ROUTING OF SHUNT TRAINS AT LOGISTICS HUBS OF NS USING CONSTRAINT PROGRAMMING

Master thesis

Industrial Engineering & Management

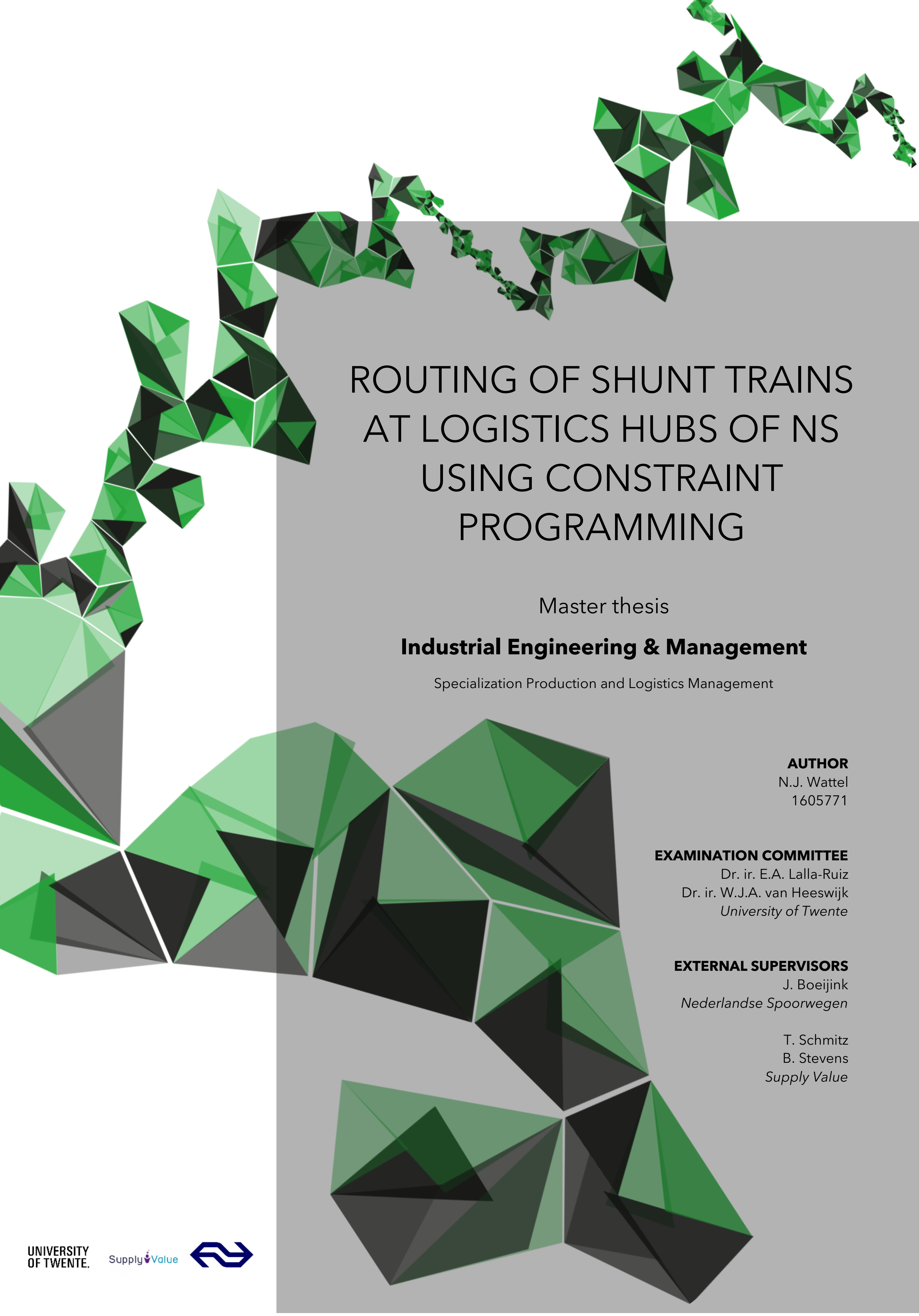
Specialization Production and Logistics Management

AUTHOR
N.J. Wattel
1605771

EXAMINATION COMMITTEE
Dr. ir. E.A. Lalla-Ruiz
Dr. ir. W.J.A. van Heeswijk
University of Twente

EXTERNAL SUPERVISORS
J. Boeijink
Nederlandse Spoorwegen

T. Schmitz
B. Stevens
Supply Value



Management summary

This research is conducted at NS, the leading public transport company in the Netherlands, and is conducted at the department of Research & Development of Hub Logistics. This department develops new approaches to solve the Train Unit Shunting Problem (TUSP). This problem can be defined as the problem of routing shunt trains to the shunting yard, parking shunt trains within the shunting yard, plan service tasks to shunt trains on allowed tracks and match incoming shunt trains to outgoing shunt trains. Currently, the problem is solved using a local search and several exact methods to solve small subproblems, which altogether are called HIP (Dutch: Hybride Integrale Planmethode, English: Hybrid Integrated Planning Method). To plan a 24-hour horizon, HIP currently can take up more than 16 hours of running time for a middle-sized hub like Eindhoven. NS wants to reduce this running time to be able to plan just a couple of hours before executing the plan. By reducing this time, NS can plan more agile, react faster to demand fluctuations and is more robust against short-term disruptions. HIP mainly faces problems of finding routes for shunt trains if there are few possibilities to route the shunt trains through the other train traffic. To overcome this problem, this research is set up. From previous research at NS, constraint programming (CP) appeared to be promising for determining routes for shunt trains. In literature, CP is not widely used for the Shunt Routing Problem. However, by seeing the problem as a joint problem of allocating resources and scheduling activities, the scheduling possibilities of CP can be highly valuable for modelling the SRP. In the literature, also hybrid methods are proposed, which start with CP and feed the output as warm start to MIPs or local searches. However, to the best of our knowledge, this hybrid approach has not yet been used for the Shunt Routing Problem and the Train Unit Shunting Problem.

The goal of this research is to develop a CP model that can determine routes and corresponding time intervals for the shunt trains from the station to the shunting yard (Shunt Routing Problem). The output of this model is then used as input for HIP, thereby overcoming the difficulty of finding those shunt routes.

The CP model is built for Eindhoven station, which is composed of a station area and two shunting yards. The model is based on the routing flow through the station that most shunt trains follow: a shunt train arrives at a platform, then should be routed towards the shunting yard. After being serviced, the shunt train leaves the shunting yard to be on time at its departure track. Some shunt trains without passengers arrive at the side of the station area instead of at a platform and need an additional route to be found by the model: to come from the side of the station area to the platform and vice-versa.

To validate the model, two data sets are used. Data set 2 is slightly bigger than data set 1 in terms of shunt trains and through trains. Both data sets yield good results in the CP model and the model solves both data sets within reasonable time (<15s) although data set 2 is slightly bigger than data set 1. Next, the output of the CP model had to be connected to HIP and serves as input for the local search. The baseline (running HIP without CP) is compared with running HIP together with CP in terms of running time and number of shunt moves. Three experiments with CP are created. The first experiment is without an objective function and experiment 2 and 3 have an objective function aligned with HIP to respectively increase the time each train spends in the shunting yard and to balance the trains over the shunting yards. Though the original goal of this research is to plan a 24-hour period, running time for this time interval appeared to be too high to conduct multiple experiments and replications. Therefore, the 24-hour period is split up into three intervals, together representing the 24-hour period: 13:00-15:00, 18:00-20:00 and 20:00-24:00. Lastly, to be able to conclude something about the 24-hour period, one replication of all three experiments and the baseline experiment is conducted for the full 24-hour period on data set 1.

All four experiments are conducted on HIP on the two different data sets for 10 replications each to prevent an error due to the random seed of the local search of HIP. To compare the results, both the algorithm running time and the number of shunt moves are taken as KPI. The results are similar for data set 1 and data set 2. Although the differences in running time with and without CP highly fluctuated, using CP as warm start for HIP resulted on average in a 70% reduction in running time compared to running HIP without CP. Especially the 24-hour resulted in a significant improvement in running time (>95%). Conclusions about the number of shunt moves are less significant, as a fair comparison can only be made when running time is equal. An important conclusion is that running CP without an objective function (experiment 1) does not necessarily guarantee good results in HIP. Balance between the two shunting yards plays an important role, however, the balance has to be fine-tuned more to be robust over other scenarios in Eindhoven. Lastly, increasing the time a shunt train stays in the shunting yard does help in most cases to reduce running time, but an excess of overtime complicates the parking problems thereby leading to a worse running time than originally achieved by HIP.

Firstly, we can conclude that the running time of the CP model without an objective is short (<15s) and that CP seems to be a promising and well-suited approach for solving the Shunt Routing Problem. Secondly, in most of the experiments, applying CP to find the shunt routes before starting the local search, yields a significant improvement on the total running time. Especially on the 24-hours scenario the results were impressive. However, we recommend to do an in-depth analysis in the future to investigate the reason of the running time reduction of the local search. This could then be used to align the objective of the CP model to always be able to guarantee good results in HIP when using CP. Furthermore, there is future work to do to develop a general model with an objective that is robust and can be generalised over other stations. Lastly, suggestions for using HIP are given, such as predetermined routes in HIP, separately solving shunting yards and the possibility to change predetermined routes created by CP.

Preface

This thesis marks the end of my master Industrial Engineering & Management and thereby the end of my time as a student. I would like to express my gratitude to several people who helped me during the process of my master thesis, but also during my time as a student in general.

First, I would like to thank Eduardo Lalla-Ruiz and Wouter van Heeswijk for their role as supervisor from the University of Twente. They helped me adding and improving the academic layer of my thesis. Although constraint programming was new for both of them, they were eager to help me and with their feedback I could lift my work to a higher level.

Next, I would like to thank Jord Boeijink for his role as supervisor at NS. Although I could visit the headquarters of NS just three times, we weekly debated about the direction of my research and how to achieve a good result, not only from a scientific view but also from a practical perspective. I would also like to thank Bob Huisman for brainstorming about a possible assignment and giving me the chance to do my research at NS.

Furthermore, I would like to thank Tom Schmitz and Bram Stevens for their supervisory role at Supply Value. They gave me practical recommendations and asked sharp questions from a side I did not expect, thereby enabling me not only to improve my thesis but also myself.

I can look back to a wonderful seven years in which I made numerous friends in Enschede, finished my bachelor at the University of Twente, did a board year at UniPartners Twente and subsequently a board year at the Dutch National Student Orchestra (NSO) and finished my master with this research. Many thanks to all of the following people: my family and girlfriend for their continuous support, my student house Club 9-2 which really felt as home (who else has a bar in his living room?), the do-group HGDBB for all *borrels* and weekends, my fellow students WatSchuurChaBroek who added a great deal of fun and *bi-ba-beunsma* to my masters, my high school friends Albufissa with whom I have shared many holidays and great evenings, my fellow board members from UniPartners for their ambition in running a business, my fellow board members from NSO with whom I have organised a fantastic month, and many others who I probably didn't include in this list.

I'm looking forward to the next step in my career at Supply Value!

Niek Wattel
Utrecht, October 2021

Contents

Management summary **i**

Preface **iii**

Contents **iv**

List of Figures **vii**

List of Tables **viii**

Glossary of Terms and Abbreviations **ix**

1 Introduction **1**

1.1 NS as a company 1

1.2 Problem description 2

1.3 Constraint programming 3

1.4 Research goal 5

1.5 Research problem 6

1.6 Research questions 6

1.7 Research design and methods 8

2 Problem context **10**

2.1 Terminology 10

2.2 Complexity of shunting 12

2.3 Subproblems of TUSP 13

2.3.1 Matching 13

2.3.2 Parking 14

2.3.3 Routing 14

2.3.4 Cleaning 14

2.3.5 Crew scheduling 15

2.4 Current situation at NS 15

2.4.1 Characteristics of the solution approach 15

2.4.2 Requirements of the local search 18

2.4.3 Assumptions of the local search 18

2.4.4 Limitations of the local search 19

2.5 Conclusion 20

3 Literature study 21

3.1 Train Unit Shunting Problem 21

3.2 Routing of trains 22

3.3 Constraint programming 25

3.4 Conclusion 26

4 Problem description and solution approach 28

4.1 Problem definition 28

4.2 Quality measurement 29

4.3 Problem example 30

4.4 Solution approach 33

4.5 Scope of the model 35

4.6 Requirements of the model 35

4.7 Model assumptions 37

4.8 Preliminaries of constraint programming 38

4.8.1 Interval variables 38

4.8.2 Sequence variables 39

4.8.3 Alternative constraint 39

4.8.4 StartAtEnd constraint 39

4.8.5 NoOverlap constraint 40

4.8.6 Synchronise constraint 40

4.9 Flow of a shunt train 41

4.10 Constraint programming model 43

4.10.1 Sets, parameters and variables 43

4.10.2 Constraints 43

4.11 Problem size reduction 45

4.12 Conclusion 45

5 Experimental setup 47

5.1 Data instances 47

5.2 Data preparation 47

5.3 Experimental design 49

5.4 Planning horizon and replications 50

5.5 Conclusion 52

6 Experimental results 53

6.1 Results from CP 53

6.2 Results from HIP 54

6.2.1 Results from HIP data set 1 55

6.2.2 Results from HIP data set 2 56

6.3 Conclusion 58

7 Conclusions and recommendations 59

7.1 Conclusions 59

7.2 Recommendations and future research 61

7.2.1 Recommendations and future research for the CP model 61

7.2.2 Recommendations and future research for HIP 63

References 65

Appendices 69

A Description of HIP input parameters 69

B Visualised shunt routes 70

C Description of parameters in the model 71

D Description of interval variables in the model 72

E Model constraints and explanation 73

F Conversion to HIP input 79

F.1 Scenario file 79

F.2 Location file 80

G Generalisation of the model 81

List of Figures

1 Railway network 2020 . . . 2
2 Partial track plan Utrecht CS . . . 3
3 Example of a sudoku . . . 4
4 Eindhoven station and shunting yards Tuin and Oostzijde . . . 6
5 Research design . . . 9
6 LIFO tracks and free tracks . . . 10
7 Caroussel layout and shuffleboard layout . . . 11
8 The three steps of the saw move . . . 11
9 Visualisation of a VIRM-4 and VIRM-6 . . . 15
10 Content of location and scenario files . . . 16
11 The six different types of tracks parts . . . 16
12 Overview of trackplan of Eindhoven station . . . 17
13 Visualisation of a local search . . . 20
14 Schematic overview of an artificial location . . . 31
15 Flowchart of HIP with and without CP . . . 34
16 Example of an interval variable . . . 38
17 Alternative constraint . . . 39
18 NoOverlap constraint . . . 40
19 Flow chart routing of shunt trains . . . 42
20 Break-down of the minimal time in a shunt yard . . . 42
21 Route dominance . . . 46
22 Track removal . . . 48
23 Summary of the experimental setup . . . 52
24 Example of HIP choosing the wrong route . . . 58
25 Adaptation to the location file . . . 80
26 Changing the model to another station . . . 83

List of Tables

1 Artificial example of combining train units and splitting trains 12
2 Summary of the literature 24
3 Feasibility constraints for HIP 30
4 Artificial scenario of the SRP 31
5 Possible shunt routes for the artificial location 31
6 Explanation of all sets in the model 43
7 Overview of all sets, parameters, variables and sequence variables 44
8 Explanation of all sequence variables 44
9 Overview of the two data instances used in the experiments. 47
10 Example of inclusion and exclusion of shunt trains in small time intervals . 51
11 Results from running data set 1 on the constraint programming model . . . 53
12 Results from running data set 2 on the constraint programming model . . . 53
13 Model characteristics of data set 1 versus data set 2 54
14 Results of HIP for data set 1 for different time intervals 55
15 Results of HIP for data set 2 for different time intervals 56

Glossary of Terms and Abbreviations

CP	Constraint Programming
COP	Constraint Optimization Problem
Crossing	Two trains using the same infrastructure element at the same time
CSP	Constraint Satisfaction Problem
Empty stock	Trains that do not carry passengers
Gateway	Point at which a train can enter and leave the shunting yard
Global constraint	Constraint which captures several if not all variables
HIP	Hybrid Integral Planning Method
Instanding trains	Trains that are present at the station or in the shunting yard at the start of the planning horizon
KPI	Key Performance Indicator
LIFO	Last In First Out
Local constraint	Constraint which captures usually at most two variables
Main line	Railway tracks outside the station area. Also called *open line*
MIP	Mixed-Integer Programming
MP	Mathematical Programming
NS	Nederlandse Spoorwegen
Oostzijde	One of the two shunting yards of Eindhoven station
Open line	Railway tracks outside the station area. Also called *main line*
Outstanding trains	Trains that are present at the station or in the shunting yard at the end of the planning horizon
Rolling stock	Umbrella term for all trains
Shunt trains	Trains that have to be shunted to the shunting yard or are routed back from the shunting yard
Shunting	The movement of trains to and from the shunting yard
Shunting yard	Yard on which trains can be temporarily stored or receive service
SRP	Shunt Routing Problem
Through trains	Trains that drive through the station and do not need to be shunted
Train unit	A train unit consists out of multiple carriages and a train consists out of one or multiple train units
Tuin	One of the two shunting yards of Eindhoven
TUSP	Train Unit Shunting Problem

1 Introduction

The first chapter of this thesis introduces NS as a company. After reading this chapter, the reader has an overview of what NS does, what research problem is addressed and what the motivation is behind the research question. In Section 1.1, NS as a company is elaborated on. In Section 1.2, the problem of NS is described as well as some characteristics of the problem. Next, the basics of constraint programming are described in Section 1.3. In Sections 1.4 - 1.6, the goal of the research and the research questions are stated. Lastly, in Section 1.7, the research design is presented together with an overview of the thesis.

1.1 NS as a company

The Dutch Railways, in Dutch *Nederlandse Spoorwegen* and abbreviated as NS, is the main passenger railway operator of the Netherlands. NS was founded in 1938 by a merger of HSM (Hollandsche Ijzeren Spoorweg-Maatschappij) and SS (Maatschappij tot Exploitatie van Staatsspoorwegen). It is a public limited company with the Dutch state as sole shareholder. The rail infrastructure was part of the portfolio of NS up to 1995, but nowadays ProRail has the right to maintain the railway network. NS has the sole right to transport passengers on the main railway network. On the decentralised railway network, there is no sole right for railway transportation and competitors such as Arriva, Blauwnet and Connexion exploit these parts of the network. With 1.3 million passengers per day and almost 20,000 employees in 2020 (NS, 2020), NS serves on one of the busiest railway systems of the world. Only operators in Switzerland drive more train kilometers per track kilometer (NS, 2019). Serving a big and diverse population from a sustainable point of view is resembled in respectively the mission and ambition of NS:

"Keeping the Netherlands accessible in a sustainable manner. For everyone."

and

"We and our partners deliver world-class mobility with a sense of responsibility towards our local environment. Always nearby, always affordable. Always sustainable."

By constantly optimizing the current railway network and adapting where necessary and possible, NS tries to improve the customer satisfaction. They aim thereby not only for the Randstad conurbation, but also for the rural regions of the Netherlands. This has resulted in a total railway length of 7,097 km and annually 165 million train kilometres are covered using the railway network as shown in Figure 1 (ProRail, 2019b).

In line with the vision and ambition to deliver nearby services for all Dutch citizens, NS does not only focus on transport between stations, but also the door-to-door service has been extended in the past few years, for example NS Zonetaxi, which is now available at 328 stations and, more commonly used, the OV Fiets (rental bike from NS). This is a sustainable door-to-door service, whose usage has more than doubled between 2016 and 2019. NS also stimulates going by bike to the train stations by offering facilities for storing bikes. The bicycle parking facility at Utrecht Central Station, opened in 2019, is the biggest bicycle parking facility in the world and can accommodate up to 12,500 bicycles (ProRail, 2019a). Focusing on the sustainability contribution, in 2020 NS managed to reuse 99% of all revised trains; old train floors became ping-pong tables, trains steps became tables and upholstery was used to make bags and laptop covers. Another example of the sustainability contribution is the use of green energy. Since 2017 all trains of NS run on 100% wind power (NS, 2020).

1.2 Problem description

To prepare for the continuously growing demand for public transport, the department Research & Development Hub Logistics at NS is responsible for innovations for logistic operations at the 40 hubs in the railway network. A hub consists of a train station and one or multiple *shunting yards*. A shunting yard consists of multiple tracks where trains, also called *rolling stock*, either can be temporarily stored or receive service such as cleaning, inspection and small maintenance. During peak hours - between 6:30 and 9:00 and between 16:00 and 18:30 - most of the rolling stock is needed to ensure seating capacity for passengers. Outside peak hours the surplus in rolling stock cannot stay on the main railways as it would interfere with through trains, so it has to be stocked elsewhere; in the shunting yard. The movement of trains from and to the shunting yard is called *shunting* and these trains are referred to as *shunt trains*. Contrary to shunt trains are *through trains*, which are defined as trains that stop by the station, either for passengers or cargo, and continue their way to another station afterwards. Beside serving as storage of rolling stock surplus, the shunting yard also provides services such as cleaning, inspection, maintenance etc. A *shunt plan* is a detailed plan of the whole shunting process; matching arriving and departing trains, scheduling service tasks, routing trains through the station and to the shunting yards and parking the trains on the shunting yards. In literature, the planning problem of a shunt plan is referred to as *Train Unit Shunting Problem (TUSP)*. The subproblems of TUSP are elaborated on in Section 2.3. In this research the focus will be on routing trains from the station to the shunting yards and vice versa. In literature, this problem is referred to as the *Shunt Routing Problem (SRP)*.

Figure 1: Railway network 2020

Railway tracks impose far more movement constraints compared to general roadways. In contrast to trains, cars can pass other cars, cars can make every turn (though sometimes

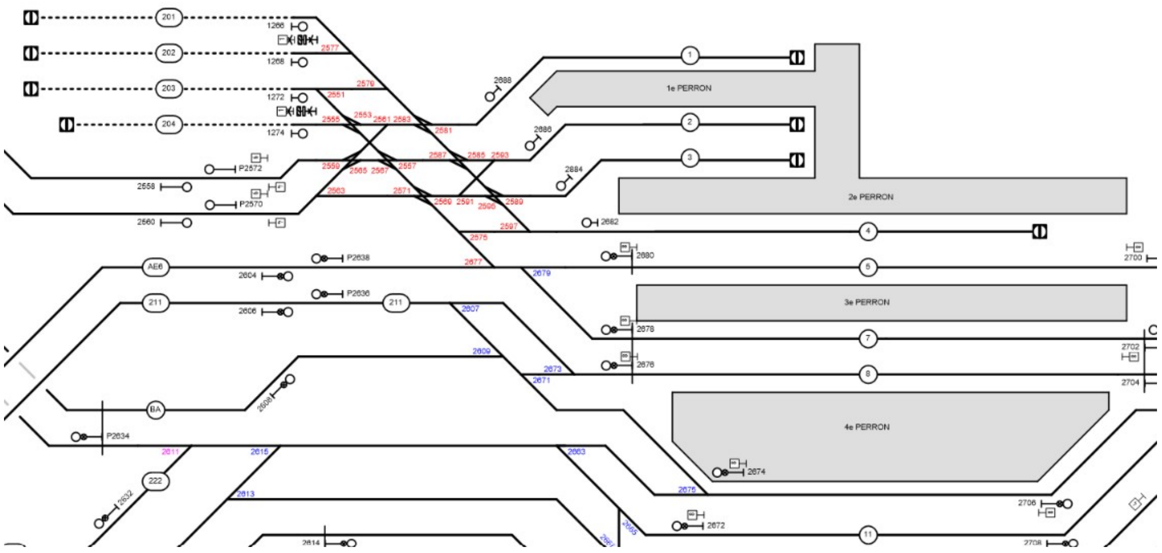


Figure 2: Partial track plan Utrecht CS, retrieved from *www.sporenplan.nl*

using a three-point turn) and - if necessary - cars can drive off the road. Each hub has its own infrastructure and depending on the infrastructure certain moves are possible and other moves are not. Figure 2 shows a partial track plan for Utrecht Central Station. In the figure you see the two types of tracks: LIFO (last in, first out) tracks and free tracks. LIFO tracks are tracks that trains can only enter and leave from one side, for example track 1 and 2 in Figure 2. Free tracks are tracks where trains can enter and leave from both sides, for example track 5 and 7 in Figure 2. Using switches trains can change from one track to another. Note that a connection between two tracks does not always mean that trains are able to drive in one movement from one track to another. For example, going from track 4 to track 5 (Figure 2) is only possible by leaving track 4 and drive backwards to track 5. This is called a saw movement and will be explained in more detail in Section 2.1.

1.3 Constraint programming

Constraint programming (CP) is a technique developed in the field of Artificial Intelligence and Computer Science and extends the power of logic programming. The concept of CP is comparable with Mathematical Programming (MP). There is a set of input parameters, there is a set of (discrete) variables and there is a set of constraints which limit the domain of the variables. CP is based on constraint propagation and domain reduction together with a constructive search strategy with backtracking. The finite domain of each variable is reduced by propagating in-between constraints and within constraints. An example where constraint propagation and domain reduction is used, is a sudoku. A sudoku requires that every 3x3 matrix, every row and every column is composed out of different numbers between 1 and 9. In Figure 3, the domain of all empty cells is {1,2,3,4,5,6,7,8,9}. Let us take cell E3 as an example. The constraint that requires different values in each 3x3 matrix reduces the domain of cell E3 to {2,3,4,6,8,9}. The constraint that requires different values in each row

	A	B	C	D	E	F	G	H	I
1	5	3			7				
2	6			1	9	5			
3		9	8					6	
4	8				6				3
5	4			8		3			1
6	7				2				6
7		6					2	8	
8				4	1	9			5
9					8			7	9

Figure 3: Example of a sudoku. When solving a sudoku, you implicitly use domain reduction by constraint propagation.

reduces the domain of cell E3 even further to {2,3,4}. Lastly, the constraint that requires different values in each column reduces the domain of cell E3 to {3,4}. This procedure is called constraint propagation and domain reduction. Next, we can start the constructive search strategy. We can, for example, assign value 3 to cell E3 and continue propagating. There are two possibilities: either we end up in a feasible final solution, or the constraint propagation detects an inconsistency. In case of the latter, the algorithm "backtracks" to cell E3 and changes its value into 4 which should then not lead to an inconsistency.

A core concept in constraint programming is the use of global constraints. Global constraints are constraints which depend on several if not all variables, whereas local constraints are constraints which mainly depend on at most two variables. The global constraint *AllDifferent* is the best known global constraint, which states that all variables captured within the constraint need to have different values. In the example of the sudoku (Figure 3), an example of the *AllDifferent* constraint would be: *AllDifferent(A1,A2,A3,B1,B2,B3,C1,C2,C3)*. If only local constraints could have been used, we would need $\binom{9}{2} = 36$ constraints (the number of combinations): $A1 \neq A2, A1 \neq A3, ..., C2 \neq C3$. Besides a more compact modelling formulation, global constraints can be more easily propagated which leads to a faster domain reduction and thereby a shorter running time.

Initially, CP was developed to solve Constraint Satisfaction Problems (CSPs). A solution for a CSP is feasible when all variables are assigned a value from its domain and all constraints are satisfied, for example the sudoku in the previous example. CP can easily be extended to a Constraint Optimization Problem (COP), which is a CSP with an objective function. CP then iteratively solves different branches until inconsistency is found and all branches are explored or pruned. Besides the domain reduction in combinatorial problems, CP's strengths arise from its flexibility in modeling and a rich set of operators. On the other hand, MP offers search techniques such as relaxation, cutting planes and duals for specific mathematical problem structures (Leung, 2004). Grossmann and Biegler (2004) point out that MP is very efficient when the relaxation is tight and the models have a structure that can be effectively exploited. CP works better for highly constrained discrete optimization problems where expressiveness of MP is a major limitation.

1.4 Research goal

In this thesis the focus will be on the Shunt Routing Problem (SRP). Matching train units, scheduling service tasks and parking is outside the scope of this research. The goal is to find an unobstructed and safe route for each train in the schedule to and from the shunting yard. From all possible routes, one should choose the route that allow easy routing for all subsequent trains. The main goal of this research is to develop an exact method that constructs plans for the routing of shunt trains for a twenty-four hour period. Given the infrastructure of a hub, the timetable of arriving and departing trains and the configuration (type, length) of arriving and departing trains, Constraint programming (CP) will be used to develop a feasible solution within reasonable time. This solution contains a schedule that shows for every train unit what route it has to take via which tracks to which shunting track in which time interval.

Since 2016 NS has been developing a hybrid solution method (HIP) that contains a local search heuristic based on the one proposed by R. van den Broek (2016) to construct solutions for the TUSP integrally; except crew scheduling all other subproblems of TUSP are solved integrally. This heuristic could solve realistic cases within reasonable time, whereas the tool used before, which was based on the state-of-the-art MIP approach (Kroon et al., 2008), could only solve small artificial instances. A drawback of the local search is its difficulty to find routes if the time interval a train can slip through the train traffic is very small. Several attempts has been made to solve the TUSP to optimality using MIP (Wolfhagen (2017), Haahr et al. (2017)), but they lacked the ability to scale well and in some cases the solutions performed worse than the manually created solutions by the planners. In 2017 a study has been conducted to investigate the possibility of using three - not yet adopted - techniques to solve the TUSP (Haahr et al., 2017). CP - and especially a variant of CP (CPMH) - showed promising results. In 2019, NS started a project to use constraint programming for routing of shunt trains. That research was first conducted on two shunting yards of Den Haag (Grote Binckhorst and Kleine Binckhorst). These two shunting yards are separated from the main railway network. Results were very promising and NS conducted subsequent research on Eindhoven, which is generally harder to solve because of the position of the shunting yards. As seen in Figure 4, shunting yard *Tuin* lies between the main railways and shunting yard *Oostzijde* lies close to the station as well, with two main railways in between. Research on Eindhoven was also promising, but was stopped in the beginning of 2020.

In the current situation, the network planning and the hub planning are highly interconnected which implies a long lead time for the hub planning. The hub planning is partially created months upfront and adapted to a greater or lesser extent, depending on the hub and the day to plan, between 4 weeks en 56 hours before executing the plan. NS wants to improve the flexibility of the hub logistics. A flexible hub logistic contributes to a more agile network logistic, a quicker response to demand fluctuations and more robust to short-term disruptions. They want to improve the flexibility of the hub logistics by planning short

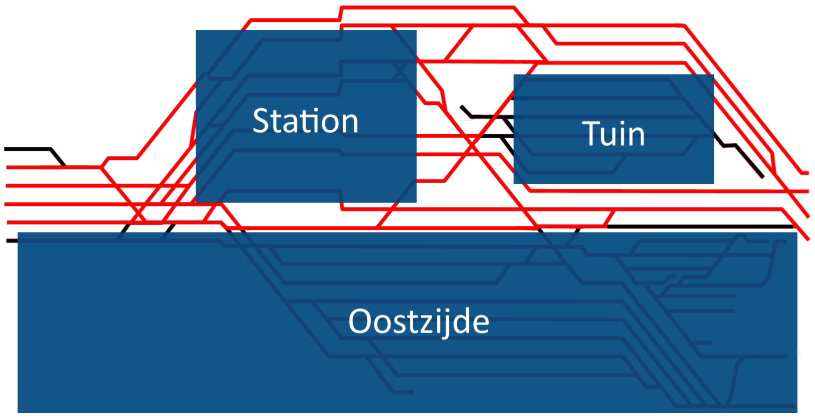


Figure 4: Eindhoven station and shunting yards Tuin and Oostzijde. The red railways mark the main railways that are used by through trains and cargo trains.

cycle (twice a day) and planning integral (a plan containing all movements and activities for both shunting and servicing). The Hybrid Integral Planning Method (HIP) is developed to both plan short cycle and plan integral. The main problem of NS is the running time of the local search within HIP, which can take up more than 16 hours of running time for a middle-sized hub like Eindhoven and is therefore not yet able to make a plan twice a day. The idea behind this research is to help the local search achieving a shorter running time. Whereas the local search has difficulty to find routes if the time intervals are very small, constraint programming should be able to identify these gaps. The output of the routing problem solved with constraint programming could then be used as input for the local search, thereby aiming for a feasible starting point of the local search with respect to routes between station and shunting yard. This focus of this research should therefore be on feasibility instead of on optimality.

1.5 Research problem

As mentioned in the previous section, NS has conducted research to constraint programming, but results were not as conclusive as they wanted. This research can be formulated as a follow-up, giving more conclusive results. The main research question can therefore be formulated as follows:

To what extent can constraint programming be used to create a feasible starting solution for the Shunt Routing Problem as warm start for the local search of NS?

To be able to answer this main research question, several research sub-questions are defined.

1.6 Research questions

The first research question focuses on the current situation at NS. The current problem approach of SRP is studied. Also the assumptions, requirements and limitations of the current problem approach are studied. The aim of this research question is to analyze the problem context and to find novelty gaps for the solution method proposed in this thesis. This leads to the first research question:

1. How does NS currently handle the SRP?
    (a) What solution approach does NS currently use to solve the SRP?
    (b) What are the requirements of the solution approach with respect to SRP?
    (c) What are the assumptions of the solution approach made by NS with respect to SRP?
    (d) What are the limitations of the solution approach?

To develop a model to solve the SRP, one should first look at the proposed and existing methods available in literature. First a broad search is conducted to identify methods to solve the SRP. Next, an in-depth search is conducted to the link between constraint programming and SRP. Lastly, scalability of the problem and running time of the algorithm is important for NS. Therefore we arrive at the following research question:

2. What is proposed in literature to solve the SRP?
    (a) What methods are used to solve the SRP?
    (b) To what extent has constraint programming been used to solve SRP?
    (c) How do the researched methods differ in terms of problem scale and running time?

After the literature phase, insight should be obtained in building the solution method for solving the SRP. This insight consists of assumptions, requirements, and other necessary information to build the solution method. This leads to the third research question.

3. How should the solution approach be developed?
    (a) What is the scope of the solution approach?
    (b) What are the requirements of the solution approach?
    (c) What are the assumptions of the solution approach?
    (d) How can we translate the assumptions and requirements to model constraints?

After the solution method has been developed, the method should be tested. The first interesting question is what experiments can be developed to validate the solution approach. Another aspect is how the current local search of NS performs when the output of the proposed solution approach is used as starting point for the current local search. This leads to the fourth research question.

4. How does the solution approach perform compared to the current situation under different experimental settings?
    (a) What are the different experimental setups that should be considered?

(b) How does the solution approach perform when the output of the proposed solution approach is used as starting point of the current method at NS?

Finally, we need to draw conclusions from the experiments and make recommendations for NS.

5. What are the conclusions and recommendations for NS?

   (a) What can be concluded about using constraint programming to create a feasible starting solution for the Shunt Routing Problem as warm start for the local search of NS?

   (b) What are recommendations and future research for using constraint programming as a feasible warm start for the local search at NS?

1.7 Research design and methods

This research will mainly focus on quantitative analysis. The global outline of this research will start with identifying the problem context and a literature study to find already existing information about the problem. Next, methods are defined to solve the problem and necessary data for the model is collected. Using the methods, different experiments are setup and executed. The results of the experiments will be verified and will lead to conclusions. Figure 5 gives an overview of the relation between the research questions, what input is needed for each research question and what chapter of the thesis is dedicated to each research question.

Using the Python API, the IBM ILOG® CP Optimizer will be used to solve the constraint programming problem. The advantage of this solver compared to other CP solvers, is that it incorporates some heuristics to boost the performance of CP. The solver makes use of the framework to process temporal constraints as described by Dechter et al. (1991). This framework is called Temporal Constraint Satisfaction Problem (TCSP) and aims to 1) to find all feasible times that a given event can occur, 2) to find all possible relationships between two given events and 3) to generate at least one scenario consistent with the information provided. This is mainly interesting in finding possible times at which shunt trains do not conflict with through trains. This can be input to the routing problem.

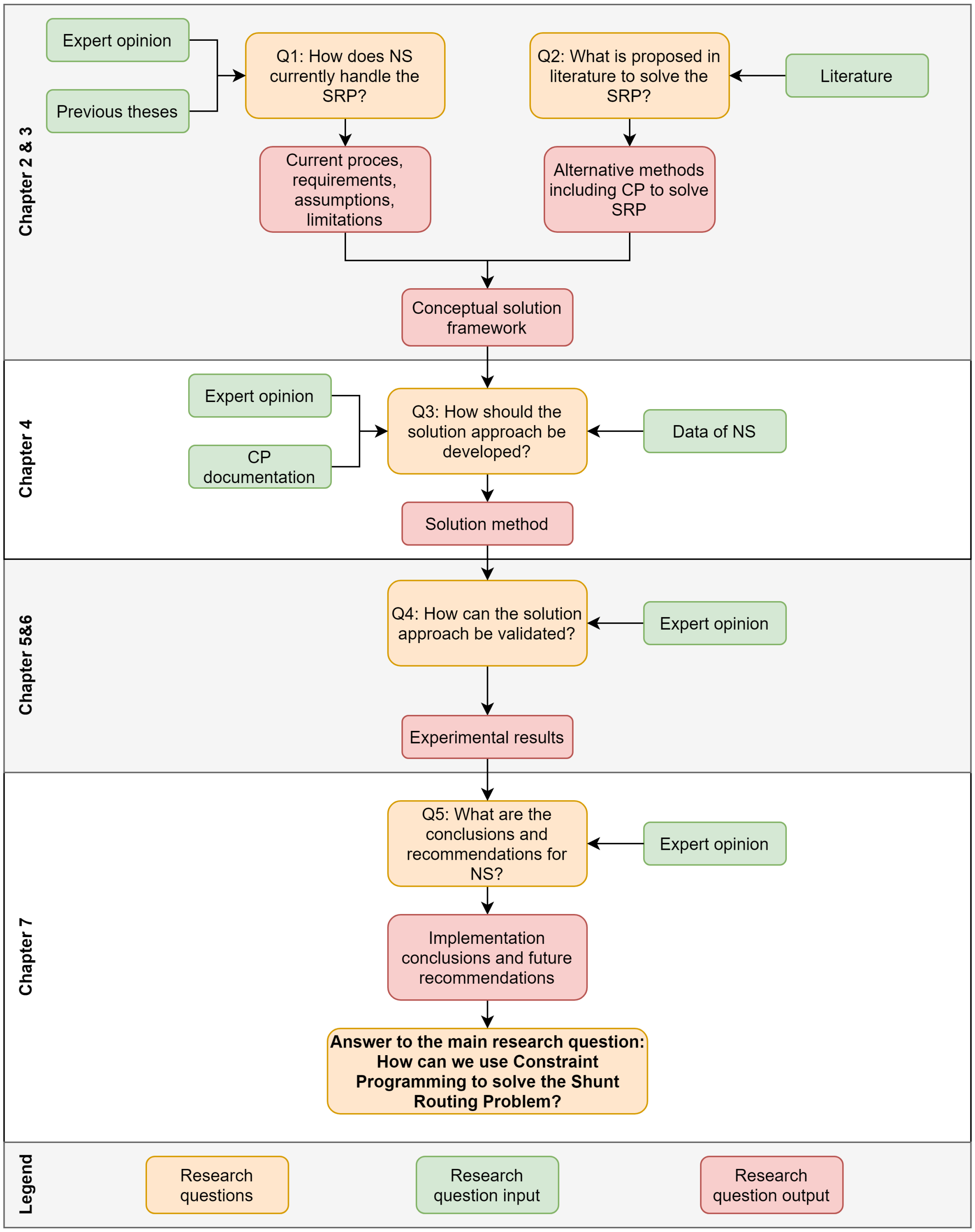

Chapter 2 & 3
Expert opinion
Previous theses
Q1: How does NS currently handle the SRP?
Q2: What is proposed in literature to solve the SRP?
Literature
Current proces, requirements, assumptions, limitations
Alternative methods including CP to solve SRP
Conceptual solution framework
Chapter 4
Expert opinion
CP documentation
Q3: How should the solution approach be developed?
Data of NS
Solution method
Chapter 5&6
Q4: How can the solution approach be validated?
Expert opinion
Experimental results
Chapter 7
Q5: What are the conclusions and recommendations for NS?
Expert opinion
Implementation conclusions and future recommendations
Answer to the main research question: How can we use Constraint Programming to solve the Shunt Routing Problem?
Legend
Research questions
Research question input
Research question output

Figure 5: Research design. This figure shows the relation between the research questions (orange), what input is needed for each research question (green) and what output is generated by each research question (red). It also shows the chapters in which the research questions are answered.

2 Problem context

In this chapter a detailed description of the problem is given. It starts with some terminology used in the problem (Section 2.1). Thereafter, in Section 2.2, insight into the complexity of the problem is given. In Section 2.3, the subproblems of TUSP are explained in more detail and lastly, an overview of the current situation at NS is given in Section 2.4. This last section will treat research question 1 and a conclusion will be given in Section 2.5.

2.1 Terminology

As introduced in Section 1.2, there are two different types of tracks: LIFO tracks and free tracks. Figure 6 provides an example of LIFO and free tracks. As can be observed, LIFO tracks are tracks through which trains can only enter and leave from one side. This behaviour can be described as a stack. On the other hand, free tracks are tracks through which trains can enter and leave from both sides. This behaviour can be modelled as a double-ended queue (deque). The example of the LIFO track in Figure 6 can only be achieved in one way: first train 2 enters the track and thereafter train 1. The example of the free track in Figure 6 can be achieved in three ways: either train 1 arrived from the left and train 2 from the right, or train 2 arrived from the left first and thereafter train 1 arrived also from the left, or train 1 arrived from the right first and thereafter train 2 arrived also from the right.

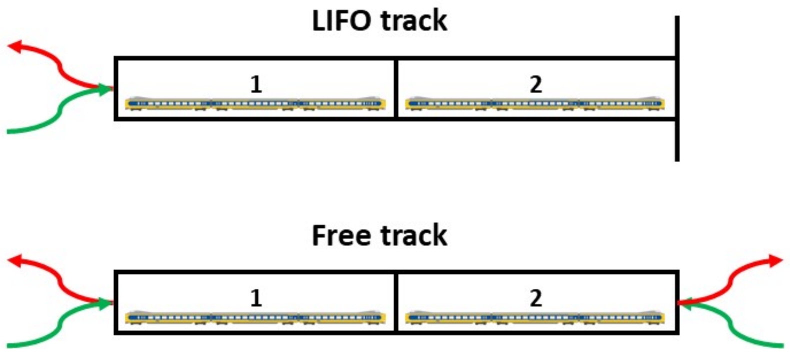


Figure 6: Difference between LIFO tracks (stack) and free tracks (deque).

Through trains arrive and leave the station using the main network. Shunt trains arrive and depart by using a combination of main network tracks and shunt tracks. There can be one or multiple possible ways to go from the station to the shunting yard. These routes are called *shunt routes* and the entrances to the shunting yard from the main network are called *entrance gateways*.

Just as a station can have multiple layouts depending on the track configuration, the shunting yard also can have multiple layouts. This depends on the number of tracks, but also on the type of tracks. We can distinguish two types of layouts in a shunting yard:

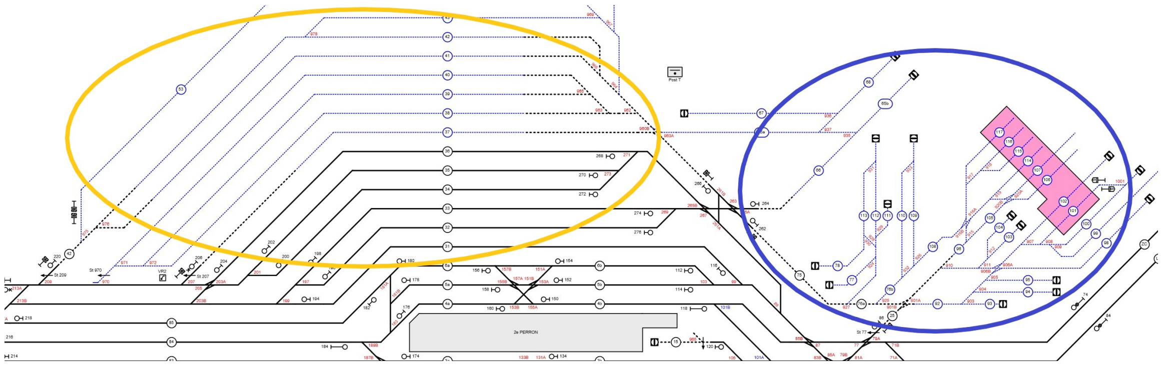
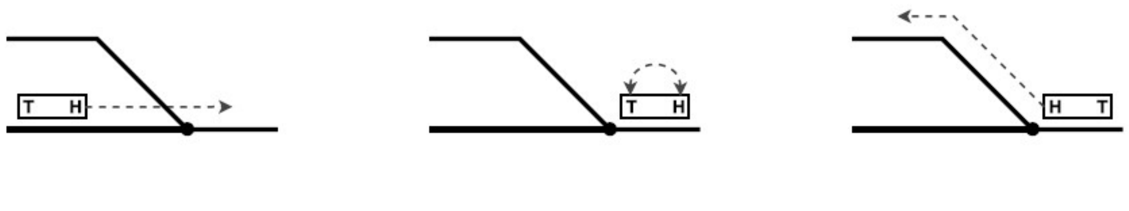


Figure 7: Caroussel layout (yellow circle) and shuffleboard layout (blue circle), retrieved from *www.sporenplan.nl*

1. *Carousel layout.* This layout consists of free tracks. A shunt train can enter the shunting yard from multiple entrance gateways and can also leave the yard from multiple sides. A shunt train is not constrained to use only one track within the shunting yard and can make a circle to return to its starting point, thereby referring to the rotating behaviour of a carousel. The yellow circle in Figure 7 shows a typical carousel layout.
2. *Shuffleboard layout.* This layout refers to the shuffleboard game where you have to shuffle pucks into scoring boxes via small doorways. This type of shunting yard consists only of LIFO tracks where the trains are represented by the pucks and the LIFO tracks by the scoring boxes. The blue circle in Figure 7 shows a typical shuffleboard layout.

If a train needs to switch between tracks, for example in the carousel layout, the train has to make a so-called *saw move*. Figure 8 shows the steps of a saw move. In this move, a train leaves a certain track (Figure 8a), then needs to change direction (Figure 8b) to drive to the next track in opposite direction (Figure 8c). To change direction, the driver needs to walk from the head to the tail of the train. The tail then becomes the new head of the train, which is called *kopmaken* and the train can drive in the other direction to the new track. This move is possible for trains of NS since all trains are bi-directional. However, saw moves are costly in terms of time and space since the driver must traverse the complete train length while the train is blocking an entrance for at least two tracks.

Trains can consist of multiple train units. And each train unit consists of multiple carriages.



(a) Leave first track. (b) Change direction. (c) Enter second track.

Figure 8: The three steps of the saw move (R. van den Broek, 2016)

For example, the two train units of Figure 9 are two trains if they travel separated. They can also be combined to form a longer train. This longer train then consists of two train units. The first train unit in Figure 9 consists of four carriages whereas the second train unit consists of six train units. Combining train units is only possible if they are of the same type (e.g. VIRM). Trains of different subtypes, but of the same type, are allowed to be combined (e.g. VIRM-4 and VIRM-6). Combining and splitting trains can be used to make longer or shorter train based on the expected amount of passengers, but also for convenience in the shunt planning. In reality, a train ID is comparable to a flight number. It specifies a certain line at a certain time and the train ID stays the same if in intermediate stations a train unit is added *(bijplaatsen)* or removed *(aftrappen)*. For modelling convenience, when train units are combined and when a train is split in intermediate stations, the train ID of the train before combining/splitting ceases to exist and the newly splitted/combined trains get another train ID. Table 1 shows an example of such a fictional splitting and combining schedule.

Train ID	**Time**	**Arrival/Departure**	**Train unit(s)**
917	09:20	A	VIRM-6, VIRM-4
9171	09:25	D	VIRM-6
102	09:27	A	VIRM-4, VIRM-4
1021	09:30	D	VIRM-4, VIRM-4, VIRM-4
104	09:39	A	ICM-3, ICM-4, ICM-4
1041	09:42	D	ICM-4
1042	09:45	D	ICM-3, ICM-4

Table 1: Artificial example of combining train units and splitting trains

The last part of the terminology is *crossing*. If train 2 leaves the LIFO track in Figure 6 before train 1, train 2 crosses train 1. In other words, train 2 would go through train 1. This is not possible in reality and solutions to the SRP containing crossings are therefore infeasible.

2.2 Complexity of shunting

To grasp the complexity of the TUSP, one should take a look at all planning phases that happen before planning of shunting takes place. Six stages can be defined (van Hove, 2019):

1. *Network planning.* In this planning phase the rail network is designed. It is decided where stations and tracks are placed, which makes it a strategic planning phase over a long-term horizon.

2. *Line planning.* This planning phase defines which lines are formed in the rail network. It includes the start and end station of each line, at which stations the line stops and what the frequency of the line is.

3. *Timetabling.* After the line planning has been made, a timetable is added to all lines. This is called the timetabling planning phase.

4. *Rolling stock scheduling.* When the timetable for every line is developed, rolling stock is assigned to all lines. This includes determining the type of train, the number of train units as well as the sub type which implies the number of carriages. Figure 9 shows two sub types of the type VIRM. Both train units can be connected to form a longer train. Note that combining train units is only possible if all train units are of the same type, but within this type different sub types can be combined. The combination of train units is referred to as *train* and it consists of two *train units*. The objective in this phase is to maximise the seating availability based on the expected number of passengers.

5. *Personnel planning.* Based on the rolling stock schedule, personnel is assigned to the planning. This planning matches the demand of rolling stock with the availability of personnel.

6. *Shunt planning.* Lastly the shunt planning is made. This includes the main four sub problems of shunt planning: matching, parking, routing and cleaning (Lentink, 2006). In Section 2.3, there will be elaborated more on the different sub problems of shunt planning.

As the shunt planning is preceded by many other planning phases, the input of the TUSP is highly constrained which makes it complicated for shunt planners to find feasible solutions. Combined with the continuously growing density of the Dutch railway network, there is a need to find feasible solutions within a small time interval and ease the job of the shunt planner. Besides the operational added value, a shunt planning can also be used on tactical or even strategic level to determine if the current capacity is enough to fulfil future demand.

2.3 Subproblems of TUSP

Lentink (2006) defined five main subproblems of TUSP: matching, parking, routing, cleaning and crew scheduling. Although the scope of this research is limited to the routing subproblem, insight is needed into the other subproblems as well. This is due to the interaction between the different subproblems. Also in part of the solution approaches, a.o. the solution approach currently used by NS, the subproblems are solved integrally. Therefore, in this section, these subproblems will be shortly elaborated on.

2.3.1 Matching

Lentink (2006) defines the matching problem as follows: ”Given a timetable of arriving and departing train services, the *Train Matching Problem (TMP)* is to find a feasible matching of arriving train units to departing train units of a minimum required number of resources.” Resources can be defined as personnel, track, time, etc. Lentink also shows that this problem in its most general form is NP-hard by a reduction from the 3 Partition Problem.

2.3.2 Parking

Lentink (2006) defines the parking problem as follows: "Given a set of blocks of shunt units, a set of shunt tracks, and estimates of the route cost for each unit to and from each shunt track, the *Track Assignment Problem (TAP)* is to assign blocks to shunt tracks in a feasible manner, thereby minimizing the cost and maximizing the robustness of the solution." Blocks of shunt units are sets of train units that arrive in the same train and also depart in the same train. A solution to this problem is feasible if the solutions does not include crossings, if the length of all parked shunt units on a shunt track never exceeds the length of the shunt track and if all train units in the problem are allowed to park. Lentink also shows that this problem in its most general form is NP-hard by a reduction from the Bin Packing Problem.

2.3.3 Routing

Lentink (2006) defines the routing problem as follows: "Given the station railway infrastructure, a set $\mathcal{X}$ of assigned infrastructure reservations, and a set $\mathcal{R}$ of route requests, the *Shunt Routing Problem (SRP)* is to find a maximum number of shunt routes without conflicts, thereby minimizing the cost of the set of routes." In this definition the term infrastructure reservation refers to the reservation of a part of the infrastructure for a certain time period. Infrastructure can be reserved for small time intervals (e.g., for through train movements, for standing rolling stock) and large time intervals (e.g., infrastructure maintenance). The term route request refers to the request of a certain train for a route from track A to track B. A solution to the SRP assigns a route to all route request without crossing. R. van den Broek (2016) points out that a shortest path can be found in polynomial time if the track occupation on the site at the time of movement is fixed. However, parked trains are allowed to be moved to make space. This closely resembles the Rush Hour problem which is known to be PSPACE-complete (Freling et al., 2005).

2.3.4 Cleaning

Lentink (2006) defines the cleaning problem as follows: "Given is a set of blocks $\mathcal{B}$, where each block $b \in \mathcal{B}$ is assigned a release time a_b, a deadline q_b and an amount of work p_b, a parameter z, and a function $c(m)$ representing the number of man-minutes available at each point in time $m \in \mathcal{M}$. The *Shunt Unit Cleaning Problem (SUCP)* is to assign p_b consecutive man-minutes to each block $b \in \mathcal{B}$ within $[a_b, q_b]$, while maximizing the number of jobs that start cleaning in $[a_b, a_b + z]$, and $c(m)$ man-minutes are available at each point $m \in \mathcal{M}$." The parameter z indicates the number of minutes after the arrival of a block, which is still considered "close" in time to its release time and gives thereby a little slack to the problem. SUCP can be seen as a single machine scheduling problem and by reduction from the problem Sequencing within Intervals it is proven to be NP-complete. In further research the cleaning problem is extended with other service tasks such as inspections and small maintenance jobs.

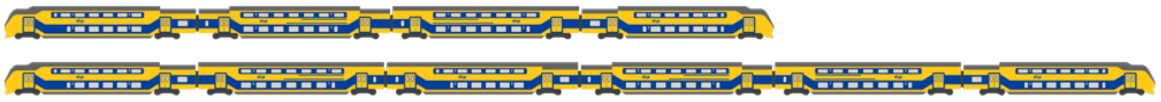


Figure 9: Two train units of the same type but with different lengths: 4 carriages (VIRM-4) and 6 carriages (VIRM-6)

2.3.5 Crew scheduling

The objective of this problem is to use the available crews as efficiently as possible. Lentink (2006) distinguishes three different types of crew: shunting drivers, shunting assistants and cleaning crews. The shunting driver is responsible for routing the train over the tracks. The shunting assistant is responsible for (de)coupling of trains and preparing for departure. The cleaning crews are responsible for the internal and external cleaning of trains. The output of the previous four subproblems is input to the crew scheduling problem. For example, depending on the output of the routing problem, a certain number of shunting drivers is needed to perform the routing actions.

2.4 Current situation at NS

In this section the goal is to answer the first research question: *How does NS currently handle the SRP?* To answer the research question, the local search of NS is described in Section 2.4.1. In Section 2.4.2, the requirements of the local searched are explained. In Section 2.4.3, the assumptions of the local search are shown and in Section 2.4.4, the limitations of the local search are explained.

2.4.1 Characteristics of the solution approach

NS currently solves the SRP integrally with the other subproblems of TUSP. The solution approach is based on the local search as proposed in R. van den Broek (2016) and it is called *HIP* (Dutch: Hybride Integrale Planmethode, English: Hybrid Integrated Planning Method). The local search in HIP is based on Simulated Annealing and many features have been added in the past few years. A complete description of the functionalities of the local search and an in-depth analysis of the local search is not required in this research. In this research the elements of the local search only related to routing are described, as well as the assumptions, requirements and limitations related to routing. The last parts are the input and output files. HIP needs two input files in JSON format: a location file which contains all infrastructure elements of a hub, and a scenario file, which contains all trains and necessary attributes. The complete decomposition of the files is shown in Figure 10. The term ”nonServiceTraffic” refers to through trains which either stop by a platform or just pass the station without stopping. ”In” and ”Out” refer to trains that are respectively incoming trains which need to be shunted to a shunting yard, and outgoing trains that need to be shunted from a shunting yard to the station. Not uncommonly these ”In” and ”Out” trains are train units that are split off or combined with through trains. Note that ”Out”-

trains do not have service tasks assigned. Each "Out"-train is matched to an "In"-train, therefore service tasks are already executed before the "In"-train becomes an "Out"-train.

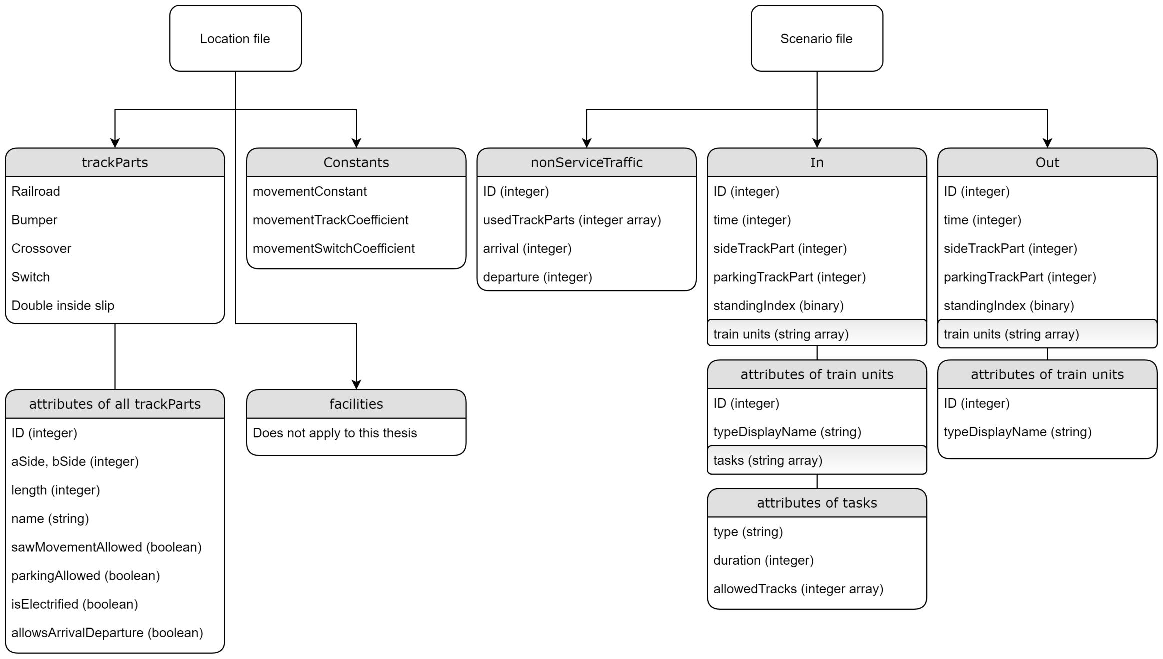
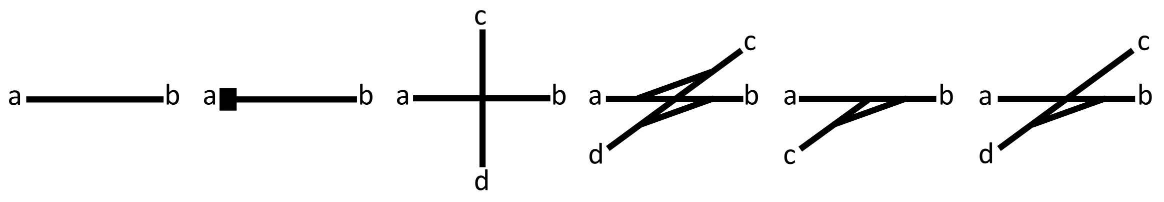


Figure 10: Overview of the content of JSON input files 'location' and 'scenario' used by HIP

An explanation of all attributes of Figure 10 can be find in Appendix A. The different type of track parts are illustrated in Figure 11.



(a) Railroad. Possible routes are: a $\leftrightarrow b$

(b) Bumper. Possible routes are: a $\mapsto b$

(c) Crossover. Possible routes are: a $\leftrightarrow b$, $c \leftrightarrow d$

(d) Double inside slip. Possible routes are: a $\leftrightarrow b$, $a \leftrightarrow c$, $b \leftrightarrow d$, $c \leftrightarrow d$

(e) Switch. Possible routes are: a $\leftrightarrow b$, $b \leftrightarrow c$

(f) Single inside slip. Possible routes are: a $\leftrightarrow b$, $b \leftrightarrow d$, $c \leftrightarrow d$

Figure 11: The six different track parts. Note that the single inside slip is not present in HIP. It is modeled by a combination of a crossover and a switch. A signal is modeled as a combination of a switch and a bumper.

To ensure safety, trains should keep a safe distance between each other. This holds for every infrastructure element; platforms, railroads and switches. These safety norms are laid down by ProRail. The size of the safety norm depends on three factors:

1. whether the two trains drive in the same direction or in opposite direction of each other,
2. whether both trains are through trains, both trains are shunt trains or one train is a through train and one train is a shunt train,
3. whether the train is arriving, departing, stopping shortly or driving through the station.

In the newest version of HIP, the safety norms are implemented. However, the version of HIP used in this research does not contain the safety norms yet.

If HIP wants to park a train or make a saw movement on a certain track, it should be allowed to do so. This is described in the *parkingAllowed*- and *sawMovementAllowed*-attributes of all track parts. On almost all tracks within the shunting yards, a train is allowed to make a saw movement. On a large part of these tracks, parking is also allowed. On all platforms, a saw movement is allowed, but parking is not allowed. Lastly, on some through tracks, such as 18A and 18B, a saw movement movement is allowed, but parking is not. Figure 12 shows an overview of Eindhoven with the tracks where parking and saw movements are allowed.

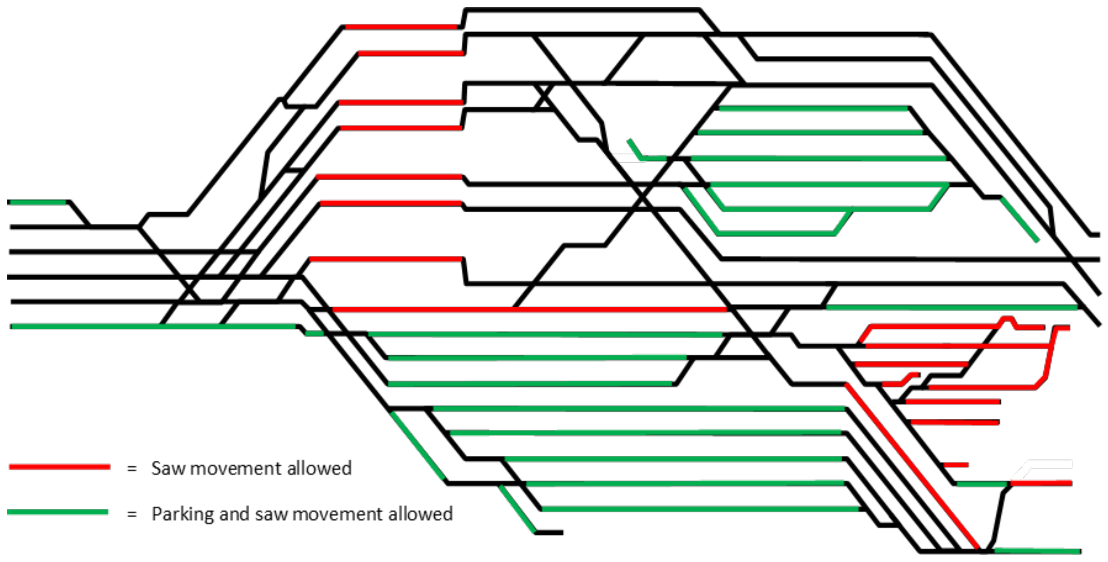


Figure 12: Overview of trackplan of Eindhoven. Green tracks indicate that parking and saw movements are allowed. Red tracks indicate that only saw movements are allowed. On black tracks neither parking nor saw movements are allowed.

Furthermore, HIP makes use of a *route-lock route-release*, which means that a train reserves a certain route for its movement and releases the infrastructure elements on the route after the train has completed the route. NS is thinking of implementing a *route-lock sectional-release*, which means that a train reserves a certain route for its movement, but releases parts of the route (sections) when the train has bypasses the section. This enables more flexibility in routing, however, the infrastructure of the stations should allow this method.

At the moment, there are only two key performance indicators to evaluate the quality of the solution created by HIP; shunt moves and running times. Shunt moves are the number of movements all shunt trains make together and running time is the algorithm running time to plan one day on a hub, which is approximately 16 hours.

As this research is focused on Eindhoven, two specific rules are described. Firstly, the intercity from Den Haag to Eindhoven (ICR Gvc-Ehv 9), which is planned every 30 minutes, arrives in Eindhoven at platform 1 and has to leave Eindhoven 29 minutes later from platform 5. For technical reasons and convenience the rule is to always send this train via shunting yard "Tuin" (see Figure 4) to platform 5. Secondly, some service tasks are only possible at shunting yard "Oostzijde". The specific tracks at which the service tasks can be executed are denoted in the *allowedTracks*-attribute of every service task. If a train needs service that can only be performed in Oostzijde, the train should, as expected, be shunted to Oostzijde.

The output of HIP is a solution file which contains the matching for each incoming train to an outgoing train, which infrastructure elements each train uses in a certain time interval and which service teams are planned on which tracks and which trains they have to give service to. Also relevant statistics, local search configurations and warnings are output of HIP.

2.4.2 Requirements of the local search

The requirements of the local search are mostly set up to guarantee a feasible schedule. A feasible schedule is a schedule where all trains are planned, trains do not cross (i.e., they never have a single infrastructure element in common at the same time), all service tasks are planned, trains arrive and depart on the provided times, workers do not provide two services at the same time, trains only park and reverse at the tracks that allow for parking and reversing, safety norms of ProRail are respected, each incoming train is matched to an outgoing train, a train cannot park on a track that is smaller than the train itself and the capacity of the shunting yards is respected. There are no hard requirements to the running time of the local search, which is currently approximately 16 hours for a medium-sized hub like Eindhoven station.

2.4.3 Assumptions of the local search

In the local search, many assumptions have been made. They can be categorised into three main assumptions: duration, movement and changeability.

Duration assumptions. Many different durations have been estimated or assumed. The first duration is movement duration. HIP assumes all sub movements to take 2 minutes. For example, a train enters the station area and drives to a platform, then goes to an intermediate parking track, then reverses and drives to the final parking track and after some time drives to the platform and leaves the station. These are five different movements that are all assumed to be 2 minutes (movementConstant in Figure 10). There is however research being conducted to estimate different movements more accurately, depending for example on weight and acceleration of the train. Another possible extension of the movement duration, which can already be applied in HIP, is to extend the movement duration with a fixed time per used switch and/or per used track (movementSwitchCoefficient and movementTrackCoefficient in Figure 10). The second duration is reversal time. This duration has been estimated more accurately than the movement duration and depends on the type of train and the train length. The third duration is the (de)coupling time. If two train units need to be coupled, or a train needs to be split into two train units, a certain time is needed to perform this action. Coupling is estimated to be 3 minutes and decoupling to be 2 minutes, not depending on the type of train. The fourth and last duration assumption is walking time. This is of interest for the mechanics and the train driver and depends on the distance between different infrastructure elements. Note that this is the current situation. NS is continuously improving the duration assumptions to obtain more realistic time intervals.

Movement assumption. There are some movements called *open movements*. These movements are shunt trains which either enter the station area without passengers and could directly move to the shunting area without stopping by a platform, or leave the station area without passengers and directly move from the shunting yard without stopping by a platform. For modelling convenience, the assumption made regarding open movements is to always direct open movements via a platform.

Changeability assumption. The planning of through trains is done in a previous phase. So it is assumed the timing, the route and the platform of through trains is fixed and cannot be changed in HIP. The same holds for shunt trains, only the route from station to shunting yard (and vice versa) and the choice of shunting yard is flexible.

2.4.4 Limitations of the local search

There are two major limitations and one minor limitation of the local search. Firstly, the local search sometimes fails to identify routing possibilities if the possible time interval to execute the movement is very small. Figure 13 gives a visual explanation of an artifical scenario. The three sub figures represent the solution space for three scenarios and the local search tries to find a feasible solution within this solution space. Above the red dotted line are feasible solutions and below the red dotted line are infeasible solution. In Figure 13a, the local search keeps searching for this feasible solution, but the chance the local search is able to find the feasible solution is very small. In Figure 13b, the chance of finding the feasible solution is already bigger due to the broader time interval in which the solution is feasible. In Figure 13c, the chance of finding the feasible solution is again bigger. Although the time interval itself is small, the rest of the graph is globally climbing towards the feasible time interval.

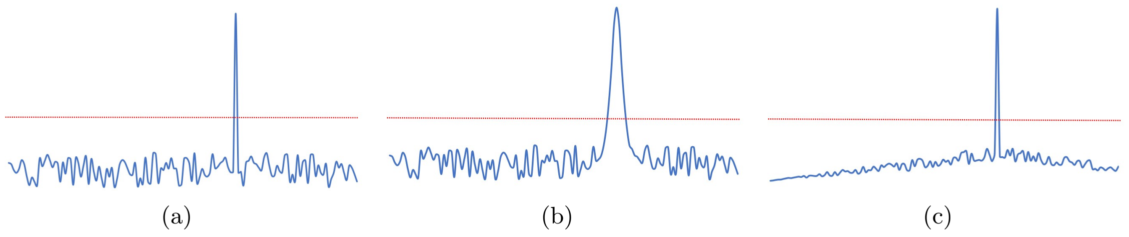


Figure 13: Visual example of a local search trying to find a feasible solution that lies in a small time interval for different scenarios: a non-increasing solution space and a very small feasible time interval (a), a non-increasing solution space and a slightly bigger time interval (b) and an increasing solution space and a very small feasible time interval (c). The x-axis represent the (in)feasible solution space, the y-axis represent the quality of the solution where the solution is feasible above the red dotted line.

Secondly, the local search cannot detect infeasibility and keeps looking for a solution whereas there might be none. Some tools have been created to assess feasibility beforehand without solving the actual problem. However, these tools do not detect infeasibility in general, but they only address the feasibility of small parts of the problem.

A minor limitation of the local search is its difficulty to allocate a shunt train to another shunting yard than proposed at first by the local search. An incoming shunt train and its matched outgoing shunt train should be allocated to the same shunting yard. This could lead to complications if the incoming shunt train is dedicated to a certain shunting yard and the local search chooses a different shunting yard at first for the matched outgoing shunt train.

2.5 Conclusion

NS currently handles the SRP with a local search integrated in a hybrid solution method called HIP, which integrally solves all sub problems of TUSP. Two input files are needed (scenario and location) and one output file is generated containing the matching and detailed time table with routing decisions and service tasks. When solving the problem, HIP needs to take into account the safety norms, whether parking and saw movements are allowed on certain tracks and two rules why specific trains should go to shunting yard "Tuin" and shunting yard "Oostzijde". To make HIP work, three major assumptions have been made: duration estimations (movement time, reversal time, (de)coupling time and walking time), movement assumptions (lead open movements via a platform) and a changeability assumption (through trains are fixed and cannot be altered). The two major limitations of the local search are the difficulty of finding feasible movements if the possible time interval is very small, and the inability to detect infeasibility. A last minor limitation of the local search is its difficulty to switch between shunting yards in the solution space.

3 Literature study

In this section existing literature is collected on three different topics. Firstly, literature on the Train Unit Shunting Problem (TUSP) is collected. Then, literature on the Shunt Routing Problem (SRP) is collected. Next, literature on constraint programming and its relation to Mixed-Integer Programming (MIP) and scheduling problems is collected. Lastly, hybrid methods using constraint programming are examined. This section aims to answer research question 2. In Section 3.4, a conclusion to answer research question 2 is drawn and the scientific contribution of this thesis is summarized.

3.1 Train Unit Shunting Problem

To the best of our knowledge, Winter and Zimmermann (2000) were the first to research something similar to TUSP: the tram-dispatching problem. On arrival of a tram at the shunting yard, it is determined to which stack the tram is allocated, to which upcoming tram schedule the tram is connected and if rearrangements within the shunting yard are necessary to make a feasible schedule. Freling et al. (2005) based their research on Winter and Zimmermann (2000) and focused on trains, thereby introducing the term Train Unit Shunting Problem (TUSP) and they solved the matching and parking problem seperately. Lentink (2006) also included the routing and cleaning of trains. He proposed to solve these subproblems sequentially. Kroon et al. (2008) built a linear programming model based on Lentink and aimed to solve the parking and matching integrally. With the help of Kroon, den Hartog (2010) further improved the model of Kroon et al. (2008) and called it the Arrival on Park Track (APT). R. van den Broek (2016) also took the MIP of Kroon et al. (2008) as starting point and improved the solution by creating a local search that outperformed the MIP of Kroon. One year later, Wolfhagen (2017) tried to capture the TUSP in a MIP again, however the instances were too big to solve to optimality using an exact method, so she used a combination of the exact solution method row generation and the heuristic method tabu search. One of the disadvantages of the local search (R. van den Broek, 2016) was that this approach sometimes was unable to find a simple detour route when the initial route of the train is blocked by another train. Recently, van Cuilenborg (2020) investigated the use of Multi-Agent Pathfinding (MAPF) to overcome this disadvantage.

Whereas literature for the routing problem is extensively collected in the next section (3.2), we present some researches into separate parts of TUSP in this section. Otto and Pesch (2017) investigated the Train-to-Yard Assignment Problem (TYAP), which is a combination of routing to and parking at the shunting yard. They were able to determine lower bounds in a fraction of a second, which makes it interesting in enumeration algorithms. Lin and Kwan (2013) did not include cleaning into the problem, but developed an integrated two-stage approach for matching, routing and parking. Van Hove (2019) approached the problem of routing and parking in her master thesis using Disjoint Paths Approach. The research showed promising results, however computation time exploded for larger instances and resulted in memory limits for CPLEX. A difference between Burggraeve and Vansteenwegen (2017) and van Hove (2019) is that the former treats the routing and timetabling problem as an integral problem thereby aiming to find a robust schedule, whereas most research assumes that routing is a succeeding problem of timetabling.

3.2 Routing of trains

Zwaneveld et al. (1996) was one of the first to research the problem of routing passenger trains through railway stations. In that paper, arrival and departure times are fixed and the objective is to find a set of routes and platforms in a one hour period based on the Node Packing Problem. This is mainly a strategic problem, aimed to prove feasibility of the problem. Also NP completeness is proven as soon as trains have more than 2 alternative paths through a station (Zwaneveld et al., 1996, Kroon et al., 1997). To that end, heuristics for routing were developed, such as a local search algorithm (D'Ariano et al., 2007) and a tabu search algorithm as described in Corman et al. (2010).

Zwaneveld et al. (2001) is a follow-up of Zwaneveld et al. (1996) and improved its model and algorithm in several ways. In particular, they improved the model by including also shunting decisions and preferences of trains for platforms and routes, and they improved the algorithm by extending the preprocessing techniques. The algorithm described in Zwaneveld et al. (1996) was not sufficient for solving the routing problem within the largest Dutch railway stations such as Amsterdam Central Station and Utrecht Central Station. The algorithm presented in Zwaneveld et al. (2001) could handle the routing problem for all railway stations in the Netherlands in 2001 efficiently.

Initial studies on the capacity for routing trains to and from shunt tracks can be found in Egbers (2001), J. van den Broek (2002), and J. van den Broek and Kroon (2007). Egbers (2001) integrated the railway infrastructure into the model and checks for feasibility of routes through and from the shunting yard. J. van den Broek (2002) noted that a conflict can be described when both time intervals and route parts of two trains overlap. He tries to assign preferred routes to all trains by determining the order of routes. However, always routing over preferred routes appeared to be impossible and planners could manually adjust the routes afterwards. J. van den Broek and Kroon (2007) further developed the model of J. van den Broek (2002) and incorporated the manual adjustments of routes by planners into a mathematical model.

In contrast to routing of passenger trains, the routing of freight trains had already been researched in the 80s. Crainic et al. (1984) and Crainic and Rousseau (1986) proposed a non-linear mixed-integer model to determine on macroscopic level the tactical freight train planning and to determine routes as well as frequencies for the transportation requests. This is closely related to routing of passenger trains, however the dynamics of the problem differ in terms of frequency, train compositions and shunting movements.

J. van den Broek (2009) completely focused on the routing between the platform area of a station and the shunting area. He assumes that the capacity and the detailed layout of the shunting area are not relevant. Moreover, shunt trains have a predefined set of possible routes: one priority route and nine alternative routes, thereby reducing the possible routes to 10. Pellegrini et al., 2014 also limited the number of routes by forbidding the use of some railway segments, simulating for example maintenance activities. They compared three different scenarios: fully functioning where all tracks are available, partially disrupted where about 70% of the tracks is available, and severely disrupted where only 40% of the tracks is available. They also researched the difference in modeling granularity. If a station has a fine granularity, a route-lock sectional-release method could best be used. However, if a station has a rough granularity, the route-lock sectional-release method gives a worse performance compared to the route-lock route-release method.

In line with the granularity of the infrastructure, Cappart and Schaus (2017) pointed out that the performance of MIP models for solving scheduling problems is highly dependant on the granularity of time chosen. Pellegrini et al. (2019) succeeded to improve his previous model (Pellegrini et al., 2014) by adding valid inequalities to the MILP model. Caimi et al. (2011) worked with a resource-tree conflict graph model instead of a standard conflict graph formulation. It is based on possible alternative train paths and because of its strong linear relaxation very quick to solve even for large instances. Generating alternative paths is outside the scope of that research. Lusby et al. (2011) showed that the problem can be formulated as a large set-packing problem with a resource-based constraint system; the constraints of the model enforcing the requirement that no trains may simultaneously claim the same piece of junction infrastructure. Although this model may contain significantly many constraints, they developed a branch-and-price framework to solve the model by focusing on a dual representation of any basic feasible solution. They also showed that the proposed model has a tighter LP relaxation than the conventional NPP approaches (Zwaneveld et al., 1996) and is more flexible. Branishtov et al. (2015) suggested a route choice method for a train within the branched track infrastructure of a station. The method bases on a partition algorithm of a station into zones for accelerating the route search procedure. The proposed method possesses a series of advantages, namely, usability and easy numerical implementation, simple computations, as well as fast analytical identification of "bottlenecks" in a railway station network.

Closely related to routing trains between station and shunting yard, is the problem of routing trains within shunting yards. Riezebos and Wezel (2009) implemented a k-best routing algorithm with the possibility for manual adjustments for shunt planners (obligatory tracks or prohibited tracks). In the first stage k-shortest paths are determined from the obligatory tracks to all other tracks. The second stage determines the sequence of visiting the obligatory tracks for any of the k best solutions. Adlbrecht et al. (2015) address this problem using Answer Set Programming (ASP). However, they concluded that ASP can be used to solve complex routing problems, but the algorithm running time is yet too high.

Table 2 gives an overview of the literature mentioned in the last two sections. For each article it shows what problems are treated, what methods are used to solve the problem, what the size of the problem instances are and what the algorithm running time is.

Article	Shunting problem	Solution method	#trains	R	A
Winter 2000	Tram dispatching	MIP and several heuristics	14-46	✓	✓
Freling 2005	Matching, parking	MIP, column generation, dynamic programming	48-68	✓	
Lentink 2006	Routing, parking, matching, cleaning	Shortest path, 2-OPT, Set Partitioning Problem, column generation, A* Search	66-538	✓	
Kroon 2008	Parking, matching	MIP	30-600	✓	
Hartog 2010	Matching, parking, routing	MIP, two-stage heuristic	20-32	✓	✓
Broek 2016	Routing, parking, matching, cleaning	Local search (simulated annealing, tabu search)	2-50	✓	✓
Wolfhagen 2017	Routing, parking, matching	MIP, Row generation, column generation	2-32	✓	
Cuilenborg 2020	Routing, parking, matching	Multi agent pathfinding	1-13	✓	✓
Lin 2013	Matching, routing, parking	Branch-and-price ILP, MILP	–	✓	
Hove 2019	Routing, parking	Disjoint Shortest Path, Successive Shortest Path	10-45	✓	✓
Burggraeve 2017	Routing	MILP	20-85	✓	
Zwaneveld 1996	Routing	Node Packing Problem, branch-and-cut algorithm	3-27	✓	✓
Ariano 2007	Routing	Branch-and-bound algorithm	54	✓	✓
Corman 2010	Routing	Tabu search	3-40	✓	✓
Zwaneveld 2001	Routing	Node Packing Problem, branch-and-cut algorithm	15-79	✓	
Broek 2002	Routing	ILP, Branch-and-bound	1-30	✓	
Broek 2007	Routing	MILP, Branch-and-bound	150-175	✓	
Broek 2009	Routing	MIP, Branch-and-bound	175-200	✓	
Pellegrini 2014	Routing	MILP-based heuristic	15-45	✓	✓
Cappart 2017	Routing	Constraint programming	5-30	✓	✓
Pellegrini 2019	Routing	MILP-based heuristic	12-101	✓	
Caimi 2011	Routing	ILP of Resource-tree conflict graph	12-67	✓	
Lusby 2011	Routing	LP of Set-parcking problem, branch-and-price algorithm, column generation	25-45	✓	
Riezebos 2009	Routing	K-best routing	1		✓
Adlbrecht 2015	Routing	Answer Set Programming	1		✓
Rodriguez 2007	Routing	Constraint programming	6-24	✓	✓
Haahr 2017	Matching, parking	Constraint programming, column generation, greedy heuristic	87-518	✓	✓

Table 2: Summary of literature. The first column shows the article, the second column shows the covered problems, the third column shows the solution method used to solve the problem, the fourth column shows the problem instances used in the articles and the fifth and sixth column denote if a realistic (R) problem instance is used and/or an artificial (A).

3.3 Constraint programming

Mixed-Integer Programming and constraint programming have the same objective: to capture the real-world in a mathematical model and solve it (Hooker and van Hoeve, 2018). However, their background is different and therefore also the solution method is different. MIP is founded in the area of Operations Research and CP is founded in the area of Artificial Intelligence. While MIP is driven by the goal of closing the dual gap, CP is driven by the aim of reducing the domains of the variables until they are reduced to single values. Maravelias and Grossmann (2004) argues that CP is particularly effective for solving feasibility problems and seems to be better than traditional MILP approaches in discrete optimization problems where finding a feasible solution is difficult. The lack of an obvious relaxation, however, makes CP worse for loosely constrained problems, where the focus is on finding the optimal solution among many feasible ones and proving optimality. In this section, CP is treated as well as its applicability to scheduling problems.

Gedik et al. (2018) point out that CP is well known to find feasible solutions in a short computation time due to the use of global constraints to express complex relationships and the use of effective domain filtering. The effectiveness of the CP models have been proven over many combinatorial problems, for example parallel machine scheduling (Edis and Oğuz, 2012), maintenance activities (Nachtmann et al., 2014), team orienteering problem (Gedik et al., 2017), sports scheduling (Trick and Yildiz, 2011) and rostering (He and Qu, 2012).

Cappart and Schaus (2017) point out that Constraint Based Scheduling, or in other words, applying CP on scheduling problems, seems to be a good alternative over MIP. Several works (Kelareva et al., 2012, Kelareva et al., 2014, Ku and Beck, 2016) showed that CP can be used for solving scheduling problems on large and realistic instances. By following this trend, Rodriguez (2007) proposes a CP model for real-time train scheduling at junctions. He shows how the problem of routing and scheduling trains through a junction can be formulated as a joint problem of allocating resources and scheduling activities. CP formulations view the movement of a train through the junction as a job (i.e., a sequence of activities linked by temporal precedence constraints). The movement over a given section can be considered an activity, with a sequence of such activities resulting in a train path. Unlike the NPP, with CP the assignment of train paths amounts to the allocation of a sequence of track sections.

Although multiple CP solvers are available, Cappart and Schaus (2017) saw a gap in the newly formed interval variables of IBM ILOG® CP Optimizer (Laborie and Rogerie, 2008) and improved the work of Rodriguez in several ways. Firstly, they made more use of the strengths of global constraints which can provide a better propagation. Secondly, their search is improved using heuristics and local search techniques. Finally, they include passengers and different categories of trains into the objective. The choice of CP solver is partially based on objective matters such as performance and modelling support, and partially based on subjective matters such as modelling language and integration with other software. Col and Teppan (2019) compared Google's OR Tools and Microsoft's CP Optimizer and concluded that CP Optimizer performs slightly better on classic benchmarks, but performs significantly better on large-scale instances. Other examples of CP solvers are MiniZinc, Chuffed and Gecode.

Several articles name as advantage of CP the type of variables and constraints CP can capture. Zarandi et al. (2020) called this "structured" variable types. One example is activities or interval variables that are appropriate to model the scheduling and sequence problem. Also Kumar et al. (2018) demonstrate the effective modeling of operational constraints using conditional time-interval variables and specialised constraints of CP. This is mainly due to recent improvements in the CP Optimizer. CP Optimizer extends classical CP on integer variables with a few mathematical concepts (intervals, functions) that make it easier to model scheduling problems while providing interesting problem structure for its automatic search algorithm (Laborie et al., 2018). Haahr et al. (2017) compared different optimization methods for parking and matching, a.o. MIP and CP. Classical CP is outperformed, but CP combined with heuristics outperforms most other methods. Also Laborie (2018) makes a comparison between CP and other state-of-the-art approaches (a.o. MIP, CIP, SMT) and shows that the standalone CP model outperforms all other methods due to recent improvements in the CP Optimizer. In the CP Optimizer model, both allocation decisions (presence status of interval variables) and scheduling decisions (interval variables start and end values) hold on the same decision variables that are efficiently pruned by constraint propagation on optional intervals. Another advantage of CP is the ability to capture and solve more realistic instances compared to MIP (Kizilay et al., 2018).

Lastly, recent articles encourage the use of hybrid methods, thereby combining the strengths of both MIP and CP. Rahimian et al. (2017) combined IP and CP methods to a hybrid solution method which outperforms each method alone. Hooker and van Hoeve (2018) researched combinations of constraint programming and techniques from Operations Research. Maravelias and Grossmann (2004) used CP to check the feasibility and create integer cuts for the master problem (MILP). The hybrid method performed two to three times faster than the standalone MILP method. Both Pour et al. (2018) and Dems et al. (2015) used CP to generate feasible initial solutions to feed as warm start to an MIP solver for further improvement. Pour et al. (2018) proved the hybrid method to be significantly superior to the MIP solver whereas Dems et al. (2015) saw possibilities but the results were not as conclusive. All authors experimenting with hybrid methods claim that both methods are not only related, but complementary and, therefore, hybrid methods could be interesting. To the best of our knowledge, hybrid methods involving CP have not been proposed in literature to solve the Shunt Routing Problem as part of TUSP.

3.4 Conclusion

The Train Unit Shunting Problem as well as the Shunt Routing Problem are both extensively researched in the past 25 years. The basis of solving TUSP as well as SRP is Mixed-Integer Programming (MIP). Variants such as Integer Linear Programming (ILP), Mixed-Integer Linear Programming (MILP), Mixed-Integer Non Linear Programming (MINLP) and Linear Programming (LP) are used as well. Some authors use concepts of well-known problems or completely see the TUSP/SRP as a well-known problem, for example, the Set Partitioning Problem, the Shortest Path Problem, the Resource Tree Conflict Graph and the Node Packing Problem. Most of the time, an algorithm is used because of the scale of the MIP.

Branch-and-price or branch-and-cut, in combination with column generation, is a frequently used algorithm. Also Pathfinding is sometimes used, specifically Multi Agent Pathfinding and A*-search. Some other, not so frequently used algorithms, are Tabu Search, Simulated Annealing, 2-OPT and k-best. Constraint programming is not widely used as method to solve the SRP. What is striking is that mostly IBM ILOG CP Optimizer is used when constraint programming is applied to train problems in general. When the SRP is seen as a joint problem of allocating resources and scheduling activities, the scheduling constraints of IBM ILOG CP Optimizer appear to be highly valuable.

When looking at problem size and algorithm running times, the methods differ substantially. Especially MILP combined with a heuristic have been researched extensively and a realistic problem size compared with a reasonable running time tend to be accomplishable. But also constraint programming, though not extensively researched, looks promising in terms of problem size and algorithm running time. Especially using powerful scheduling constraints and feeding CP as warm start for MIPs or local searches seem to be recommended.

The contribution of this thesis to the scientific literature is twofold. Firstly, developing a hybrid method by applying constraint programming to the Shunt Routing Problem and feeding the solution as warm start to solve the Train Unit Shunting Problem using a local search. Secondly, the application of our hybrid solution approach to solve two realistic Train Unit Shunting Problems.

4 Problem description and solution approach

This chapter will answer the third research question: How should the solution approach be developed? This question is answered using the following sub-questions:

1. What is the scope of the solution approach? (Section 4.5)
2. What are the requirements of the solution approach? (Section 4.6)
3. What are the assumptions of the solution approach? (Section 4.7)
4. How can we translate the assumptions and requirements to model constraints? (Section 4.10)

Before the sub-questions of the third research question are treated, the problem is described into detail in Section 4.1. Thereafter, an analysis of possible KPIs to measure the quality of the solution is made (Section 4.2). Next, an example of the problem is given (Section 4.3). After the scope, requirements and assumptions are defined (Section 4.5-4.7), some basic modelling concepts of CP are given in Section 4.8. The flow that the majority of the shunt trains follow within the station is explained (Section 4.9) and using this flow, Section 4.10 explains the CP model of this research. After the model has been explained, Section 4.11 explains how the problem size can be reduced a priori.

4.1 Problem definition

In this section, the sets, parameters, and variables are introduced. In the next section, an example of the Shunt Routing Problem (SRP) is given.

A hub, as explained in Section 1.2, consists of a train station and one or multiple *shunting yards*. A train station consists of multiple platforms $p \in \mathcal{P}$ and multiple infrastructure elements $v \in \mathcal{V}$ (only switches are taken into account, see Section 4.11). In a station there are two sets of trains: through trains $f \in \mathcal{F}$ and shunt trains $n \in \mathcal{N}$. Through trains are trains going through the station, either stopping by (passenger trains) or only driving through (cargo trains). They are fixed in terms of route and time. Shunt trains are trains that either arrive at the station and have to be shunted to the shunting yard, or have to depart from the station after being shunted in the shunting yard. There are multiple entrance gateways to the shunting yards, defined by the set $\mathcal{Z}$. Most of the time shunt trains arrive at a platform transporting passengers and have to be shunted afterwards (called regular shunt trains), but sometimes empty shunt trains without passengers arrive at the side of the station area and can be shunted immediately (called empty stock shunt trains). The possible entrance points at the side of the station area are defined by the set $\mathcal{B}$.

All possible routes from the side of the station area to the station are captured in the set $\mathcal{E}$ and all possible routes from the station to the entrance of the shunting yards are captured in the set $\mathcal{R}$. Routes in the sets $\mathcal{E}$ and $\mathcal{R}$ are characterised by a starting point , an endpoint and the switches $v \in \mathcal{V}$ that are used to get from the starting point to the endpoint. The starting point for set $\mathcal{E}$ are tracks at the side of the station area $b \in \mathcal{B}$ and the starting point for set $\mathcal{R}$ is a platform $p \in \mathcal{P}$. The endpoint for set $\mathcal{E}$ is a platform $p \in \mathcal{P}$ and the endpoint for set $\mathcal{R}$ is an entrance gateway $z \in \mathcal{Z}$ to a shunting yard.

The SRP aims to find a route and a time interval for every shunt train to the shunting yard and vice-versa, which prevents crossing of trains and makes sure that every train can depart at the right time. Note that there should be enough time between entering the shunting yard and leaving the shunting yard to make movements and to receive service in the shunting yard (e.g., inspection, cleaning, maintenance). We can formally define the decision variable x_{1ntr} as the interval t at which shunt train n takes route r from the station to the shunting yard and the decision variable x_{2ntr} as the interval t at which shunt train n takes route r from the shunting yard back to the station. Other decision variables are y_{1ntp} which defines the time interval t shunt train n is parked on platform p after arriving at the station and decision variables are y_{2ntp} which defines the time interval t shunt train n is parked on platform p before departing from the station. In case of an empty stock shunt train, there is an additional decision variable z_{nte} which defines the time interval t at which shunt train n takes route route e to get from the side of the station area to the station itself. All mentioned variables are optional variables (see Section 4.8.1 for a detailed explanation) which means they both take a boolean value for their existence in the solution and an interval value (start and end time) if the variable exists in the solution.

4.2 Quality measurement

In this section, the KPIs to measure the quality of the solution from HIP are derived. As described in Section 1.4, NS wants to reduce the running time of HIP. So the first measurement is running time. This includes both the running time of the CP model and running time of HIP. However, a short running time does not guarantee a good-quality solution. Therefore, a quality measurement is chosen as well. There are 11 feasibility constraints which need to be complied with to obtain a feasible plan in HIP. An overview of these constraints is given in Table 3. Besides making a feasible plan, HIP aims to minimise the total number of movements the shunt trains make. This is referred to as *shunt moves*. The number of shunt moves is directly related to the number of drivers needed and therefore directly impacts the personnel expenses. This KPI will be chosen as the quality measurements, together with the measurement of running time.

Feasibility constraints	**Explanation**
Crossings	The number of trains crossing each other.
ArrivalDelays	The number of shunt trains having a delayed arrival.
DepartureDelays	The number of shunt trains having a delayed departure.
TrackLengthViolations	The number of trains parking/reversing on a track that is shorter than the train length.
CombinesOnDeparture	The number of incorrect combinations, for example a train should depart in the sequence X-Y but departs in the sequence Y-X.
ServiceTimeViolation	The number of trains not complying with their required service time.
DelayedReservedMovements	The number of through trains that are not scheduled on the predetermined time.
NonElectrifiedTrackUsages	The number of trains a shunt train requiring electrification uses a non-electrified track.
GettingOffForReversalViolations	The number of times a driver needs to get off the train for reversing on a track where that is not allowed.
IllegalParkingTime	The number of times a train parks on a track where it is not allowed to park on.
ActivityOutsideDriversShifts	The number of times a driver is scheduled outside working hours of its shift.

Table 3: Feasibility constraints for HIP. All feasibility constraints should be 0 to obtain a feasible plan.

4.3 Problem example

To illustrate the SRP described in the previous section, a toy problem which contains sets $\mathcal{F}$ and $\mathcal{N}$ is given in Table 4. In the table, the arrival time and end time are provided as well as the necessary infrastructure. Finally, the operation of the shunt train is provided in the column "Operation". This way, it can be observed that train F1 (the first through train) makes its first movement towards platform 2 at 8:00 and arrives at 8:02 at platform 2. At 8:05 it departs from platform 2 and that movement ends at 8:07. The first shunt train (N1) arrives at 8:07 at platform 2 and after being shunted it should depart at 10:16 from platform 1. Figure 14 gives an overview of the artificial location and all corresponding shunt routes $\mathcal{R}$ between the two platforms and the two entrance gateways are given in Table 5. Table 4 and 5 together are input for the model. There are two platforms at the artificial location, so $\mathcal{P} = \{1, 2\}$. Nodes 13 and 14 are the entrance gateways to the shunting yard, so $\mathcal{Z} = \{13, 14\}$. Lastly, in this artificial example, the shunt train (N1) has three properties:

1. The train should spend at least 2 hours in the shunting yard. We assume that this includes movement time, reversal time and service time in the shunting yard.
2. The train has a movement time of 2 minutes.
3. The train has a reversal time of 3 minutes.

The SRP of this artificial example can be described as the allocation of a route and time from platform 2 to the shunting yard and a route and a time from the shunting yard to platform 1 for shunt train N1. The configuration of routes and times should take into account that N1 stays at least 2 hours in the shunting yard, that the movement and reversal time are respected and that it does not include crossings between trains. As shunt train N1 arrives at platform 2, N1 can either take route R3 or route R4 to the shunting yard, since they depart from platform 2 instead of platform 1. As N1 should depart from platform 1, either route R1 or R2 can be chosen to go from the shunting yard back to the station. This results in the following four possible configurations: [R3, R1], [R3, R2], [R4, R2] and [R4, R1]. On the next page, the four possible solutions are computed and checked on feasibility.

Train	**Start time**	**End time**	**Used infrastructure**	**Operation**
F1	08:00	08:02	4-8-2	Arrive at platform 2
F1	08:02	08:05	2	Park on platform 2
F1	08:05	08:07	2-10-6	Depart from platform 2
N1	08:05	08:07	4-8-2	Arrive at platform 2
N1	08:07	08:09	2	Park at platform 2
F2	08:10	08:12	5-9-1	Arrive at platform 1
F3	08:11	08:13	11-12	Cargo
F2	08:12	08:13	1	Park on platform 1
F2	08:13	08:15	1-7-3	Depart from platform 1
F4	10:11	10:13	6-10-2	Arrive at platform 2
F4	10:13	10:20	2	Park at platform 2
F5	10:13	10:15	11-12	Cargo
N1	10:16	10:18	1-7-3	Depart from platform 1
F4	10:20	10:22	2-8-4	Depart from platform 2

Table 4: Artificial scenario of the SRP. The column *Train* describes the type of train (through train F or shunt train N) and the number.

Route	**Starting point**	**Endpoint**	**Used infrastructure**
R1	1	13	1-7-8-11-13
R2	1	14	1-9-10-12-14
R3	2	13	2-8-11-13
R4	2	14	2-10-12-14

Table 5: Possible shunt routes for the location as described in Figure 14.

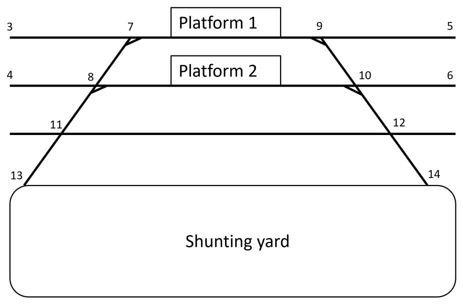


Figure 14: Artificial location for the example of the SRP. The numbers denote the different infrastructure elements and the box at the bottom of the figure denotes the shunting yard. The exact configuration of the shunting yard is not important for the SRP.

R3 and R1. In this solution configuration, the route to the shunting yard will be R3 and the route from the shunting yard back to the station will be R1. Shunt train N1 arrives from the left side of platform 2, so a route departing also at the left side of platform 2 (R3) has to include a reversal on platform 2. Since a reversal for N1 takes 3 minutes, the train should be parked on platform 2 in the interval [8:07,8:10] instead of [8:07,8:09]. A movement of N1 takes 2 minutes and through train F3 uses 11 and 12 in the interval [8:11,8:13] which means that N1 can only take the route to the shunting yard after 8:13. Let us assume N1 takes route R3 on the interval [8:13,8:15] and arrives at 8:15 at the shunting yard. As N1 should be at least 2 hours in the shunting yard, the earliest time to start route R1 is 10:15. Through trains F4 and F5 do not occupy route R1 in the interval [10:15,10:17], so that interval is possible for N1. As N1 has to depart from platform 1 in the left direction and R1 arrives at platform 1 from the left direction as well, a reversal is needed again which takes 3 minutes and will be executed in the interval [10:17,10:20]. However, this solution configuration is not feasible, because N1 should depart from platform 1 at 10:16 and the solution configuration proposes a departure at 10:20.

R3 and R2. In this solution configuration, the route to the shunting yard will be R3 and the route from the shunting yard back to the station will be R2. Using the same logic as in the previous paragraph, N1 will arrive at the shunting yard at 8:15 using route R3 and the earliest start time of R2 is 10:15. Through trains F4 and F5 do not occupy route R2 in the interval [10:15,10:17], so that interval is possible for N1. As N1 has to depart from platform 1 in the left direction and R2 arrives at platform 1 from the right direction, a reversal is not needed and N1 can depart from platform 1 at 10:17. However, this solution configuration is not feasible, because N1 should depart from platform 1 at 10:16 and the solution configuration proposes a departure at 10:17.

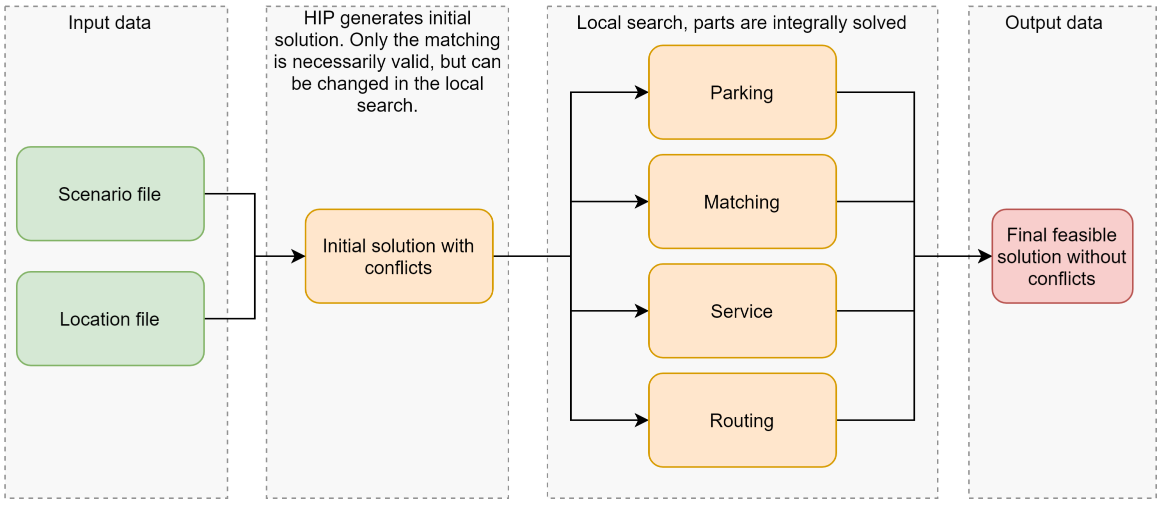
R4 and R2. In this solution configuration, the route to the shunting yard will be R4 and the route from the shunting yard back to the station will be R2. As shunt train N1 arrives from the left side of platform 2 and route R4 departs from the right side of platform 2, a reversal is not needed and N1 can take route R4 in the interval [8:09,8:11]. Taking route R4 one minute later is not possible because that would imply a crossing between N1 and F3 on node 12. N1 arrives in the shunting yard at 8:11 which means that the earliest start of route R2 back to the station is 10:11. However, train F4 blocks route R2 in the interval [10:11,10:13] and train F5 blocks route R2 in the interval [10:13,10:15]. The earliest possible interval for R2 is therefore [10:15,10:17]. As N1 has to depart from platform 1 in the left direction and R2 arrives at platform 1 from the right direction, a reversal is not needed and N1 can depart from platform 1 at 10:17. However, this solution configuration is not feasible, because N1 should depart from platform 1 at 10:16 and the solution configuration proposes a departure at 10:17.

R4 and R1. In this solution configuration, the route to the shunting yard will be R4 and the route from the shunting yard back to the station will be R1. As shunt train N1 arrives from the left side of platform 2 and route R4 departs from the right side of platform 2, a reversal is not needed and N1 can take route R4 in the interval [8:09,8:11]. N1 arrives in the shunting yard at 8:11 which means that the earliest start of route R2 back to the station is 10:11. This time, the interval [10:11,10:13] for route R1 is not blocked by any train so N1 can take route R1 in the interval [10:11,10:13]. Note that deviating from this interval is not possible because of train F5 occupying the route in the interval [10:13,10:15]. As N1 has to depart from platform 1 in the left direction and R1 arrives at platform 1 from the left direction as well, a reversal is needed again which takes 3 minutes and will be executed in the interval [10:13,10:16] and N1 can depart from platform 1 at 10:16. This is a feasible solution.

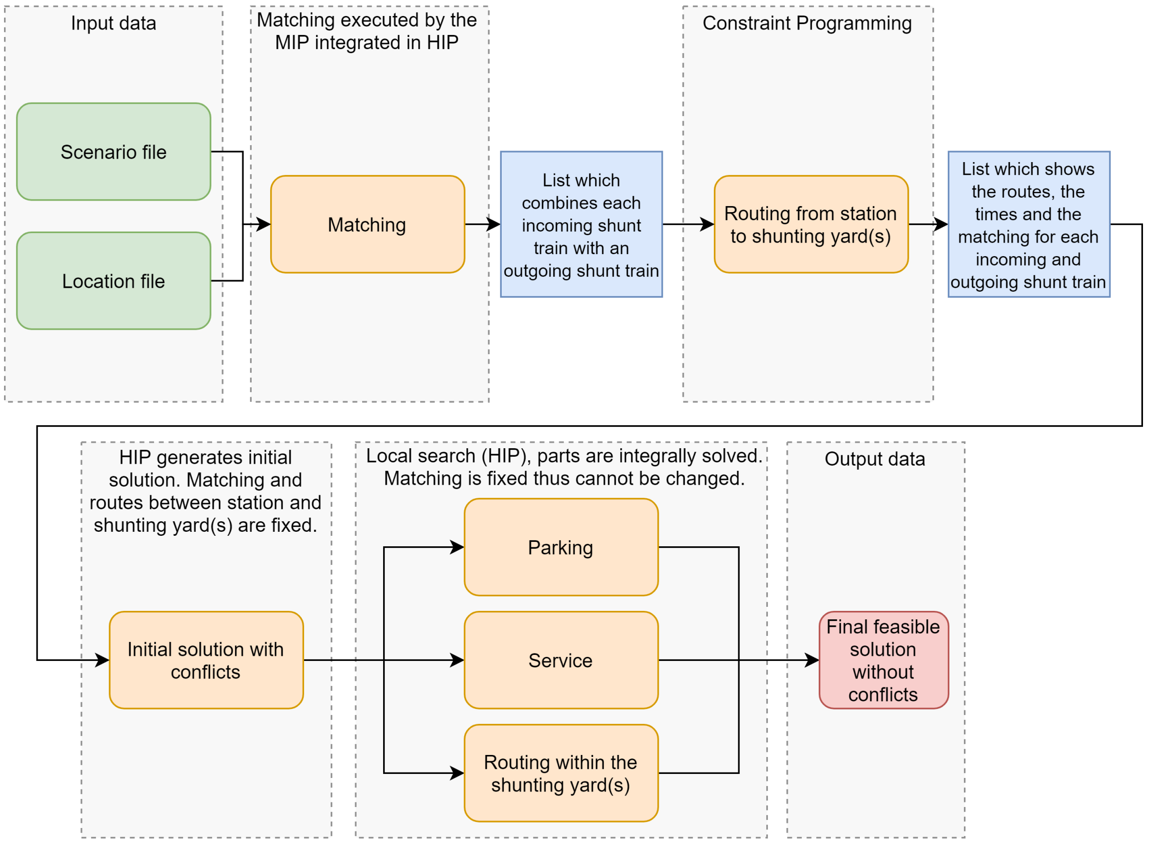
This toy example makes the problem tangible and shows how difficult it is to find an appropriate route through the train traffic. In realistic cases there are multiple shunt trains which also influence each other, making the problem even more complex.

4.4 Solution approach

In this section the solution approach is worked out. As mentioned in Section 1.4, the objective is to create a constraint programming model that determines the routes between the station and shunting yard. The output of this model will be input for the local search which can subsequently solve the parking, servicing and routing problem within the shunting yards. Figure 15a shows the current solution approach HIP conducts and Figure 15b shows the flow chart of the proposed solution, where CP functions as a warm start for the local search in HIP. As the figure points out, currently an initial (infeasible) solution is generated and the local search integrally solves the problems of parking, matching, servicing and routing. In the proposed solution approach, first the matching is solved, which is input for the CP model. Next, the CP model solves the SRP between the station and the shunting yard. This results in a list that contains a starting time, an endtime, the matching and the routes for every shunt train unit. Using this input data, the local search creates an (infeasible) initial solution. Lastly, the local search integrally solves the three problems of parking, servicing and routing within the shunting yards. More practically, the routes found in the CP model will be fixed routes in the input for HIP. This means that HIP might not only profit from a warm start, but also from a reduced search space. The downside is that HIP might not be able to find a feasible solution due to the a possibly too tight search space.


Input data
Scenario file
Location file
HIP generates initial solution. Only the matching is necessarily valid, but can be changed in the local search.
Initial solution with conflicts
Local search, parts are integrally solved
Parking
Matching
Service
Routing
Output data
Final feasible solution without conflicts

(a) Current solution approach


Input data
Scenario file
Location file
Matching executed by the MIP integrated in HIP
Matching
List which combines each incoming shunt train with an outgoing shunt train
Constraint Programming
Routing from station to shunting yard(s)
List which shows the routes, the times and the matching for each incoming and outgoing shunt train
HIP generates initial solution. Matching and routes between station and shunting yard(s) are fixed.
Initial solution with conflicts
Local search (HIP), parts are integrally solved. Matching is fixed thus cannot be changed.
Parking
Service
Routing within the shunting yard(s)
Output data
Final feasible solution without conflicts

(b) Proposed solution approach

Figure 15: Flow chart of difference between the current solution approach of HIP (a) and the proposed solution approach where CP functions as warm start for HIP (b). Green boxes mark the input, orange boxes mark a process, blue boxes mark what is transferred between the processes and red boxes mark the output.

4.5 Scope of the model

The SRP, as described in Section 4.1, will be translated into a model (see Section 4.10). In doing so, the scope of the model is defined in this section and in the next two sections the requirements and the assumptions of the model are covered.

Firstly, the scope of the model only contains the station area and the area between station and shunting yard. The areas within the shunting yards and outside the station area are not part of this research. Next, the scope of the model is limited to routing of shunt trains, i.e., allocating a route and a time window for each shunt train. Finding routes is not within the scope of the model, but allocating shunt trains to routes is. While allocating trains to routes, through trains should be taken into account and the model should guarantee to have no physical overlap between any train. Note that both routes and departure times are fixed for through trains. For this research, the station Eindhoven is considered. This means that the model is not completely generic for every station in the Netherlands, but most concepts are. Appendix G shows how the model can be made generalised over all stations. Lastly, the model will consider a 24-hour planning horizon.

4.6 Requirements of the model

In this section, all requirements to the model are defined. Since the output of CP will be used as input for HIP, the requirements of the CP model has to comply with the requirements of HIP. Most requirements are therefore corresponding with and extracted from the requirements of HIP (see Section 2.4.2).

1. The biggest requirement of the model is to make a feasible schedule. That means firstly that trains never cross while driving, while parking and on platforms. It also means that all trains are planned and that the planning does not contain idle times which implies jumping in time or place. For example, a train cannot enter shunting yard Oostzijde and leave shunting yard Tuin. Next, the track lengths should be respected, which means that some trains cannot be allocated to short tracks. As the scope of the model is to plan a 24-hour period, all movements should also be planned within the 24-hour period. There is an exception for trains that arrive just before the end of the 24-hour horizon. These trains are allowed to end in the middle of their movement.

2. The second requirement is that all constraints on time windows are hard constraints. For example, a train cannot make a saw movement in a time window that is shorter than its reversal time. Also arrival, departure, service and movement times should be respected and cannot be deviated from. Altogether, this also means that there should be enough time between entering the shunting yard and leaving the shunting yard for service and (saw) movements. Otherwise, when HIP tries to solve the problem with this models' output, it will never find a feasible solution. Note that walking times are not part of the input, since personnel planning is not part of this research.

3. The third requirement is that trains only stop and make saw movements at the tracks that allow so (see Figure 12).

4. The fourth requirement is that train units that are combined to form a longer train, take exactly the same route. Otherwise, the train would not operate as one in real life.

5. The fifth requirement is that the model uses a route-lock route-release method. This means that a train reserves all switches when starting a route and releases all used switches after completing the route. The NS is thinking to adapt in the future a route-lock sectional-release method, where trains reserve all switches at the start of the route, but release a part of the route when the train has passed that part. However, for the time being, the route-lock route-release method is in line with current processes at NS and is therefore a requirement to the model.

6. The sixth requirement is regarding *empty stock* trains (shunt trains that do not carry passengers and do not arrive at a platform but at an open line at the border of the station area). As these ingoing empty stock trains arrive on the main railway network, routing towards the station should directly start at arrival as the train cannot wait at the border of the station. The same applies to outgoing empty stock trains. The routing of these trains should end exactly at the time at which the trains should leave the border of the station area.

7. The seventh requirement is specifically focused on Eindhoven and specifies that trains of type ICR Gvc-Ehv 9 should always be routed to shunting yard Tuin (see Section 2.4.1).

8. The eighth requirement is that there should be enough time between trains that are shunted subsequently to Oostzijde via gateway Z5. This is a requirement which makes it easier for HIP to solve the routing and parking problem within shunting yard Oostzijde.

9. The ninth requirement is about the model running time. Currently, HIP needs approximately 16 hours to solve a 24-hour period at Eindhoven. As the model of this research will be made to aid HIP, the running time of the model combined with the running time of HIP should be less than 16 hours. For the model specifically, the running time should then be far less than 16 hours. For example, if the model has a running time of 2 minutes and the output of the model helps HIP to achieve a running time of 15 hours, this can be seen as an improvement. However, if the running time of the model is 3 hours and HIP achieves a running time of 14 hours, the model does not successfully aid HIP.

10. Lastly, a requirement not necessarily of the model, but for the model, is that the pre-generated shunt routes should end at the shunting yard gateways. In that way, the scope of the model is limited to the routing between station and shunting yard and does not contain the shunting yards.

4.7 Model assumptions

This section explains the assumptions that are made to develop the CP model.

1. Matching of ingoing shunt trains to outgoing shunt trains is not within the scope of this research. Therefore, the first assumption of the model is that matching has already been done and is implemented in the input data. An assumption to the matching is that the matching is chosen in such a way that the service time is abundantly respected, which guarantees that the input data is not infeasible.

2. Just as matching, parking and routing within the shunting yards is not part of this research. Therefore we assume infinite capacity of both shunting yards.

3. Another assumption is that each train has a finite number of routes to choose from. As mentioned previously, these routes are pre-defined and can be found in Appendix B.

4. There are two related assumptions regarding through trains. First, the through trains are fixed, thus they cannot be changed by the model, and second, the through trains mutually do not cause errors in the model. That means that through trains should never occupy the same switches at the same time. Since that is not possible according to the model and since through trains are fixed, the model would immediately give the status *infeasible*.

5. There are three assumptions regarding empty stock movements.
   (a) The first assumption is that all empty stock movements from the left side of the station are directed to a platform, from where they can continue their movement towards the shunting yard. The reason why these trains are not forced to go to the shunting yard directly, is that we expect it to tighten the planning too much which makes it infeasible. By giving the choice of a platform, the possible solution space is broadened. Also, the possibility to wait at a platform for an opportunity to go to the shunting yard broadens the solution space.
   (b) The second assumption is that empty stock movements from the right side are directed to Oostzijde and only use switch 42. They are not directed via a platform. The reason behind this assumption is that most trains in practice take this route and by entering Oostzijde through the backdoor, the tight search space is relieved.
   (c) The last empty stock assumption is that the empty stock movements coming from the right side do not interfere with through trains at the right side.

6. The next assumption is regarding movement duration. Just as HIP (see Section 2.4.3, we assume that each train has a movement duration of two minutes. The safety norms as required by ProRail (Section 2.4.1) will not be within the scope of this project, as the local search of NS also does not incorporate these safety norms yet. Also, NS is researching a new safety policy which is less complicated than the current safety policy and has more similarities to the current approach in the local search.

7. Another assumption about movement is that we assume a maximum of two movements between station and shunting yard. The number of movements is fixed depending on the route. For example, if a train goes directly to Tuin, only one movement is made. Though a movement via reversal track 40 requires two movements.

8. The last assumption is about coupling and decoupling of train units and is coherent with the fourth requirement in the previous section. Since all routes between station and shunting yard of coupled train units should be the same, we assume that coupling and decoupling takes place in the shunting yards. For example, if two train units separately enter the a shunting yard and should later depart from a certain platform as one train, there are two choices to do so: either the train units are sent separately to the platform where the train units are coupled, or the train units are coupled in the shunting yard and sent together to the platform. The latter is consistent with the fourth requirement as all routes between station and shunting yard of coupled train units should be the same.

4.8 Preliminaries of constraint programming

Before the model itself is explained (Section 4.10), first some basic (scheduling) concepts of constraint programming in CP Optimizer are explained in this section. Two types of variables and three types of constraints are explained. These concepts are also used in the proposed model (see Appendix E).

4.8.1 Interval variables

An *interval variable* is a local variable that represents an interval in time in which a particular property holds. For example, the interval a machine is occupied, the interval personnel is scheduled or the interval an activity is scheduled. Interval variables are defined by a lower bound, an upper bound and the size of the interval. The size of the interval can be either fixed or variable. Figure 16 gives a visual representation of an interval variable with a variable size between 2 and 4 that should lie between 0 and 10. Possible domains of this interval variable are [0,2), [0,3), [0,4), [1,3), ..., [6,10). An additional property of interval variables is *optionality*; an interval variable can be optional. In case of an optional interval variable, this variable does not need to be present in the solution. In other words, it is part of the decisions of the problem to decide whether the interval will be present or absent in the

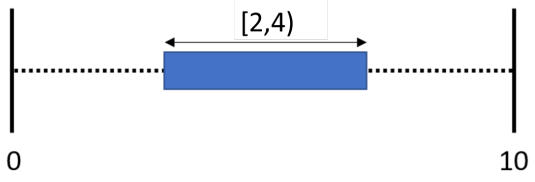


Figure 16: Example of an interval variable. Lower bound = 0, upper bound = 10, variable size = [2,4)

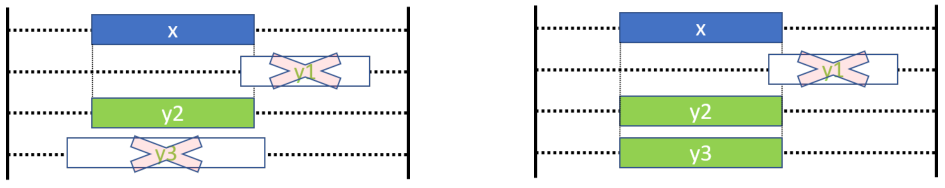
solution using constraints. This is a very powerful concept when for scheduling problems on multiple resources.

4.8.2 Sequence variables

The second variable is the *sequence variable*. This global variable contains a set of interval variables. For example, a sequence variable could be all activities a certain worker should do during a work shift. Or all activities that are scheduled on a certain machine. The sequence itself does not imply temporal ordering. Several constraints can be applied to a sequence to ensure temporal ordering. For example the constraint *prev(p,x,y)*, which means that - if interval variables x and y are present in the solution - interval variable x appears right before interval variable y in sequence p.

4.8.3 Alternative constraint

The *alternative*-constraint is a global constraint which ensures that, given an initial interval variable and a set of interval variables, one alternative of a set of alternative interval variables is chosen and it is synchronised with the initial interval variable. For example, given an interval variable for a job and a set of intervals for machines where this job can be executed on, the constraint will select one interval variable on a certain machine and synchronises the start time and end time of the job interval and machine interval. An additional parameter can be given to the alternative constraint, which is the cardinality. This number constrains how many alternatives should be chosen. When cardinality is not given, one alternative is chosen.



(a) Cardinality = 1 (Default) (b) Cardinality = 2

Figure 17: Example of *alternative(x,[y1..y3])* and *alternative(x,[y1..y3],2)*. Interval x is given and in the first picture y2 is chosen and synchronised with x, and in the second picture y2 and y3 are chosen and synchronised with x.

4.8.4 StartAtEnd constraint

The local *StartAtEnd*-constraint ensures a direct temporal relation between two intervals. *StartAtEnd(x,y)* means that, if both interval variables are present, interval variable x has to start immediately after interval y has ended. There is an optional input parameter *delay*, which denotes the time between the end of interval y and the start of interval x.

StartAtEnd(x,y,10) means that, if both intervals are present, interval variable x has to start 10 time steps after interval y has ended. The delay can also be negative.

4.8.5 NoOverlap constraint

The *NoOverlap*-constraint is also a global constraint. This constraint has a sequence variable as input and constrains all present interval variables in the sequence not to overlap in time. An example where this could be useful is to put a NoOverlap constraint over a sequence of all job interval variables executed on a certain machine. An optional input parameter is the transition matrix. If all interval variables in the sequence are given a certain integer type, the transition matrix denotes the minimum time between each pair of interval variables in the sequence. This is comparable with the *delay* parameter in the StartAtEnd constraint. Figure 18 shows an example of a NoOverlap constraint on a sequence with and without a transition matrix depending on the type of interval (T1 or T2).

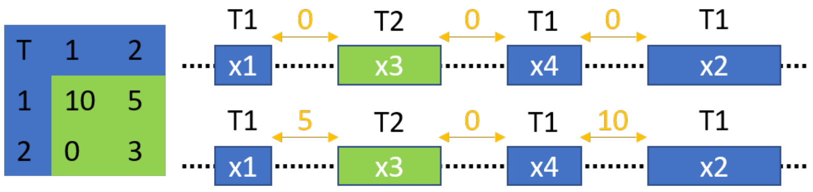


Figure 18: Example of a NoOverlap with and without transition matrix (left). The upper part shows a NoOverlap without a transition matrix and the lower part shows a NoOverlap with transition matrix. Note that the inter-intervals times (orange) are minimal times.

4.8.6 Synchronise constraint

The last constraint is the *Synchronise*-constraint. This constraint between an interval variable a and a set of interval variables $\{b_1...b_n\}$ makes all present intervals $\{b_1...b_n\}$ start and end together with interval variable a. If interval variable a is not present in the solution, this constraint is automatically satisfied and no synchronization between a and $\{b_1...b_n\}$ occurs.

4.9 Flow of a shunt train

In this section, the flow of a shunt train is described, as visualised in Figure 19. As explained in Section 4.1, a shunt train can either arrive at a platform (regular shunt train) or at the side of the station area (empty stock shunt train). In case of the latter, first a route from the side of the station area to a platform within the station should be chosen (*Vrijebaan1*). This variable is an interval variable that indicates both the chosen route and the interval in which the route is taken. When the train arrives at a platform, the interval variable *Perroninterval1* indicates at which time interval the train stays at a certain platform. For example, the model can determine that a train should stay at the platform 4 minutes longer than initially proposed, because then there will be a route to the shunting yard without other trains blocking the route. The chosen route from the station to the shunting yard is captured in the variable *Route1*. This interval variable indicates both the chosen route and the interval in which the route is taken. Some routes does not go directly to the shunting yard, but a reversal at reversal track 40 is needed. If the chosen route goes to the shunting yard via reversal track 40, then the model should determine how long the train stays at reversal track 40 (interval variable *k40interval1*) and at what subsequent interval the train moves from reversal track 40 to the shunting yard (interval variable *k40toSB*).

Depending on the service time, the entrance gateway to the shunting yard and the exit gateway from the shunting yard, the minimum time the shunt train should be in the shunting yard is calculated. Figure 20 gives a break-down of the calculation of this minimum time. The leaving process of a shunt train is the same as the arrival process, but then the other way around. So first a route and a corresponding time interval is chosen in which the train moves from the shunting yard to a platform at the station (interval variable *Route2*). If this route goes via reversal track 40, the train should first go from the shunting yard to reversal track 40 (interval variable *SBtok40*), then it should be determined how long the train stays at reversal track 40 (interval variable *k40interval2*) and then the train can continue its route to the platform. Arrived at the platform, the model has to determine how long the train stays at the platform (interval variable *Perroninterval2*). A train cannot stay at the platform after it should have been departed, but it can arrive at the platform earlier if that implies no crossings with other trains. Lastly, if a train should depart from the platform, the flow ends at the platform. If the train should depart from the side of the station area (empty stock shunt trains), the appropriate route and corresponding time interval to the side of the station area should also be determined (interval variable *Vrijebaan2*).

The model is based on this flow. Next, constraints are used to guarantee the flow, to prevent crossings and to reserve enough time in the shunting yards.

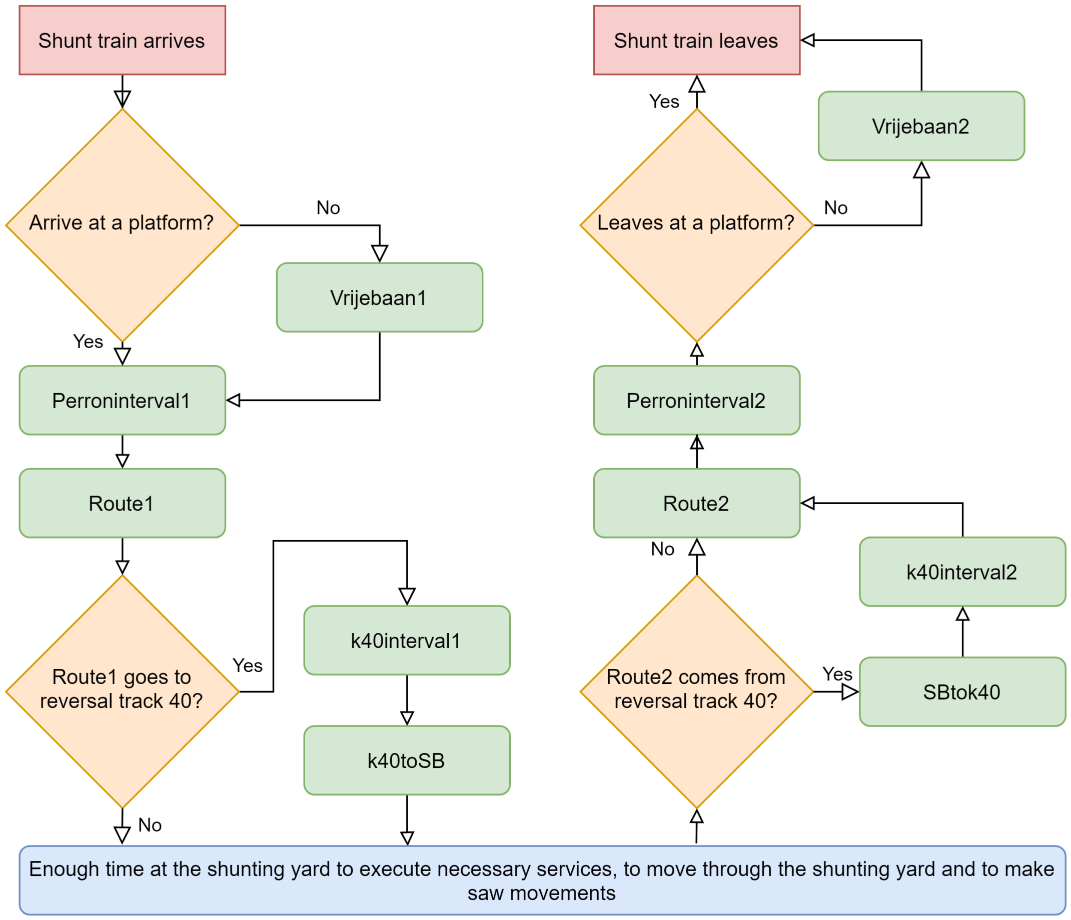


Figure 19: Flow chart of the relation between the model variables, and an overview of the shunt routing sequence in Eindhoven

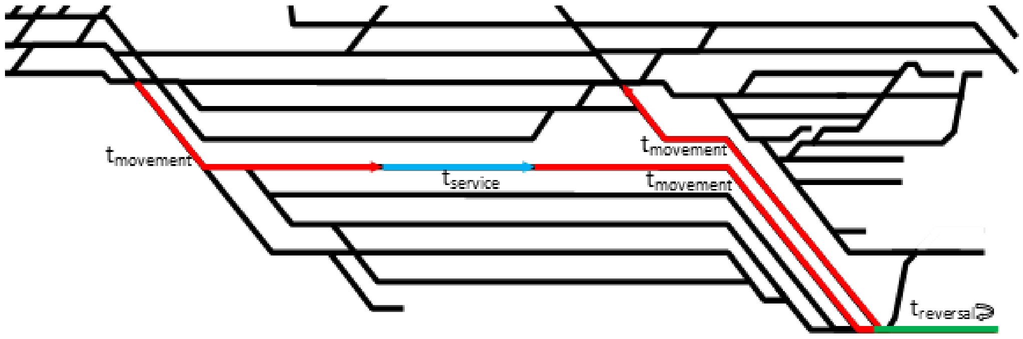


Figure 20: An example of the break-down of the minimal time a shunt train should be in Oostzijde: $t_{movement} + t_{service} + t_{movement} + t_{reversal} + t_{movement}$

4.10 Constraint programming model

In this section the model is explained. The model is made based on the process of a shunt train from arrival until departure as described in the previous section. In this section, an overview of all sets, parameters and variables is given (Section 4.10.1) and then based on the relation between the variables and the requirements to the variables, the constraints are explained (Section 4.10.2). The global objective of the model is to find a feasible schedule. The requirements to achieve a feasible schedule can be found in Section 4.6. Next, Section 5.3 describes what experiments will be conducted and what corresponding objective is given as input for the model. In this section only a brief overview of all used constraints is given. For an overview of all constraints and a detailed explanation per constraint, see Appendix E.

4.10.1 Sets, parameters and variables

The model makes use of seven different sets: $\mathcal{N}$, $\mathcal{T}$, $\mathcal{F}$, $\mathcal{V}$, $\mathcal{P}$, $\mathcal{R}$ and $\mathcal{E}$. The sets are described in Table 6.

Set name	**Description**
$\mathcal{N}$	All shunt train units. Matching is included, i.e., every element in N an incoming shunt train unit and its matched outgoing shunt train unit.
$\mathcal{T}$	Fixed parking movements of all through trains.
$\mathcal{F}$	Fixed train movements of all through trains. All elements in F contain either an in-going movement or an outgoing movement, thus set F is twice as big as set T.
$\mathcal{V}$	All switches at the station area.
$\mathcal{P}$	All platforms at the station.
$\mathcal{R}$	All possible shunt routes between station and shunting yard, from each platform to each possible shunting yard gateway.
$\mathcal{E}$	All *open line* entries/exits at the border of the station area

Table 6: Explanation of all sets in the model

As there are many parameters and variables, just an overview of all used parameters and variables is given in this section (see Table 7). Appendix C gives a full description of the parameters of the model and Appendix D gives a full description of the interval variables used in the model.

Lastly, there are three sequence variables, as described in Table 8.

4.10.2 Constraints

This section describes the sets of constraints created for the CP model.

1. To ensure the routing as described in the flowchart of Figure 19, the first set of constraints (1 - 30 in Appendix E) is made. They make sure that each train is routed according to the flowchart, without idle time between activities and respecting all times, such as arrival time and reversal time.

Parameters		**Variables**	
lengths	movementduration	fixedmovement	fixedperron
atob	reversaltimes	perroninterval1	perroninterval2
standing	fixedtuin	movement1	movement2
parent	mintimes	route1	route2
fixedperron	vrijebaan	movementVB1	movementVB2
fixednodes	times	routeVB1	routeVB2
fixedstarttime	fixedendtime	k40interval1	k40interval2
fixedparkingstart	fixedparkingperron	k40toSB	SBtok40
fixedparkingend	ShuntRoutes		
VBRoutes	routenodesKtoSB		

Table 7: Overview of all sets, parameters, variables and sequence variables used in the model

Sequence name	**Sequence variable for..**	**Set**
nodesequence	All intervals in which trains use a certain switch v	∀V
perronsequence	All intervals in which trains use a certain platform p	∀P
k40sequence	All intervals in which trains use reversal track 40	-

Table 8: Explanation of all sequence variables

2. The second set of constraints (31 - 40 in Appendix E) make sure that all combined train units follow exactly the same path.

3. The third set of constraints (41 in Appendix E) makes sure that there is enough time in the shunting yard reserved for each train unit. This time should include the service time (if necessary), the time needed to make movements within the shunting yard and the reversal times. See Figure 20 for an example of the break-down of the minimal time a shunt train should be in Oostzijde.

4. The fourth set of constraints (42 in Appendix E) makes sure that matched train units enter and leave the same shunting yard. It is not possible that an incoming shunt train enters for example Oostzijde and its matched outgoing shunt train leaves from Tuin.

5. The fifth set of constraints (43-46 in Appendix E) makes sure that no track violation occur at reversal track 40 (maximal length of 366 meters) and at a reversal track in Oostzijde (maximal length of 283 meters).

6. The last set of constraints (47-50 in Appendix E) are the global constraints that allow no overlap in time windows for each switch, each platform and the reversal track 40. In other words, these constraints prevent crossing of trains for infrastructure element.

The alternative constraint below gives an example of how the constraints are set up. For the complete set of constraints, we refer to Appendix E.

$$alternative(movement1_{n,p}, [route1_{n,r} \;\; \forall R \textit{ if route r comes from platform p}]) \quad \forall N, \forall P$$

This constraint ensures that for every shunt train n making a movement to the shunting yard ($movement1_{n,p}$), a route r ($route1_{n,r}$) is chosen from the possible set of routes. The possible set of routes is limited to routes starting at platform p. Note that the alternative-constraint only works if $movement1_{n,p}$ is present in the solution. As $movement1_{n,p}$ is only present if the shunt train arrives at platform p, a route from the correct platform to the shunting yard is chosen by the model.

4.11 Problem size reduction

We describe several preprocessing techniques to reduce the problem size a priori: route dominance, node removal and track removal. The first is route dominance. In most of the cases there are multiple routes from point A to point B. But there are routes that reserve certain paths through the station and other routes also reserve these paths but also more paths. We define A as a dominated route over B if the set of blocked paths of B is a subset of the set of blocked paths of A. Figure 21 gives an example of a dominated route. The orange route has two routing possibilities in the middle; green and red. Green only blocks paths between entrance 4 on the left and the platforms on the right, while the red route also blocks paths between entrance 3 on the left and all platforms on the right. We say that route red is dominated by route green. If a route is dominated by another route, the dominated route will not be used as input for the model. This prevents unnecessary expansion of the search space. Another preprocessing techniques is node removal. As the scope of this research is the area between station and shunting yard, all nodes outside of this area will not be taken into account in the model. Those are the nodes within the shunting yards and the nodes outside the station that do not lie in between station and shunting yard. The last preprocessing technique is track removal. The model only takes switches into account and not the tracks in between switches.

4.12 Conclusion

In this section we have answered the research question *How should the solution approach be developed?* We see that the scope of the model is limited to routing between shunting yard and station and it should find a schedule for a 24-hour period on Eindhoven station. The requirements are mostly linked to HIP, which has to be the case as the output of this model will be input for HIP. To sum up, the model should find (i) a feasible schedule, (ii) constraints on time windows have to be hard constraints, (iii) stopping and saw movements can only be executed at tracks that allow so, (iv) combined train units take the same route, (v) the model opts for a route-lock route-release method, (vi) empty stock movements cannot stop at the border of the station, (vii) ICR Gvc-Ehv 9 should always go to Tuin, (viii) there should be a balance between the two shunting yards and enough time between

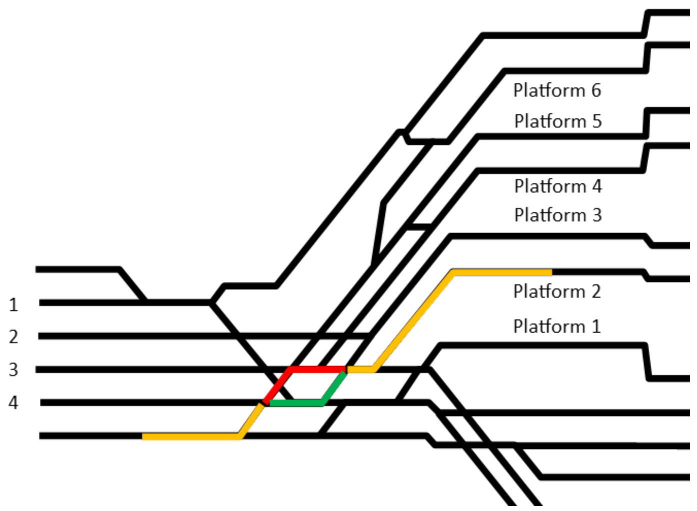


Figure 21: Route dominance. The orange route has two routing possibilities in the middle; green and red. Green only blocks paths between entrance 4 on the left and the platforms on the right, while the red route also blocks paths between entrance 3 on the left and all platforms on the right. Route red is dominated by route green.

ingoing trains to Oostzijde via gateway Z5, (ix) the running time should be reasonable and (x) routes should end at the shunting yard gateways.

The assumptions are not completely overlapping with the assumptions from HIP. Matching is assumed to be input for the model and in the matching enough time is reserved at the shunting yards. Through trains are fixed and are assumed to not cause any errors. There are also some assumptions about empty stock movements. The last assumption, for modelling convenience, is that (de)coupling always take place in the shunting yards.

The strength of the CP Optimizer are the global constraints, which allow a powerful propagation (Focacci et al., 2002). By viewing the problem as a joint problem of allocating resources and scheduling activities, and by using the (global) constraints described in Section 4.8 in the model, to potential of using CP Optimizer is exploited. Based on pre-generated routes, a routing scheme has been developed which is implemented in the model.

5 Experimental setup

This chapter will answer sub-question a) of the fourth research question: What are the different experimental setups that should be considered? (Section 5.3)

First, Section 5.1 elaborates on the data sets used in the experiments and Section 5.2 discusses how the data should be preprocessed to be valid input for the model. After the experiments are explained (Section 5.3), the planning horizon of 24 hours is discussed (Section 5.4). Lastly, a conclusion is drawn in Section 5.5.

5.1 Data instances

In this section, we describe the two different data instances used for our experiments. Both instances are real life data from Eindhoven station. Table 9 shows the main characteristics of the two data sets.

Characteristic	**Data set 1**	**Data set 2**
Number of shunt trains ($\mathcal{N}$)	92	106
Time interval shunt trains	08:00-08:00	08:00-08:00
Number of through trains ($\mathcal{F}$)	158	318
Time interval through trains	13:00-08:00	08:00-08:00
Open line entries	4	8
Day	Monday	Monday
Types of train units	7	7

Table 9: Overview of the two data instances used in the experiments.

The main difference between the two data sets is that data set 2 has a higher density in trains compared to data set. This is mostly due to the fact that through trains in data set 1 only drive in the interval between 13:00 and 08:00, while through trains in data set 2 drive the complete 24 hours (08:00-08:00). Moreover, data set 2 has cargo through trains while data set 1 has not. Another difference is that trains without passengers (empty stock shunt trains) in data set 1 only arrive from the left side of the station (which has 4 open line tracks), while empty stock shunt trains in data set 2 arrive from both the left and the right side (8 open line tracks).

5.2 Data preparation

There are several steps for data preparation. The first step is to extract the necessary information from the JSON files (see Figure 10 in Section 2.4.1) to the input parameters as described in Table 7. To convert a JSON scenario file to array values for all input parameters, we wrote a Python script. This script takes for example only switches and no tracks into account whereas the JSON scenario file takes also includes tracks. An overview of all used switches and the corresponding numbers is given in Figure 22. The second step is to guarantee the fourth assumption (Section 4.7) which states that through trains mutually do not cause errors in the model. In some input data, two through trains appear to use the same track or switch at the same time. This could be caused by the fixed movement duration of 2 minutes which cannot properly identify the real time granularity, or just by errors in the input data. As explained in Section 4.7, these inputs should be changed to prevent an inevitable *infeasible*-status. To that end, if two trains share one infrastructure element, that infrastructure element is deleted in the route for one of the two trains.

Another step for data preparation is regarding the location file. The location file should be checked to make sure it correctly displays which tracks it is allowed to park on and on which tracks it is only allowed to make a saw movement (see Figure 12). This is achieved by changing the parameters *sawMovementAllowed* and *parkingAllowed* to the right value. The last step is also regarding the location file and is to check the lengths of the reversal tracks. For an unknown reason, the lengths differ between different location files of Eindhoven and mostly do not correspond with the correct lengths obtained from Donna. Donna is the system used by ProRail where all schedules from different public transport operators are verified and processed. If a schedule is not correct according to Donna, the schedule cannot be operated and should be adjusted. Therefore, Donna gives the most reliable track lengths. All important reversal tracks (35A, 40, 166 and 172) are verified and changed if necessary to the lengths in Donna.

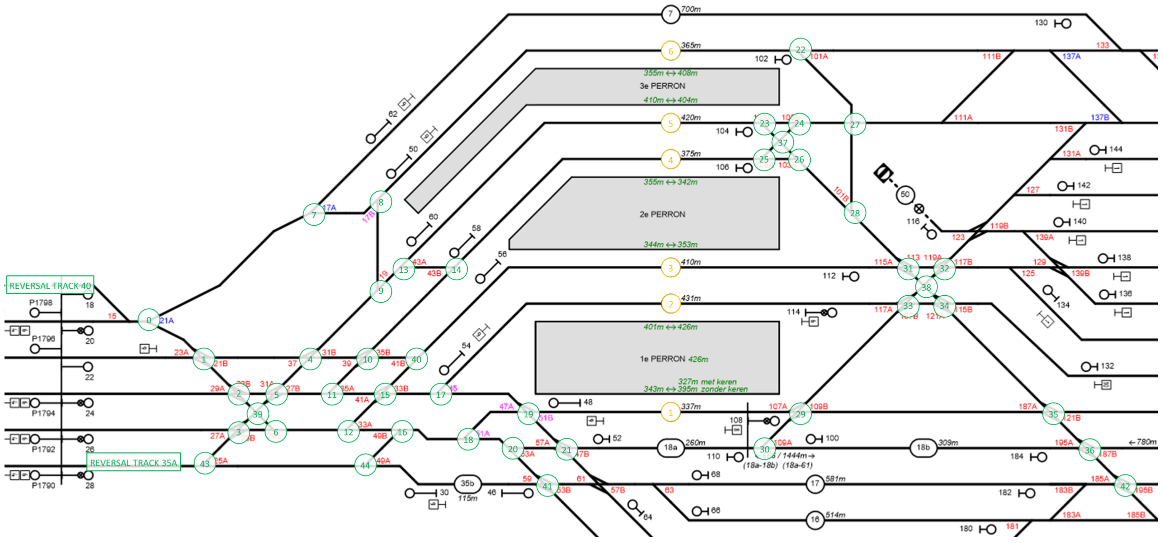


Figure 22: Track removal. This figure shows the infrastructure elements that are embedded in the model; the switches (green circles), the reversal tracks (green boxes) and the platforms (yellow circles).

5.3 Experimental design

We performed all experiments on a computer with an i7-8750H processor of 2.20 GHz and 16.0 GB RAM, which is representative for the computers used by the department R&D Hub Logistics. The experiments consist of two phases: i) the experiment is conducted on the constraint programming model and ii) the output of the experiment in the CP model is used to run on HIP. The results from running the experiments (experiment 1-3) on HIP are compared with the baseline; running HIP without adapted input from CP (experiment 0). See Appendix F for an extended guideline on how to converse the output from CP to input for HIP.

First, we define the experiments conducted in the constraint programming model.

Experiment 1. As CP is mainly powerful in finding feasible solutions instead of optimal solutions, the first experiment is to run the constraint programming model without an objective function. The model will thereby only look for a feasible solution without maximizing or minimizing a certain function. We expect that this experiment will be the fastest in CP since it is a feasibility problem instead of an optimization problem. However, incorporating an objective function in the CP model which is better aligned with the problems HIP has to solve, might help HIP in creating either a solution faster or a better solution, or a combination of both.

Experiment 2. The second experiment is to incorporate an objective function to the CP model which maximises the time each train stays in the shunting yard. This objective might positively influence the solvability of HIP. The local search within HIP can have difficulties to solve the parking problem in the shunting yard if there is no flexibility in time to move shunt trains more than absolutely necessary. In other words, HIP might find a solution faster if the time a train spends in the shunting yard is not too tight. For example, if train X has to be in the shunting yard for at least 2 hours and in experiment 1 train X appears to be exactly 2 hours in the shunting yard, HIP must find a solution to the parking and servicing problem without buffer time for train X which possibly makes the solution space too tight to find a feasible solution. As the CP model cannot find an optimal solution within reasonable time, the running time is limited to 10 minutes. A possible problem we foresee to this experiment might be that the parking problem in the shunting yards gets harder and even too hard when more trains stay in the shunting yard at the same time.

Experiment 3. The third and last experiment is to add an objective function which divides the trains 50:50 over the two shunting yards. Just as the second experiment, this objective function might positively influence the solvability of HIP. The local search within HIP can have difficulties to solve the parking problem in the shunting yard if there are too many trains in one shunting yard at the same time. In other words, HIP might benefit from a proper division of trains over the two shunting yards. Having 90% of the trains stalled at shunting yard 1 and 10% at shunting yard 2 will probably cause a hard or even infeasible problem at the overused shunting yard 1. Unfortunately, there is (not yet) a perfect known ratio between shunting yards Oostzijde and Tuin at Eindhoven. From prior small experiments we see that a ratio outside the interval [60:40 - 40:60] (Oostzijde:Tuin) causes bad running times for HIP or no feasible solution at all. Next, if you look at maximum capacity of both shunting yards, the ratio is approximately 60:40 (Oostzijde:Tuin). However, approximately 40% of all shunt trains are of type *ICR Gvc-Ehv 9* which should always be shunted to Tuin and most of them stay only for 15 minutes at the shunting yard. Shunt trains that have to stay for a longer period on the shunting yard cause more complexity for the parking problem, so aiming for a ratio corresponding to the maximum capacity (60:40, Oostzijde:Tuin) will probably make Oostzijde too hard. We therefore decided to make a compromise and aim for experiment 3 to divide the trains 50:50 over the two shunting yards. This is accomplished by adding an objective function to the CP model which minimises the difference in trains shunted to Oostzijde and trains shunted to Tuin.

5.4 Planning horizon and replications

In Section 4.5, the scope of the model was defined to be able to plan a 24-hour planning horizon. Although the constraint programming model is able to find a solution within seconds or minutes, HIP needs hours and sometimes days to make a plan for a 24-hour planning horizon. Due to time limitations, the experiments in HIP will not be executed completely for the 24-hour planning horizon. Instead, three smaller time intervals are chosen which represent the 24-hour period as accurate as possible. Experiments on the three smaller time intervals are replicated 10 times to reduce the experimental error caused by the randomness of the local search within HIP. On top of that, to gain at least some knowledge about the full 24-hour period with and without adaptations from the constraint programming model, all experiments with one replication each will be executed on the complete 24-hour planning horizon for data set 1. Due to time limitations, the 24-hour horizon will not be run for data set 2.

The following three smaller intervals are defined:

Interval 1: 13:00-15:00. The first interval is expected to be an easy interval. The interval does not lie in the peak hours, so there is a regular flow of traffic within the station. Also, few trains need to be shunted to or moved within the shunting yards. This interval is meant to represent most day hours that are not peak hours. As the local search within HIP encounters most problems finding routes through a high density of trains, the expectation is that HIP with input from CP will come slightly faster to a solution compared to running HIP without input from CP.

Interval 2: 18:00-20:00. The second interval lies partially in and partially just after the peak hours. This means that the density of trains within the station is relatively high compared to the first interval. Also, just after the peak hours, many rolling stock need to be shunted because less capacity is needed on the railways after the peak hours. The combination of a high density of through trains and a big number of shunt moves is expected to cause a big difference between running HIP with and without input from the CP model.

Interval 3: 20:00-24:00. The last interval is an interval that represents the night. During the night, there is little traffic in the station area, but the shunting yards are largely occupied. Besides, most service tasks are performed during the night, so the service problem is the hardest in the last interval. As finding routes through the traffic in the station is relatively easy if there are few trains in the station, we expect that the difference between HIP with and without CP will be negligible. HIP without CP might even benefit from the broad solution space compared to the tightened solution space caused by the CP model.

To create the scenario files for the three intervals, the scenario files with the 24-hour period as described in the previous section are used. All through trains are kept in the scenario file and all shunt trains of which the interval they stay in the shunting yard does not overlap with the experiment's interval, are deleted from the scenario file. Table 10 gives an example of which shunt trains are deleted from and which shunt trains are kept in the scenario file for interval 1 (13:00-15:00).

Arrival time	**Departure time**	**Kept in scenario file**
12:45	13:15	Yes
10:30	11:30	No
14:00	14:10	Yes
11:00	16:00	Yes
14:30	15:30	Yes
15:30	16:30	No

Table 10: Example of shunt trains that are deleted from and that are kept in the scenario file for interval 1 (13:00-15:00).

5.5 Conclusion

This chapter aimed to answer the question *What are the different experimental setups that should be considered?*

There are two data sets which mainly differ in number of shunt trains and through trains. Three experiments have been defined which are compared against the baseline; running HIP without CP input (experiment 0). In the first experiment CP is run without an objective, in the second experiment CP is run with an objective to increase the time each shunt train stays in the shunting yard, and in the third experiment CP is run with an objective to balance the shunt trains equally between shunting yards Oostzijde and Tuin. Experiment 1 is the easiest to implement and experiment 2 and 3 are developed to prevent foreseen problems with HIP. Lastly, three time intervals within the 24-hour planning horizon are defined to conduct experiments on due to the long running time of a 24-hour planning in HIP. These three time intervals are meant to represent the 24-hour period as good as possible, thereby giving the opportunity to draw preliminary conclusions about the 24-hour period. To strengthen conclusions about the 24-hour period, the full 24-hour period is run once for all experiments on data set 1. Due to time limitations, the full 24-hour period is not run on data set 2.

Figure 23 gives a summary of the experimental setup.

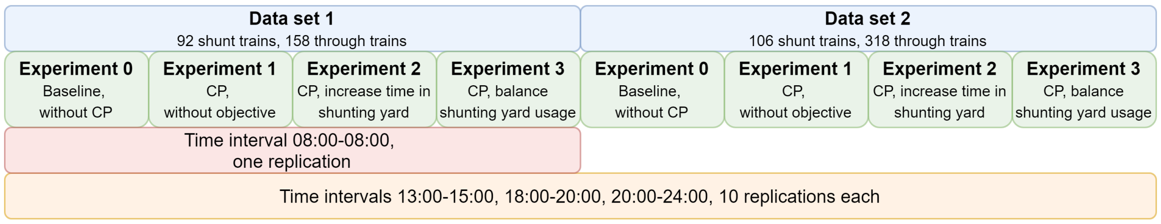

Data set 1
92 shunt trains, 158 through trains
Data set 2
106 shunt trains, 318 through trains
Experiment 0
Baseline, without CP
Experiment 1
CP, without objective
Experiment 2
CP, increase time in shunting yard
Experiment 3
CP, balance shunting yard usage
Experiment 0
Baseline, without CP
Experiment 1
CP, without objective
Experiment 2
CP, increase time in shunting yard
Experiment 3
CP, balance shunting yard usage
Time interval 08:00-08:00, one replication
Time intervals 13:00-15:00, 18:00-20:00, 20:00-24:00, 10 replications each

Figure 23: Summary of the experimental setup. All experiments are conducted on the two data sets using 10 replications for the three small time intervals. Additionally, one replication of the 24-hour planning horizon is run for all experiments on data set 1.

6 Experimental results

This section aims to answer sub-question b) of the fourth research question: *How does the current solution approach perform when the output of the proposed solution approach is used as starting point of the current method at NS?*

The results from running the CP model are presented (Section 6.1) and thereafter the results of HIP with and without CP input are presented (Section 6.2). Lastly, a conclusion is drawn in Section 6.3.

6.1 Results from CP

In this section, the results from running the CP model will be presented. Besides running time of CP, two more results are added. As experiment 2 aims to increase the total time all shunt trains stay in the shunting yard, the KPI *Total shunt time* is added to all experiments. As experiment 3 aims for a 50-50 balance of shunt trains over the two shunting yards, the number of shunt trains routed to Oostzijde and to Tuin is also presented for all experiments (#Tuin, #Oostzijde). Table 11 and Table 12 show the results for the CP model when running respectively data set 1 and data set 2. An asterisk (*) indicates a maximum running time set to the model.

	Experiment 1	**Experiment 2**	**Experiment 3**
Objective	No objective	Increase time in yard	Balance yards 50-50
Running time (s)	9.1	600*	12.39
Status	Feasible	Feasible (9.3% gap)	Optimal
#Tuin	49	52	46
#Oostzijde	43	40	46
Total shunt time (min)	18706	23546	18850

Table 11: Results from running data set 1 on the constraint programming model. An asterisk (*) indicates a maximum running time.

	Experiment 1	**Experiment 2**	**Experiment 3**
Objective	No objective	Increase shunt time	Balance yards 50-50
Running time (s)	11.51	600*	14.49
Status	Feasible	Feasible (22.2% gap)	Optimal
#Tuin	39	51	53
#Oostzijde	67	55	53
Total shunt time (min)	7256	16951	7779

Table 12: Results from running data set 2 on the constraint programming model. An asterisk (*) indicates a maximum running time.

The first observation is that that experiment 2 clearly increases the total time shunt trains stay in the shunting yard and that experiment 3 achieves the fifty-fifty balance over the shunting yards.

The second observation is that the total time all shunt trains stay in the shunting yard is approximately the same for experiment 1 and 3 in both data sets (18706 versus 18850 and 7256 versus 7779). In data set one, an explanation is the small difference in shunt yard usage of experiment 3 compared to experiment 1 (46:46 versus 49:43), which makes the solutions of both experiments much alike. However, this does not count for data set two, where the division in experiment 1 is 39:67 and in experiment 3 53:53. When the output of the CP model is analyzed into more detail, the difference regarding total shunt time between experiment 2, and experiments 1 and 3, appears to be mainly due to the difference in departure time from the shunting yard back to the station. The arrival times at the shunting yard in all three experiments are roughly the same.

The third observation is that the increased total time all shunt trains stay in the shunting yard in experiment 2 with respect to experiment 1 is much higher in data set 2 compared to data set 1. Data set 1 achieved a 26% increase in shunt time while data set 2 achieved a 134% increase. However, results for experiment 2 might therefore be different between data set 1 and data set 2.

A fourth observation is that the division of trains over Tuin and Oostzijde in experiment 1 of data set 2 is very skewed, which will probably influence the running time of HIP. Contrary to the skewed division in experiment 1 of data set 2, is the relatively well-balanced experiment 2 of data set 2 (51:55 in Tuin and Oostzijde).

Lastly, Table 13 shows the model characteristics for data set 1 and data set 2. As can be observed, the difference in running time between data sets 1 and 2 is just 2 seconds which is acceptable for the difference in instance size. This is an important result, though it is just based on two data sets.

	Data set 1	**Data set 2**
Constraints	30242	34880
Interval variables	8575	10670
Sequence variables	52	52
Running time experiment 1 (s)	9.1	11.51
Running time experiment 2 (s)	600*	600*
Running time experiment 3 (s)	12.39	14.49

Table 13: Model characteristics of data set 1 versus data set 2. An asterisk (*) indicates a maximum running time.

6.2 Results from HIP

In this section the results from running HIP with and without input from the CP model will be presented. As described in Section 4.2, the running time as well as the number of shunt moves is given. Lastly, to reduce the experimental error caused by the randomness from the local search within HIP, 10 replications are run for every experiment. For these experiments, each replication is run sequentially. The results also show the solvability rate (number of solved replications). However, a newer version of HIP allows to run multiple replications in parallel all using a different core of the computer. This way, always the best replication will be chosen out of n replications, where n is the amount of cores HIP is run on in parallel. Therefore, and to reduce the impact of outliers caused by randomness, the median instead of the average of the 10 replications is reported. Note that the median only covers the running time of the feasible solutions. The next section discusses the results of data set 1 and Section 6.2.2 discusses the results of data set 2 for all experiments and the three time intervals as described in Section 5.4.

6.2.1 Results from HIP data set 1

Interval	KPIs	Exp. 0	Exp. 1	Exp. 2	Exp. 3
13:00-15:00	HIP Running time (hh:mm:ss)	00:00:26	00:00:01	00:00:02	00:00:01
	# Shunt moves	48	50	50	52
	Instances solves	10/10	10/10	10/10	10/10
18:00-20:00	HIP Running time (hh:mm:ss)	01:18:48	00:14:40	00:04:32	00:03:17
	# Shunt moves	82	85	92	87
	Instances solves	7/10	10/10	10/10	10/10
20:00-24:00	HIP Running time (hh:mm:ss)	00:17:44	00:16:31	00:04:22	00:03:05
	# Shunt moves	135	138	164	150
	Instances solves	4/10	7/10	10/10	9/10
08:00-08:00	HIP Running time (hh:mm:ss)	24:00:00*	24:00:00*	00:17:06	00:08:27
	# Shunt moves	242	245	277	259
	Instances solves	0/1	0/1	1/1	1/1

Table 14: Results of HIP for data set 1 without input from CP (experiment 0) and with input from CP (experiment 1-3) for different time intervals. An asterisk (*) indicates a maximum running time.

Table 14 shows the results of data set 1 when running HIP for all experiments on the three small time intervals and the full 24-hour period . The first and most important observation is that the 24-hour period in experiment 2 and 3 can be solved by HIP in respectively 8:27 minutes and 17:06 minutes. This is a reduction of over 95%. However, experiment 1 could not find a solution after 24 hours.

The second observation is that running HIP with CP (experiment 1-3) has a strictly shorter running time compared to running HIP without input from CP (experiment 0), although the difference between the running time of Experiment 0 and Experiment 1 are negligible for the time interval 20:00-24:00.

Thirdly, the expectations of difference in running time between the time intervals, as explained in Section 5.4, appear to be correct. The interval 13:00-15:00 is easy to compute, as all experiments have a 100% solvability rate and a low running time. Although the relative difference in running time is high, the absolute difference is very small. Also the number of shunt moves between the experiments does not lie in a big interval.

The interval 18:00-20:00 was expected to give big differences between experiment 0 and experiments 1-3. This appears to be true. Experiments 1-3 need a running time less than 20% of the running time of experiment 0. However, the number of shunt moves needed in experiments 1-3 is slightly higher than the number of shunt moves needed in experiment 0. However, a longer running time gives more time to search for a lower number of shunt moves. To make a fair comparison in shunt moves between the experiments, all experiments should have been run for 1:18:48 hour.

Lastly, interval 20:00-24:00 was expected to have approximately the same running time. This expectation holds for experiment 1, but experiment 2 and 3 have a significantly lower running time. Besides, experiments 2 and 3 have a better solvability rate compared to experiment 0 and experiment 1. The number of shunt moves needed for both experiments is significantly higher as well, though the same argument as mentioned previously about running time and number of shunt moves holds in this case. However, experiment 2 needs a lot more shunt moves (164) compared to experiment 3 (150). This observation shows that increasing the total time trains stay in the shunting yard can also have negative effects on the solution. A closer look to the solutions of experiment 0 in the time interval 20:00-24:00 reveals that the bad solvability rate is mostly due to difficulties with open line movements from the left side of the station. Those open line movements from the open line to the station interfere easily with shunt trains that have to go from the station to the open line at the same time. Just one routing possibility is feasible which is hard for the local search to find.

Lastly, the running time of experiments 2 and 3 compared to experiment 1 is much lower in the time intervals 18:00-20:00 and 20:00-24:00. Based on this data set, we could carefully say that flexibility in time in the shunting yard (experiment 2) and well-balanced shunting yards (experiment 3) are indeed important factors influencing feasibility.

6.2.2 Results from HIP data set 2

Interval	KPIs	Exp. 0	Exp. 1	Exp. 2	Exp. 3
13:00-15:00	HIP Running time (hh:mm:ss)	00:00:06	00:00:01	00:00:01	00:00:04
	# Shunt moves	35	38	38	36
	Instances solves	10/10	10/10	10/10	10/10
18:00-20:00	HIP Running time (hh:mm:ss)	00:22:13	00:01:51	00:02:18	00:03:10
	# Shunt moves	74	70	72	73
	Instances solves	9/10	10/10	10/10	10/10
20:00-24:00	HIP Running time (hh:mm:ss)	01:20:49	01:23:40	01:28:00	00:24:42
	# Shunt moves	132	138	139	126
	Instances solves	6/10	7/10	5/10	10/10

Table 15: Results of HIP for data set 2 without input from CP (experiment 0) and with input from CP (experiment 1-3) for different time intervals.

Table 15 shows the results for the second data set. The first observation is that the difference between experiment 0 and experiments 1-3 is comparable with the differences as found in data set 1. However, the results for the last time interval (20:00-24:00) are deviating from results of data set 1.

Also, the expectations, as explained in Section 5.4, appear to be correct for this data set.

The first time interval (13:00-15:00) is rather simple and conclusions are similar to the conclusions from data set 1. The difference in running time between experiment 0 and experiments 1-3 is big in relative sense but small in absolute matters. Also the number of shunt moves are comparable and the solvability is 100% for all experiments.

The time interval 18:00-20:00 was expected to cause the biggest differences in running HIP with and without CP. This expectation also holds for data set 2. Though HIP without CP has less difficulties finding a solution in data set 2 (running time of 00:22:13) compared to data set 1 (running time of 01:18:48), the difference in running time between experiment 0 and experiments 1-3 is big in this data set as well. Remarkable is the number of shunt moves, which is in this data set generally lower for experiments 1-3 compared to experiment 0, whereas data set 1 shows contrary results.

The biggest difference between data set 1 and data set 2 is time interval 20:00-24:00. Experiment 0 and 1 are comparable between both data sets, though the running time of experiment 1 is now slightly higher than the running time of experiment 0. Also the solvability of experiment 0 is better compared to data set 1. Experiment 1 might encounter problems in this time interval with the skewed balance between Oostzijde and Tuin (67:39). Experiment 2 performs inferior to all other experiments in this data set in terms of running time and especially in terms of solvabilility. The running time is the highest of all experiments and only 5 instances could be solved. This could be explained by the observation made in Section 6.1 that the total shunt time of experiment 2 in data set 2 is more than twice the total shunt time of experiment 1 and 3. A detailed research into the infeasible solutions of this experiment points indeed out that almost all conflicts arise from a lot of movements shifts due to a hard parking problem in the shunting yards. Remarkable is the fact that this is not reflected by the number of shunt moves, which was significantly above average in data set 1 (164) and is slightly above average in this data set (139). Experiment 3 performs well in time interval 20:00-24:00 in terms of running time, shunt moves and solvability. Well-balanced shunting yards are important, however, experiment 2 showed that well-balanced (51:55) does not always perform good. As experiment 3 might be a lucky shot of the CP model - not only creating a fifty-fifty balance, but also routing each train to the apparently best shunting yard - improvements of experiment 3 should be considered in future work.

Lastly, HIP makes a bad decision in some cases regarding the choice of routes. Figure 24 gives an example of a fixed train and a shunt train from data set 2. The fixed train has reserved the blue line in the interval [21:47-21:49]. The shunt train needs to reserve a route from platform 6 to Open line 2 in that same time interval. HIP mostly chooses the red route, which results in a crossing with the fixed train and therefore an infeasible solution. If however the green route was chosen by HIP, no crossing would have occurred and HIP would have found a feasible solution. Note that CP was able to find the green route and thereby a feasible solution. The reason HIP chooses the red path is because paths are chosen using Dijkstra's shortest-path algorithm which does not include blockades by fixed trains. This can be an indication for HIP for using predetermined routes instead of Dijkstra's algorithm, or to include fixed trains the algorithm.

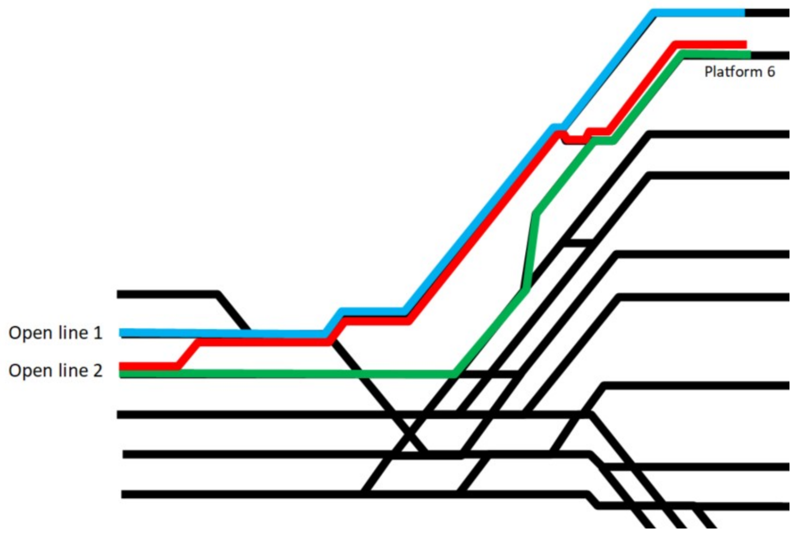


Figure 24: The blue line represents a fixed train moving over the line towards Open line 1. At the same time a shunt train needs to find a route from platform 6 to Open line 2. There are two possibilities: the red one which crosses with the fixed train, and the green one, which does not cross with a train.

6.3 Conclusion

This chapter aimed to answer the following research question: How does the current solution approach perform when the output of the CP model is used as starting point of the current method at NS? To answer this question, two data sets have been used and different experiments are run with 10 replications each to prevent an error due to the random seed of the local search of HIP. The results from the CP model are promising. The model solves both data sets within reasonable time although data set 2 is slightly bigger than data set 1. Furthermore, the increase in total time at the shunting yard is much higher in data set 2 compared to data set 1, of which the results are visible in HIP later. To measure the performance of both data sets in HIP, the algorithm running time and the number of shunt moves are chosen. The conclusion for both data sets are generally speaking similar. Although the differences in running time with and without CP highly fluctuate, using CP as warm start for HIP results on average in a 70% reduction in running time compared to running HIP without CP. Especially the 24-hour period results in a significant improvement in running time (>95%). Conclusions about the number of shunt moves are less significant, as a fair comparison can only be made when running time is equal. An important conclusion is that running CP without an objective function (experiment 1) does not necessarily guarantee good results in HIP, as can be seen in the results for the 24-hour period. Balance between the two shunting yards plays an important role, however, the balance could be fine-tuned more. Lastly, increasing the time a shunt train stays in the shunting yard does help in most cases, but an excess of overtime complicates the parking problems thereby leading to a worse running time than originally achieved by HIP as can be seen in data set 2.

7 Conclusions and recommendations

In this chapter, we summarise the research findings, draw conclusions and summarize both the practical and scientific contribution of this thesis. Next, we give recommendations for future research, both based on the CP model as on HIP.

7.1 Conclusions

For this research, we explored the possibility of using constraint programming as warm start for the local search of NS, thereby aiming for a shorter running time. The local search of NS has difficulties finding routes from the station to the shunting yard if there is only a small gap in time where the routes are unobstructed by other trains. The goal of the research was to develop a CP model that could determine the routes and the timing of all shunt trains from the station to the shunting yard, thereby keeping enough time between incoming and outgoing shunt trains to perform service tasks in the shunting yards and make the necessary movements and reversals. The model was built to solve Eindhoven station and was based on the routing flow through the station which most shunt trains follow: a shunt train arrives at a platform, then should be routed towards the shunting yard. After being serviced the shunt train leaves the shunting yard to be on time at its departure track. Some shunt trains without passengers arrive at the side of the station area instead of a platform and need an additional route to be found: to come from the side of the station area to the platform and vice-versa.

To validate the model, two data sets were used. Data set 2 was slightly bigger than data set 1 in terms of shunt trains and through trains. Both data sets yielded good results in the CP model and the model solved both data sets within reasonable time (<15s) although data set 2 is slightly bigger than data set 1. Next, the output of the CP model had to be connected to HIP (Hybrid Integral Planning Method) and serve as input for the local search. Besides running HIP without input from CP (experiment 0) and running HIP with input from CP (experiment 1), two additional experiments were developed to help HIP. The original CP model was extended with two objectives which represented the two experiments: one objective that increased the time a shunt train could stay in the shunting yard (experiment 2), and one objective that tried to balance the usage of the two shunting yards equally (experiment 3). The goal of experiment 2 was to aid HIP by giving some flexibility in the service and parking problem. The goal of experiment 3 was to aid HIP by preventing one over-capacitated shunting yard and one under-capacitated shunting yard. Though the original goal of this research was to plan a 24-hour period, running time for this time interval appeared to be too high to conduct multiple experiments and replications. Therefore, the 24-hour period was split up into three intervals, together representing the 24-hour period. The first interval was 13:00-15:00, which represents the time at which both the station and the shunting yard are normally capacitated. The second interval was 18:00-20:00, which represents the time at which the station is rather busy with trains at the end of the peak hour and many trains need to be shunted to the shunting yards. The third and last interval was 20:00-24:00, which represents the time at which the station is almost empty, but the shunting yards are filled up and many service tasks need to be conducted. Lastly, to be able to conclude something about the 24-hour period, one replication of all four experiments has been conducted for the full 24-hour period on data set 1.

All four experiments are conducted on HIP on the two different data sets for 10 replications each to prevent an error due to the random seed of the local search of HIP. To compare the results, both the algorithm running time and the number of shunt moves are taken as KPI. The results are similar for data set 1 and data set 2. Although the differences in running time with and without CP highly fluctuated, using CP as warm start for HIP resulted on average in a 70% reduction in running time compared to running HIP without CP. Especially the 24-hour period resulted in a significant improvement in running time (>95%). Conclusions about the number of shunt moves are less significant, as a fair comparison can only be made when running time is equal. An important conclusion is that running CP without an objective function (experiment 1) does not necessarily guarantee good results in HIP. Balance between the two shunting yards play an important role, however, the balance has to be fine-tuned more to be robust over other scenarios in Eindhoven. Lastly, increasing the time a shunt train stays in the shunting yard does help in most cases to reduce running time, but an excess of overtime complicates the parking problems thereby leading to a worse running time than originally achieved by HIP.

Firstly, we can conclude that the running time of the CP model without an objective is short (<15s) and that CP seems to be a promising and well-suited approach for solving the Shunt Routing Problem. Secondly, in most of the experiments, applying CP to find the shunt routes before starting the local search, yields a significant improvement on the total running time. Especially on the 24-hours scenario the results were impressive. However, we recommend to do an in-depth analysis in the future to investigate the reason of the running time reduction of the local search. This could then be used to align the objective of the CP model, to always be able to guarantee good results in HIP when using CP. Furthermore, there is future work to do to develop a general model with an objective that is robust and a model that can be more easily generalised over other stations.

Summarizing, the contribution of this thesis to the scientific literature is twofold. Firstly, we developed a hybrid method by applying constraint programming to the Shunt Routing Problem and feeding the solution as warm start to solve the Train Unit Shunting Problem using a local search. Secondly, the applied our hybrid solution approach to solve two realistic Train Unit Shunting Problems. The contribution of this thesis to practice is that the results show the potential of using constraint programming in a hybrid method with a local search. This result contributes to the objective of NS to make logistic hubs more flexible by planning on a shorter term.

7.2 Recommendations and future research

Based on this research, we formulate several recommendations for NS which might be good to explore in future research. This section is split up in recommendations regarding the CP model and recommendations regarding the use of HIP. First the recommendations regarding the CP model are discussed.

7.2.1 Recommendations and future research for the CP model

The first recommendation is about the running time limit of CP. In this research, the running time limit is 10 minutes, which is only reached by experiment 2. As argued in this thesis, the parking problem in HIP might get easier if there is enough flexibility in time for trains to move within the shunting yard. However, if a train stays longer than necessary in the shunting yard, HIP might get difficulties with the parking problem. We see this happen in interval 20:00-24:00 of data set 2. It would be worth investigating how the model can increase the total time shunt trains stay in the shunting yard without making the parking problem unnecessary harder. A possibility would be to maximise the amount of trains that stay X longer than necessary in the shunting yard where X is to be determined in future experiments. X could be a percentage as well as a fixed value. This way, you prevent trains staying too long on the shunting yard and you capture each train individually instead of looking at all trains together.

Another recommendation is regarding the capacity of the two shunting yards and the way the trains are balanced between them. In this research, an equal division is pursued (experiment 3) over the 24-hour horizon. This means that within the 24-hour period, the same number of trains are shunted to both shunting yards. However, this approach does not take the duration of shunt trains or the moment they are in the shunting yard into account. This could lead in the worst case to a situation in which all trains that have to stay in the shunting yard for multiple hours in approximately the same hours are routed to one shunting yard, whereas trains that are staying just for half an hour at the shunting yard and are perfectly spread over the day are routed to the other shunting yard. In future experiments it is therefore interesting to aim for an hourly balance, or to aim for a balance in time on the shunting yard instead of number of trains on the shunting yard. Such a precise balance criteria does only make sense when there is enough information about an optimal balance between the two shunting yards. Currently, there is knowledge about the maximum capacity of both shunting yards, but this does not necessarily reflect the optimal ratio when the shunting yards are not completely occupied. This could be something interesting to investigate for NS.

The next recommendation is about which shunt trains to focus on in the CP model. Though this research has focused on Eindhoven station, it is particularly interesting for NS how to generalise this model over other stations. Most shunt trains follow the common routing flow (arrive at a platform, shunted to the shunting yard, serviced, shunted back to a platform) and can thereby be captured by a general model. There are some exceptions that cannot be captured by this model. The first exception are trains that are present at the start of the

24-hour horizon (InStanding) and trains that are present at the end of the 24-hour horizon (OutStanding). Those trains can be either on a platforms or on a track in the shunting yard. They only follow part of the flow chart and therefore cannot be captured easily and generalised in the model. An option is not to consider these trains in the CP model and let HIP deal with this exception. A similar situation is created when shunt trains arrive at the open line without passengers and have to leave within a short time from a platform, or the other way around. For example, in data set 2, there is a train that arrives at 21:41 at platform 6 and needs to depart at 21:49 from one of the open lines. In that case, there is no time to follow the routing chart. It will stay on platform 6 up to 21:47 and then leaves for the open line where it will leave 2 minutes later at 21:49. This is also an exception that cannot be captured by a general CP model. More specifically, trains that have less than 25 minutes between arrival and departure, cannot be captured properly by the CP model. An option to deal with those trains, is again to not consider these trains in the CP model and let HIP deal with this exception.

There are also two recommendations regarding possible improvements of the CP model. The first is regarding open line movements. In this research, it is assumed that all open line shunt trains are shunted via a platform at the station to the shunting yard. However, it is possible that these trains enter the shunting yard directly without stopping at a platform. The possible sets of routes for open line shunt trains could then be expanded with all allowed set of routes from the open line directly to the shunting yard. This may reduce the number of moves needed to get into the shunting yard and therefore reduced the amount of personnel needed. Another possible improvement is regarding matching. If, in the original scenario files without adaptations from the CP model, an outgoing shunt train is composed out of multiple train units, it is stated which types should depart in which order. For example, at 8:00 a train which is composed out of a VIRM-6 and a VIRM-4 should depart from platform 1. The CP model requires a matching, so the VIRM-6 and VIRM-4 are matched to a specific train unit number. In this case, there are different subtypes needed (VIRM-6 and VIRM-4), so there is no flexibility in modeling the order of train units. However, if there are multiple train units of the same subtype needed, HIP could have the flexibility to switch the predetermined matching. For example, at 8:00 a train which is composed out of a VIRM-4 and a VIRM-4 should depart from platform 1 and train units 9406 and 9405 are matched to this train. In this case it does not matter in which order they depart, because both train unit types are the same. By fixing an order [9406,9405], HIP does not see the order [9405,9406] as a feasible option, whereas in practice this does not matter. We saw this in experiment 3 of data set 1 for the 24-hour period which was infeasible in first instance but switching the order resulted in a feasible solution. A possible solution to this problem is to give no matching to this type of trains. However, this will lead to multiple trains having no matching and thereby needlessly making the problem harder for HIP. A better solution, which cannot be implemented in HIP yet, is to be able to say if the order of the train units matters. This would give HIP more freedom in finding a feasible solution without enlarging the solution space.

The last recommendation for the CP model is to explore the possibilities of "refine conflicts". This is a module in Python that gives the opportunity to examine an infeasible solution and to investigate why the solution is infeasible. The module exists in Python, but is not user-friendly and hard to understand. Exploring this module gives the opportunity to find the reason behind infeasible solutions and give an insight in what to change to the input data to make a feasible solution.

7.2.2 Recommendations and future research for HIP

In this section recommendations regarding HIP and the connection between CP and HIP are given. The first recommendation is about the connection between CP and HIP. In this research, the routes determined by the CP model are fixed in the input for HIP, which means that HIP cannot change the routes. This results firstly in a smaller search space for HIP instead of a warm start, and secondly HIP might benefit if it can alter some routes if infeasibilities arise within the shunting yards. There is no possibility in HIP at the moment to give a route as a warm start, so therefore the shunt routes are fixed in this research. It is however recommended for NS to look into the possibility of giving routes as warm start to HIP.

There are two recommendations about search space reduction. The first recommendation is to solve both shunting yards in HIP separately. In the current version of HIP, a train can make a movement from one shunting yard to the other. If the balance between the shunting yards is not correct, this should definitely be a possibility to prevent infeasibility. However, if the balance between the shunting yards is correct, HIP might benefit from a reduced search space. Not only will the problem be solved faster, but also using less shunt moves as redundant shunt moves outside the shunting yard will not be made anymore. A crucial point is that a shunt train does not have to leave the shunting yard to guarantee feasibility and a good balance between the shunting yards is therefore necessary. The second recommendation is to predefine routes in HIP as well. In the current version of HIP, routes are determined using Dijkstra's algorithm. In some cases this algorithm chooses a route that is firstly an obstructed route, and secondly a route that is not preferred by ProRail. In those cases we can argue for predefined routes in HIP, but in general would predefined routes also reduce the search space of HIP. A drawback of predefined routes is that they should be created. However, there is an overview for every station with all possible routes between every infrastructure element and which route is preferred over other routes. This could be transferred to input for HIP.

Lastly, there are two recommendations regarding baseline comparison. Firstly, CP does not only chooses which routes in what time interval each shunt train has to take, but by choosing the route, it also chooses the shunting yard for each shunt train. To make a fair comparison between running HIP with and without CP, the choice of shunting yard should be predefined when running HIP without CP. This way, it can be verified whether the difference in running time is mostly due to the routing of CP or mostly due to the choice of shunting yard of CP. In case of the latter, NS could use easier methods than CP. Secondly, running HIP with CP results in a break-up of the route from the station to a track within the shunting yard. First a route from the station to the entrance of the shunting yard is chosen by CP with a duration of 2 minutes, and then HIP chooses a route from the entrance of the shunting yard to a track within the shunting yard, also with a duration of 2 minutes. Running HIP without CP would only require one route from the station to a track within the shunting yard with a duration of 2 minutes. To make a fairer baseline comparison, HIP should see the movement from the station to a track within the shunting yard as one movement of 2 minutes when running HIP with input from CP. This could be programmed by simply stating that an incoming shunt train arrives 2 minutes earlier at the entrance of the shunting yard and that an outgoing shunt train depart 2 minutes later from the entrance of the shunting yard. HIP would still see this as two movements, but both movements together will have a duration of 2 minutes. This will not be a problem for HIP, as we currently fix all routes from the station to the shunting yard gateway. That way, HIP does not know that the movement from the station to the shunting yard gateway and the movement from the gateway to a track within the shunting yard belong to the same train. This makes it possible to make both movements within the same time interval.

References

Adlbrecht, J.-A., Hüttler, B., Ilo, N., & Gronalt, M. (2015). Train routing in shunting yards using answer set programming. *Expert Syst. Appl.*, *42*(21), 7292–7302.

Branishtov, S., Shirvanyan, A., & Tumchenok, T. (2015). Formation of train routes in a station. *Automation and Remote Control*, *76*, 1655–1663.

Burggraeve, S., & Vansteenwegen, P. (2017). Robust routing and timetabling in complex railway stations. *Transportation Research Part B: Methodological*, *101*, 228–244.

Caimi, G., Chudak, F., Fuchsberger, M., Laumanns, M., & Zenklusen, R. (2011). A new resource-constrained multicommodity flow model for conflict-free train routing and scheduling. *Transportation Science*, *45*, 212–227.

Cappart, Q., & Schaus, P. (2017). Rescheduling railway traffic on real time situations using time-interval variables. *Lecture Notes in Computer Science: Vol. 10335*, 312–327.

Col, G., & Teppan, E. (2019). Google vs IBM: A constraint solving challenge on the job-shop scheduling problem. *Electronic Proceedings in Theoretical Computer Science*, *306*, 259–265.

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

Crainic, T., Ferland, J., & Rousseau, J. (1984). A tactical planning model for rail freight transportation. *Transportation Science*, *18*(2), 165–184.

Crainic, T., & Rousseau, J.-M. (1986). Multicommodity, multimode freight transportation: A general modeling and algorithmic framework for the service network design problem. *Transportation Research Part B: Methodological*, *20*(3), 225–242.

D'Ariano, A., Pacciarelli, D., & Pranzo, M. (2007). A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research*, *183*, 643–657.

Dechter, R., Meiri, I., & Pearl, J. (1991). Temporal constraint networks. *Artificial Intelligence*, *49*(1), 61–95.

Dems, A., Rousseau, L.-M., & Frayret, J.-M. (2015). A hybrid constraint programming approach to a wood procurement problem with bucking decisions. *Constraints*, *21*.

den Hartog, M. (2010). *Shunt planning : An integral approach of matching,parking and routing* (Master's thesis). University of Twente. Enschede, the Netherlands.

Edis, E., & Oğuz, C. (2012). Parallel machine scheduling with flexible resources. *Computers & Industrial Engineering*, *63*, 433–447.

Egbers, J. (2001). *Knelpunten op knooppunten : Het ontwerp van een toets op uitvoerbaarheid bij de productie van basisdagen.* (Doctoral dissertation). Eindhoven University of Technology. Eindhoven, the Netherlands.

Focacci, F., Lodi, A., & Milano, M. (2002). Optimization-oriented global constraints. *Constraints*, *7*, 351–365.

Freling, R., Lentink, R. M., Kroon, L. G., & Huisman, D. (2005). Shunting of passenger train units in a railway station. *Transportation Science*, *39*(2), 261–272.

Gedik, R., Kalathia, D., Egilmez, G., & Kirac, E. (2018). A constraint programming approach for solving unrelated parallel machine scheduling problem. *Computers & Industrial Engineering*, *121*.

Gedik, R., Kirac, E., Milburn, A., & Rainwater, C. (2017). A constraint programming approach for the team orienteering problem with time windows. *Computers & Industrial Engineering*, *107*.

Grossmann, I., & Biegler, L. (2004). Part II. future perspective on optimization. *Computers & Chemical Engineering*, *28*, 1193–1218.

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

He, F., & Qu, R. (2012). A constraint programming based column generation approach to nurse rostering problems. *Computers & Operations Research*, *39*, 3331–3343.

Hooker, J., & van Hoeve, W.-J. (2018). Constraint programming and operations research. *Constraints*, *23*.

Kelareva, E., Tierney, K., & Kilby, P. (2014). CP methods for scheduling and routing with time-dependent task costs. *EURO Journal on Computational Optimization*, *2*, 147–194.

Kelareva, E., Brand, S., Kilby, P., Thiebaux, S., & Wallace, M. (2012). CP and MIP methods for ship scheduling with time-varying draft. *ICAPS 2012 - Proceedings of the 22nd International Conference on Automated Planning and Scheduling*.

Kizilay, D., Eliiyi, D. T., & Van Hentenryck, P. (2018). Constraint and mathematical programming models for integrated port container terminal operations. *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, 344–360.

Kroon, L. G., Lentink, R. M., & Schrijver, A. (2008). Shunting of passenger train units: An integrated approach. *Transportation Science*, *42*(4), 436–449.

Kroon, L. G., Romeijn, H., & Zwaneveld, P. (1997). Routing trains through railway stations: Complexity issues. *European Journal of Operations Research*, *98*, 485–498.

Ku, W.-Y., & Beck, J. (2016). Mixed-Integer programming models for job shop scheduling: A computational analysis. *Computers & Operations Research*, *73*, 165–173.

Kumar, R., Sen, G., Kar, S., & Tiwari, M. K. (2018). Station dispatching problem for a large terminal: A constraint programming approach. *INFORMS Journal on Applied Analytics*, *48*(6), 510–528.

Laborie, P. (2018). An update on the comparison of MIP, CP and hybrid approaches for mixed resource allocation and scheduling. *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, 403–411.

Laborie, P., & Rogerie, J. (2008). Reasoning with conditional time-intervals. *Proceedings of the 21th International Florida Artificial Intelligence Research Society Conference, FLAIRS-21*, 555–560.

Laborie, P., Rogerie, J., Shaw, P., & Vilím, P. (2018). IBM ILOG CP optimizer for scheduling: 20+ years of scheduling with constraints at IBM/ILOG. *Constraints*, *23*, 210–250.

Lentink, R. M. (2006). *Algorithmic decision support for shunt planning* (Doctoral dissertation). Erasmus University Rotterdam. Rotterdam, the Netherlands.

Leung, J. Y.-T. (2004). *Handbook of scheduling: Algorithms, models, and performance analysis*. Chapman & Hall/CRC.

Lin, Z., & Kwan, R. (2013). A two-phase approach for real-world train unit scheduling. *Public Transport*, *6*, 35–65.

Lusby, R., Larsen, J., Ryan, D., & Ehrgott, M. (2011). Routing trains through railway junctions: A new set-packing approach. *Transportation Science*, *45*, 228–245.

Maravelias, C., & Grossmann, I. (2004). A hybrid MILP/CP decomposition approach for the continuous time scheduling of multipurpose batch plants. *Computers & Chemical Engineering*, *28*, 1921–1949.

Nachtmann, H., Mitchell, K., Rainwater, C., Gedik, R., & Pohl, E. (2014). Optimal dredge fleet scheduling within environmental work windows. *Transportation Research Record Journal of the Transportation Research Board*, *2426*, 11–19.

NS. (2019). Nederlandse spoorwegen, 2019 annual report. https://www.nsjaarverslag.nl

NS. (2020). Nederlandse spoorwegen, 2020 annual report. https://www.nsjaarverslag.nl

Otto, A., & Pesch, E. (2017). Operation of shunting yards: Train-to-yard assignment problem. *Journal of Business Economics*, *87*, 465–486.

Pellegrini, P., Marlière, G., & Rodriguez, J. (2014). Optimal train routing and scheduling for managing traffic perturbations in complex junctions. *Transportation Research Part B: Methodological*, *59*, 58–80.

Pellegrini, P., Pesenti, R., & Rodriguez, J. (2019). Efficient train re-routing and rescheduling: Valid inequalities and reformulation of RECIFE-MILP. *Transportation Research Part B: Methodological*, *120*, 33–48.

Pour, S. M., Drake, J., Ejlertsen, L. S., Rasmussen, K., & Burke, E. (2018). A hybrid constraint programming/mixed integer programming framework for the preventive signaling maintenance crew scheduling problem. *European Journal of Operations Research*, *269*, 341–352.

ProRail. (2019a). *Grootste fietsenstalling ter wereld in utrecht geopend* [accessed 23 April 2021]. https://www.prorail.nl/nieuws/grootste-fietsenstalling-ter-wereld-in-utrecht-geopend

ProRail. (2019b). Prorail, 2019 annual report. https://www.jaarverslagprorail.nl

Rahimian, E., Akartunalı, K., & Levine, J. (2017). A hybrid integer and constraint programming approach to solve nurse rostering problems. *Computers & Operations Research*, *82*, 83–94.

Riezebos, J., & Wezel, W. (2009). K-shortest routing of trains on shunting yards. *OR Spectrum*, *31*, 745–758.

Rodriguez, J. (2007). A constraint programming model for real-time train scheduling at junctions. *Transportation Research Part B: Methodological*, *41*, 231–245.

Trick, M. A., & Yildiz, H. (2011). Benders' cuts guided large neighborhood search for the traveling umpire problem. *Naval Research Logistics (NRL)*, *58*(8), 771–781.

van den Broek, J. (2002). *Toets op inplanbaarheid van rangeerbewegingen.* (Master's thesis). Eindhoven University of Technology. Eindhoven, the Netherlands.

van den Broek, J., & Kroon, L. G. (2007). A capacity test for shunting movements. *Algorithmic Methods for Railway Optimization*, 108–125.

van den Broek, J. (2009). *MIP-based approaches for complex planning problems* (Doctoral dissertation). Eindhoven University of Technology. Eindhoven, the Netherlands.

van den Broek, R. (2016). *Train shunting and service scheduling: An integrated local search approach* (Master's thesis). University of Utrecht. Utrecht, the Netherlands.

van Cuilenborg, D. (2020). *Train unit shunting problem, a multi-agent pathfinding approach* (Master's thesis). Delft University of Technology. Delft, the Netherlands.

van Hove, E. (2019). *Train routing at the shunt yard : A disjoint paths approach* (Master's thesis). University of Twente. Enschede, the Netherlands.

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

Wolfhagen, F. E. (2017). *The train unit shunting problem with reallocation.*

Zarandi, M., Sadat Asl, A. A., Sotudian, S., & Castillo, O. (2020). A state of the art review of intelligent scheduling. *Artificial Intelligence Review*, *53*, 501–593.

Zwaneveld, P., Kroon, L., & Hoesel, S. V. (2001). Routing trains through a railway station based on a node packing model. *European Journal of Operations Research*, *128*, 14–33.

Zwaneveld, P., Kroon, L. G., Romeijn, H., Salomon, M., Dauzère-Pérès, S., Hoesel, S., & Ambergen, H. (1996). Routing trains through railway stations: Model formulation and algorithms. *Transportation Science*, *30*, 181–194.

Appendices

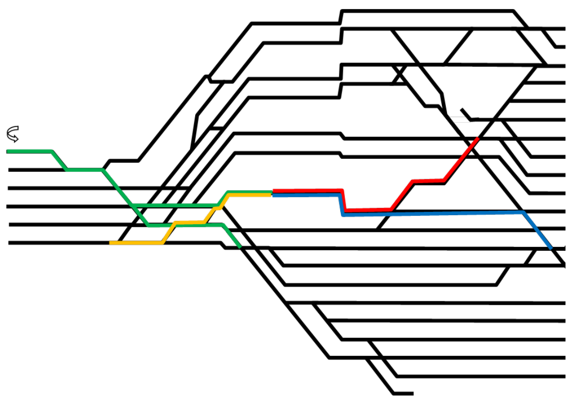
A Description of HIP input parameters

This appendix shows an alphabetically sorted list with a description of every input parameter as shown in Figure 10.

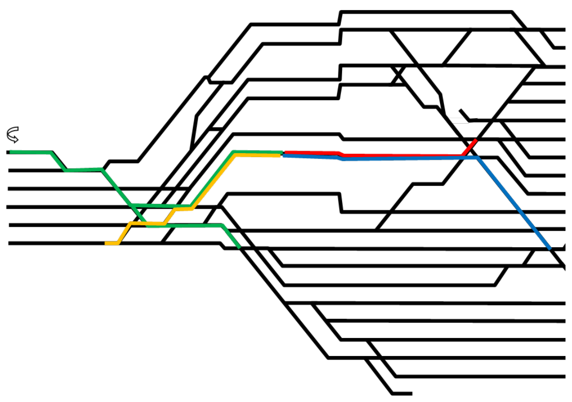
allowedTracks: the allowed tracks to perform the service task.
allowsArrivalDeparture: whether arrival and departure of a train is allowed on the track part. This is the case for all platforms and all outer points of the station area.
arrival: the start time of the movement of nonServiceTraffic (measured from the start time of the scenario).
aSide, bSide: which other track part(s) are connected to the a side and the b side of the track part.
departure: the end time of the movement of nonServiceTraffic (measured from the start time of the scenario).
duration: the duration of the service task in seconds.
ID: the specific identification number of the infrastructure element or train (unit).
isElectrified: whether the track part is electrified (True) or not (False).
length: length of the track part in meters.
movementConstant: constant duration in seconds of each movement. Default setting is 120.
movementSwitchCoefficient: constant duration in seconds per movement over a switch. Default setting is 0.
movementTrackCoefficient: constant duration in seconds per movement over a track. Default setting is 0.
name: name of the track part.
parkingAllowed: whether parking on the track part is allowed (True) or not (False).
parkingTrackPart: the parking track part on which the train arrives or from which the train leaves.
sawMovementAllowed: whether a saw movement on the track part is allowed (True) or not (False).
sideTrackPart: the side from which the train arrives or leaves. It refers to the signal at that side of the track.
standingIndex: whether the train is already in the station / stays at the station (1) or not (0).
time: the time at which a train arrives on or leaves from the parking track part.
type: the type of service task needed to perform, e.g., B-controle, external cleaning.
typeDisplayName: type name of the train, e.g., VIRM-4, ICM-3, FLIRT-3.
usedTrackParts: which track parts (IDs) are used during the movement of nonServiceTraffic.

B Visualised shunt routes

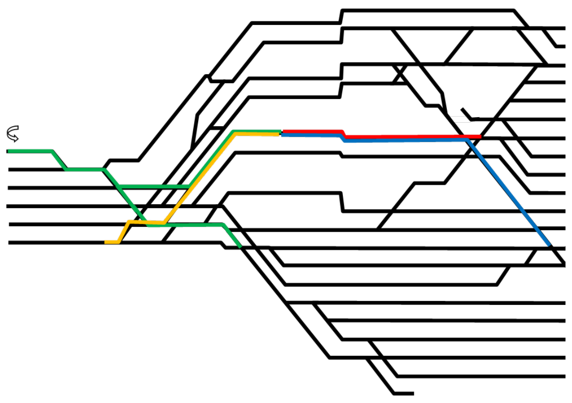
The six images below show all possible shunt routes per platform. The red route goes to Tuin, the blue route goes to Oostzijde via gateway Z5, the green route goes to Oostzijde via reversal track 40 and the yellow track goes to reversal track 35A after which HIP can find a route to Oostzijde.



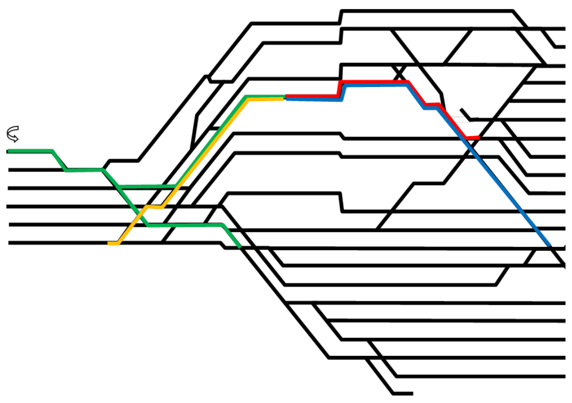
a) Routes from platform 1



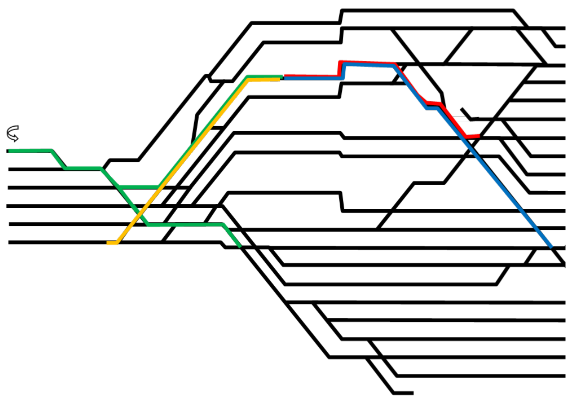
b) Routes from platform 2



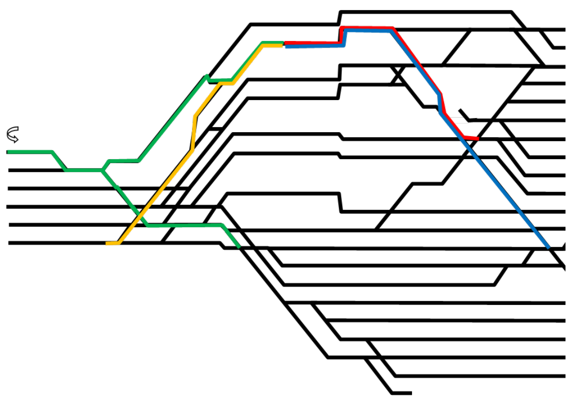
c) Routes from platform 3



d) Routes from platform 4



e) Routes from platform 5



f) Routes from platform 6

C Description of parameters in the model

Name	Description	Set
movementduration	The assumed duration of every movement, set to two minutes	-
lengths	The length in meter of each train unit. In case the train unit is combined, the total length of all combined train units is given	∀N
reversaltimes	The time in minutes for a train unit to make a reversal movement. In case the train unit is combined, the total reversal time of all combined train units is given	∀N
atob	Whether the train unit drives in the direction from the A-side of the station to the B-side of the station (1) or the other way around (0)	∀N
fixedtuin	Whether the train unit has to be shunted to shunting yard "Tuin" (1) or does not necessarily need to go to "Tuin" (0). Shunt train units of type ICR Gvc-Ehv 9 should go to "Tuin", as described in Section 2.4.1	∀N
standing	Whether the train unit is present at the station at the start/end of the 24-hour period (1) or arrives and departs within the 24-hour period (0)	∀N
mintimes	The time in minutes of the total service time of each train unit, i.e., the minimal time a train unit should be in a shunting yard. In case the train unit is combined, the total service time of all combined train units is given	∀N
parent	Whether the train unit is the main train unit in a (combined) train (-1) or it is a child train unit of a parent train unit (number parent train, in set N)	∀N
vrijebaan	Whether the train unit arrives at a platform (0) or arrives on one of the open line tracks at the border of the station without passengers (the number of the border)	∀N
fixedperron	Whether the train unit has a pre-specified platform to go to (number in set P) or does not have a pre-specified platform, usually for *empty stock*-trains (0)	∀N
times	The time at which a train arrives and departs from the station	∀N
fixednodes	The switches that a movement of through trains occupies when arriving or leaving the station	∀F
fixedstarttime	The start time of a movement of through trains	∀F
fixedentime	The end time of a movement of through trains	∀F
fixedparkingperron	The platform to which through trains have to go	∀T
fixedparkingstart	The start time of a parking interval at a platform of through trains	∀T
fixedparkingend	The end time of a parking interval at a platform of through trains	∀T
ShuntRoutes	All possible routes between station and shunting yards, specified by a name, a platform, a shunting yard gateway and the switches used during the movement	∀R
VBRoutes	All possible routes between the borders of the station area and the platforms within the station, specified by a name, a platform, a border *(open line)* number and the switches used during the movement	∀E∀P
routenodesKtoSB	The switches a train uses when driving from reversal track 40 to shunting yard "Oostzijde"	-

D Description of interval variables in the model

Name	**Interval variable for..**	**Characteristics**	**Set**
fixedmovement	The movement of each through train	start=fixedstarttime, end = fixedendtime	$\forall F$
fixedperron	The platform parking time for each through train	start=fixedparkingstart, end=fixedparkingend	$\forall T$
perroninterval1	The platform parking time for each incoming shunt train	Optional, start=arrival time, size $\geq$ *movementduration*	$\forall N$, $\forall P$
perroninterval2	The platform parking time for each outgoing shunt train	Optional, end=departure time, size $\geq$ *movementduration*	$\forall N$, $\forall P$
movement1	The movement from station to shunting yard gateway	Optional, size=movementduration	$\forall N$, $\forall P$
movement2	The movement from shunting yard gateway to station	Optional, size=movementduration	$\forall N$, $\forall P$
route1	The movement from station to shunting yard gateway specifying the shunt route	Optional	$\forall N$, $\forall R$
route2	The movement from shunting yard gateway to station specifying the shunt route	Optional	$\forall N$, $\forall R$
movementVB1	The presence of a movement from the border of the station area to the station	Optional, start=arrival time, size=movementduration	$\forall N$
movementVB2	The presence of a movement from the station to the border of the station area	Optional, end=departure time, size=movementduration	$\forall N$
routeVB1	The movement from the border of the station area to a platform in the station	Optional	$\forall N$, $\forall P$
routeVB2	The movement from a platform in the station to the border of the station area	Optional	$\forall N$, $\forall P$
k40interval1	The time that an incoming shunt train spends at reversal track 40 if routemovement1 leads to reversal track 40	Optional, start=end of movement1, end=start of k40toSB	$\forall N$
k40interval2	The time that an outgoing shunt train spends at reversal track 40 if routemovement2 departs from reversal track 40	Optional, start=end of SBtok40, end=start of movement2	$\forall N$
k40toSB	The movement of an incoming shunt train from reversal track 40 to the shunting yard	Optional, start=end of k40interval1	$\forall N$
SBtok40	The movement of an outgoing shunt train from the shunting yard to reversal track 40	Optional, end=start of k40interval2	$\forall N$

E Model constraints and explanation

This appendix contains all constraints used in the model. First, the constraints are shown and afterwards an explanation per constraint is given.

$$presenceOf(movementVB1_n) \quad if\ vrijebaan1_n \neq 0 \quad \forall N \tag{1}$$

Constraint 1 ensures that, if train n arrive at the border of the station area (i.e., it is *empty stock*), there has to be a movement from the border of the station to a platform (*movementVB1*).

$$alternative(movementVB1_n, [routeVB1_{n,p}\ \ \forall P]) \quad \forall N \tag{2}$$

Constraint 2 ensures that for every incoming empty stock movement, a route from the open line to a platform is chosen.

$$presenceOf(perroninterval1_{n,p}) \quad if\ p = fixedperron1_n \quad \forall N, \forall P \tag{3}$$

Constraint 3 ensures that, if train n has a fixed platform to be located to (*fixedperron1*), the corresponding platform interval is made present in the solution (*perroninterval1*).

$$\begin{aligned} presenceOf(perroninterval1_{n,p}) = presenceOf(routeVB1_{n,p}) \\ if\ vrijebaan1_n \neq 0 \quad \forall N, \forall P \end{aligned} \tag{4}$$

Constraint 4 ensures that, if platform p is chosen in Constraint 2, a platform interval at platform p is reserved for train n.

$$presenceOf(perroninterval1_{n,p}) = presenceOf(movement1_{n,p}) \quad \forall N, \forall P \tag{5}$$

Constraint 5 ensures that the movement for train n from platform p to the shunting yard is made present if train n had a stop at platform p.

$$startAtEnd(movement1_{n,p}, perroninterval1_{n,p}) \quad \forall N, \forall P \tag{6}$$

Constraint 6 ensures that the movement of train n from platform p starts directly after the interval train n is standing on platform p.

$$\begin{aligned} alternative(movement1_{n,p}, [route1_{n,r}\ \ \forall R\ \textit{if route r comes from platform p}]) \\ if\ fixedtuin_n = 0 \quad \forall N, \forall P \end{aligned} \tag{7}$$

Constraint 7 ensures that, if train n is not limited to go to shunting yard Tuin, a route is chosen that starts at platform p. Note that the alternative-constraint only works if $movement1_{n,p}$ is present in the solution.

$$\begin{aligned} alternative(movement1_{n,p}, [route1_{n,r}\ \ \forall R\ \textit{if route r comes from platform p} \\ \textit{and goes to Tuin}\ ])if\ \ fixedtuin_n = 1 \quad \forall N, \forall P \end{aligned} \tag{8}$$

Constraint 8 ensures that, if train n is limited to go to shunting yard Tuin, a route is chosen that starts at platform p and ends at Tuin. Note that the alternative-constraint only works if $movement1_{n,p}$ is present in the solution.

$$\begin{aligned} sizeOf(peroninterval1_{n,p}) \geq reversaltimes1_n \quad if\ atob1_n \\ is\ reverse\ direction\ of\ route1_{n,r} \quad \forall N, \forall P \end{aligned} \tag{9}$$

Constraint 9 ensures that, if the train movement from platform p to shunting yard is in opposite direction of its arrival movement at the platform, the interval it stays at the platform should be at least as long as it takes to make a reversal.

$$\begin{aligned} sizeOf(peroninterval1_{n,p}) \geq movementduration \quad if\ atob1_n \\ is\ same\ direction\ of\ route1_{n,r} \quad \forall N, \forall P \end{aligned} \tag{10}$$

Constraint 10 ensures that, if the train movement from platform p to shunting yard is in the same direction as its arrival movement at the platform, the interval it stays at the platform should be at least a pre-defined movement duration to get passengers off the train.

$$presenceOf(k40interval1_n) \quad if\ route1_{n,r}\ goes\ to\ reversal\ track\ 40 \quad \forall N, \forall R \tag{11}$$

Constraint 11 ensures that, if the route of train n from the platform goes to reversal track 40, the interval train n stays at reversal track 40 is made present in the solution.

$$\begin{aligned} startAtEnd(k40interval1_n, route1_{n,r}) \quad if\ route1_{n,r} \\ goes\ to\ reversal\ track\ 40 \quad \forall N, \forall R \end{aligned} \tag{12}$$

Constraint 12 ensures that the interval train n stays at reversal track 40 starts directly after its arrival at reversal track 40.

$$sizeOf(k40interval1_n) \geq reversaltimes1_n \quad \forall N \tag{13}$$

Constraint 13 ensures that the time train n stays at reversal track 40 is at least as long as it takes to make a reversal.

$$presenceOf(k40toSB_n) = presenceOf(k40interval1_n) \quad \forall N \tag{14}$$

Constraint 14 connects a movement from reversal track 40 to the shunting yard. If train n is standing at reversal track 40 (*k40interval1*), it should have a movement from reversal track 40 to the shunting yard (*k40toSB*).

$$startAtEnd(k40toSB_n, k40interval1_n) \quad \forall N \tag{15}$$

Constraint 15 ensures that the movement from reversal track 40 to the shunting yard starts directly after the interval train n is standing at reversal track 40.

$$presenceOf(movementVB2_n) \quad if\ vrijebaan2_n \neq 0 \quad \forall N \tag{16}$$

Constraint 16 ensures that, if train n departs at the border of the station area (i.e., it is *empty stock*), there has to be a movement from a platform to the border of the station (*movementVB2*).

$$alternative(movementVB2_n, [routeVB2_{n,p}\ \forall P]) \quad \forall N \tag{17}$$

Constraint 17 ensures that for every outgoing empty stock movement, a route from a platform to the open line is chosen.

$$presenceOf(perroninterval2_{n,p}) \quad if\ p = fixedperron2_n \quad \forall N, \forall P \tag{18}$$

Constraint 18 ensures that, if train n has a fixed platform to depart from (*fixedperron2*), the corresponding platform interval is made present in the solution (*perroninterval2*).

$$\begin{aligned} presenceOf(perroninterval2_{n,p}) = presenceOf(routeVB2_{n,p}) \\ if\ vrijebaan2_n \neq 0 \quad \forall N, \forall P \end{aligned} \tag{19}$$

Constraint 19 ensures that, if platform p is chosen in Constraint 17, a platform interval at platform p is reserved for train n.

$$presenceOf(perroninterval2_{n,p}) = presenceOf(movement2_{n,p}) \quad \forall N, \forall P \tag{20}$$

Constraint 20 ensures that the movement for train n from the shunting yard to platform p is made present if train n will depart from platform p.

$$endAtStart(movement2_{n,p}, perroninterval2_{n,p}) \quad \forall N, \forall P \tag{21}$$

Constraint 21 ensures that the movement of train n from shunting yard to platform p ends directly at the start of the interval train n is standing on platform p.

$$\begin{aligned} alternative(movement2_{n,p}, [route2_{n,r} \ \ \forall R \textit{ if route r goes to platform p}]) \\ if\ fixedtuin_n = 0 \quad \forall N, \forall P \end{aligned} \tag{22}$$

Constraint 22 ensures that, if train n is not limited to come from shunting yard Tuin, a route is chosen that ends at platform p. Note that the alternative-constraint only works if $movement2_{n,p}$ is present in the solution.

$$\begin{aligned} alternative(movement2_{n,p}, [route2_{n,r} \ \ \forall R\ if\ route2_{n,r} \textit{ comes from Tuin} \\ \textit{and goes to platform p]) } if\ \ fixedtuin_n = 1 \quad \forall N, \forall P \end{aligned} \tag{23}$$

Constraint 23 ensures that, if train n is limited to come from shunting yard Tuin, a route is chosen that ends at platform p and starts at Tuin. Note that the alternative-constraint only works if $movement2_{n,p}$ is present in the solution.

$$\begin{aligned} sizeOf(perroninterval2_{n,p}) \geq reversaltimes2_n \quad if\ atob2_n \\ \textit{is reverse direction of } route2_{n,r} \quad \forall N, \forall P \end{aligned} \tag{24}$$

Constraint 24 ensures that, if the train movement from shunting yard to platform p is in opposite direction of its departure movement from the platform, the interval it stays at the platform should be at least as long as it takes to make a reversal.

$$\begin{aligned} sizeOf(perroninterval2_{n,p}) \geq movementduration \quad if\ atob2_n \\ \textit{is same direction of } route2_{n,r} \quad \forall N, \forall P \end{aligned} \tag{25}$$

Constraint 25 ensures that, if the train movement from the shunting yard to platform p is in the same direction as its departure movement from the platform, the interval it stays at the platform should be at least a pre-defined movement duration to get passengers on the train.

$$presenceOf(k40interval2_n) \quad if\ route2_{n,r}\ \textit{comes from reversal track 40} \quad \forall N, \forall R \tag{26}$$

Constraint 26 ensures that, if the route of train n to the platform comes from reversal track 40, the interval train n stays at reversal track 40 is made present in the solution.

$$\begin{aligned} endAtStart(k40interval2_n, route2_{n,r}) \quad if\ route2_{n,r} \\ \textit{comes from reversal track 40} \quad \forall N, \forall R \end{aligned} \tag{27}$$

Constraint 27 ensures that the interval train n stays at reversal track 40 ends directly after its departure from reversal track 40 to the station.

$$sizeOf(k40ainterval2_n) \geq reversaltimes2_n \quad \forall N \tag{28}$$

Constraint 28 ensures that the time train n stays at reversal track 40 is at least as long as it takes to make a reversal.

$$presenceOf(SBtok40_n) = presenceOf(k40interval2_n) \quad \forall N \tag{29}$$

Constraint 29 connects a movement from the shunting yard to reversal track 40. If train n is standing at reversal track 40 (*k40interval2*), it should be preceded by a movement from the shunting yard to reversal track 40 (*SBtok40*).

$$endAtStart(SBtok40_n, k40interval2_n) \quad \forall N \tag{30}$$

Constraint 30 ensures that the movement from the shunting yard to reversal track 40 ends directly after the start of the interval train n is standing at reversal track 40.

$$synchronize(routeVB1_{n,p}, routeVB1_{parent1_n,p}) \quad if\ parent1_n \neq -1 \quad \forall N, \forall P \tag{31}$$

$$\begin{aligned} synchronize(perroninterval1_{n,p}, perroninterval1_{parent1_n,p}) \\ if\ parent1_n \neq -1 \quad \forall N, \forall P \end{aligned} \tag{32}$$

$$synchronize(route1_{n,r}, route1_{parent1_n,r}) \quad if\ parent1_n \neq -1 \quad \forall N \tag{33}$$

$$synchronize(k40interval1_n, k40interval1_{parent1_n}) \quad if\ parent1_n \neq -1 \quad \forall N \tag{34}$$

$$synchronize(k40toSB_n, k40toSB_{parent1_n}) \quad if\ parent1_n \neq -1 \quad \forall N \tag{35}$$

$$synchronize(routeVB2_{n,p}, routeVB2_{parent2_n,p}) \quad if\ parent2_n \neq -1 \quad \forall N, \forall P \tag{36}$$

$$synchronize(perroninterval2_{n,p}, perroninterval2_{parent2_n,p}) \\ if\ parent2_n \neq -1 \quad \forall N, \forall P \tag{37}$$

$$synchronize(route2_{n,r}, route2_{parent2_n,r}) \quad if\ parent2_n \neq -1 \quad \forall N \tag{38}$$

$$synchronize(k40interval2_n, k40interval2_{parent2_n}) \quad if\ parent2_n \neq -1 \quad \forall N \tag{39}$$

$$synchronize(SBtok40_n, SBtok40_{parent2_n}) \quad if\ parent2_n \neq -1 \quad \forall N \tag{40}$$

Constraints 31 - 40 make sure that, if train n in part of a combined train and train n is not the main train unit ($parent_n \neq -1$), all intervals are synchronised with the main train unit of the combined train ($parent_n$).

$$min(SBtok40_n, [route2_{n,r} \forall R]) - max(k40toSB_n, [route1_{n,r} \forall R]) \geq \\ mintimes_n + movements\ depending\ on\ entry\ and\ exit\ from\ shunting\ yard \quad \forall N \tag{41}$$

Constraint 41 ensures that the time between entering and leaving the shunting yard is bigger than the service time at the shunting yard (*mintimes*) and additional time depending on the movements and reversals to be made. Note that the interval leaving the shunting yard is *SBtok40* if the route goes to reversal track 40 and *route2* if the route goes directly to a platform. The same holds for the interval for entering the shunting yard.

$$sum([presenceOf(route1_{n,r})\ \forall R\ if\ route1\ goes\ to\ Tuin]) = \\ sum([presenceOf(route2_{n,r})\ \forall R\ if\ route2\ comes\ from\ Tuin]) \quad \forall N \tag{42}$$

Constraint 42 ensures that, if a train is directed to shunting yard Tuin, it also choose a route that leaves from shunting yard Tuin. This immediately implies the same for shunting yard Oostzijde.

$$presenceOf(route1_{n,r}) = 0 \quad if\ route\ r\ goes\ to\ reversal\ track\ 40 \\ and\ train\ length > 366 \quad \forall R, \forall N \tag{43}$$

The length of reversal track 40 is 366 meters. Constraint 43 ensures that trains longer than 366 meters cannot go to reversal track 40.

$$presenceOf(route1_{n,r}) = 0 \quad if\ route\ r\ goes\ to\ gateway\ Z5 \\ and\ train\ length > 283 \quad \forall R, \forall N \tag{44}$$

The length of the reversal track in Oostzijde that you need to use if you enter Oostzijde via gateway Z5, is 283 meters. Constraint 44 ensures that trains longer than 283 meters cannot enter Oostzijde via gateway Z5.

$$presenceOf(route2_{n,r}) = 0 \quad if\ route\ r\ comes\ from\ reversal\ track\ 40 \\ and\ train\ length > 366 \quad \forall R, \forall N \tag{45}$$

The length of reversal track 40 is 366 meters. Constraint 45 ensures that trains longer than 366 meters cannot come from reversal track 40.

$$presenceOf(route2_{n,r}) = 0 \quad \begin{array}{l} \textit{if route r comes from gateway Z5} \\ \textit{and train length} > \textit{283} \quad \forall R, \forall N \end{array} \tag{46}$$

The length of the reversal track in Oostzijde that you need to use if you leave Oostzijde via gateway Z5, is 283 meters. Constraint 46 ensures that trains longer than 283 meters cannot leave Oostzijde via gateway Z5.

$$noOverlap(nodesequence_v) \quad \textit{if } v \neq \textit{36} \quad \forall V \tag{47}$$

Constraint 47 ensures for every switch (except switch 36) that there is no overlap of the interval variables using that switch. In other words, this constraint prevents trains from using the same switch at the same time.

$$noOverlap(nodesequence_v, [[0,0],[0,x]]) \quad \textit{if } v = \textit{36} \quad \forall V \tag{48}$$

Constraint 48 is similar to Constraint 47. The difference is that, besides overlap is not allowed between all interval variables, there should be a gap of at least x minutes between all interval variables of type 1 using switch 36. In other words, this constraint prevents all trains from using switch 36 at the same time and ensures a time of x minutes between all incoming shunt trains (defined as type 1). This is consistent with the ninth requirement in Section 4.6.

$$noOverlap(perronsequence_p) \quad \forall P \tag{49}$$

Constraint 49 ensures for every platform in the station that all interval variables linked to platform p ($nodesequence_p$) cannot overlap. In other words, only one train at the time can stand at a platform.

$$noOverlap(k40sequence) \tag{50}$$

Constraint 50 ensures for reversal track 40 that all interval variables linked to reversal track 40 ($k40sequence$) cannot overlap. In other words, only one train at the time can stand on reversal track 40.

F Conversion to HIP input

To be able to convert the output of the constraint programming model to an input for HIP, some adjustments have to be made. This Appendix will explain the steps needed for conversion to HIP input. In Section 2.4.1, the configuration of the two input files is explained and Appendix A describes the input parameters in more detail. Both input files (scenario and location), need to be adjusted for a valid link with HIP.

F.1 Scenario file

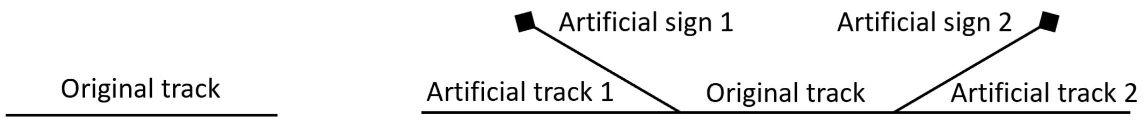
The first addition to the scenario file, is the route each shunt train takes from the station to the shunting yard and vice-versa. This is the output from the CP model. Each route is characterised by a start time, an end time and the used infrastructure elements. The start time and end time can easily be extracted from the output of the constraint programming model. However, the used infrastructure elements cannot be extracted directly. As explained in Section 4.11, only switches are part of the constraint programming model, whereas HIP needs all infrastructure elements to be in the route. By manually defining all infrastructure elements per route, the chosen route can easily be linked to the used infrastructure elements and added to the scenario file together with the start time and end time. Note that if a shunt train goes to the shunting yard via reversal track 40, two routes should be added; one route from the station to reversal track 40 and one route from reversal track 40 to the shunting yard.

The second adaptation to the scenario file, is to change the location and time of arriving shunt trains. In the original scenario file, shunt trains arrive either at a platform or at the side of the station area. In the adapted scenario file, shunt trains should arrive at the border of the shunting yard. To accomplish this, three elements need to be adapted: parkingTrackPart, sideTrackPart and time. The element time describes at which time the shunt train arrives. The elements parkingTrackPart and sideTrackPart respectively describe on which infrastructure element the shunt train arrives and from which side (described by a railway signal) the shunt train arrives at the parkingTrackPart.

The last adaptation is to add the matching which the constraint programming model used as input. This means that for every outgoing shunt train, the train unit ID of the matched incoming train unit should be added.

F.2 Location file

In the original location file, all platforms are configured such that shunt trains can arrive and depart from the platforms. As the constraint programming model requires shunt trains to arrive and depart from the border of the shunting yard, the tracks at the entrance gateways to the shunting yard should be configured the same as the platforms. Two things should be added or adapted: i) artificial railway signals should be added, and ii) artificial tracks should be added. Figure 25 gives an overview of the layout with added artificial tracks and railway signals and an example of the implementation into the scenario file.



(a) Original layout. (b) Adapted layout.

Figure 25: The original and adapted layout for the tracks at the entrance gateways to the shunting yards. For example, a shunt train follows a route from the station to the shunting yard that ends at *Artificial track 2*. In this case, in the scenario file, the parkingTrackPart and sideTrackPart would have to be be adapted to respectively *Original track* and *Artificial railway signal 2*.

G Generalisation of the model

This Appendix describes how the model can be generalised to be used for other stations than Eindhoven. It also describes what steps need to be taken in the current CP model to program other stations.

In general, two things need to be adjusted when modelling other stations: the model and the input data. Figure 26 shows the steps needed to perform to change both the model (left) and the input data (right). Note that first the steps of the location should be performed before creating the input data, as the conversion to the input data depends on which infrastructure elements are incorporated in the model.

The first step when modelling a new location, is to determine the numbering of the open line gateways. Those are the gateways at the side of the station area where empty stock trains arrive. The second step is to determine the station area, the shunting yard area(s) and the reversal tracks used within the shunting yard(s) of the new location. The station area is usually defined by the platforms that are input for the shunt routes and the empty stock routes. The reversal tracks are important for the next step. In the next step, the gateways to the shunting yard are chosen. The gateways to the shunting yard are based on the shunting yard area(s) and are input for the shunt routes. In some cases, an incoming shunt route could goes via a reversal track to the shunting yard. If this reversal track is outside the shunting yard area, we can incorporate this in the routes. If the reversal track is within the shunting yard, the gateway should be placed just before the reversal track so that shunt routes from the station to the gateway will end at the reversal track. This way, HIP can determine for how long the shunt trains stay on the reversal track depending on other trains within the shunting yard that want to use the reversal track for movements within the shunting yard.

The next step is to determine the shunt routes and empty stock routes. The shunt routes are defined by a starting platform, an ending shunting yard gateway and the used infrastructure. The empty stock routes are defined by a starting open line gateway, an ending platform and the used infrastructure. Based on the station area, the shunting yard area(s), the shunt routes and the empty stock routes, one should determine which infrastructure elements of the hub to include in the model as explained in Section 4.11. This selection typically consists of the switches of the station area and the area between the station area and the shunting yard(s). Also important reversal tracks outside the shunting yard are taken into account. Three steps can be performed in parallel. The first is to check which reversal tracks are a limitation in terms of track length. If a reversal track is shorter than the longest possible train composition, it should be taken into account in the model. Once the gateways to the shunting yard are defined (step 2), one can also start defining the minimal time that is needed for a shunt train to make movements and reversals within the shunting yard (see Figure 20). This time should not yet include the service time, which is automatically added in the model. The third parallel step is to determine if certain follow-up times between shunt trains on gateways should be incorporated in the model. For example to ensure no crossing within the shunting yard or to make it possible for a shunt train to reverse in the shunting yard. The last step is to include all input of the previous steps in the model where the code is commented to add changes.

Once the new location is modelled, the input data can be configured from the scenario file. This process has just one step, which is to check whether specific shunt trains need to be shunted to specific shunting yards. For example, in the case of Eindhoven, all trains of type ICR Gvc-Ehv need to be shunted to shunting yard Tuin. Also, external cleaning can only be performed in shunting yard Oostzijde, so shunt trains needing external cleaning should go to Oostzijde. Once this is incorporated in the input data, the Python script that converts the scenario to input data can be executed.

To generalise the model to other stations, a recommendation (see Section 7.2) can be followed that mitigates trains from the model that are hard to capture by a general model. The trains to mitigate from the model are shunt trains that does not follow the flow as presented in Figure 19. As the model is based on this flow, these trains are not captured in the model and should be modelled as an exception. An example of a train that does not follow the routing flow, are trains that arrive at a platform and have too little time to be shunted to a shunting yard, so they rather stay at the platform. Another example of shunt train to mitigate from the model, are trains that are already present at the start of the 24-hour horizon, or are still present at the end of the 24-hour horizon. These trains only partially follow the flow of Figure 19 and should therefore also be modelled as an exception. Mitigating these two types of shunt trains from the model prevents manual additions to the model to capture these exceptions.

Another part of the generalization, would be to be able to convert the preferred routes from Donna to an array of used infrastructure used in the location files. This way, only the start and end point of the shunt routes and empty stock routes have to be determined manually and the used infrastructure could be extracted from Donna. However, this is not possible yet and should be developed. This connection could also be used in HIP, as we recommend to use the preferred fixed routes from Donna in the local search as well.

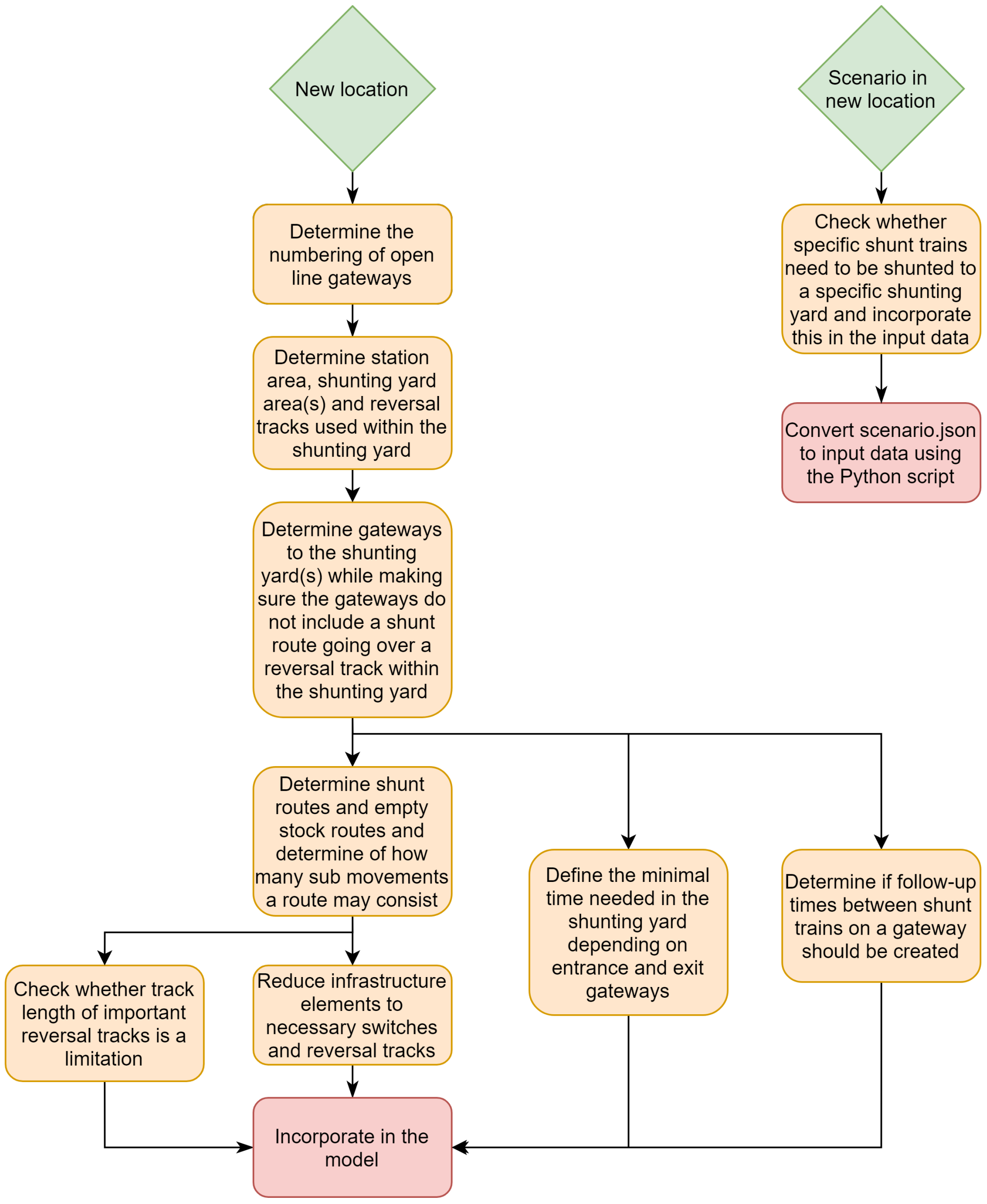

New location
Determine the numbering of open line gateways
Determine station area, shunting yard area(s) and reversal tracks used within the shunting yard
Determine gateways to the shunting yard(s) while making sure the gateways do not include a shunt route going over a reversal track within the shunting yard
Determine shunt routes and empty stock routes and determine of how many sub movements a route may consist
Check whether track length of important reversal tracks is a limitation
Reduce infrastructure elements to necessary switches and reversal tracks
Define the minimal time needed in the shunting yard depending on entrance and exit gateways
Determine if follow-up times between shunt trains on a gateway should be created
Incorporate in the model
Scenario in new location
Check whether specific shunt trains need to be shunted to a specific shunting yard and incorporate this in the input data
Convert scenario.json to input data using the Python script

Figure 26: The process flows of customizing the current model to a specific location (left) and the process flow of creating the input data from a new scenario file (right).