Mas-Utzl-Ut-Utvr-Bnk-Db-Mrn-Vndw
71	B7400-H-1	Sprinter	SGMm06 (3+3)	158	Asd-Asdm-Asa-Dvd-Asb-Ashd-Ac-Bkl-Mas-Utzl-Ut-Utvr-Bnk-Db-Mrn-Vndw
72	B7500-H-1	Sprinter	Flirt03 FFF	63	Ed-Wf-Otb-Ah
73	B7600-H-1	Sprinter	SGMm05	131	Va-Ahp-Ah-Ahz-Est-Nml-Nm-Nmgo-Nmd-Wc
74	B7700-H-1	Sprinter	SGMm03	79	Rtd
75	B11600-H-1	Intercity	ICMm09	243	Ledn-Ldl
76	B11700-H-1	Intercity	ICMm09	243	Shl-Asdz-Dvd
77	B14600-H-1	Sprinter	SLT10	170	Asdm-Asd
78	B15800-H-1	Sprinter	SLT10	170	Asd-Asdm-Assp
79	B20000-H-1	Stop train	Flirt05 AB	91	Ah-Zv-El-Em
80	B30700-H-1	Stop train	GTW03	56	Ah-Ahp-Wtv-Dvn-Zv
81	B30900-H-1	Stop train	GTW03	56	Ah-Ahp-Wtv-Dvn-Zv
82	B32300-H-1	Stop train	Lint 41 04 bakken	82	Nm-Nmh

	Train	Train Type	Rolling stock	Length [m]	Route
83	B32400-H-1	Stop train	Flirt04 TAG	91	Mt-Bde-Bk-Lut-Std-Srn-Ec-Rm
84	B32500-H-1	Stop train	GWE02	41	Hrl-Hb-Nh-Sn-Sbk-Gln-Std
85	B71600-H-1	Empty rolling stock	ICMm09	243	Hfdo-Shl
86	B73100-H-1	Empty rolling stock	VIRM10	270	Hfdo-Shl
87	B73500-H-1	Empty rolling stock	VIRM10	270	Hfdo-Shl
88	B74300-H-1	Empty rolling stock	SLT06	101	Hfdo-Hfd
89	B75700-H-1	Empty rolling stock	SLT10	170	Hfdm-Hfd
90	B75800-H-1	Empty rolling stock	SLT10	170	Hfdm-Hfd
91	B79200-H-1	Empty rolling stock	BR186 2x + 7 HST prio	223	Asdmw-Asd
92	B79300-H-1	Empty rolling stock	Thalys 10 bakken	200	Asdmw-Asd
93	B91400-H-1	Fast train	Flirt10 AB	182	Kn-Vl
94	C800-H-1	Intercity	VIRM08	216	Mt-Std-Rm-Wt-Ehv-Ht-Ut-Asa-Asd
95	C1500-H-1	Intercity	ICMm08	214	Asdm-Asd
96	C2400-H-1	Intercity	VIRM10	270	Rtd-Sdm-Dt-Gv-Laa-Ledn-Shl-Asdz-Dvd
97	C2600-H-1	Intercity	ICMm08	214	Asdm-Asd
98	C3100-H-1	Intercity	VIRM10	270	Nm-Ah-Ed-Db-Ut-Asb-Asdz-Shl
99	C3300-H-1	Sprinter	SNG10	195	Ledn-Ssh-Nvp-Hfd-Shl-Asdl
100	C3500-H-1	Intercity	VIRM10	270	Vl-Br-Hrt-Dn-Hm-Ehv-Ht-Ut-Asb-Asdz-Shl
101	C3900-H-1	Intercity	VIRM10	270	Hrl-Std-Rm-Wt-Ehv-Ht-Ut-Asa-Asd
102	C4300-H-1	Sprinter	SLT06	101	Dvaw-Rai-Asdz-Shl-Hfd
103	C4400-H-1	Sprinter	SLT08	138	Dn-Hmbh-Hm-Hmh-Hmbv-Ehv-Ehs-Bet-Btl-Vg-Ht-O
104	C5000-H-1	Sprinter	SGMm06 (3+3)	158	Rtd-Sdm-Dtcp-Dt-Rsw-Gvmw-Gv
105	C5100-H-1	Sprinter	SGMm06 (3+3)	158	Rtd-Sdm-Dtcp-Dt-Rsw-Gvmw-Gv
106	C5200-H-1	Sprinter	Flirt07 FFF	144	Ehv-Ehs-Bet-Btl-Ot-Tb-Tbu
107	C6300-H-1	Sprinter	SNG07	135	Laa-Gvm-Vst-Dvnk-Ledn-Vh
108	C6400-H-1	Sprinter	Flirt07 FFF	144	Wt-Mz-Hze-Gp-Ehv
109	C6500-H-1	Sprinter	SLT06	101	Htnc-Htn-Utln-Utvr-Ut
110	C6900-H-1	Sprinter	SLT10	170	Gdm-Cl-Htnc-Htn-Utln-Utvr-Ut
111	C7200-H-1	Stop train	GWE03	56	Gdm
112	C7300-H-1	Sprinter	SGMm06 (3+3)	158	Har-Mrn-Db-Bnk-Utvr-Ut-Utzl-Mas-Bkl
113	C7400-H-1	Sprinter	SGMm06 (3+3)	158	Har-Mrn-Db-Bnk-Utvr-Ut-Utzl-Mas-Bkl-Ac-Ashd-Asb-Dvd-Asa-Asdm-Asd
114	C7500-H-1	Sprinter	Flirt03 FFF	63	Ah-Otb-Wf-Ed
115	C7600-H-1	Sprinter	SGMm05	131	Wc-Nmd-Nmgo-Nm-Nml-Est-Ahz-Ah-Ahp-Ahpr-Vp-Rh
116	C7700-H-1	Sprinter	SGMm03	79	Wspl-Rtd
117	C14600-H-1	Sprinter	SLT10	170	Asd-Asdm-Assp
118	C15800-H-1	Sprinter	SLT10	170	Asdm-Asd
119	C30700-H-1	Stop train	GTW03	56	Zv-Dvn-Wtv-Ahp-Ah
120	C30900-H-1	Stop train	GTW03	56	Zv-Dvn-Wtv-Ahp-Ah
121	C31100-H-1	Stop train	GTW02	41	Ah-Est-Za-Hmn-Op
122	C31300-H-1	Stop train	Flirt03 FFF	63	Ed-Edc

	Train	Train Type	Rolling stock	Length [m]	Route
123	C32400-H-1	Stop train	Flirt04 TAG	91	Rm-Ec-Srn-Std-Lut-Bk-Bde-Mt
124	C32500-H-1	Stop train	GWE02	41	Std-Gln-Sbk-Sn-Nh-Hb-Hrl
125	C71600-H-1	Empty rolling stock	ICMm09	243	Shl-Hfdo
126	C73100-H-1	Empty rolling stock	VIRM10	270	Shl-Hfdo
127	C73500-H-1	Empty rolling stock	VIRM10	270	Shl-Hfdo
128	C74300-H-1	Empty rolling stock	SLT06	101	Hfd-Hfdo
129	C75700-H-1	Empty rolling stock	SLT10	170	Hfd-Hfdm
130	C75800-H-1	Empty rolling stock	SLT10	170	Hfd-Hfdm
131	D800-H-1	Intercity	VIRM08	216	Asd-Asa-Ut-Ht-Ehv-Wt-Rm-Std-Mt
132	D1500-H-1	Intercity	ICMm08	214	Asd
133	D2400-H-1	Intercity	VIRM10	270	Dvaw-Asdz-Shl-Ledn-Laa-Gv-Dt-Sdm-Rtd
134	D2600-H-1	Intercity	ICMm08	214	Asd
135	D3000-H-1	Intercity	VIRM10	270	Asd-Asa-Ut-Klp-Ed-Ah-Nm
136	D3500-H-1	Intercity	VIRM10	270	Shl-Asdz-Asb-Ut-Ht-Ehv-Hm-Dn-Hrt-Br-Vl
137	D3900-H-1	Intercity	VIRM10	270	Asd-Asa-Ut-Ht-Ehv-Wt-Rm-Std-Hrl
138	D4300-H-1	Sprinter	SLT06	101	Hfd-Shl-Asdz-Rai-Dvd
139	D4400-H-1	Sprinter	SLT08	138	O-Ow-Rs-Hto-Ht-Vg-Btl-Bet-Ehs-Ehv-Hmbv-Hmh-Hm-Hmbh-Dn
140	D5000-H-1	Sprinter	SGMm06 (3+3)	158	Gv-Gvmw-Rsw-Dt-Dtcp-Sdm-Rtd
141	D5100-H-1	Sprinter	SGMm06 (3+3)	158	Gv-Gvmw-Rsw-Dt-Dtcp-Sdm-Rtd
142	D5200-H-1	Sprinter	Flirt07 FFF	144	Tbu-Tb-Ot-Btl-Bet-Ehs-Ehv
143	D6000-H-1	Sprinter	SLT10	170	Ut-Utvr-Utln-Htn-Htnc-Cl-Gdm-Zbm-Ht
144	D6300-H-1	Sprinter	SNG07	135	Ledn-Dvnk-Vst-Gvm-Laa
145	D6400-H-1	Sprinter	Flirt07 FFF	144	Ehv-Gp-Hze-Mz-Wt
146	D6900-H-1	Sprinter	SLT10	170	Ut-Utvr-Utln-Htn-Htnc-Cl-Gdm
147	D7300-H-1	Sprinter	SGMm06 (3+3)	158	Bkl-Mas-Utzl-Ut-Utvr-Bnk-Db-Mrn-Vndw
148	D7600-H-1	Sprinter	SGMm05	131	Va-Ahp-Ah-Ahz-Est-Nml-Nm-Nmgo-Nmd-Wc
149	D7700-H-1	Sprinter	SGMm03	79	Rtd
150	D8900-H-1	Sprinter	SGMm03	79	Ledn-Ldl
151	D14600-H-1	Sprinter	SLT10	170	Asdm-Asd
152	D15800-H-1	Sprinter	SLT10	170	Asd-Asdm-Assp
153	D30700-H-1	Stop train	GTW03	56	Ah-Ahp-Wtv-Dvn-Zv
154	D30900-H-1	Stop train	GTW03	56	Ah-Ahp-Wtv-Dvn-Zv
155	D32300-H-1	Stop train	Lint 41 04 bakken	82	Nm-Nmh
156	D32400-H-1	Stop train	Flirt04 TAG	91	Mt-Bde-Bk-Lut-Std-Srn-Ec-Rm
157	D32500-H-1	Stop train	GWE02	41	Hrl-Hb-Nh-Sn-Sbk-Gln-Std
158	D71600-H-1	Empty rolling stock	ICMm09	243	Hfdo-Shl
159	D73100-H-1	Empty rolling stock	VIRM10	270	Hfdo-Shl
160	D73500-H-1	Empty rolling stock	VIRM10	270	Hfdo-Shl
161	D74300-H-1	Empty rolling stock	SLT06	101	Hfdo-Hfd
162	D75700-H-1	Empty rolling stock	SLT10	170	Hfdm-Hfd

	Train	Train Type	Rolling stock	Length [m]	Route
163	D75800-H-1	Empty rolling stock	SLT10	170	Hfdm-Hfd
164	E2000-H-1	Intercity	ICMm08	214	Ut
165	F6500-H-1	Sprinter	SLT06	101	Ut-Htnc
166	G2000-H-1	Intercity	ICMm08	214	Ut
167	H3100-H-1	Intercity	VIRM10	270	Shl-Asdz-Asb-Ut-Db-Ed-Ah-Nm
168	H6500-H-1	Sprinter	SLT06	101	Ut-Htnc
169	H7400-H-1	Sprinter	SGMm06 (3+3)	158	Asd-Asdm-Asa-Dvd-Asb-Ashd-Ac-Bkl-Mas-Utzl-Ut-Utvr-Bnk-Db-Mrn-Vndw
170	H7500-H-1	Sprinter	Flirt03 FFF	63	Ed-Wf-Otb-Ah
171	M6000-H-1	Sprinter	SLT10	170	Ht-Zbm-Gdm-Cl-Htnc-Htn-Utln-Utvr-Ut
172	N800-H-1	Intercity	VIRM08	216	Asd-Asa-Ut-Ht-Ehv-Wt-Rm-Std-Mt
173	O3000-H-1	Intercity	VIRM10	270	Nm-Ah-Ed-Klp-Ut-Asa-Asd
174	O6000-H-1	Sprinter	SLT10	170	Ht-Zbm-Gdm-Cl-Htnc-Htn-Utln-Utvr-Ut

F

Case study: Infrastructure

Microscopic overview of the infrastructure.

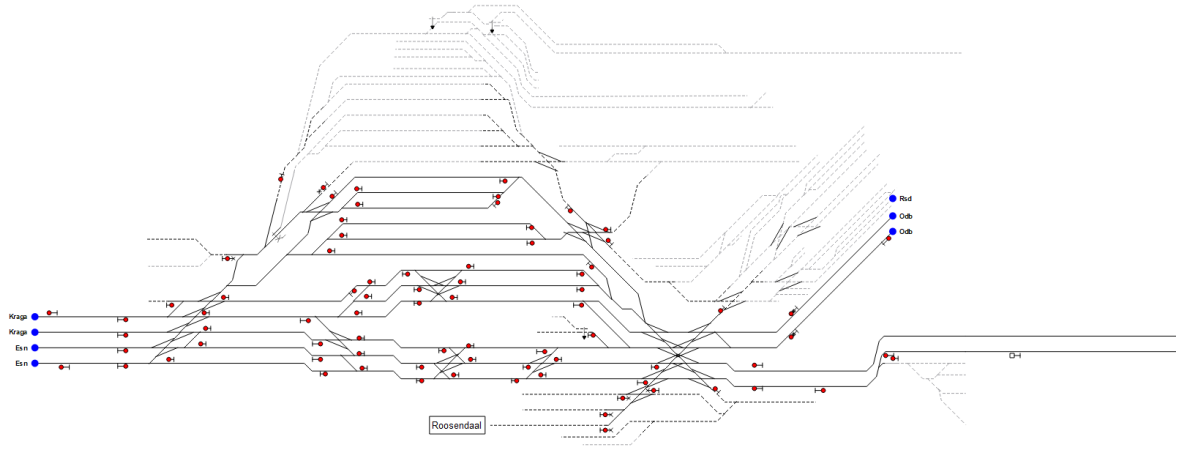


Figure F.1: Roosendaal → Ettenleur

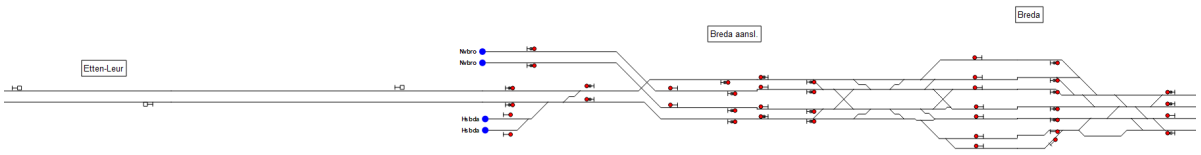


Figure F.2: Roosendaal ← Ettenleur ↔Breda → Gilzerijen

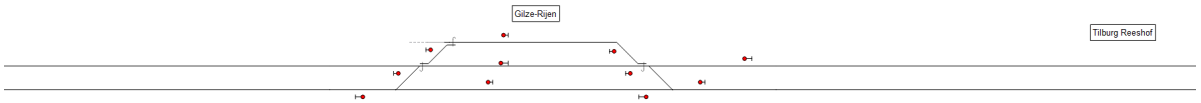


Figure F.3: Breda ← Gilzerijen → Tilburg

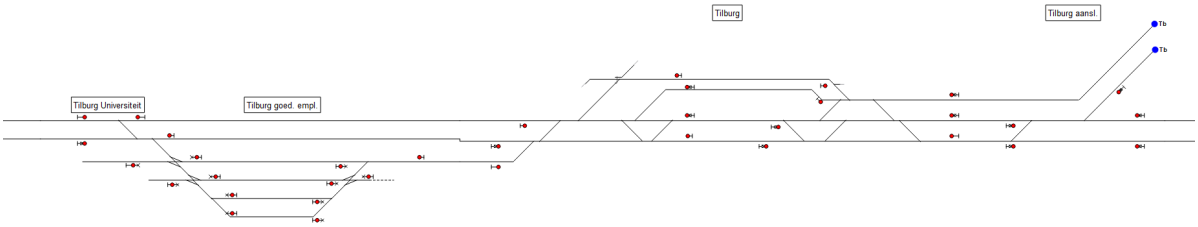


Figure F.4: Gilzerijen ← Tilburg → Boxtel

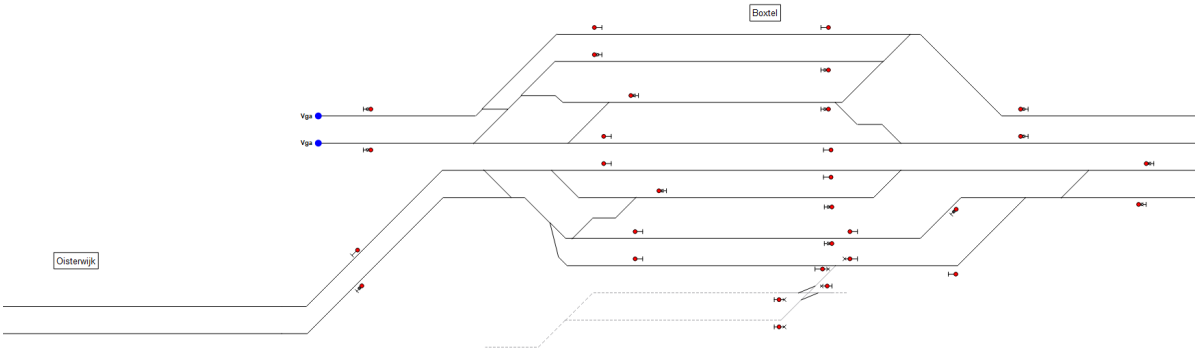


Figure F.5: Tilburg/'s-Hertogenbosch ← Boxtel → Liempde

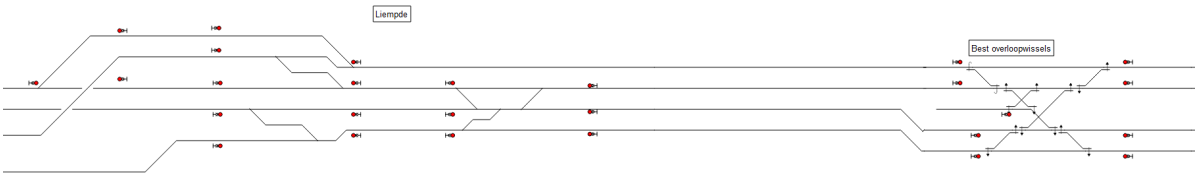


Figure F.6: Boxtel ← Liempde ↔Best → Eindhoven Strijp-S

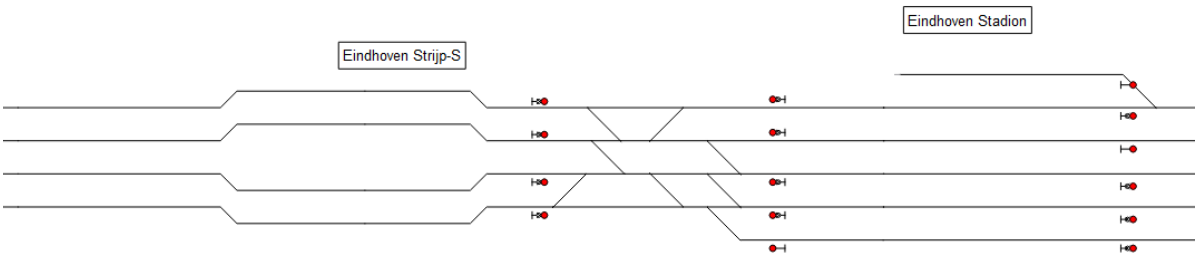


Figure F.7: Best ← Eindhoven Strijp-S ↔Eindhoven Stadion → Eindhoven Centraal

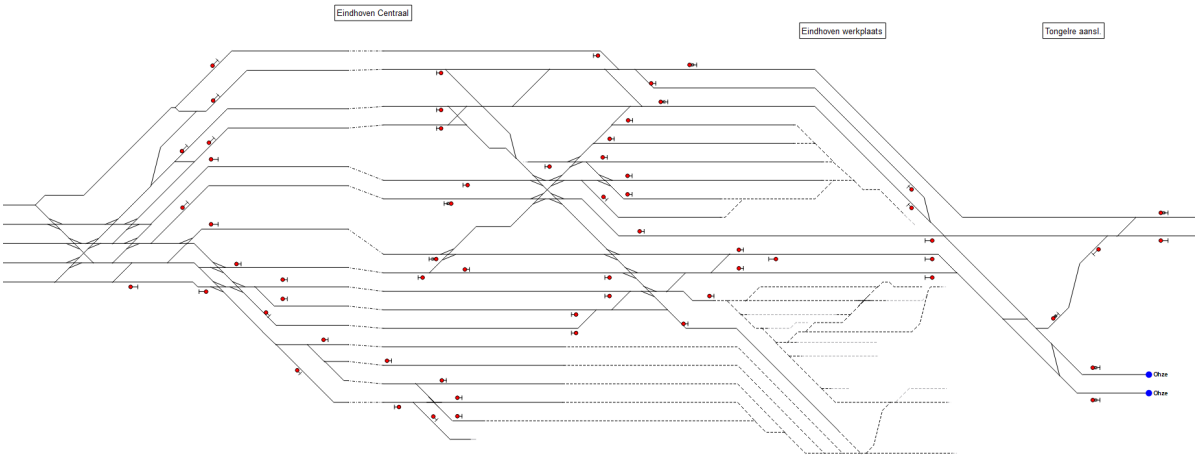


Figure F.8: Eindhoven Station ← Eindhoven Centraal → Helmond/Geldrop

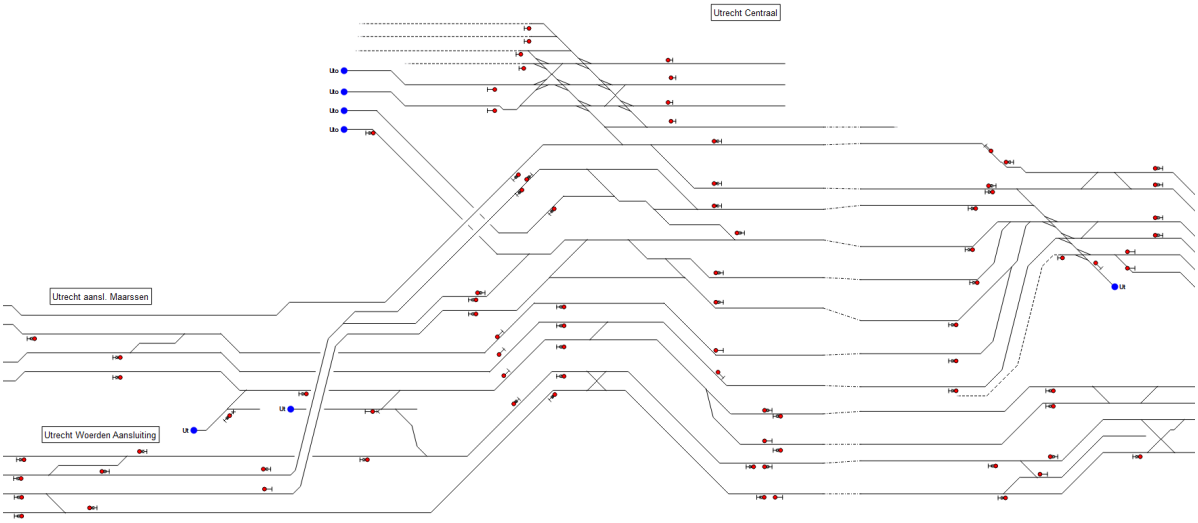


Figure F.9: Utrecht CS → Utrecht Vaartsche Rijn

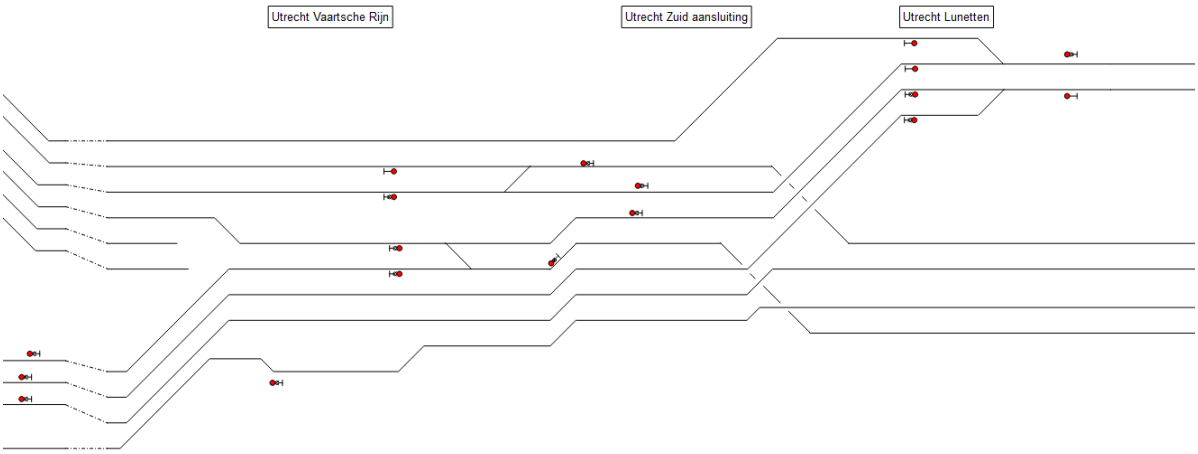


Figure F.10: Utrecht CS ← Utrecht Vaartsche Rijn → Utrecht Lunetten/Bunnik

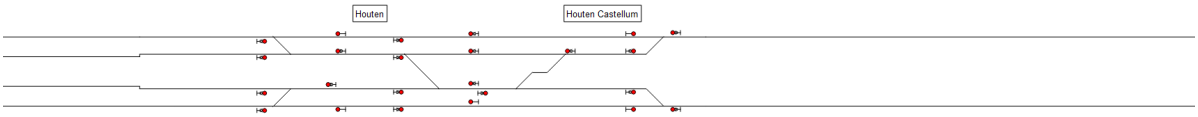


Figure F.11: Utrecht Lunetten ← Houten → Culemborg

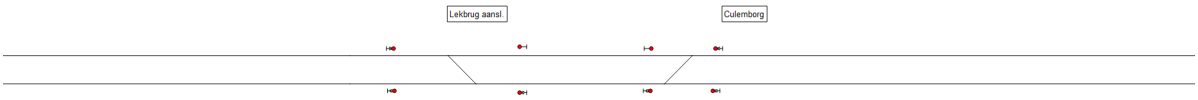


Figure F.12: Houten ← Culemborg → Geldermalsen

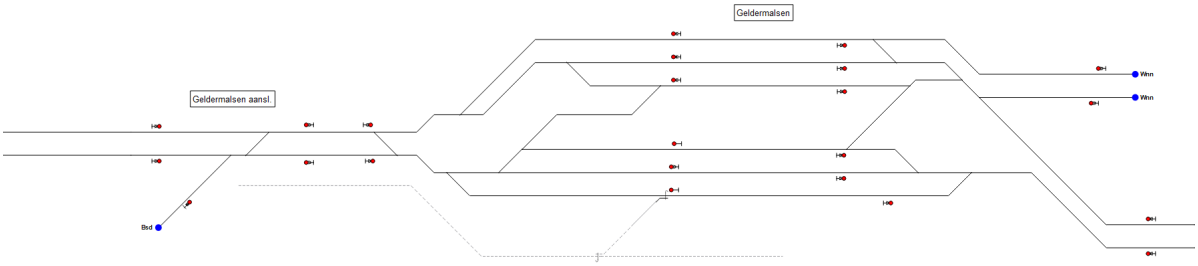


Figure F.13: Culemborg ← Geldermalsen → Meteren Betuweaansluiting

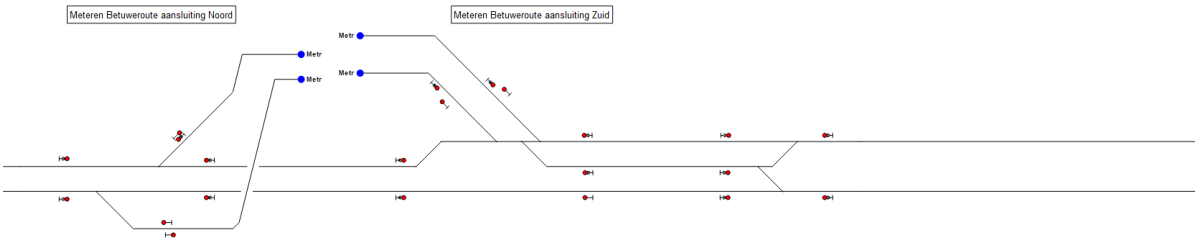


Figure F.14: Geldermalsen ← Meteren betuweaansluiting →Zaltbommel

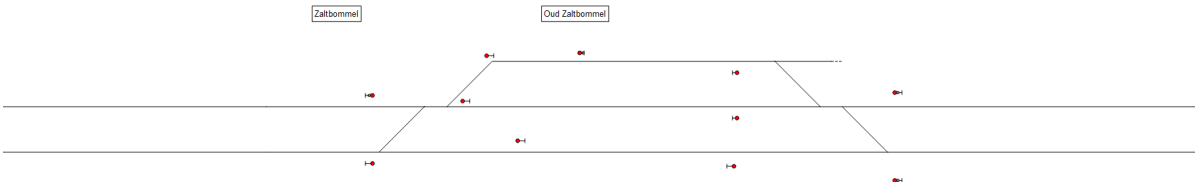


Figure F.15: Meteren betuweaansluiting ← Zaltbommel → Hedel

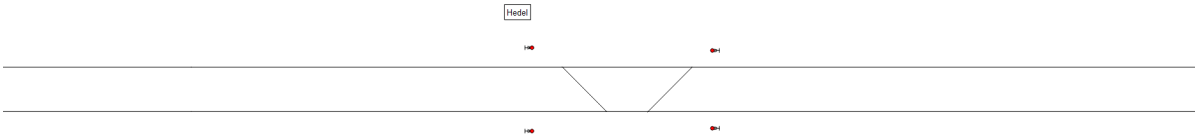


Figure F.16: Zaltbommel ← Hedel → 's-Hertogenbosch

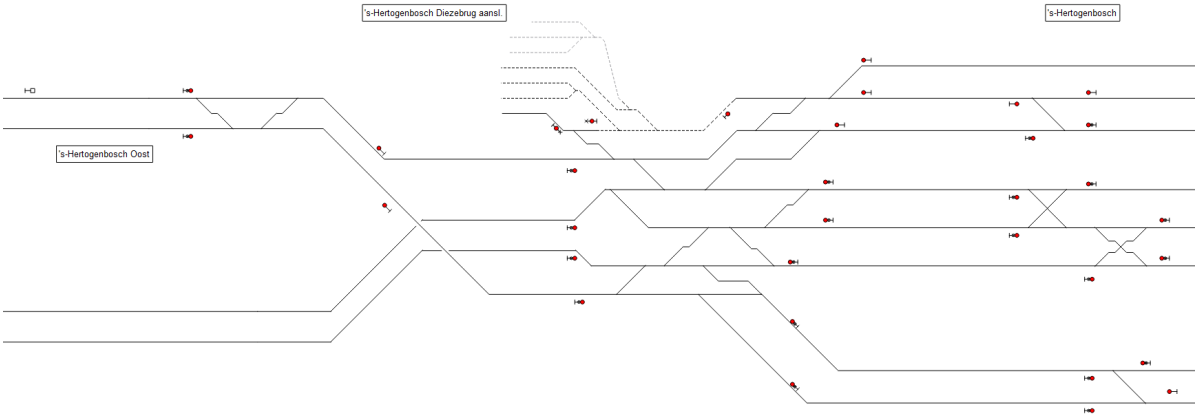


Figure F.17: Rosmalen →'s-Hertogenbosch Oost/Hedel ↔'s HertogenBosch

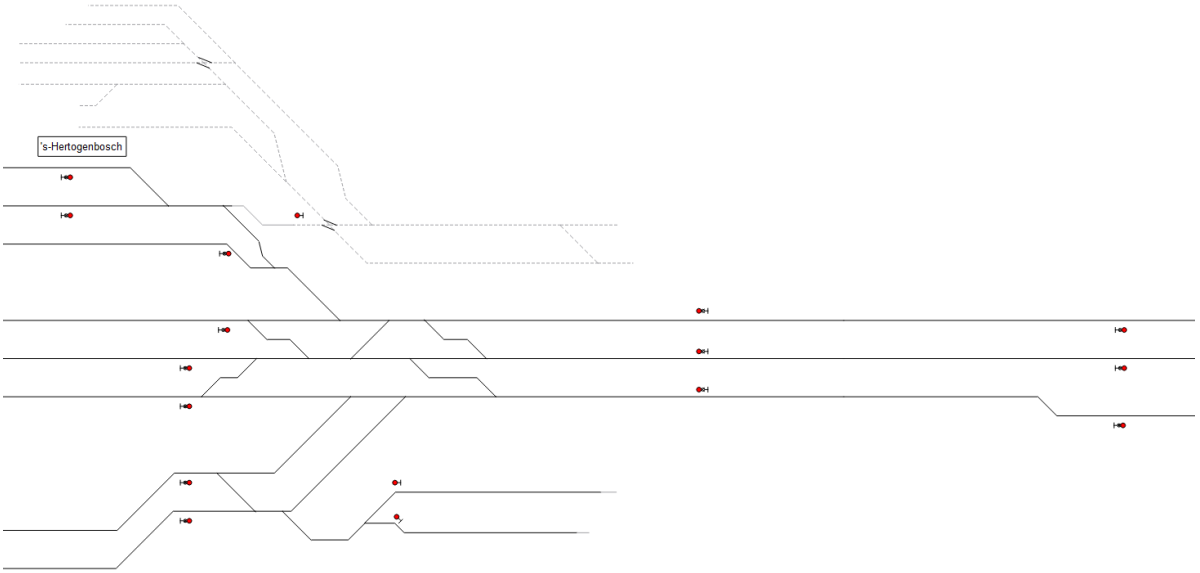


Figure F.18: 's-Hertogenbosch → Vught

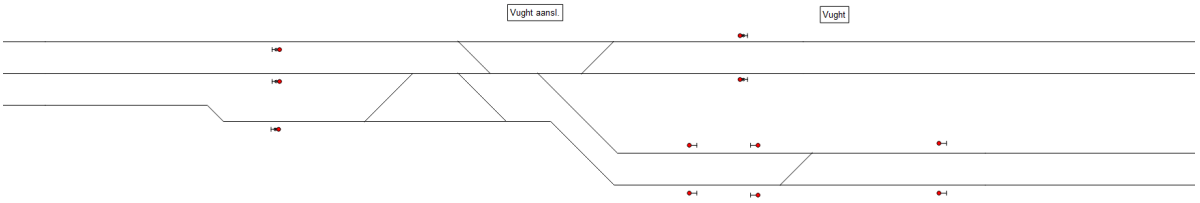


Figure F.19: 's-Hertogenbosch ← Vught → Tilbrug/Boxtel

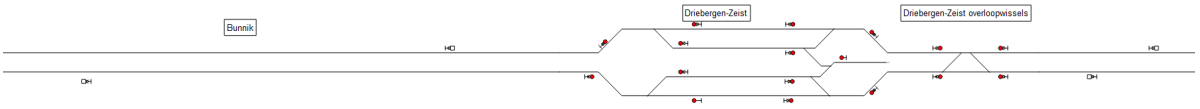


Figure F.20: Utrecht Vaartsche Rijn ← Bunnik ↔ Driebergen Zeist → Maarn

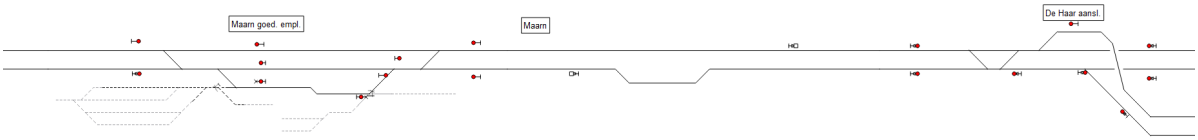


Figure F.21: Driebergen Zeist ← Maarn → Veenendaal

Figure F.22: Maarn ← Veenendaal West ↔ Rhenen

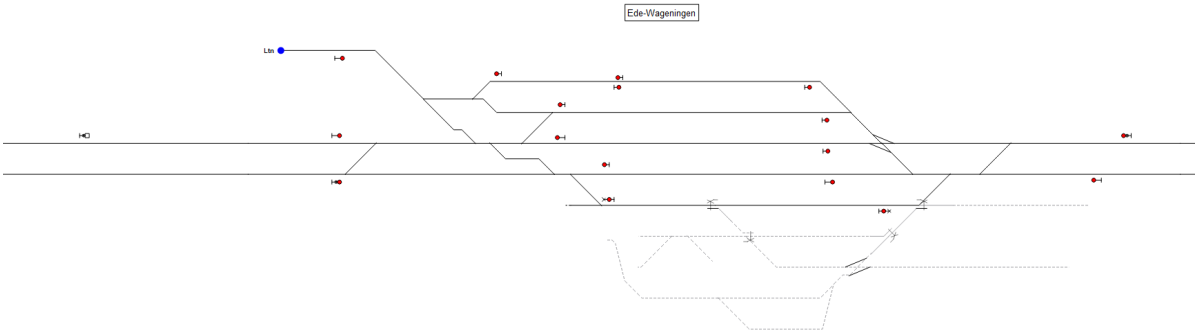


Figure F.23: Veenendaal De Klomp ← Ede-Wageningen → Wolfheze

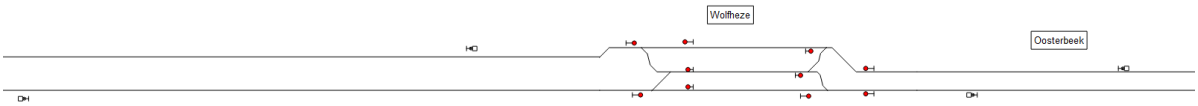


Figure F.24: Ede-Wageningen ← Wolfheze → Oosterbeek

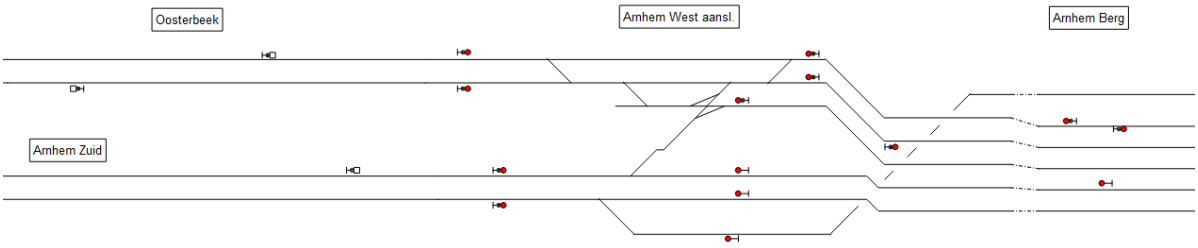


Figure F.25: Wolfheze/Elst ← Arnhem West aansluiting → Arnhem Centraal

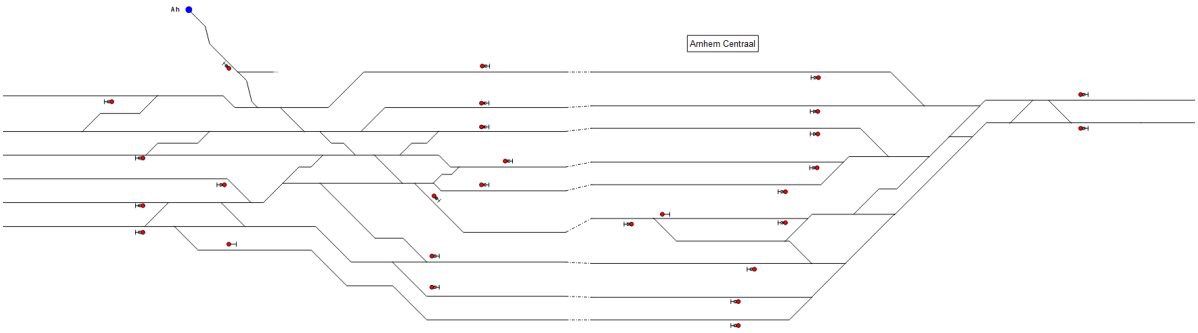


Figure F.26: Arnhem West aansluiting ← Arnhem centraal → IJsselbrug

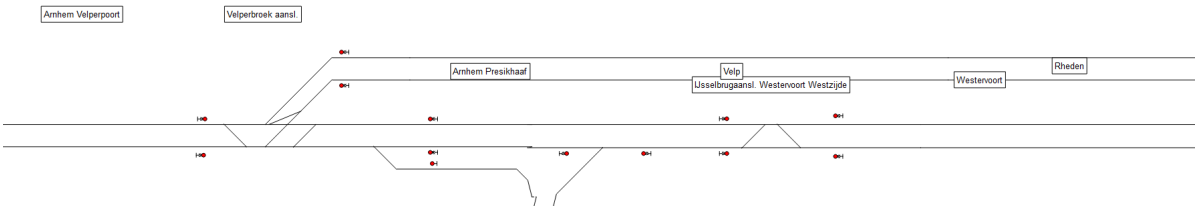


Figure F.27: Arnhem Centraal ← IJsselbrug →Zevenaar

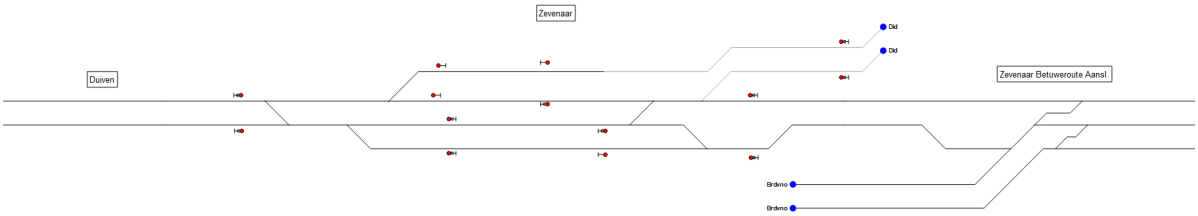


Figure F.28: IJsselbrug ← Zevenaar → Zevenaar Oost

Figure F.29: Zevenaar ← Zevenaar Oost ↔ Elten → Emmerik

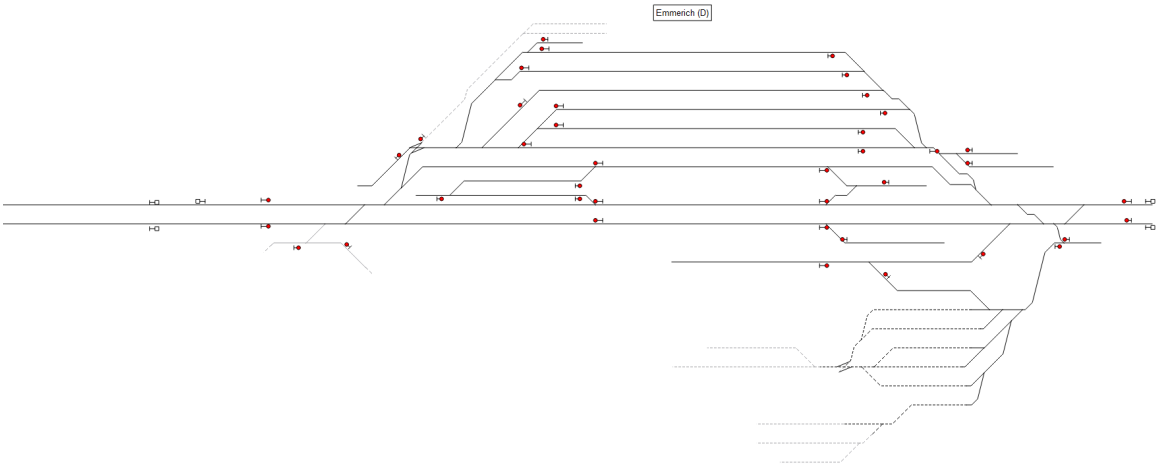


Figure F.30: Elten ← Emmerich (→ Germany)

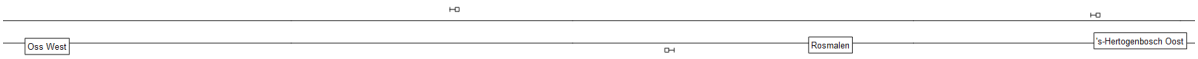


Figure F.31: Oss ←Oss West ↔ Rosmalen ↔ 's-Hertogenbosch Oost → 's Hertogenbosch

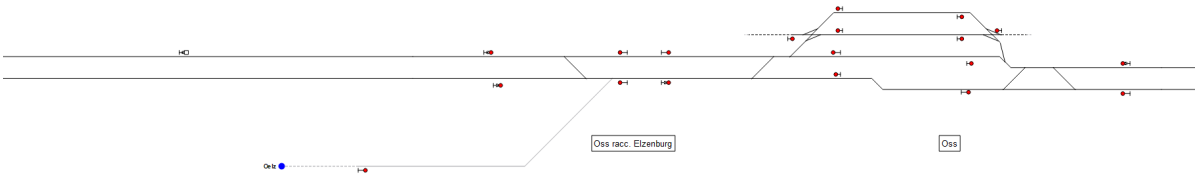


Figure F.32: Ravenstein ← Oss → Oss West

Figure F.33: Nijmegen Dukenburg ← Wijchen ↔ Ravenstein →Oss

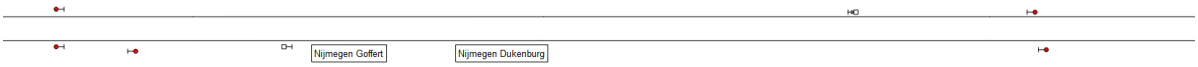


Figure F.34: Nijmegen ← Nijmegen Goffert ↔ Nijmegen Dukenburg → Wijchen

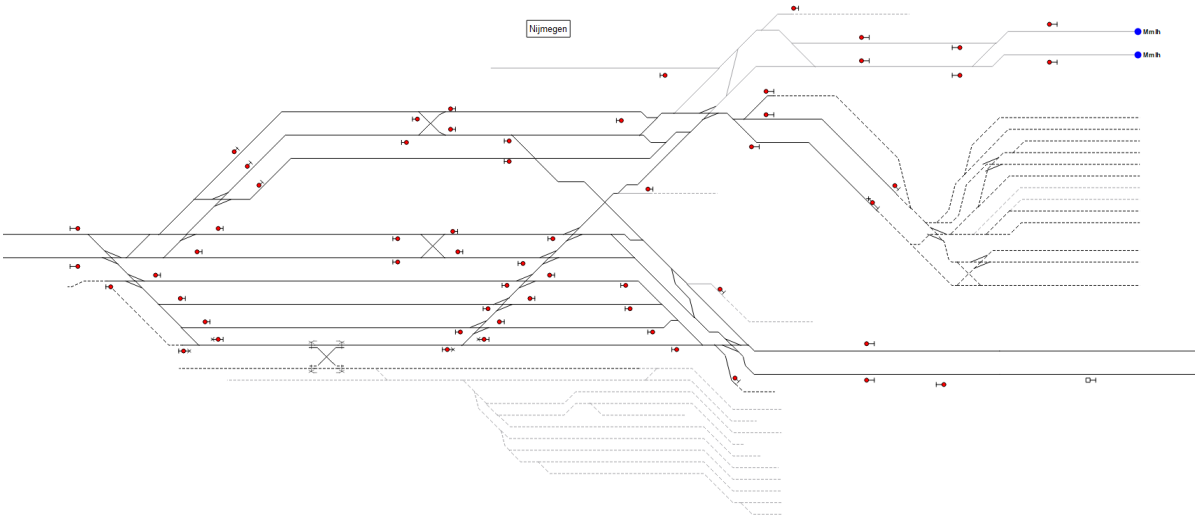


Figure F.35: Nijmegen Lent ← Nijmegen → Nijmegen Dukenburg

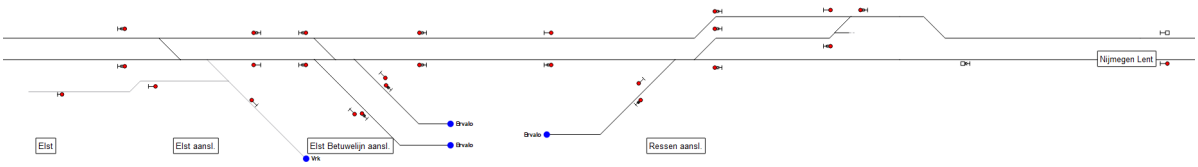


Figure F.36: Arnhem Zuid ← Elst ↔ Nijmegen Lent → Nijmegen

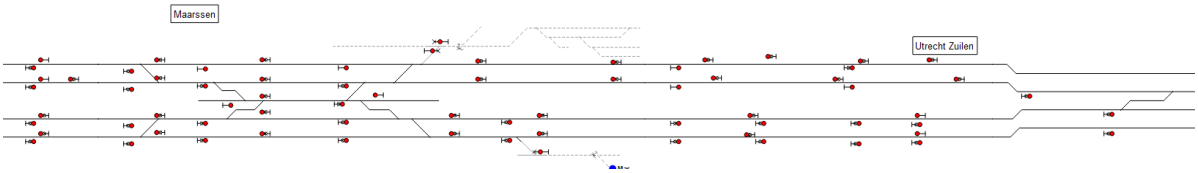


Figure F.37: Breukelen ← Maarsen ↔ Utrecht Zuilen → Utrecht CS

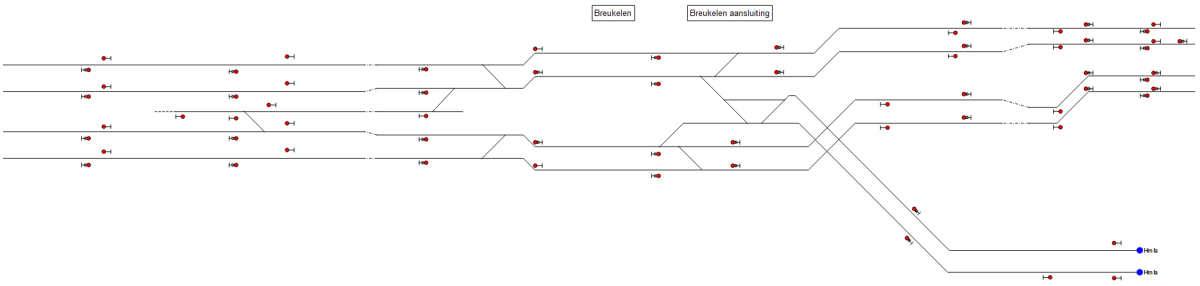


Figure F.38: Abcoude ← Breukelen → Maarsen

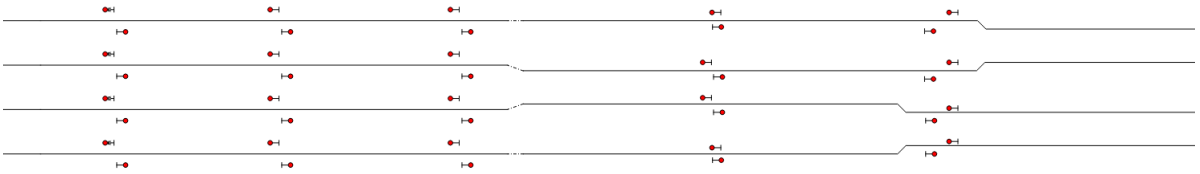


Figure F.39: Abcoude ↔ Breukelen

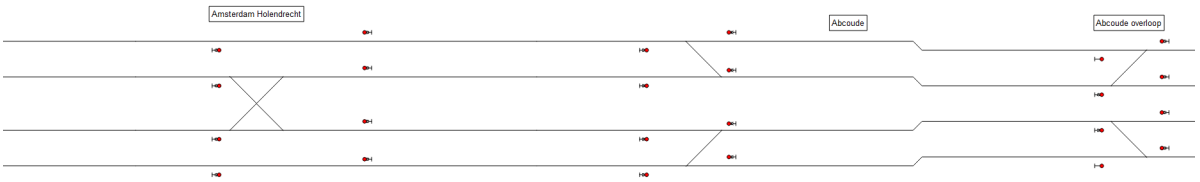


Figure F.40: Amsterdam Bijlmer ArenA ← Abcoude → Breukelen

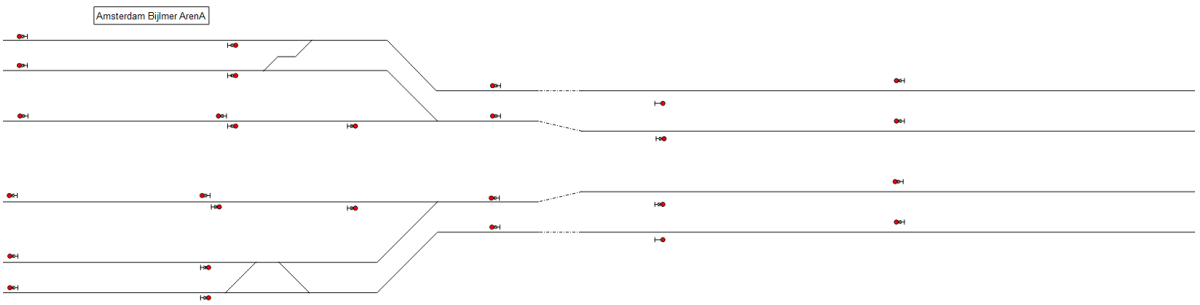


Figure F.41: Duivendrecht ← Amsterdam Bijlmer ArenA → Abcoude

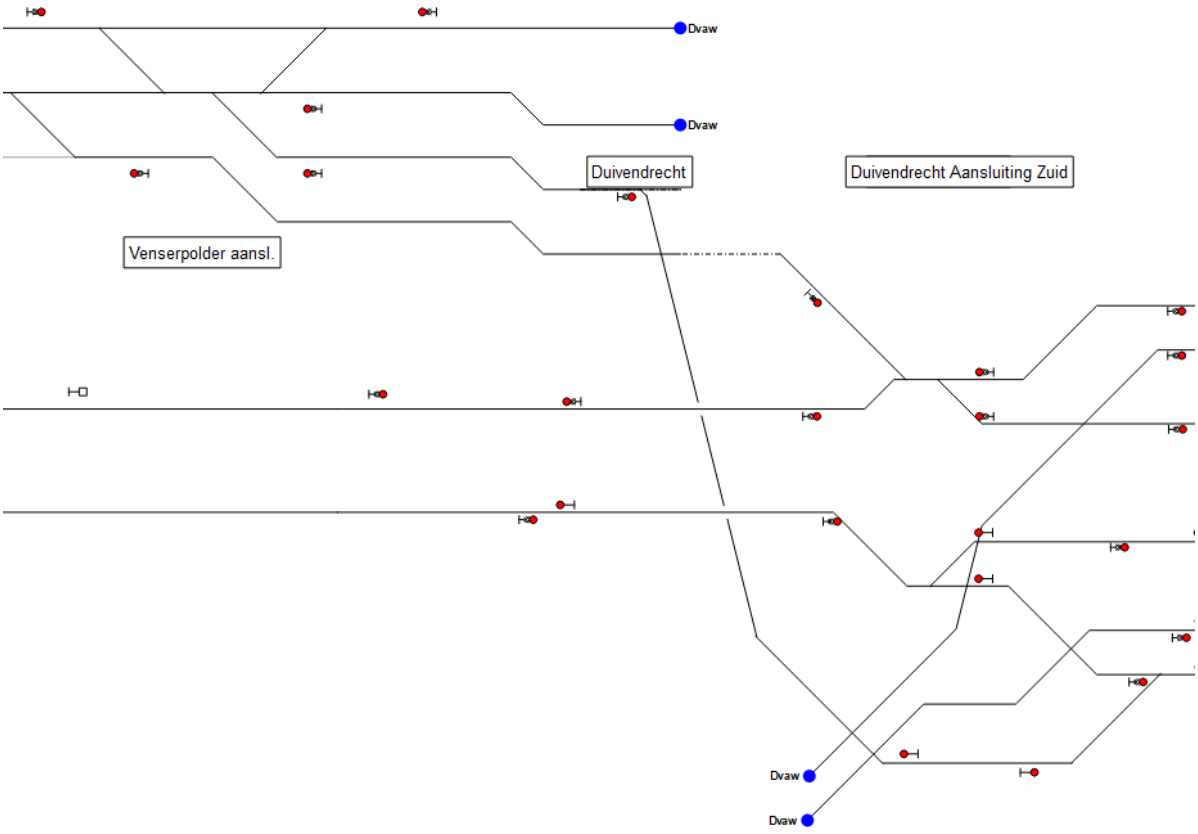


Figure F.42: Amsterdam RAI/Amsterdam Amstel/Diemen Zuid ← Duivendrecht → Amsterdam Bijlmer ArenA

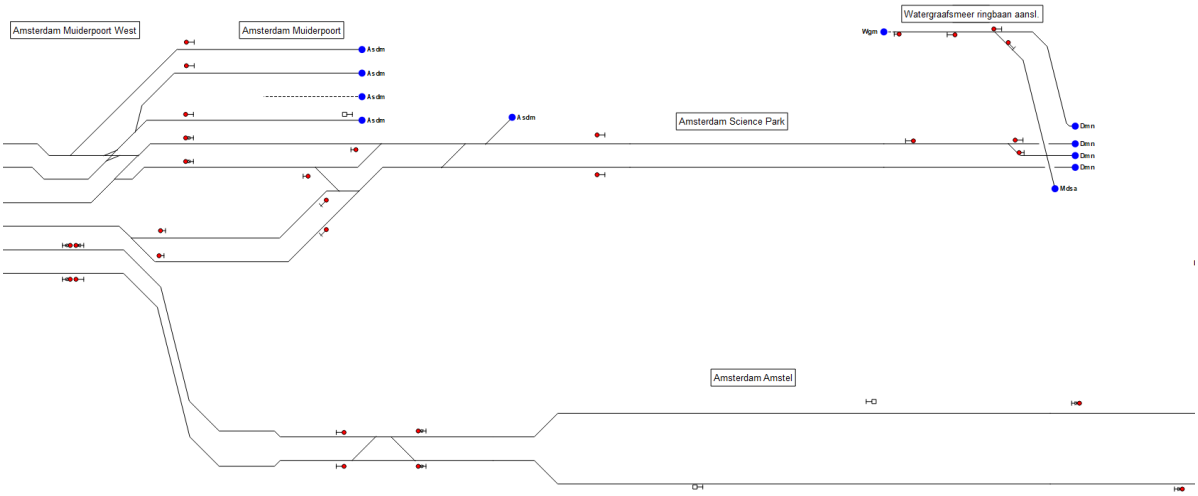


Figure F.43: Amsterdam Muiderpoort aansluiting ← Amsterdam Muiderpoort West ↔ Amsterdam Amstel → Duivendrecht/Diemen

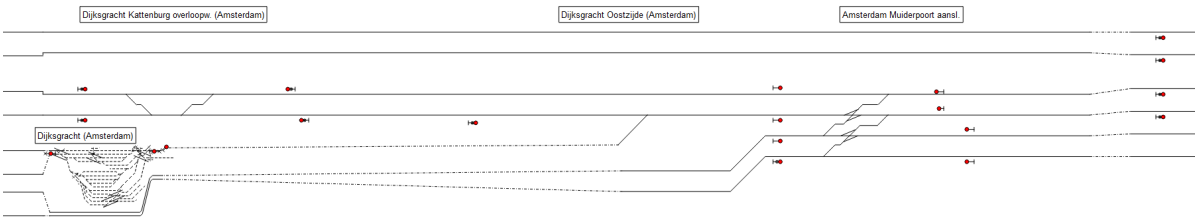


Figure F.44: Amsterdam Centraal ← Amsterdam Muiderpoort aansluiting → Amsterdam Muiderpoort West

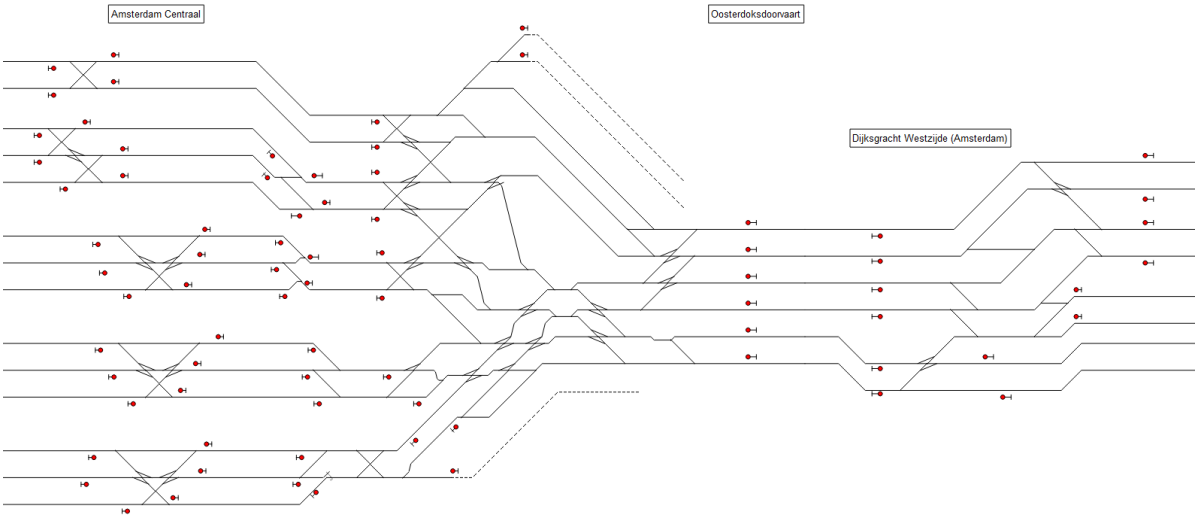


Figure F.45: Singelgracht ← Amsterdam CS → Amsterdam Muiderpoort aansluiting

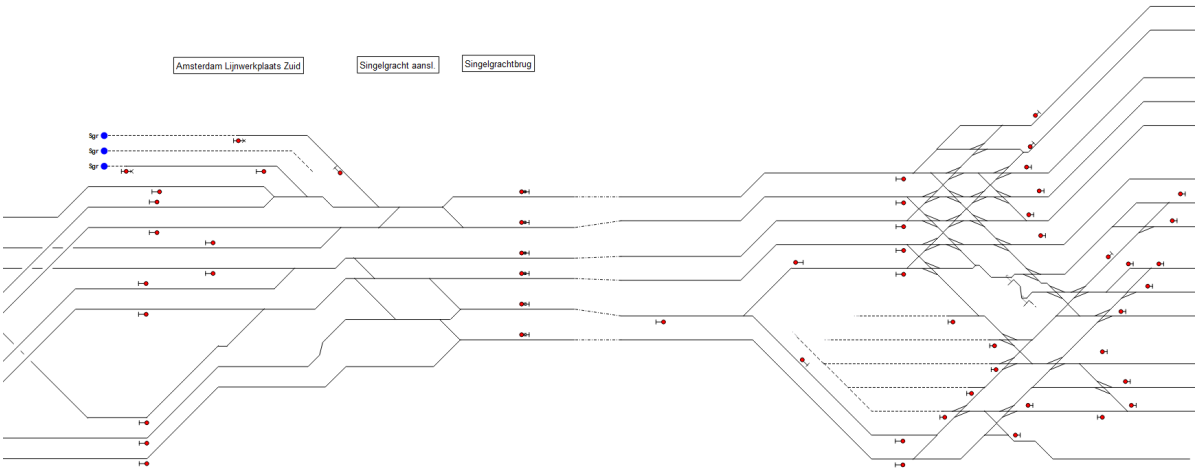


Figure F.46: Lijnwerkplaats Amsterdam ← Singelgracht → Amsterdam CS

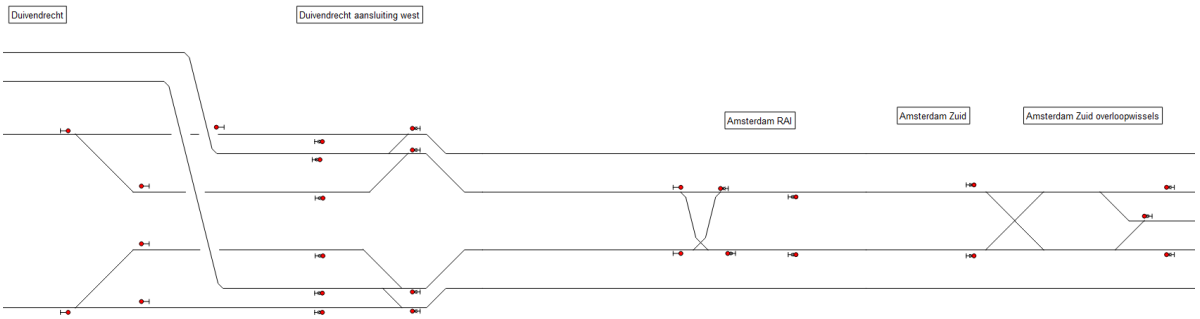


Figure F.47: Duivendrecht ← Amsterdam RAI ↔ Amsterdam Zuid → Amsterdam Riekpolder aansl.

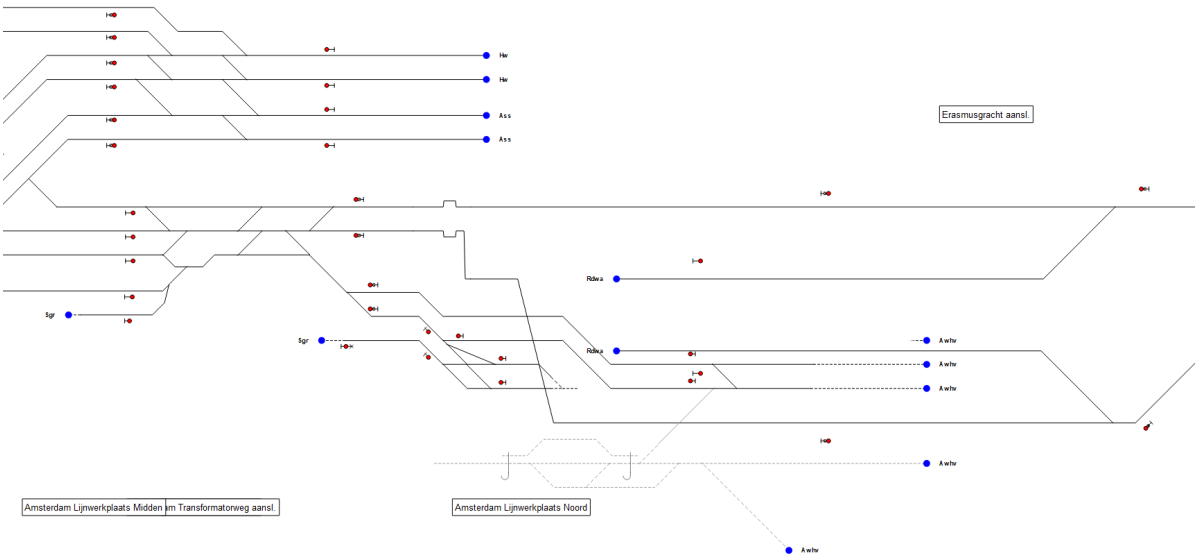


Figure F.48: Singelgracht ← Lijnwerkplaats Amsterdam → Amsterdam Lelylaan

Figure F.49: Lijnwerkplaats Amsterdam ← Amsterdam Lelylaan → Amsterdam Riekpolder aansl.

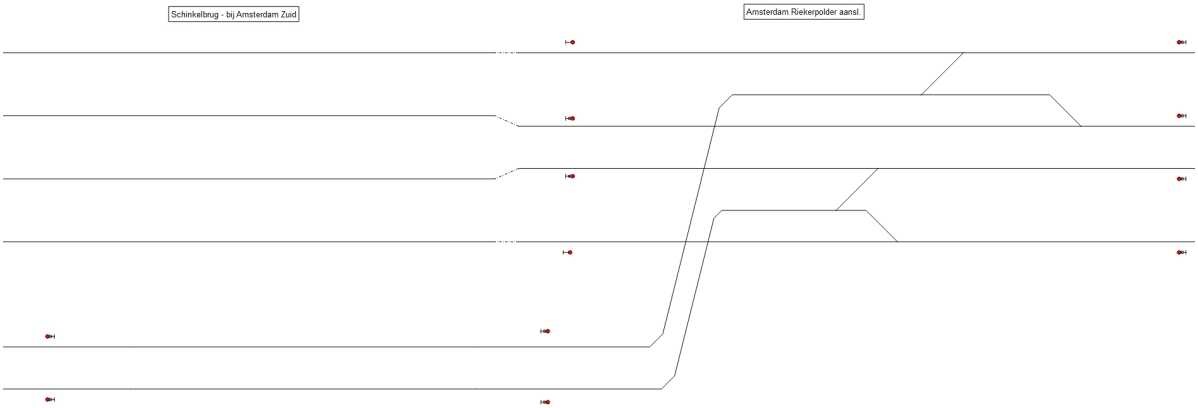


Figure F.50: Amsterdam Zuid/Lelylaan ← Amsterdam Riekpolder aansl. → Schiphol

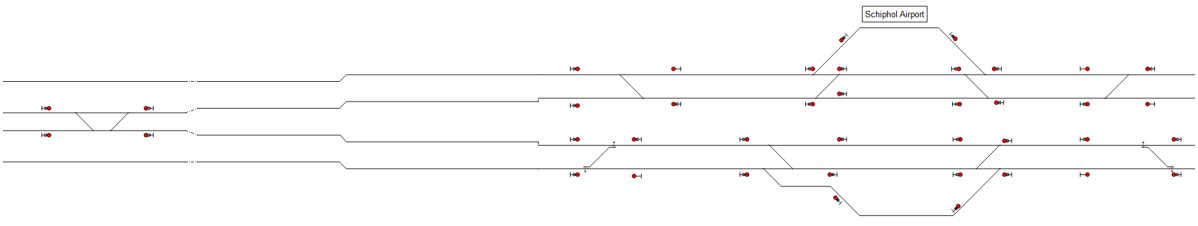


Figure F.51: Amsterdam Riekpolder aansl. ← Schiphol → Hoofddorp

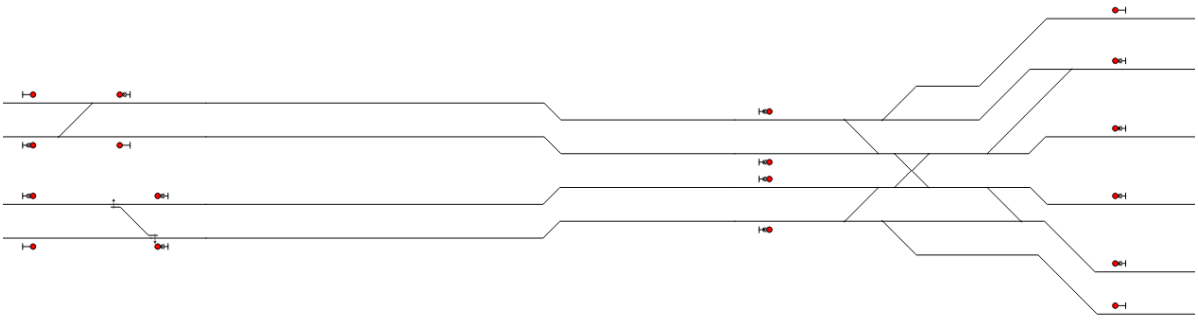


Figure F.52: Schiphol ↔ Hoofddorp

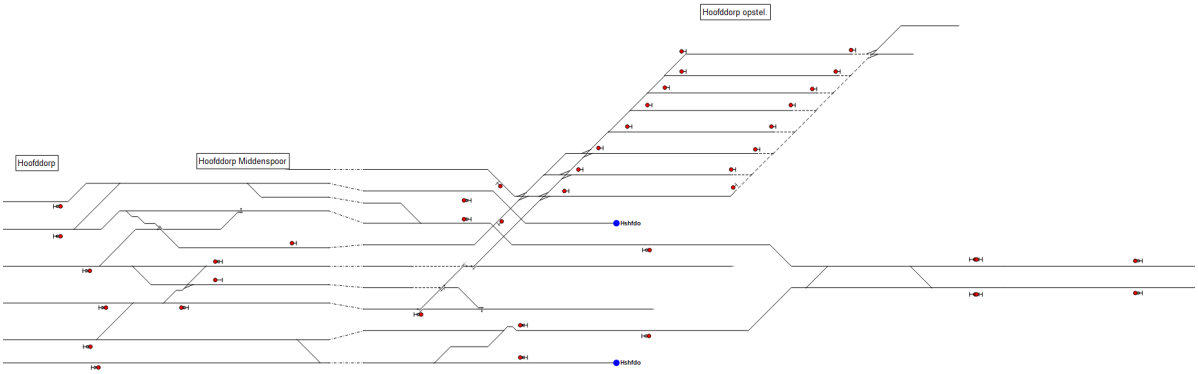


Figure F.53: Schiphol ← Hoofddorp →Nieuw-Vennep

Figure F.54: Hoofddorp ← Nieuw vennep ↔ Sassenheim →Voorhout

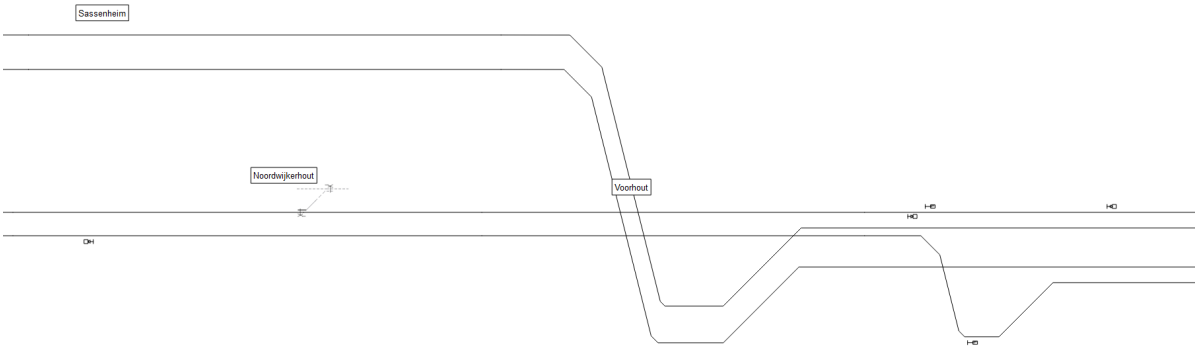


Figure F.55: Nieuw Vennep ←Sassenheim ↔ Voorhout → Leiden

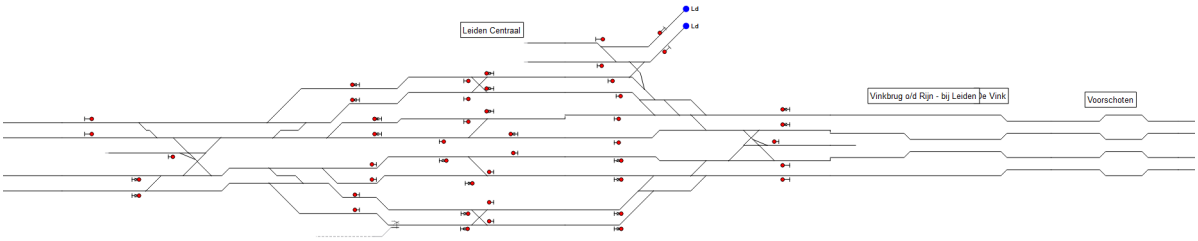


Figure F.56: Voorhout ← Leiden → Den Haag Laan van NOI

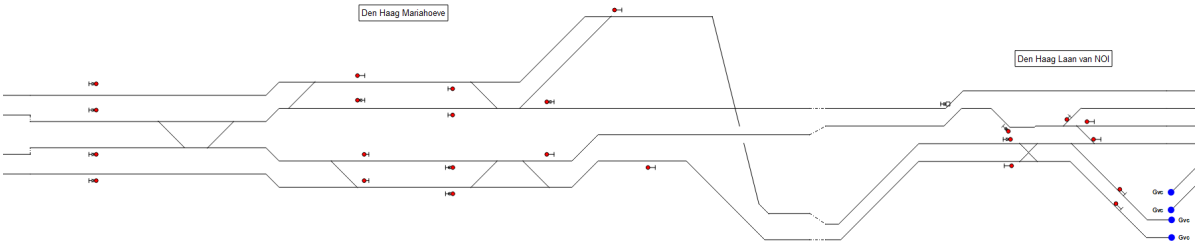


Figure F.57: Leiden ← Den Haag Laan van NOI → Den Haas HS

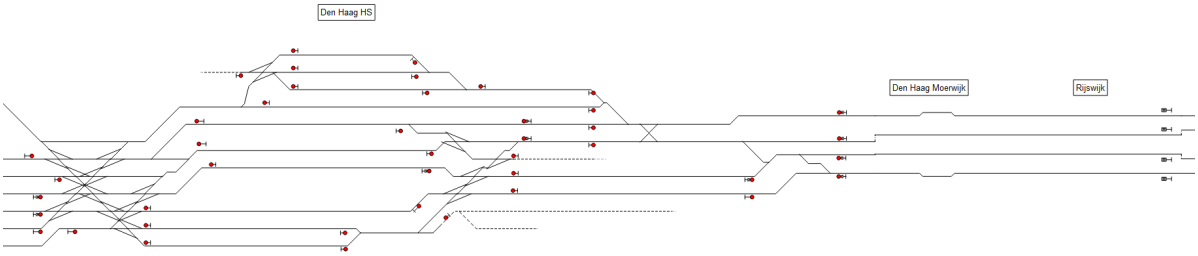


Figure F.58: Den Haag Laan van NOI ← Den Haag HS ↔ Den Haag Moerwijk ↔ Rijswijk → Delft

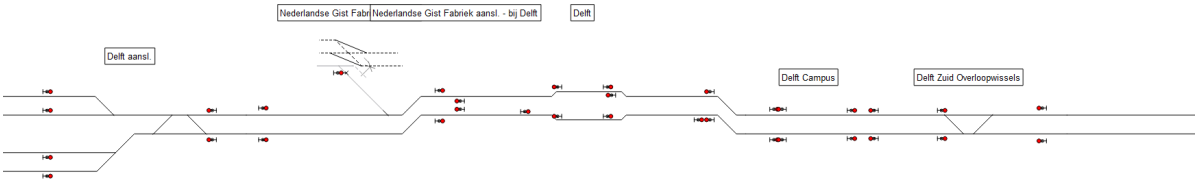


Figure F.59: Rijswijk ← Delft ↔ Delft Campus → Schiedam Centrum

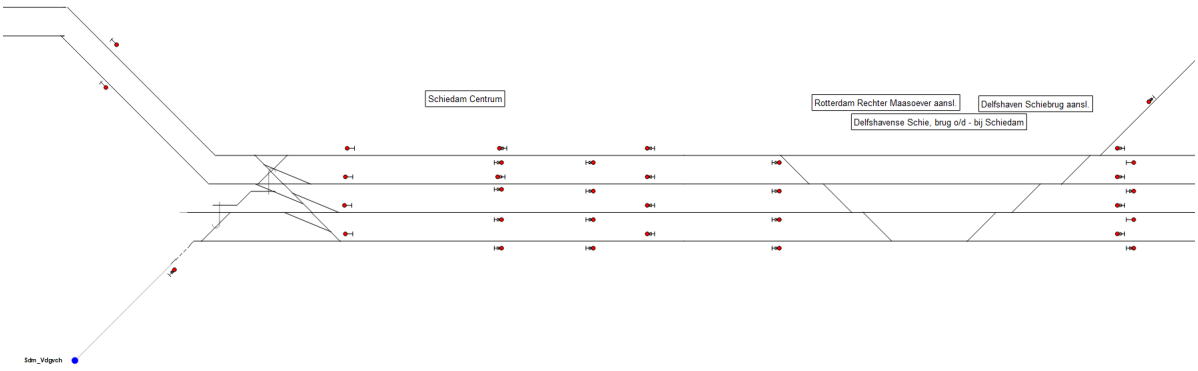


Figure F.60: Delft Campus ← Schiedam Centrum → Rotterdam Centraal

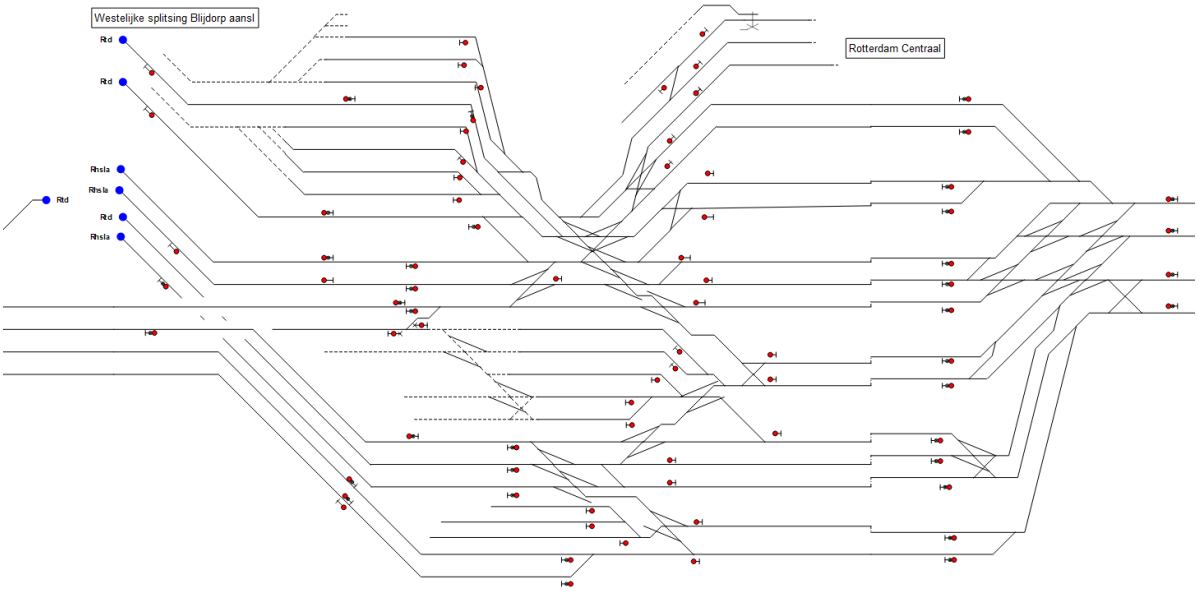


Figure F.61: Schiedam Centrum ← Rotterdam Centraal

G

Case study: trajectories

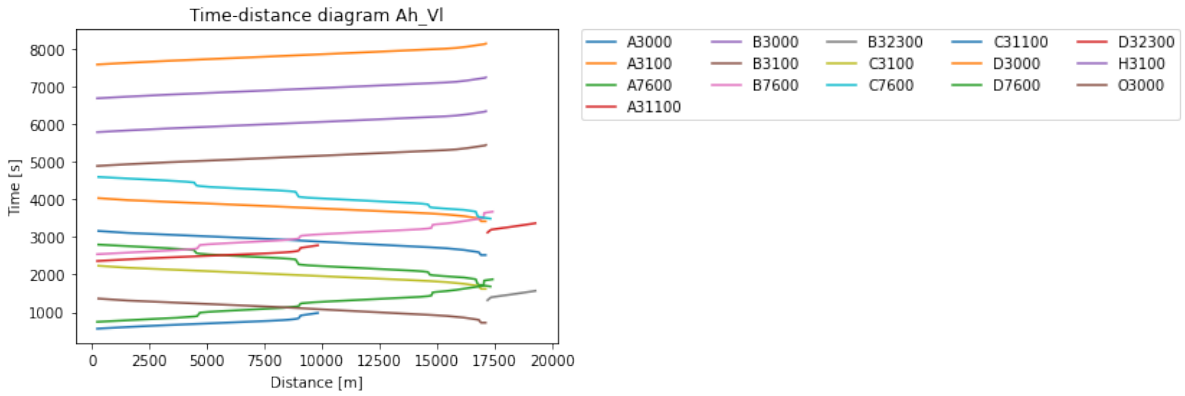


Figure G.1: Arnhem Centraal (Ah) - Venlo (Vl) Trajectory

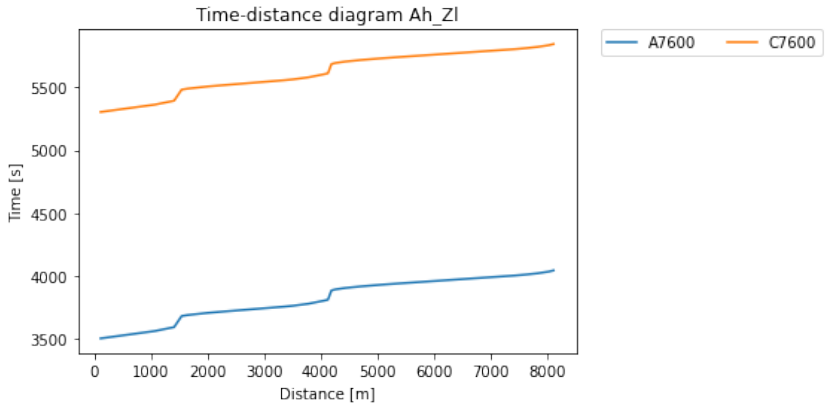


Figure G.2: Arnhem Centraal (Ah) - Zwolle (Zl) Trajectory

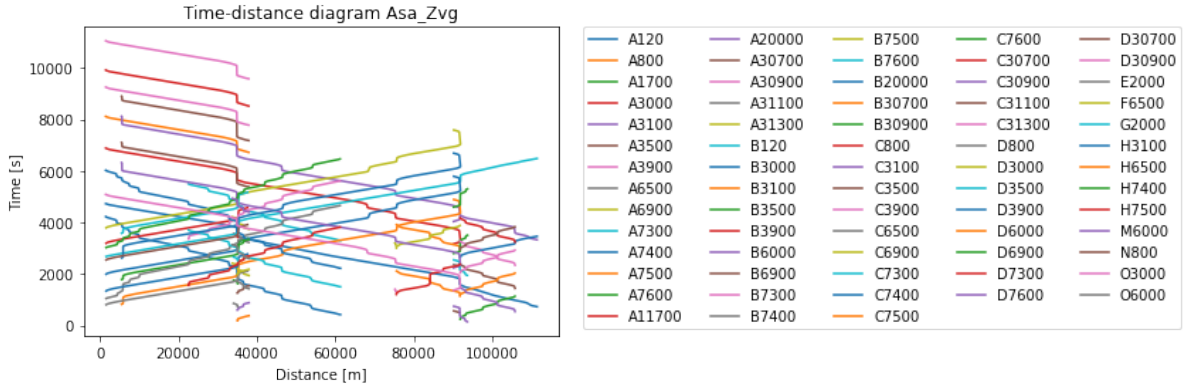


Figure G.3: Amsterdam Amstel (Asa) - Zevenaar grens (Zvg) Trajectory

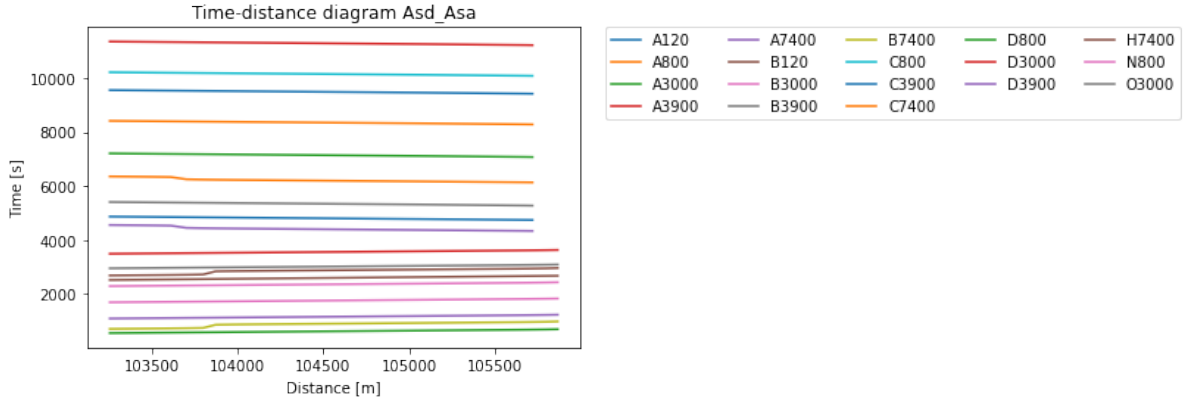


Figure G.4: Amsterdam Centraal (Asd) - Amsterdam Amstel (Asa) Trajectory

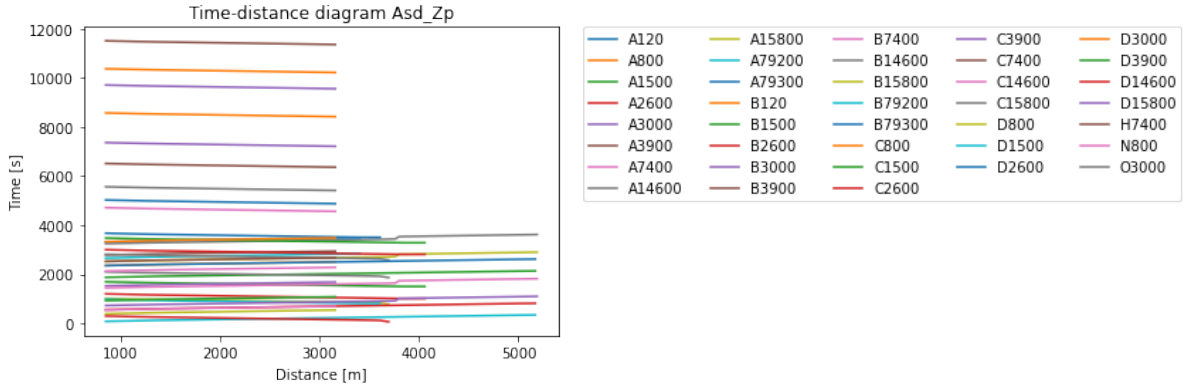


Figure G.5: Amsterdam Centraal (Asd) - Zutphen (Zp) Trajectory

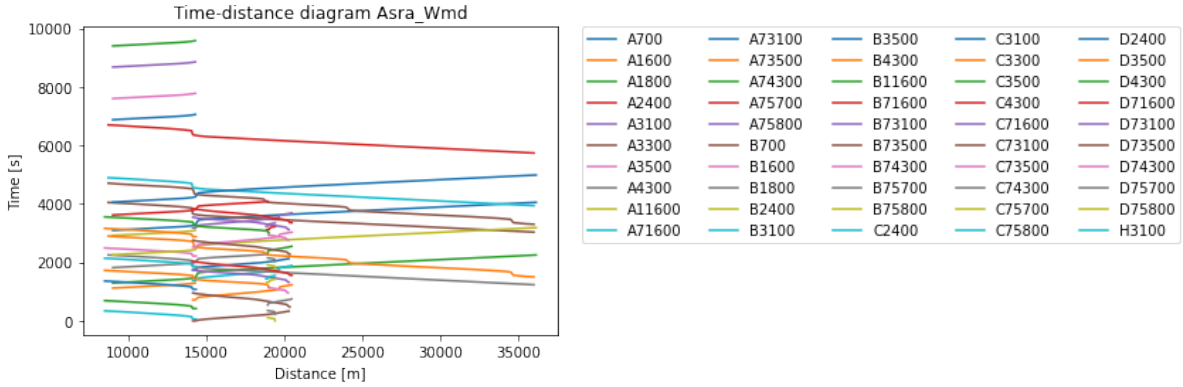


Figure G.6: Amsterdam Riekerpolder aansl. (Asra) - Wmd Trajectory

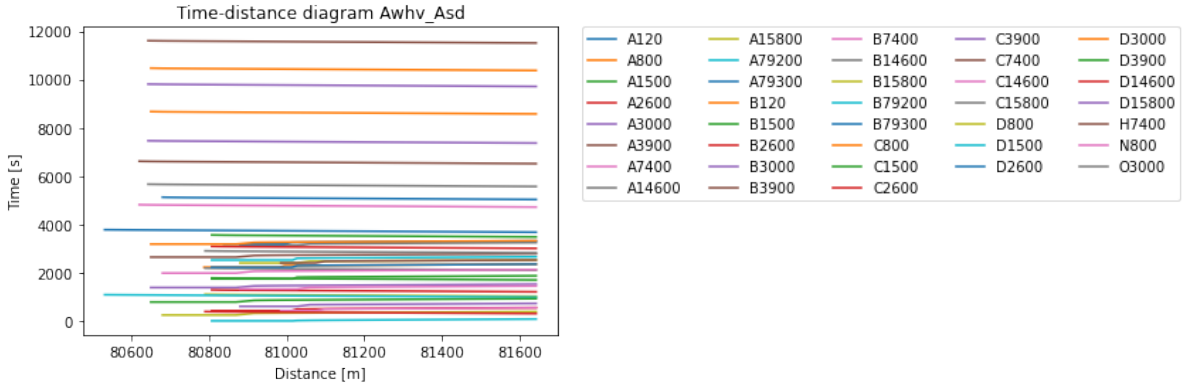


Figure G.7: Amsterdam Westhaven (Awhv) - Amsterdam Centraal (Asd) Trajectory

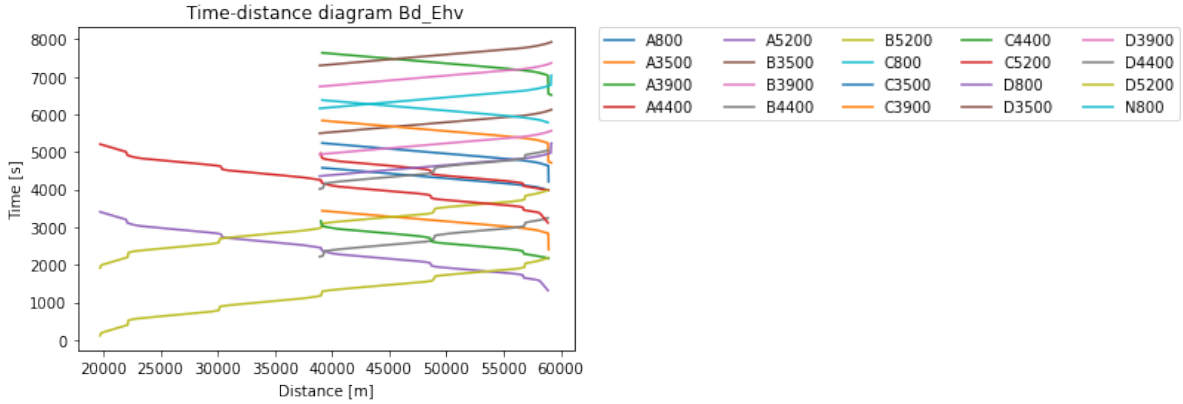


Figure G.8: Breda (Bd) - Eindhoven (Ehv) Trajectory

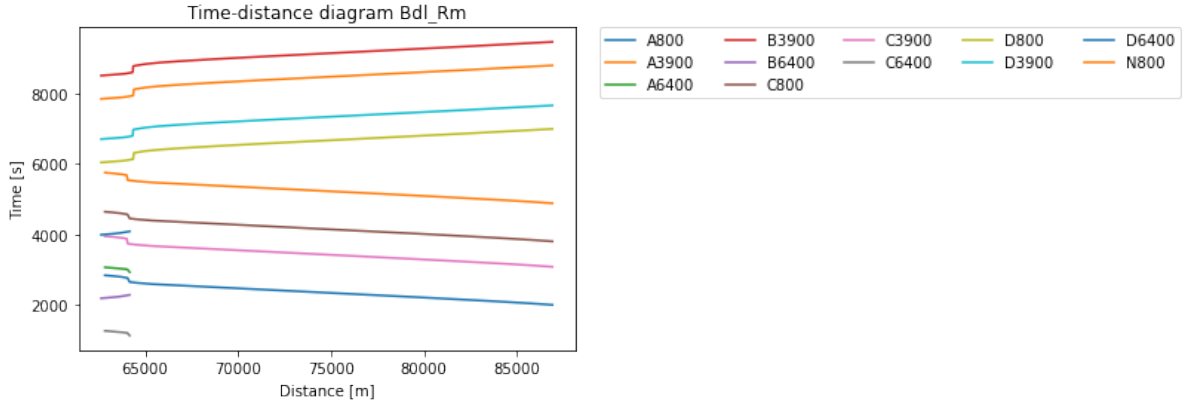


Figure G.9: Budel (Bdl) - Roermond (Rm) Trajectory

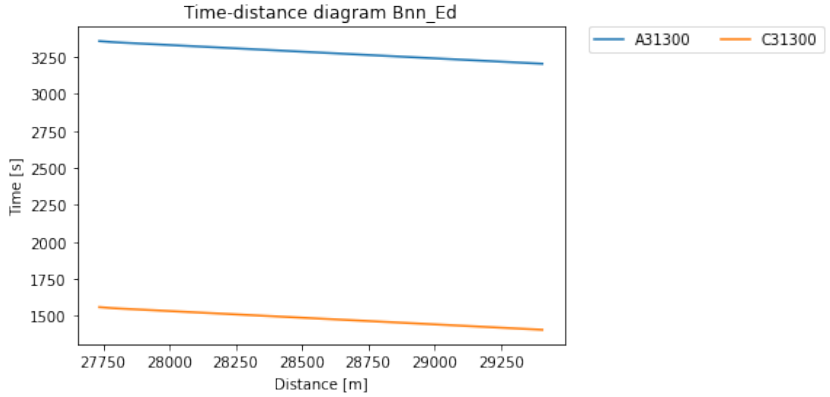


Figure G.10: Barneveld Noord (Bnn) - Ede Wageningen (Ed) Trajectory

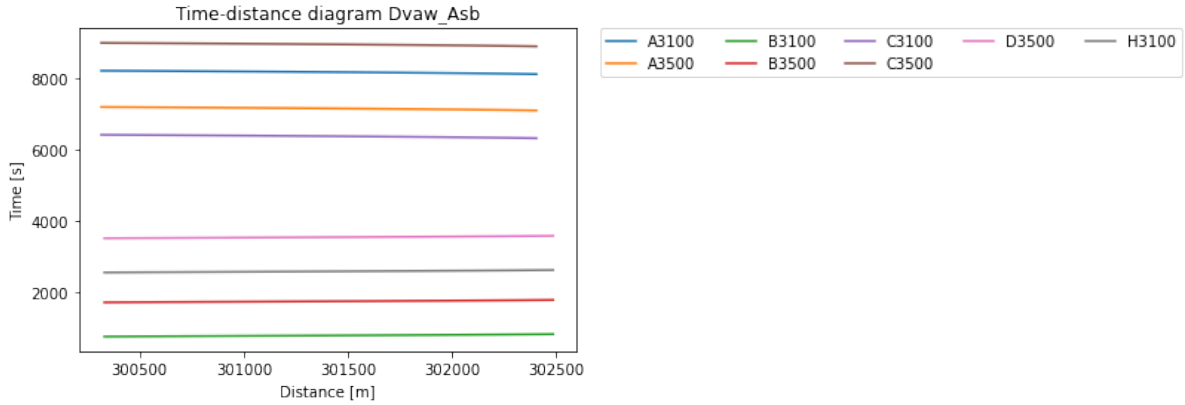


Figure G.11: Duivendrecht aansluiting west (Dvaw) - Amsterdam Bijlmer ArenA (Asb) Trajectory

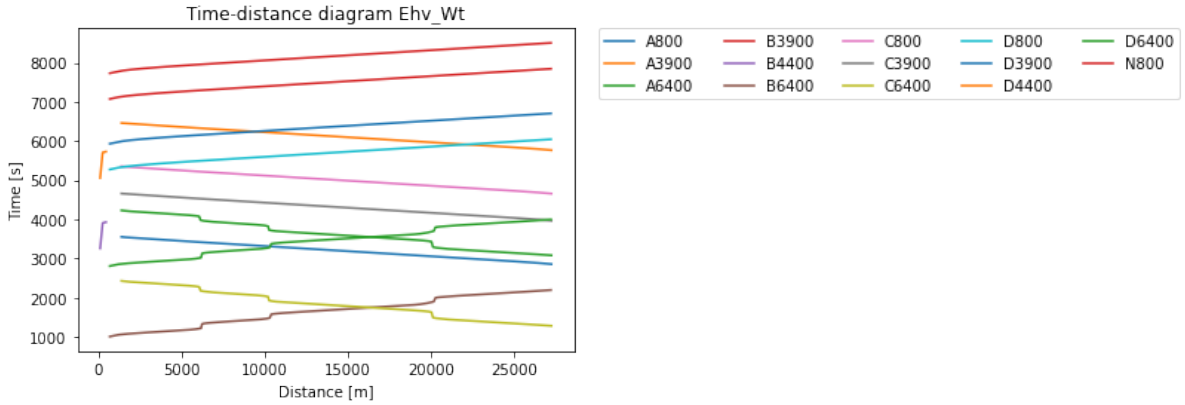


Figure G.12: Eindhoven (Ehv) - Weert (Wt) Trajectory

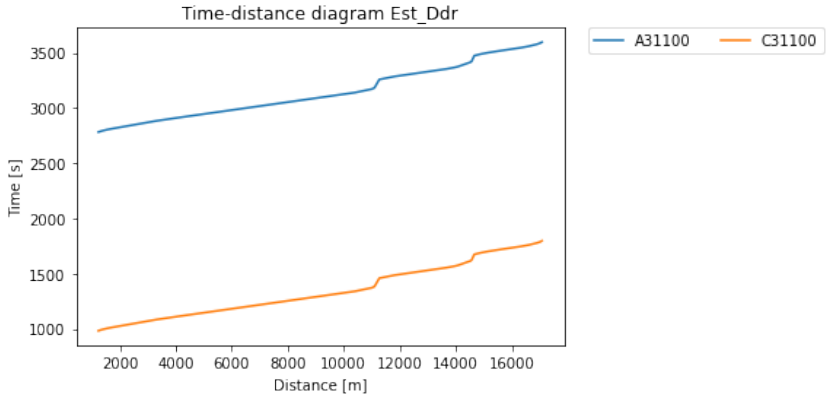


Figure G.13: Elst (Est) - Dordrecht (Ddr) Trajectory

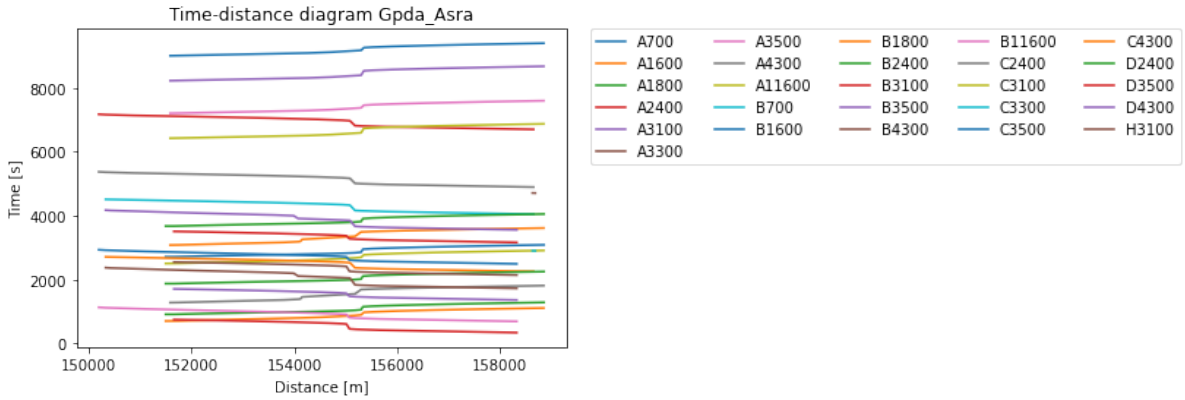


Figure G.14: Gaasperdammerweg aansl. (Gpda) - Amsterdam Riekpolder aansl. (Asra) Trajectory

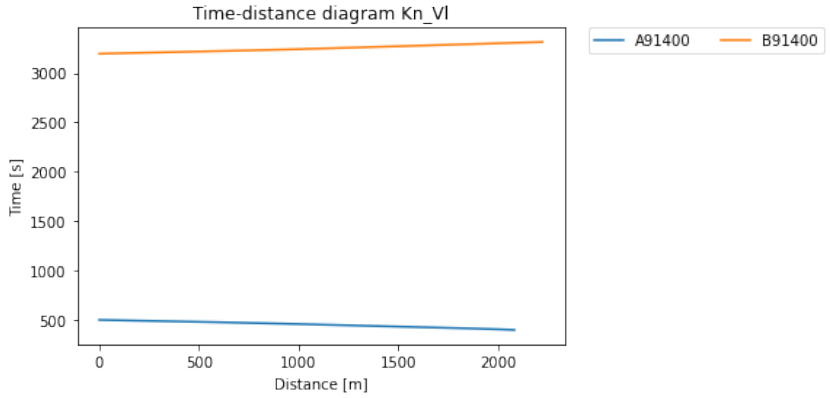


Figure G.15: Kaldenkirchen (Kn) - Venlo (Vl) Trajectory

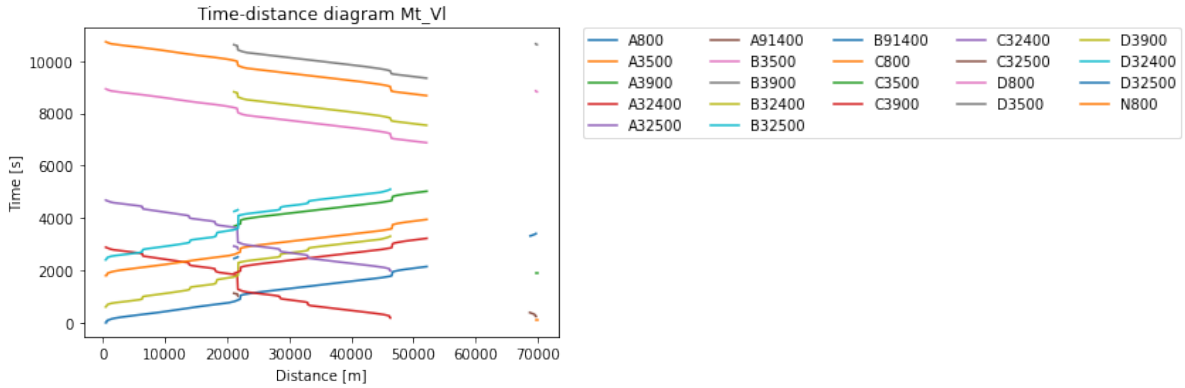


Figure G.16: Maastricht (Mt) - Venlo (Vl) Trajectory

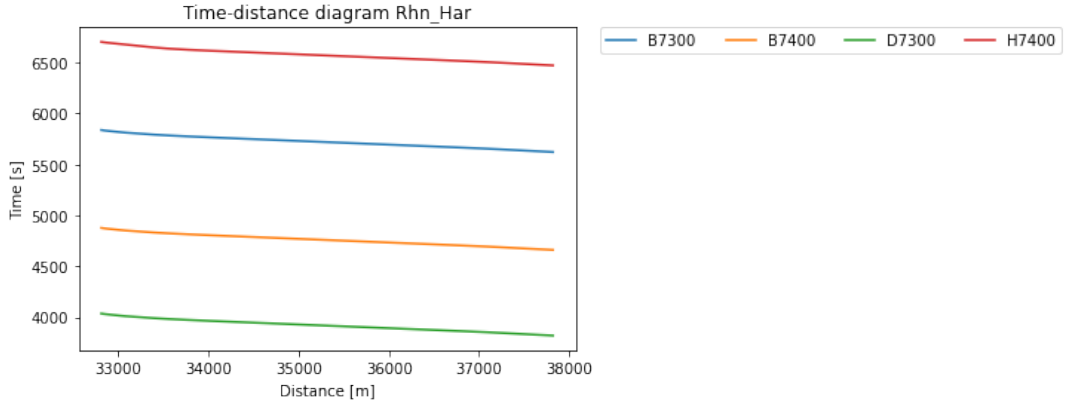


Figure G.17: Rhenen (Rhn) - De Haar aansl. (Har) Trajectory

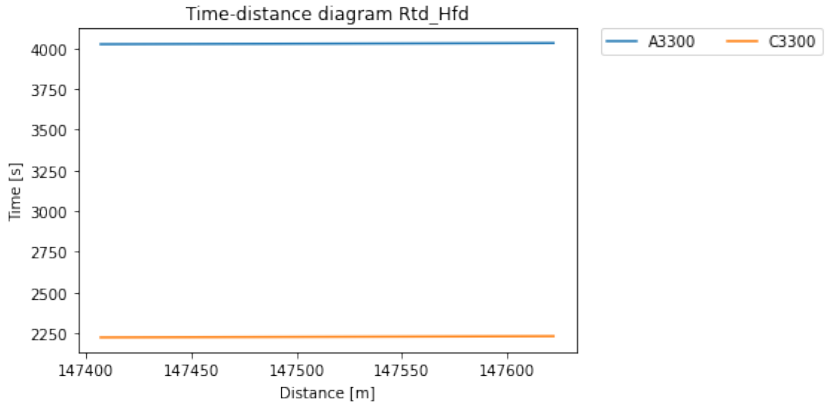


Figure G.18: Rotterdam (Rtd) - Hoofddorp (Hfd) Trajectory

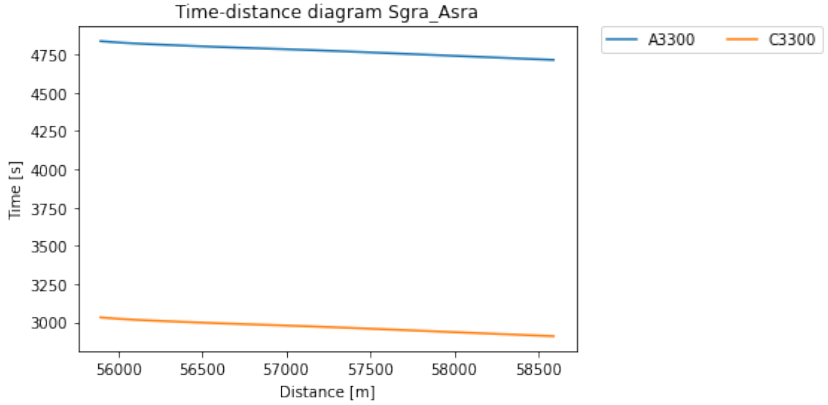


Figure G.19: Singelgracht aansl. (Sgra) - Amsterdam Riekpolder aansl. (Asra) Trajectory

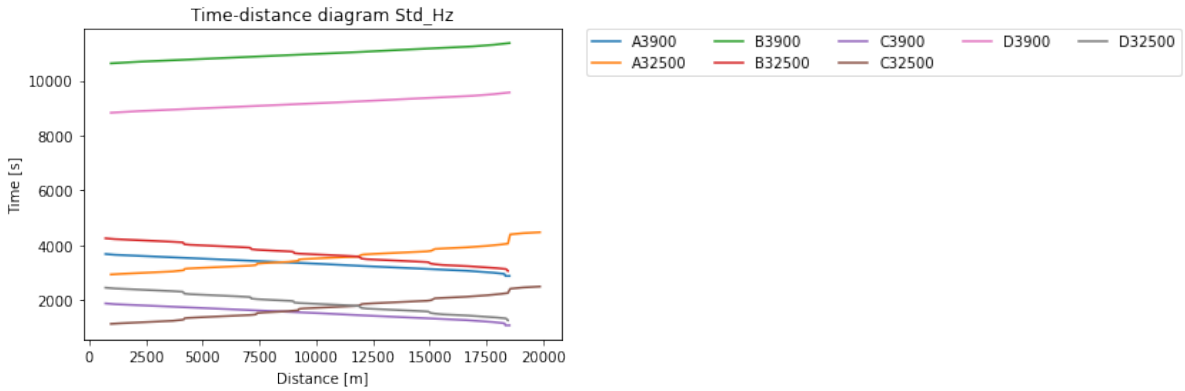


Figure G.20: Sittard (Std) - Herzogenrath (Hz) Trajectory

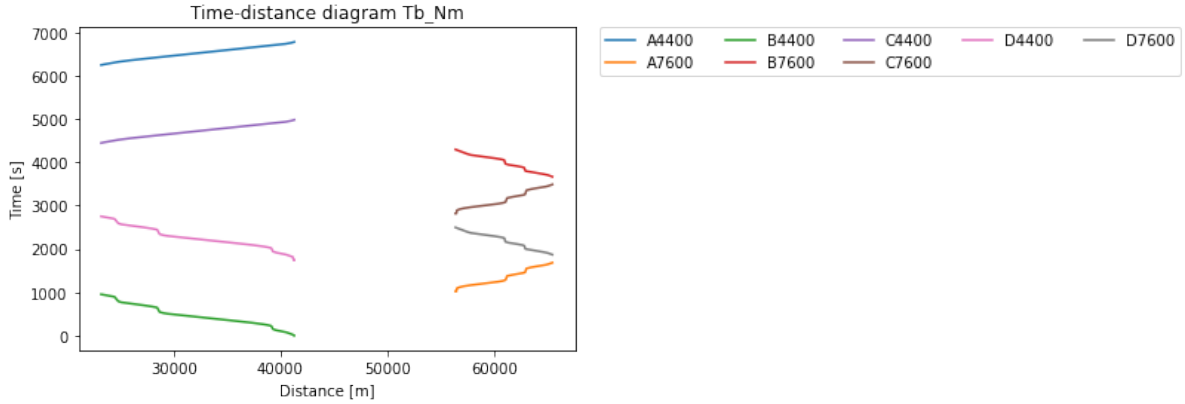


Figure G.21: Tilburg (Tb) - Nijmegen (Nm) Trajectory

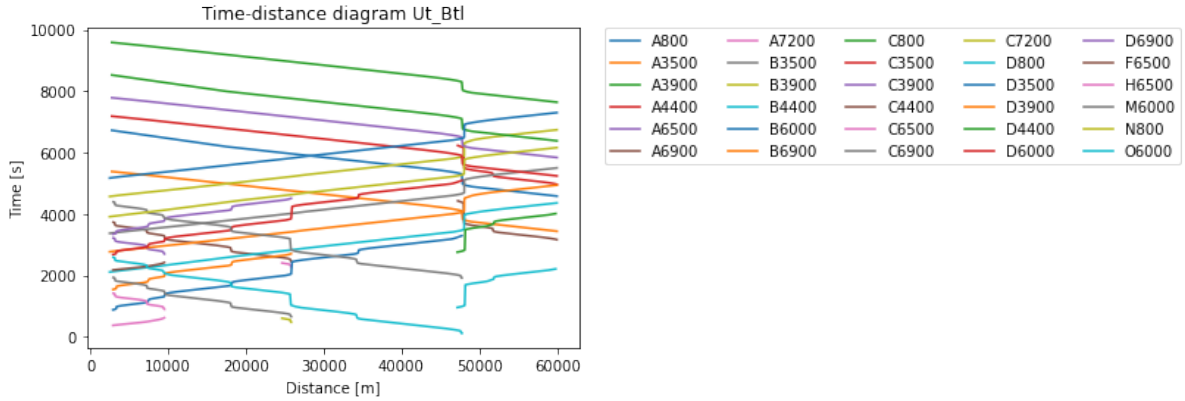


Figure G.22: Utrecht Centraal (Ut) - Boxtel (Btl) Trajectory

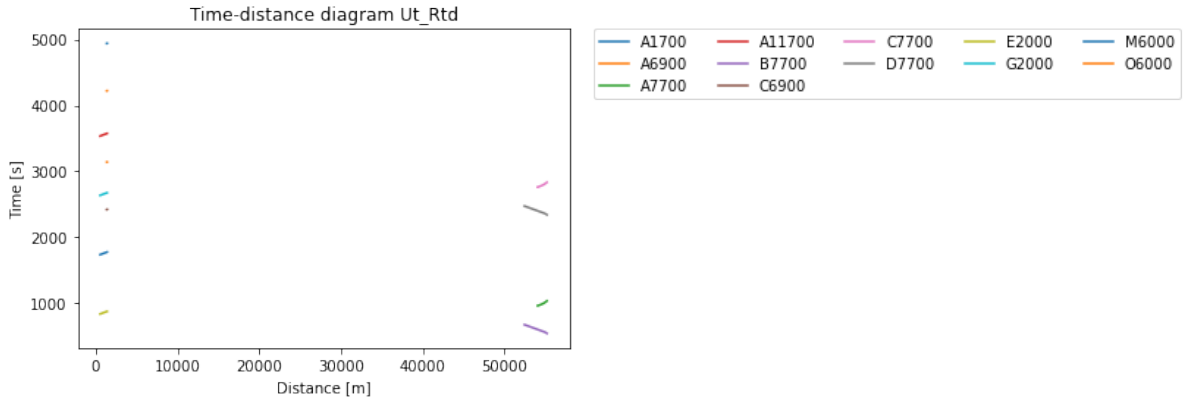


Figure G.23: Utrecht Centraal (Ut) - Rotterdam Centraal (Rtd) Trajectory

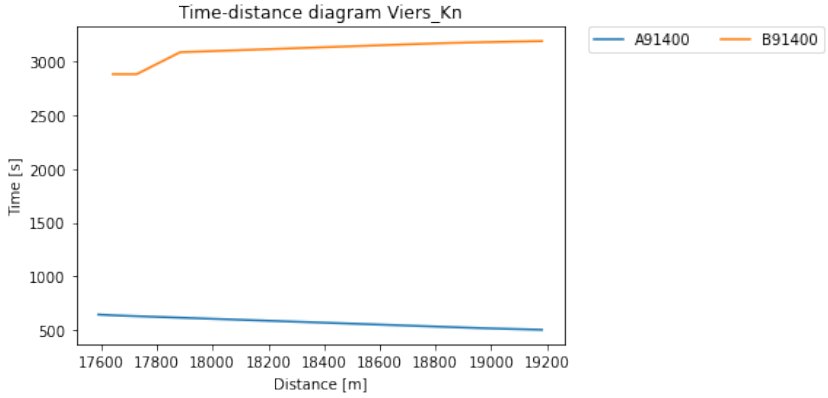


Figure G.24: Viers - Kaldenkirchen (Kn) Trajectory

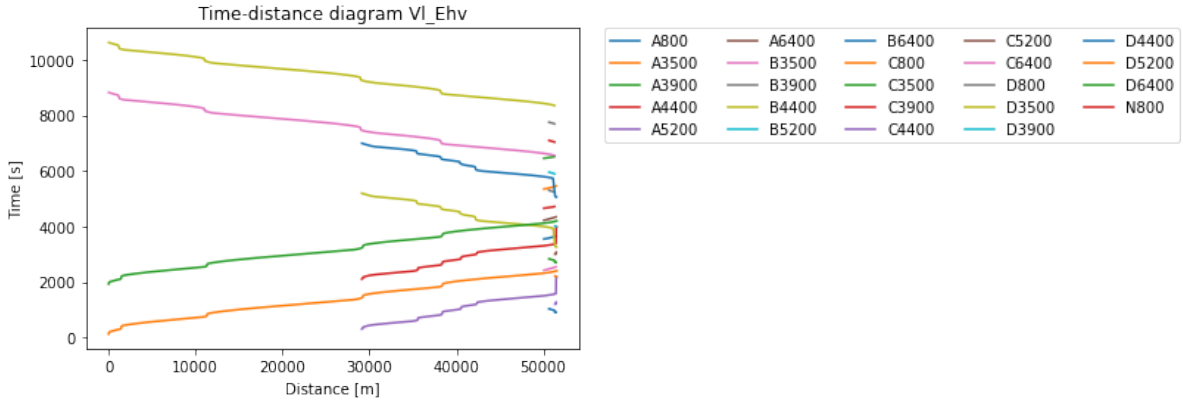


Figure G.25: Venlo (Vl) - Eindhoven (Ehv) Trajectory

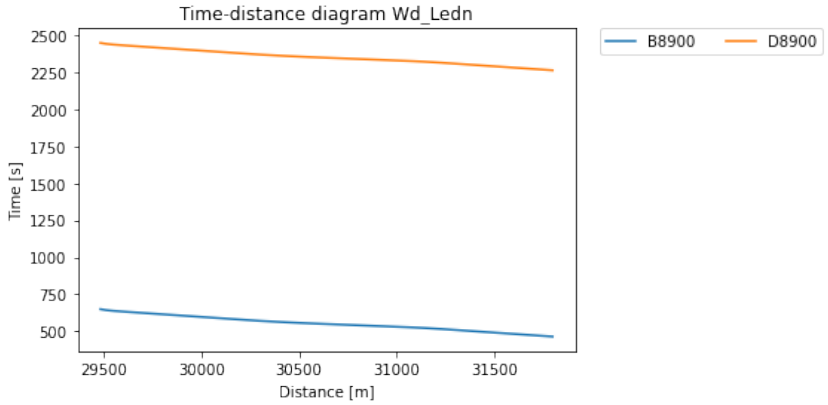


Figure G.26: Woerden (Wd) - Leiden (Ledn) Trajectory

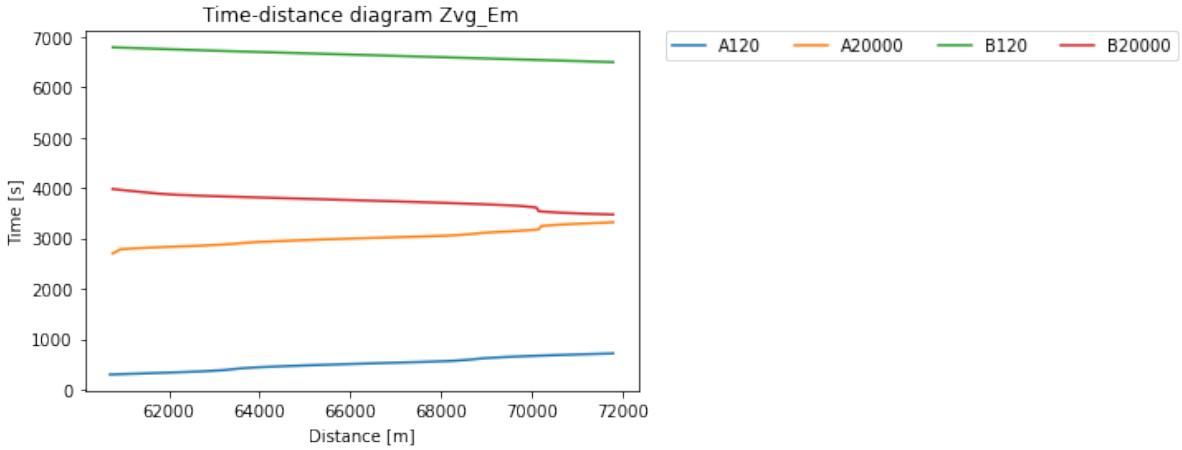


Figure G.27: Zevenaar grens (Zvg) - Emmerich (Em) Trajectory