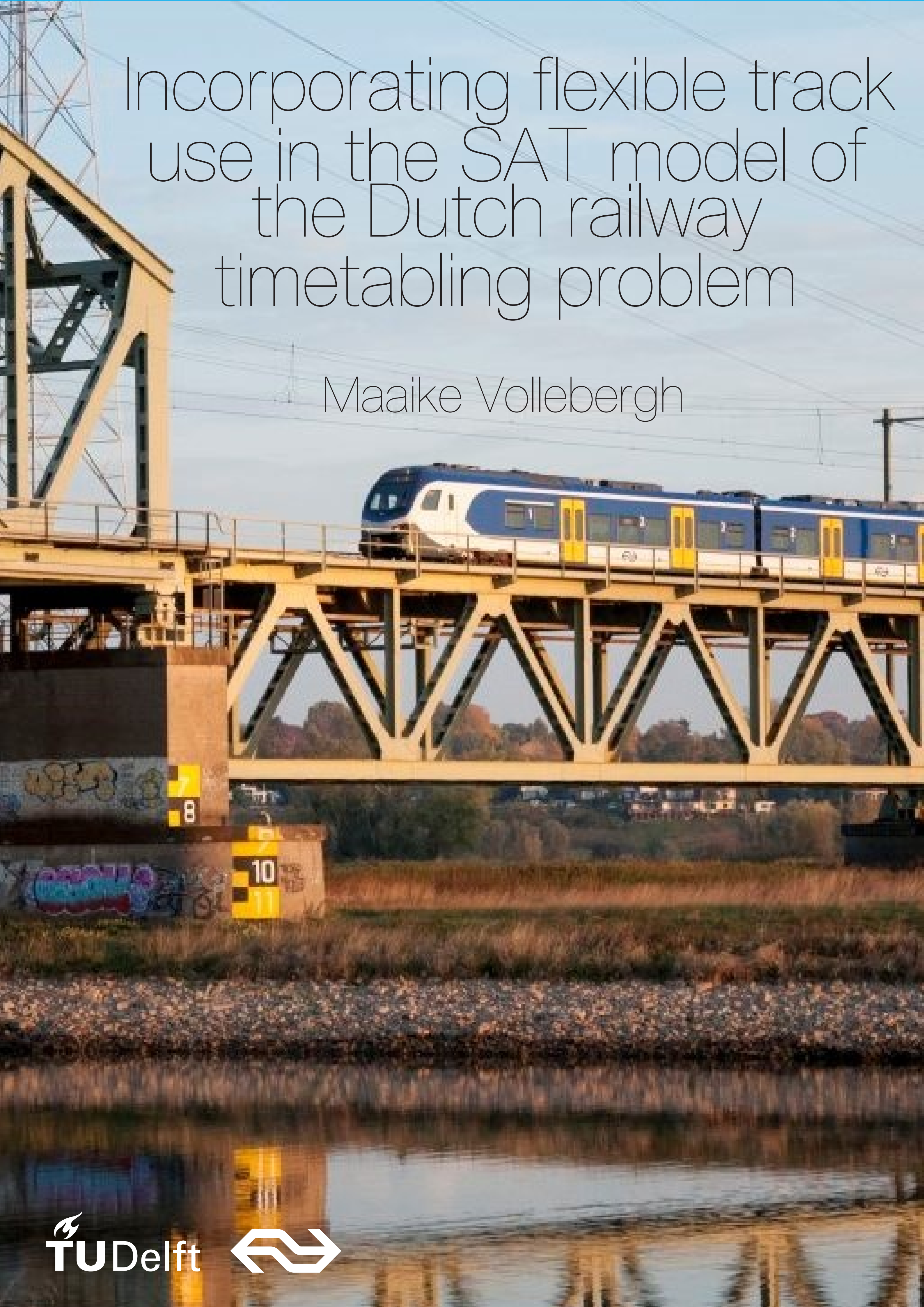


Incorporating flexible track use in the SAT model of the Dutch railway timetabling problem

Maaike Vollebergh

Incorporating flexible track use in the SAT model of the Dutch railway timetabling problem

by

Maaike Vollebergh

to obtain the degree of

Master of Science
in
Applied Mathematics

with a specialisation in
Optimisation

at the Faculty of Electrical Engineering, Mathematics and Computer Science
of Delft University of Technology.

To be defended publicly on Friday September 25, 2020 at 2:00 PM.

Student number:	4972023	
Project duration:	January 13, 2020 – September 25, 2020	
Thesis committee:	Prof. dr. ir. K. I. Aardal,	TU Delft
	Dr. D. de Laat,	TU Delft, supervisor
	Prof. dr. D. Huisman,	Erasmus University Rotterdam & Netherlands Railways
	P. J. Fioole,	Netherlands Railways, daily supervisor

;

Abstract

The construction of cyclic railway timetables is an important task for Netherlands Railways (NS).This construction can be formulated as a Periodic Event Scheduling Problem (PESP). The most powerful technique for solving cyclic railway timetabling problems is constraint programming, especially via SAT solvers when PESP instances are encoded as SAT instances. SAT solvers can determine the feasibility of problem instances of NS quickly and reliably. However, in previous implementations the problem specification must explicitly indicate the track use within stations and on four-track sections. As a result, the solver also reports infeasibility if a small adjustment of the track allocation could lead to a feasible timetable. In this thesis, the Open Periodic Event Scheduling Problem (OPESP) is introduced, which is used in a new method to incorporate flexible track use in the SAT formulation. This method yields promising results that could help improve the timetabling process at NS.

Preface

This thesis marks the end of my graduation project, carried out at NS, which is the final requirement to obtain my Master's degree in Applied Mathematics. I would like to thank some people for their contribution to this thesis.

First of all, I would like to thank my daily supervisor Pieter-Jan Fioole from NS and my supervisor David de Laat from TU Delft. Pieter-Jan, our weekly meetings helped me to stay focused and make continuous progress. David, your suggestions and accurate proofreading were highly appreciated. It was great to work with both of you. I also would like to thank Karen Aardal and Dennis Huisman, for taking place in my graduation committee.

Being a graduation intern at NS was an insightful experience. I would like to thank my colleagues for the interesting talks about the projects they are working on. Especially, I would like to thank the other interns. Our weekly online meetings, which always ended up in playing games, were a welcome variety from working on my thesis.

As a large part of my thesis is written from my parents' place, I would like to thank my parents, my brother and my sister for the special time during our lockdown together and for all the support. Working from home was hard sometimes, but you helped me through. Last but not least, thank you Martijn, for encouraging me and knowing the right things to say.

Maaike Vollebergh
Amsterdam, September 2020

Contents

1 Introduction

Netherlands Railways (Nederlandse Spoorwegen, NS) is the largest provider of mobility services in the Netherlands. Approximately 62% of all people in the Netherlands use the services of NS on a yearly basis, with 10.7 million unique passengers every year [1]. In 2019, NS provided approximately 1.3 million train journeys every day.[1]

NS holds a franchise for the main rail network of the Netherlands. The franchise authority, which is the Ministry of Infrastructure and Water Management, will evaluate the performance of NS in midterm reviews. Aspects such as customer satisfaction, punctuality, personal safety and seat availability are the focus of these reviews [1]. The aim of NS is therefore to maintain a high level of performance and to achieve further growth and quality improvements. The challenge is to offer higher frequencies, more trains and longer trains, while the capacity limits of the infrastructure are approached [1].

The challenge of meeting the needs of transport within the infrastructure capacity is captured in the complicated process that precedes the transport of passengers. This process consists of the construction of a timetable, the planning of rolling stock and the planning of personnel on the trains. In this thesis, the problem of constructing a timetable is discussed and a method to incorporate flexible track use in the SAT model of this problem is presented and tested. In the upcoming sections the planning process at NS and the reasons why this is complicated are discussed.

1.1. The Planning Process at Netherlands Railways

The planning process at NS consists of several stages. The first step of the process is the line planning. In this stage train lines are specified by their routes and subsequent stops, as well as by their hourly frequency [3]. This is a decision for about twenty years and it is made five years in advance. The line planning is based on the infrastructure and a trade-off between desirability for passengers and operational costs for NS, as maximizing the number of direct travelers results in an inefficient use of rolling stock [3].

[1]Because of COVID-19, passenger numbers have fallen to only 40% of the original number of passengers and expectations are that passenger numbers will be back at the level of 2019 after 2024 [2].

Based on the line plan a timetable is constructed. The timetable of NS is cyclic with a cycle time of an hour, which means that the timetabling process starts with the design of a basic hour pattern. The setting at the end of this pattern must agree with the setting at the start of the pattern. In this way the basic hour pattern could be repeated to create a timetable for a longer period. The design of a basic hour pattern is done a year in advance. Around four weeks in advance this pattern is used to create specific day patterns. Here, for example, maintenance on the infrastructure, for which trains have to be rescheduled, or events, for which extra trains are needed, are taken into account. In this thesis, the construction of a basic hour pattern is discussed.

The next stage consists of the rolling stock scheduling, after which the crew is scheduled. In rolling stock scheduling, the length of the trains is determined. This is done before crew scheduling, as the costs for rolling stock are much higher than crew costs [1]. An extensive description of the planning process can be found in [4]. Crew scheduling consists of assigning duties to the personnel. For this assignment, there are labor rules to take into account. A detailed description of the crew planning at NS can be found in [5].

For timetabling, as well as rolling stock and crew scheduling, changes in real-time could be needed when, for example, accidents take place. We now discuss why the construction of a basic hour pattern is a complex problem.

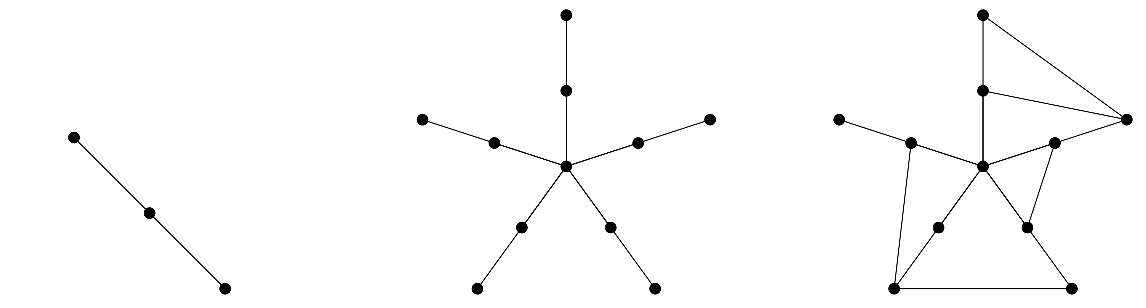
1.2. Complexity of the Dutch Railway Timetabling Problem

There are three main reasons why the Dutch railway timetabling problem is a complex problem: the density of the infrastructure, the alternation of the line planning and the planning principles of NS. In this section, we discuss these reasons in more detail.

1.2.1. Infrastructure

The infrastructure of a train network corresponds to a network structure. Figure 1.1 shows examples of different network structures. For a bus network, as in Figure 1.1a, it is easy to construct a timetable, as all stations are connected in just one line. For a star network, as in Figure 1.1b, it is also relatively easy to construct a timetable. All train lines start or end at the central node, but apart from this node they are independent from each other. Therefore the departure times of the trains have to match at this central node and the rest of the timetable can be derived from these departure times. The infrastructure in Denmark is an example of this.

The infrastructure in the Netherlands looks more like a hybrid network, of which a small example is shown in Figure 1.1c. A hybrid network is a combination of two or more other network types, in case of this example a star network and multiple bus networks. The infrastructure in the Netherlands is and will be created to match the demand of the passengers and to have travel times that are as low as possible. This results in a dense network that contains a lot of cycles. These cycles break the independence of the different lines as in a star network, making the construction of a timetable more difficult. This is because cycles in the infrastructure result in cycles of constraints between departure times of trains. Suppose we traverse such a cycle between departure times and add up the difference between the departure times. Then, to prevent a conflict for these times, the sum of the differences should be a multiple of the timetable's period. As stated in [6], these cycles between constraints are the challenging part in periodic scheduling.



(a) Example of a bus network (b) Example of a star network (c) Example of a hybrid network

Figure 1.1: Examples of networks

1.2.2. Line planning

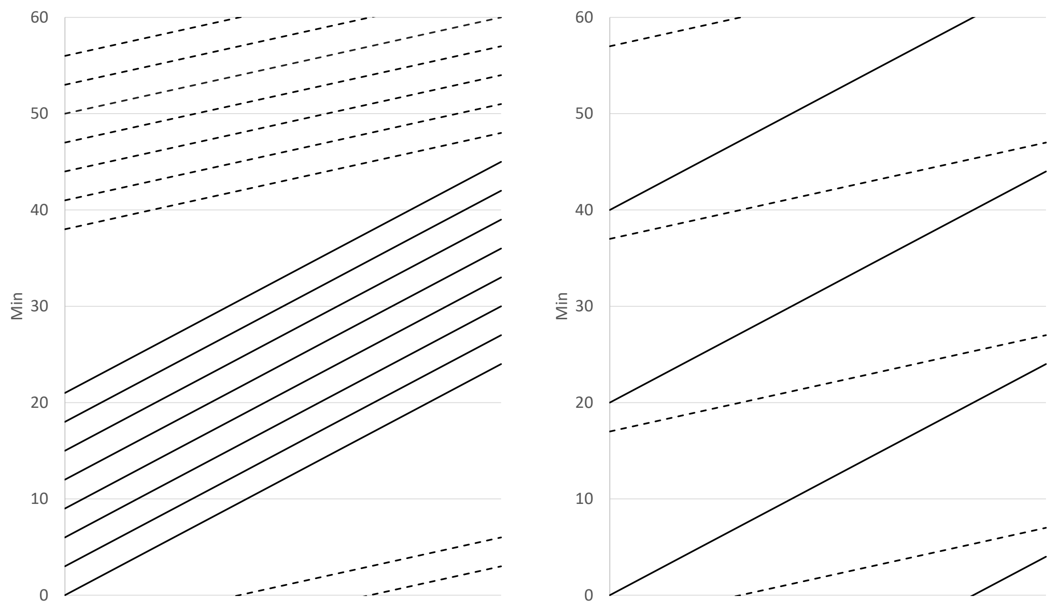
Another aspect that makes the Dutch railway timetabling problem a complex problem is the alternation of the line planning. This alternation emerges on two main points: the types and the routes of trains.

First, the type of the trains in the line planning should be alternating. In the Netherlands, there are three types of trains: sprinter trains, intercity trains and freight trains. Sprinter trains stop at each station on their route, while intercity trains only stop at particular stations. Freight trains are slow compared to passenger trains. Figure 1.2 shows two time-space diagrams, where the horizontal axis represents space, with a difference in track use for two types of trains with a different speed. In Figure 1.2a, the trains are grouped by type. In this way, the follow-up time between trains of the same type can be low, and therefore 15 trains can be scheduled. However, this does not give the best service to the customers. If one needs a train from one of the types, there is just a small period of time where trains go regularly and a risk of having to wait for a long period of time.

In Figure 1.2b, the trains are alternately driven per type. On the one hand, this is a more attractive timetable to customers as trains of the same type are now spread over the hour. On the other hand, only six trains can be scheduled in this way. The bigger the difference in speed between trains, the lower the capacity of a track. When the capacity of the track is lower, it is more difficult to schedule trains in case of a high frequency. Therefore, alternating the type of train lines contributes to the complexity of the Dutch railway timetabling problem.

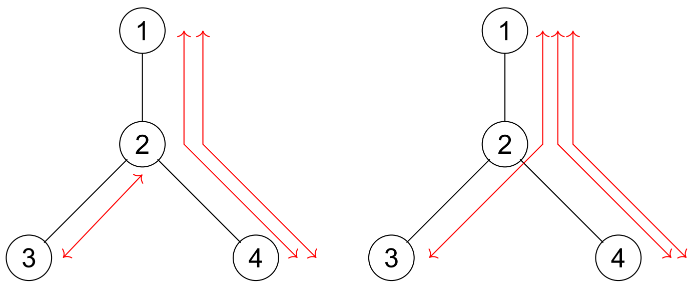
Second, the routes of the train lines should be alternating. An example is shown in Figure 1.3. Figure 1.3a is an example of a homogeneous line planning. On each part of the network, train lines have the same route: each train departing from node 1 is going to node 4 via node 2, for example. In Figure 1.3b, two train lines with alternating routes are shown. From node 1 to node 2, both lines have the same route, after which they split up. In this way, direct connections are possible between node 1 and node 3 and between node 1 and node 4.

In both cases, we want an equal distribution of trains over the hour between each pair of nodes. In the first case, the train between node 2 to node 3 is separately and the two trains between node 1 and node 4 can be scheduled 30 minutes apart. In the second case, the three trains between node 1 and node 2 should be scheduled 20 minutes apart. However, the two of those trains between node 2 and node 4 should be scheduled 30 minutes apart. Therefore it is not possible to schedule these trains with the right frequencies. This example illustrates that alternating the routes of train lines contributes to the complexity of the Dutch railway timetabling problem.



(a) Example of a time-space diagram with grouped trains by train speed (b) Example of a time-space diagram with alternating trains by train speed

Figure 1.2: Examples of track capacity



(a) Example of homogeneous routes (b) Example of alternating routes

Figure 1.3: Examples of line planning

1.2.3. Planning principles of NS

NS has certain planning principles that contribute to the complexity of the railway timetabling problem: the resulting timetable should be completely free of conflicts and should contain travel times that are as low as possible. In some other countries, for example, it is allowed to 'bend' trains (*uitbuigen* in Dutch). This means that a train has a lower speed than its maximum speed on some part of its route. In this way, the speed difference between trains can be lowered, which results in a higher track capacity (see Figure 1.2). However, as 'bending' of trains results in longer travel times in many cases, it is not allowed by NS in a conflict-free timetable.

1.3. Timetabling at Netherlands Railways

In 1969, NS introduced a new timetable to make the train more competitive [7]. The timetable was cyclic with a fixed line planning, better connections and increased frequencies. Until now, railway timetables in the Netherlands satisfy these properties, although frequencies are still increasing.

A cyclic timetable has the advantage to be transparent to the customer, as there is no need for passengers to remember complex timetables and there are no gaps in the train service throughout the day [8]. This advantage makes a cyclic timetable also relatively easy to handle within the different departments of NS.

Until 1992, timetabling was done by hand [7]. Automatic tools did not exist and frequencies were still low. But as the requirements for the timetable were increasing by creating more connections and increasing frequencies of train lines, the need for a decision support system increased. Therefore, NS started to create DONS ('Designer of Network Schedules) [9]. First, DONS was used in combination with the solver CADANS [10]. Based on the infrastructure and the line planning, a model is built in DONS, including the frequency constraints and connections. Here, the track that each train uses between stations should be defined as well. In case a solution for this rail network exists, CADANS can generate a timetable. A timetable generated by DONS and CADANS is used to create the Dutch railway timetable of 2007. In subsequent years, the Dutch railway timetable was based on this timetable.

The problem of constructing a timetable became more and more difficult in the past years, as frequencies are rising up to 6 trains per hour and better connections are sought. This results in an overdetermined problem, which CADANS reports as unsatisfiable. However, human planners are able to solve this problem with only small deviations such as changing track assignment, omitting some connection constraints or shortening a follow-up time when this is allowed. Therefore, the construction of the timetable is nowadays still mainly a human planning process, although it is supported by computer aided design tools [8]. DONS is now used as a support system, mainly for conflict signaling. Time-space diagrams can be drawn in DONS and these diagrams show if conflicts occur.

1.4. Problem Description

Although human planners are able to solve the Dutch railway timetabling problem with only small deviations, the solver CADANS reports unsatisfiability for this problem. As described above, the track that each train uses should be predefined to use these systems. This means a fixed detailed route for each train is used. The following example illustrates that this implies a lot of restrictions and could easily result in an unsatisfiable problem.

Example 1.4.1. Let us consider a railway network with three stations, station 1, 2 and 3 respectively. Station 1 has one platform, station 2 has two platforms and station 3 has one platform. Therefore there are two routes from station 1 to station 2, say route

α and route β and from station 2 to station 3 there are two routes as well, say route γ and route δ. Say there is a train standing still at the second platform of station 2. Figure 1.4 is a representation of this network and situation.

Say there is another train, which should be scheduled from station 1 to station 3 and which has a fixed route going on route β and δ. In this case, a solver would consider the problem *unsatisfiable*, while it is easy to see that we can choose an alternative route for this train to make the problem satisfiable.

The example above shows that a timetabling problem that is considered unsatisfiable using fixed routes might be sastisfiable when using another combination of detailed routes. This might be easy to see in a small example as above, but in an extensive network as the Dutch railway network support is needed.

In this thesis, we want to develop an alternative method for constructing timetables that is able to incorporate multiple route options. We consider the problem of constructing a cyclic railway timetables as a satisfiability problem. Therefore we don't use an objective function, but we try to find a feasible timetable rather than the best timetable. When incorporating multiple route options for each train instead of using a preferred route, a larger part of the solution space is considered and this might make it possible to find a feasible timetable for the Dutch railway timetabling problem.

As the currently most efficient approach for solving timetabling problems is conducted by using state-of-the-art SAT solvers [11], the goal of this thesis is to **incorporate flexible track use in the SAT formulation of the cyclic railway timetabling problem**.

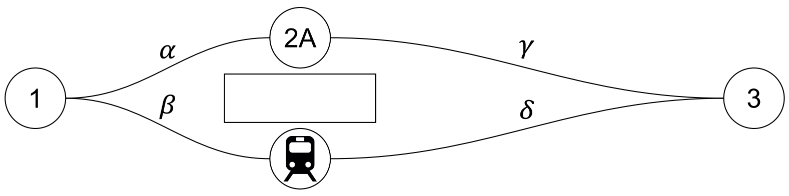


Figure 1.4: Example of a railway network

1.5. Literature Review

The construction of a cyclic railway timetable is often modeled as a Periodic Event Scheduling Problem (PESP), introduced in [12]. PESP will be described in Chapter 2. It is an NP-complete problem [12], for which currently the most efficient solving approach is to encode a PESP as a SAT problem and to use state-of-the-art SAT solvers [11]. SAT will be described in Chapter 3. It is an NP-complete problem as well [13], however, very efficient solvers in practice exist [14].

The SAT approach is used in several articles to solve PESP instances, where the track allocation is fixed. A polynomial reduction from PESP to SAT is presented in [11]. In [14], the algorithms that are implemented in the modular timetabling software system TAKT are described, where a SAT approach is combined with methods for automatic resolving of conflicts. The problem of finding a periodic timetable and the problem of finding optimal passengers' paths are integrated in [15], to improve the quality of the resulting timetables. In [16] a binary search procedure which uses a

SAT solver is proposed. A linear objective function is used, which is modeled as a constraint where the total sum of the objective function is bounded by an integer constant. In this way global minimal solutions with respect to travel time can be found. Finally, promising results are presented in [17], using an approach based on reinforcement learning, multi-agents and SAT.

Railway track allocation is a separate process and it is evident that this is an integral component in the problem of timetabling [18]. An extensive overview of the problem of routing trains through railway junctions in more detail and the different solution methods for this problem is given in [18]. The most widely used models for this routing problem are conflict graph approaches, while there is potential in adopting resource based constraint systems [18]. In [19], the complexity of the timetabling problem, solved by a SAT approach, is reduced by ignoring selected minimum headway constraints. In a post-process, the optimised track allocation is determined. It turns out that this approach is useful and fast, although it it not always possible to calculate a feasible solution in the post-process [19].

The construction of the timetable and the railway track allocation are mostly performed separately. It is suggested to integrate these two problems to investigate potential additional benefits [18]. In [20], the literature's first mixed-integer linear programming model of a cyclic, combined train timetabling and platforming problem is presented for a simple network. Here, a simple network is considered to consist of a single-track main line with discrete stations that each contain one or more sidings in parallel. According to [20] there are no papers that consider cyclic, combined train timetabling and routing on a more complex network, which, to the best of our knowledge, is still the case.

As is stated in [14], the improvements in SAT-based PESP solving permit additional extensions to the PESP model, such as dynamic track allocation for passenger trains. According to [16], “the potential of SAT approaches is not yet fully explored”.

1.6. Outline

In this thesis, an implementation for incorporating multiple route options in the cyclic railway timetabling problem is proposed. First of all, the problem of constructing a cyclic railway timetable while using fixed routes will be presented as a Periodic Event Scheduling Problem (PESP) in Chapter 2. Next, the SAT Problem and the encoding from PESP to SAT will be discussed in Chapter 3. In Chapter 4, the method for incorporating multiple route options is presented. The results of experiments with this method are shown in Chapter 5. In Chapter 6, the results are discussed and recommendations for further research are done.

2
The Periodic Event Scheduling Problem

In this chapter we first give a formal definition of the Periodic Event Scheduling Problem (PESP) and discuss some properties of PESP constraints. Second, we will explain the details of this problem in case of cyclic railway timetabling when using a fixed route for each train.

2.1. Formulation of the PESP

The PESP is introduced by Serafinin and Ukovich in [12] as a method to formulate periodic timetabling problems. We follow the definitions from [21].

Definition 2.1.1. Let $a, b \in \mathbb{R}$ and $T \in \mathbb{N}_+$. With a and b being the lower and upper bound, respectively,

$$[a,b]_T := \bigcup_{z\in\mathbb{Z}} [a + zT, b + zT] \subseteq \mathbb{Z}$$

is called *interval modulo* T.

Let $\mathcal{I}_T$ be the set of intervals modulo T.

Definition 2.1.2. A *periodic event network* (PEN) is a triple $\mathcal{N} = (\mathcal{V}, T, a)$, with $\mathcal{V}$ the set of periodic events, $T \in \mathbb{N}$ the period and $a : \mathcal{V} \times \mathcal{V} \to 2^{\mathcal{I}_T}$ a function, which assigns to each pair of nodes $(i, i') \in \mathcal{V} \times \mathcal{V}$ a set of intervals modulo T. Then for each $i, i' \in \mathcal{V}$, $a(i, i')$ is their set of constraints.

A PEN can be visualized by a directed multigraph, where $\mathcal{V}$ is the set of nodes and every interval $c \in a(i, i')$, $i, i' \in \mathcal{V}$, denotes the labeling of a directed edge (i, i').

Definition 2.1.3. A *schedule* is a function $d : \mathcal{V} \to \{0, \dots, T-1\}$. A schedule d is *valid* with respect to a if and only if

$$d(i') - d(i) \in \bigcap_{c \in a(i,i')} c$$

for all $i, i' \in \mathcal{V}$.

The PESP can be formally defined as in the following definition.

Definition 2.1.4. Given a PEN $\mathcal{N} = (\mathcal{V}, T, a)$, find a *valid schedule* $d : \mathcal{V} \to \mathbb{N}$ with respect to a.

2.1.1. Properties of PESP Constraints

As defined above, the constraint function a assigns to each pair of nodes $(i,i') \in \mathcal{V}\times\mathcal{V}$ a set of intervals modulo T. The set $a(i,i')$ can be the empty set for some $i,i' \in \mathcal{V}$, which means there are no constraints between these periodic events. Note that we can reverse a constraint $[l,u]_T$ for the pair (i,i') to a constraint for the pair (i',i) by adding the interval $[-u,-l]_T = [-u+T,-l+T]_T$ to $a(i',i)$.

PESP constraints can also model a choice between multiple disjoint intervals for a given event pair (i,i'). Suppose we have a constraint between two events i,i' that says that their event time difference should lie *either* in the interval $[l_1,u_1]_T$, *or* in the interval $[l_2,u_2]_T$, for some $l_1 \le u_1 < l_2 \le u_2$, then we want

$$d(i') - d(i) \in [l_1,u_1]_T \cup [l_2,u_2]_T.$$

As shown in Figure 2.1, this choice between two intervals is equivalent to the following constraints

$$\begin{aligned} [l_1,u_2]_T &\in a(i,i'), \\ [l_2,u_1+T]_T &\in a(i,i'). \end{aligned}$$

This idea is generalized in the following lemma from [8], adapted to our notation. A proof of this lemma can be found in Appendix A.

Lemma 2.1.1. *Suppose that for some edge $(i,i') \in \mathcal{E}$ we want to impose the constraint*

$$d(i') - d(i) \in [l_1,u_1]_T \cup [l_2,u_2]_T \cup \ldots \cup [l_k,u_k]_T,$$

where the k intervals are disjoint and ordered:

$$0 \le l_1 \le u_1 < l_2 \le u_2 < \ldots < l_k \le u_k < l_1 + T.$$

Then this union of k intervals is equivalent to the intersection of k intervals modulo T given by the constraints

$$\begin{aligned} [l_1,u_k]_T &\in a(i,i'), \\ [l_2,u_1+T]_T &\in a(i,i'), \\ &\vdots \\ [l_k,u_{k-1}+T]_T &\in a(i,i'). \end{aligned}$$

We will now discuss the details of PESP in case of cyclic railway timetabling.

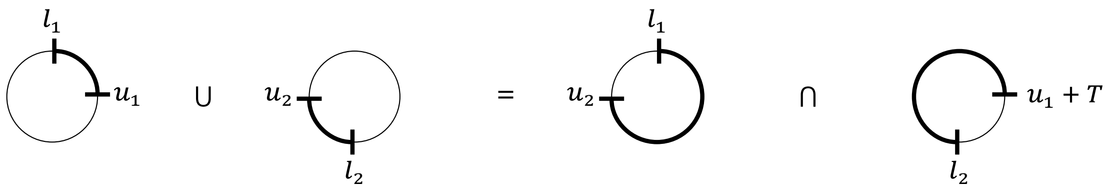


Figure 2.1: Equivalence of the union and intersection of periodic time windows

2.2. Railway Timetabling as a PESP

The input for the railway timetabling problem consists of the infrastructure and the line planning. These tell us the number of trains per hour and their detailed routes, including the stations where the train stops. On these routes we define *stage points* in such way that there is only one route option between the specified tracks at two consecutive stage points and that there is no crossing with another track in between two consecutive stage points. In this way the route of every train, specified in the line planning, is split in small stages.

Each stage of each train is characterized by the stage point and the track it departs from. Let $I \subseteq \mathbb{Z} \times \mathbb{Z}$ be the set representing the infrastructure, where, for $(i_1 i_2) \in I$, i_1 indicates the stage point and i_2 indicates the track at that stage point. Let N be the number of trains and let $r_t \in I^{s_t}$ be the route of train $t \in \{1, \dots, N\}$, where $s_t \in \mathbb{N}$ is the number of stage points the train passes, including its end point. So for train t, stage $i = 1, \dots, s_t - 1$ departs from $(r_t)_i$ and arrives at $(r_t)_{i+1}$. Let $y_i^t \in \mathbb{N}$ be the technical travel time of stage i of train t. Lastly, each stage has a code $z_i^t \in \{0,1\}$, where

$$z_i^t = \begin{cases} 1 & \text{if train } t \text{ has a stop between stage } i \text{ and stage } i+1, \\ 0 & \text{otherwise.} \end{cases}$$

The periodic event network in the railway timetabling case has period $T = 60$ when determining the timetable in minutes. Each periodic event corresponds to the periodic departure of a train t from a point on its route, say at $(r_t)_i$. Therefore, from now on, we will denote each periodic event as ϵ_i^t for some train t and some route point $(r_t)_i$. The set of periodic events then is $\mathcal{V} = \{\epsilon_i^t : t = 1, \dots, N; i = 1, \dots, s_t - 1\}$.

Example 2.2.1. Let us consider a railway network with three stations, station 1, 2 and 3 respectively. There is one track from station 1 to 2 and one track from station 2 to 3. Figure 2.2 is a representation of this network. Let $T = 60$ and let the line planning consist of two trains per hour going from station 1 to station 3, where one train, say train t, has a stop at station 2 and one train, say train t', is not stopping in between. In this case the stations are the only stage points, so we have $s_t, s_{t'} = 3$, $r_t, r_{t'} = (1\ 1; 2\ 1; 3\ 1)$, $z_1^t = 1$ and $z_2^t, z_1^{t'}, z_2^{t'} = 0$. Let the technical travel times be as follows: $y_1^t = 11$, $y_1^{t'} = 10$, $y_2^t = 22$ and $y_2^{t'} = 20$, where the difference in travel times between the trains is due to the fact that train t has a stop at station 2. We have four periodic events, $\mathcal{V} = \{\epsilon_1^t, \epsilon_2^t, \epsilon_1^{t'}, \epsilon_2^{t'}\}$.

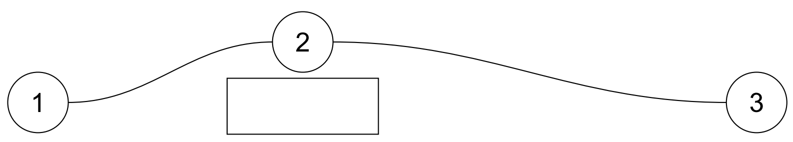


Figure 2.2: Example of a railway network

To define the constraint function a we use our own, generic norms. We will explain each type of constraint and its norms in the following sections.

2.2.1. Constraints for stages from the same train

Stages from the same train only have to be connected when they are consecutive. Two periodic events, say ϵ_i^t and $\epsilon_{i'}^{t'}$, correspond to two consecutive stages of the same train if and only if $t = t'$ and $i' = i + 1$. There are two different constraints for consecutive stages, which will be discussed below.

D-constraints When there is no stop in between, the difference between the departure times at route points $(r_t)_i$ and $(r_t)_{i+1}$ should be the technical travel time y_i^t. This is stated in a *D-constraint* (Drive constraint). A minimal and maximal slack time is added to the technical travel time to create extra time to compensate for small delays.

So for all $t = 1, \dots, N$ and $i = 1, \dots, s_t - 2$ we have

$$z_i^t = 0 \quad \Rightarrow \quad [y_i^t + minSlackTime, y_i^t + maxSlackTime]_{60} \in a(\epsilon_i^t, \epsilon_{i+1}^t).$$

S-constraints When there is a stop in between consecutive stages, the difference between the departure times should be the technical travel time y_i^t plus a minimal and maximal stop time. This is stated in an *S-constraint* (Stop constraint).

So for all $t = 1, \dots, N$ and $i = 1, \dots, s_t - 2$ we have

$$z_i^t = 1 \quad \Rightarrow \quad [y_i^t + minStopTime, y_i^t + maxStopTime]_{60} \in a(\epsilon_i^t, \epsilon_{i+1}^t).$$

2.2.2. Constraints for stages from different trains

We continue with infrastructure and frequency constraints for stages from different trains. Apart from these constraints, operating constraints could be implemented to create connections between trains.

Let ϵ_i^t and $\epsilon_{i'}^{t'}$ again be two periodic events, where $t \neq t'$. The routes of the corresponding trains are used to determine whether the following constraints apply between these events.

UU-constraints When two stages of different trains have the same stage point and track as departure point, the departure times of the corresponding periodic events should be at least three minutes apart. This is stated in an *UU-constraint* (Out-Out constraint, *Uit-Uit* in Dutch).

So for all $t, t' = 1, \dots, N$, $t \neq t'$, $i = 1, \dots, s_t - 1$ and $i' = 1, \dots, s_{t'} - 1$, if

$$(r_t)_i = (r_{t'})_{i'},$$

we want

$$\begin{aligned} &(d(\epsilon_{i'}^{t'}) - d(\epsilon_i^t) \in [3, 59]_{60}) \ \wedge \ (d(\epsilon_i^t) - d(\epsilon_{i'}^{t'}) \in [3, 59]_{60}), \\ \Leftrightarrow \ & d(\epsilon_{i'}^{t'}) - d(\epsilon_i^t) \in [3, 59]_{60} \cap [-59, -3]_{60}, \\ \Leftrightarrow \ & d(\epsilon_{i'}^{t'}) - d(\epsilon_i^t) \in [3, 57]_{60}, \end{aligned}$$

as $[-59, -3]_{60} \equiv [1, 57]_{60}$. Therefore we have

$$(r_t)_i = (r_{t'})_{i'} \quad \Rightarrow \quad [3, 57]_{60} \in a(\epsilon_i^t, \epsilon_{i'}^{t'}).$$

II-constraints When two stages of different trains arrive at the same stage point and track, the arrival times at this point should be at least three minutes apart. This is stated in an *II-constraint* (In-In constraint).

So for all $t, t' = 1, \dots, N$, $t \neq t'$, $i = 1, \dots, s_t - 1$ and $i' = 1, \dots, s_{t'} - 1$, if

$$(r_t)_{i+1} = (r_{t'})_{i'+1},$$

we want

$$\begin{aligned}(d(\epsilon_{i'}^{t'}) + y_{i'}^{t'}) - (d(\epsilon_i^t) + y_i^t) &\in [3, 57]_{60},\\ \Leftrightarrow\ d(\epsilon_{i'}^{t'}) - d(\epsilon_i^t) &\in [3 + y_i^t - y_{i'}^{t'}, 57 + y_i^t - y_{i'}^{t'}]_{60}.\end{aligned}$$

Therefore we have

$$(r_t)_{i+1} = (r_{t'})_{i'+1} \ \Rightarrow\ [3 + y_i^t - y_{i'}^{t'}, 57 + y_i^t - y_{i'}^{t'}]_{60} \in a(\epsilon_i^t, \epsilon_{i'}^{t'}).$$

UI-constraints An *UI-constraint* (Out-In constraint, *Uit-In* in Dutch) connects the periodic events corresponding to two stages of different trains where the first stage departs from the same stage point and track as where the second stage arrives. The arrival time of the second stage should be at least three minutes after the departure time of the first stage.

So for all $t, t' = 1, \dots, N$, $t \neq t'$, $i = 1, \dots, s_t$ and $i' = 1, \dots, s_{t'} - 1$, if

$$(r_t)_i = (r_{t'})_{i'+1},$$

we want

$$\begin{aligned}(d(\epsilon_{i'}^{t'}) + y_{i'}^{t'}) - d(\epsilon_i^t) &\in [3, 59]_{60},\\ \Leftrightarrow\ d(\epsilon_{i'}^{t'}) - d(\epsilon_i^t) &\in [3 - y_{i'}^{t'}, 59 - y_{i'}^{t'}]_{60}.\end{aligned}$$

Therefore we have

$$(r_t)_i = (r_{t'})_{i'+1} \ \Rightarrow\ [3 - y_{i'}^{t'}, 59 - y_{i'}^{t'}]_{60} \in a(\epsilon_i^t, \epsilon_{i'}^{t'}).$$

IU-constraints An *IU-constraint* (In-Out constraint, *In-Uit* in Dutch) is used for the periodic events corresponding to two stages of different trains where the first stage arrives at the same stage point and track as where the second stage departs. The arrival time of the first stage should be at least three minutes after the departure time of the second stage.

So for all $t, t' = 1, \dots, N$, $t \neq t'$, $i = 1, \dots, s_t - 1$ and $i' = 1, \dots, s_{t'}$, if

$$(r_t)_{i+1} = (r_{t'})_{i'},$$

we want

$$\begin{aligned}d(\epsilon_i^t) - d(\epsilon_{i'}^{t'}) &\in [3 - y_i^t, 59 - y_i^t]_{60},\\ \Leftrightarrow d(\epsilon_{i'}^{t'}) - d(\epsilon_i^t) &\in [1 + y_i^t, 57 + y_i^t]_{60}.\end{aligned}$$

Therefore we have

$$(r_t)_{i+1} = (r_{t'})_{i'} \ \Rightarrow\ [1 + y_i^t, 57 + y_i^t]_{60} \in a(\epsilon_i^t, \epsilon_{i'}^{t'}).$$

FB-constraints Two periodic events that correspond to stages of different trains that take the same single track part in opposite direction, are connected by an *FB-constraint* (Frontal Crash constraint, *Frontaal Botsen* in Dutch). The departure time of the first stage should be later than the arrival time of the second stage and the other way around.

So for all $t, t' = 1, \dots, N$, $t \neq t'$, $i = 1, \dots, s_t - 1$ and $i' = 1, \dots, s_{t'} - 1$, if

$$(r_t)_i = (r_{t'})_{i'+1} \quad \wedge \quad (r_t)_{i+1} = (r_{t'})_{i'},$$

we want

$$\begin{aligned} d(\epsilon_{i'}^{t'}) &\in [d(\epsilon_i^t) + y_i^t, 60a]_{60}, & a \in \mathbb{Z} \text{ s.t. } d(\epsilon_i^t) + y_i^t \in [60(a-1), 60a-1], \\ d(\epsilon_i^t) &\in [d(\epsilon_{i'}^{t'}) + y_{i'}^{t'}, 60b]_{60}, & b \in \mathbb{Z} \text{ s.t. } d(\epsilon_{i'}^{t'}) + y_{i'}^{t'} \in [60(b-1), 60b-1]. \end{aligned}$$

Combining these restrictions gives

$$d(\epsilon_{i'}^{t'}) - d(\epsilon_i^t) \in [y_i^t, 60 - y_{i'}^{t'}]_{60}.$$

Therefore we have

$$\big((r_t)_i = (r_{t'})_{i'+1} \;\wedge\; (r_t)_{i+1} = (r_{t'})_{i'}\big) \;\Rightarrow\; [y_i^t, 60 - y_{i'}^{t'}]_{60} \in a(\epsilon_i^t, \epsilon_{i'}^{t'}).$$

F-constraints Two different trains take the same route when the stage points they leave from are the same. We would like these trains to be (almost) equally distributed over the period of the timetable, say with a margin m. Let n be the number of trains taking this same route. An *F-constraint* (Frequency constraint) connects the periodic events corresponding to the first stages of these trains to get the right distribution.

So for all $t, t' = 1, \dots, N$, $t \neq t'$, if

$$s_t = s_{t'} \quad \wedge \quad ((r_t)_i)_1 = ((r_{t'})_i)_1 \;\forall i = 1, \dots, s_t,$$

we want

$$d(\epsilon_{i'}^{t'}) - d(\epsilon_i^t) \in \bigcup_{k=1}^{n-1} \left[\frac{k}{n}60 - m, \frac{k}{n}60 + m\right]_{60}.$$

This requirement consists of $n-1$ disjunct intervals, if and only if

$$\frac{k}{n}60 + m < \frac{k+1}{n}60 - m \;\Leftrightarrow\; 2m < \frac{60}{n}.$$

Using Lemma 2.1.1 to model this choice between multiple intervals, we get

$$\begin{aligned} & s_t = s_{t'} \;\wedge\; ((r_t)_i)_1 = ((r_{t'})_i)_1 \;\forall i = 1, \dots, s_t \\ \Rightarrow & \begin{cases} \left[\frac{1}{n}60 - m, \frac{n-1}{n}60 + m\right]_{60} \in a(\epsilon_1^t, \epsilon_1^{t'}), \\ \left[\frac{k}{n}60 - m, \frac{k+n-1}{n}60 + m\right]_{60} \in a(\epsilon_1^t, \epsilon_1^{t'}) \;\; \forall k = 2, \dots, n-1. \end{cases} \end{aligned}$$

Example 2.2.2. Let $\mathcal{V} = \{\epsilon_1^t, \epsilon_2^t, \epsilon_1^{t'}, \epsilon_2^{t'}\}$ be the set of periodic events corresponding to the network of Example 2.2.1, where $r_t, r_{t'} = (1\ 1; 2\ 1; 3\ 1)$, $z_1^t = 1$, $z_1^{t'} = 0$, $y_1^t = 11$, $y_1^{t'} = 10$, $y_2^t = 22$ and $y_2^{t'} = 20$. Moreover, let $minSlackTime = 0$, $maxSlackTime = 1$, $minStopTime = 1$ and $maxStopTime = 4$.

As $z_1^t = 1$, there is an *S-constraint* between the events ϵ_1^t and ϵ_2^t. As $z_1^{t'} = 0$, there is a *D-constraint* between the events $\epsilon_1^{t'}$ and $\epsilon_2^{t'}$.

There are departures from the same station as $(r_t)_i = (r_{t'})_i$ for $i = 1, 2$. Therefore we have *UU-constraints* between the events ϵ_1^t and $\epsilon_1^{t'}$ and between the events ϵ_2^t and $\epsilon_2^{t'}$. Furthermore there are arrivals at the same station as $(r_t)_i = (r_{t'})_i$ for $i = 2, 3$. This means we have *II-constraints* between the events ϵ_1^t and $\epsilon_1^{t'}$ and between the events ϵ_2^t and $\epsilon_2^{t'}$.

Summarizing, the constraint function $a : \mathcal{V} \times \mathcal{V} \to 2^{\mathcal{I}_T}$ is defined as follows

$$\begin{aligned}
a(\epsilon_1^t, \epsilon_2^t) &= [y_1^t + minStopTime, y_1^t + maxStopTime]_{60} = [12, 16]_{60} \\
a(\epsilon_1^t, \epsilon_1^{t'}) &= \{[3, 57]_{60}, [3 + y_1^t - y_1^{t'}, 57 + y_1^t - y_1^{t'}]_{60}\} = \{[3, 57]_{60}, [4, 58]_{60}\} \\
a(\epsilon_2^t, \epsilon_2^{t'}) &= \{[3, 57]_{60}, [3 + y_2^t - y_2^{t'}, 57 + y_2^t - y_2^{t'}]_{60}\} = \{[3, 57]_{60}, [5, 59]_{60}\} \\
a(\epsilon_1^{t'}, \epsilon_2^{t'}) &= [y_1^{t'} + minSlackTime, y_1^{t'} + maxSlackTime]_{60} = [10, 11]_{60} \\
a(\zeta, \eta) &= \emptyset \qquad \text{for } (\zeta, \eta) \notin \{(\epsilon_1^t, \epsilon_2^t), (\epsilon_1^t, \epsilon_1^{t'}), (\epsilon_2^t, \epsilon_2^{t'}), (\epsilon_1^{t'}, \epsilon_2^{t'})\}.
\end{aligned}$$

This periodic event network $\mathcal{N} = (\mathcal{V}, T, a)$ is visualized in Figure 2.3.

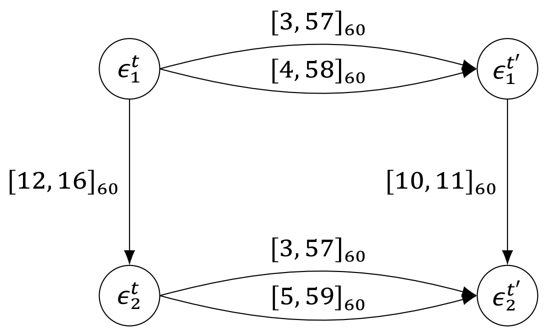


Figure 2.3: Periodic event network of Example 2.2.1

3 The SAT Problem

At the crossroads of logic, graph theory, computer science, computer engineering, and operations research stands Satisfiability. There are often multiple translations from problems in one of these fields to Satisfiability, and many problems are being solved faster by SAT solvers than other means [22]. Interesting SAT instances nowadays arise from areas such as AI planning and model checking [23], circuit testing and software verification [24], automatic test generation, logic synthesis and automatic theorem proving [25]. These applications in many areas have motivated the research in practically efficient SAT solvers [24].

In this chapter we first give a formal definition of the SAT Problem. Second, we explain how to reduce a PESP to a SAT problem in case of cyclic railway timetabling when using a preferred route for each train.

3.1. Formulation of SAT

The SAT problem can be classified in several ways, but we focus on SAT being a decision problem. Before we give the definition of the SAT problem, we first introduce the essential propositional logic. In this section we follow the definitions from [11].

Definition 3.1.1. The *alphabet of propositional logic* Σ_{SAT} consists of a countably infinite set of propositional variables $\mathcal{R} = \{p_1, p_2, ..., \}$, the brackets "(" and ")", as well as the binary connectives "∧" and "∨" and the unary connective "¬".

The connectives $\{\wedge, \vee, \neg\}$ are called conjunction ("and"), disjunction ("or") and negation ("not"), respectively.

Definition 3.1.2. A string F, that consists only of letters of the alphabet Σ_{SAT} is called a *propositional formula*, if and only if it fulfills one of the properties

1. $F = p$ for some $p \in \mathcal{R}$,
2. $F = \neg G$ for some propositional formula G,
3. $F = (G \circ H)$ for some binary connective $\circ$ and propositional formulas G and H.

Let $\mathcal{L}(\Sigma_{SAT})$ be the smallest set that contains each propositional formula under the alphabet Σ_{SAT}.

Definition 3.1.3. A propositional formula $L \in \mathcal{L}(\Sigma_{SAT})$ is called a *literal*, if and only if $L = p$ or $L = \neg p$, with $p \in \mathcal{R}$.

Definition 3.1.4. A propositional formula $C \in \mathcal{L}(\Sigma_{SAT})$ is called a *clause*, if it is a disjunction of literals. Hence, with $n \geq 0$

$$C = (\dots(L_1 \vee L_2) \vee \dots) \vee L_n),$$

where L_i $(i \in \{1, \dots, n\})$ are literals.

Definition 3.1.5. A propositional formula $F \in \mathcal{L}(\Sigma_{SAT})$ is in *conjunctive normal form* (denoted as CNF), if it is a conjunction of clauses. Thus, with $m \geq 0$

$$F = (\dots(C_1 \wedge C_2) \wedge \dots) \wedge C_m),$$

where C_i $(i \in \{1, \dots, m\})$ are clauses.

Definition 3.1.6. A propositional formula $F \in \mathcal{L}(\Sigma_{SAT})$ is in *disjunctive normal form* (denoted as DNF), if it is a disjunction of conjunctions of literals. Thus, with $m \geq 0$

$$F = (\dots(C_1 \vee C_2) \vee \dots) \vee C_m),$$

where, for $i \in \{1, \dots, m\}$ and some $n_i \geq 0$

$$C_i = (\dots(L_1 \wedge L_2) \wedge \dots) \wedge L_{n_i}),$$

where L_j $(j \in \{1, \dots, n\})$ are literals.

We use the following definition from [26], adapted to our notation, to get an interpretation of the variables.

Definition 3.1.7. An *assignment* is a map $\beta : \mathcal{R} \to \{true, false\}$ of the set of variables into the domain $\{true, false\}$.

Now we can define the following mapping to evaluate each formula.

Definition 3.1.8. Let $F \in \mathcal{L}(\Sigma_{SAT})$ be a propositional formula. Then the mapping

$$I : \mathcal{L}(\Sigma_{SAT}) \to \{true, false\}$$

$$F^I = \begin{cases} \beta(p) & \text{if } F = p \text{ for some } p \in \mathcal{R} \\ \neg(G^I) & \text{if } F = \neg G \text{ for some } G \in \mathcal{L}(\Sigma_{SAT}), \circ \in \{\wedge, \vee\} \\ G^I \circ H^I & \text{if } F = G \circ H \text{ for some } G, H \in \mathcal{L}(\Sigma_{SAT}), \circ \in \{\wedge, \vee\} \end{cases}$$

is called an *Interpretation*.

Using these definitions the SAT Problem can be defined as follows.

Definition 3.1.9. Let $F \in \mathcal{L}(\Sigma_{SAT})$ be a propositional formula. The *SAT Problem* is the problem of deciding whether

1. F is unsatisfiable or

2. F is satisfiable, if there exists an interpretation I such that $F^I = true$.

The interpretation I is the interpretation sought.

The SAT problem is trivial when the propositional formula F is in DNF. A formula in DNF is satisfiable unless each of its conjunctions contains both L and $\neg L$ for some literal L. However, conversion from a general propositional formula to DNF can lead to an exponentially sized formula.

The standard input format for most current SAT solvers is CNF [27]. This is not a limitation, as there exist polynomial algorithms to convert any propositional formula to a formula in CNF that has the same satisfiability [24]. Moreover, the CNF has advantages in terms of efficiency of the solver. For a formula in CNF to be satisfiable, each individual clause must be satisfiable, which helps prune the search space and speed up the search process [24]. It makes is easier to detect a conflict and in case of a partial assignment that did not work, a 'conflict' clause can easily be added to the formula [23].

To use modern, efficient SAT solvers, we create SAT problem instances in CNF. In the next section the encoding from PESP instances to SAT instances in CNF will be discussed.

3.2. PESP to SAT Encoding

There are several ways to reduce a PESP to a SAT Problem. We use order encoding as introduced in [28] for constraint satisfaction problems. The order encoding from PESP to SAT is described in [11], where this method is shown to be an efficient encoding for this reduction. First, we will discuss the encoding of the variables of the PESP in the cyclic railway timetabling case. Second, we will discuss the encoding of the constraints of this problem. Last, we conclude with the complete encoding of the PESP.

3.2.1. Encoding Variables

As the decision variables $d(\epsilon_i^t) \in \Phi = \{0, \dots, T-1\}$ are in a finite ordered domain, we can use the following order encoding function as defined in [11].

Definition 3.2.1 (Order Encoding Function for Variables of Finite Domains). Let $x \in \mathcal{X}$ be a variable with $dom(x) = [l, u] \subset \mathbb{N}$ and $\mathcal{X}$ the variable space. Then the function

$$encode\_ordered : \mathcal{X} \to \mathcal{L}(\Sigma_{SAT})$$

is the order encoding function for a variable of the variable space $\mathcal{X}$ with

$$encode\_ordered : x \mapsto \bigwedge_{i' \in [l+1, u-1]} (\neg q_{x,i'-1} \vee q_{x,i'})$$

with $q_{x,i'} \in \mathcal{R}$ for $x \in \mathcal{X}$ and $i' \in [l, u-1]$.

Here $q_{x,i'}$ has the meaning

$$q_{x,i'} \Leftrightarrow x \leq i',$$

and for a given interpretation I we have the following two cases

$$q^I_{x,i'} \Leftrightarrow x \leq i',$$
$$\neg q^I_{x,i'} \Leftrightarrow x \nleq i'.$$

If $encode\_ordered(x)^I$ is true, there is a unique $k \in dom(x) = [l, u]$ such that

$$\forall i \in [l, k-1] : \neg q^I_{x,i},$$
$$\forall i' \in [k, u-1] : q^I_{x,i'}.$$

If $k \in [l, u-1]$, this means $x \nleq k-1$ and $x \leq k$, so $x = k$. If $k = u$, we have $x \nleq u-1$ and thus, as $x \in [l, u]$, $x = u$. In this way we can extract the value of a variable.

Given a railway timetabling periodic event network $\mathcal{N} = (\mathcal{V}, T, a)$, we have the variables $d(\epsilon^t_i)$ for $t = 1, \ldots, N$ and $i = 1, \ldots, s_t - 1$. The propositional formula which encodes all variables is

$$\Omega^{\mathcal{N}}_{ordered} := \bigwedge_{t=1,\ldots,N, i=1,\ldots,s_t-1} encode\_ordered(d(\epsilon^t_i)).$$

This formula is in CNF, so it can be used as input for SAT.

3.2.2. Encoding Constraints

The idea of encoding PESP constraints in SAT is to describe the infeasible regions of every PESP constraint in propositional formulas in CNF. Here we follow the method of [11], but for practical coding reasons we use vertical line segments instead of rectangles to describe the infeasible regions. We start off with an example to show the base idea.

Example 3.2.1. Let $\mathcal{N} = (\mathcal{V}, T, a)$ be the periodic event network from Example 2.2.2. We have $\epsilon^t_1, \epsilon^{t'}_1 \in \mathcal{V}$ with departure times $d(\epsilon^t_1)$, $d(\epsilon^{t'}_1)$, respectively, and we have PESP constraint $[3, 57]_{60} \in a(\epsilon^t_1, \epsilon^{t'}_1)$. In Figure 3.1 one can see the feasible and infeasible regions for this PESP constraint.

The idea is to encode this constraint by describing the vertical line segments in the infeasible region. An example of a vertical line segment in the infeasible region of this particular constraint is $(d(\epsilon^t_1), d(\epsilon^{t'}_1)) \in \{10\} \times [7, 13]$, which is highlighted in Figure 3.1. In fact, what we want to say is

$$\nexists (d(\epsilon^t_1), d(\epsilon^{t'}_1)) : d(\epsilon^t_1) \leq 10, d(\epsilon^t_1) \geq 10, d(\epsilon^{t'}_1) \leq 13, d(\epsilon^{t'}_1) \geq 7,$$

which is equivalent to

$$\neg((d(\epsilon^t_1) \leq 10) \wedge (d(\epsilon^t_1) \geq 10) \wedge (d(\epsilon^{t'}_1) \leq 13) \wedge (d(\epsilon^{t'}_1) \geq 7))$$
$$\Leftrightarrow \neg((d(\epsilon^t_1) \leq 10) \wedge \neg(d(\epsilon^t_1) \leq 9) \wedge (d(\epsilon^{t'}_1) \leq 13) \wedge \neg(d(\epsilon^{t'}_1) \leq 6))$$
$$\Leftrightarrow \neg(q_{d(\epsilon^t_1),10} \wedge \neg q_{d(\epsilon^t_1),9} \wedge q_{d(\epsilon^{t'}_1),13} \wedge \neg q_{d(\epsilon^{t'}_1),6})$$
$$\Leftrightarrow (\neg q_{d(\epsilon^t_1),10} \vee q_{d(\epsilon^t_1),9} \vee \neg q_{d(\epsilon^{t'}_1),13} \vee q_{d(\epsilon^{t'}_1),6}).$$

As this formula is a clause, the resulting formula for multiple line segments will be in CNF.

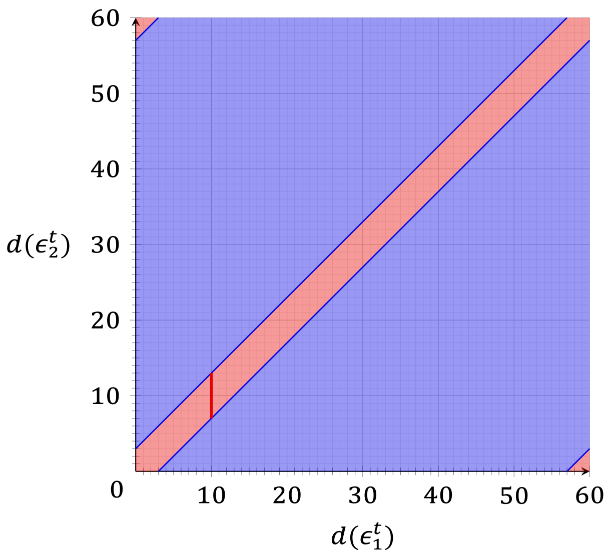


Figure 3.1: Feasible (in blue) and infeasible regions (in red) for the PESP constraint of Example 3.2.1

The following example shows the method of [11] using rectangles and the difference in number of clauses.

Example 3.2.2. Let again $\mathcal{N} = (\mathcal{V}, T, a)$ be the periodic event network from Example 2.2.2 where $\epsilon_1^t, \epsilon_1^{t'} \in \mathcal{V}$. We have departure times $d(\epsilon_1^t)$, $d(\epsilon_1^{t'})$, respectively, and PESP constraint $[3, 57]_{60} \in a(\epsilon_1^t, \epsilon_1^{t'})$.

In order to describe the middle infeasible region of Figure 3.1 by rectangles, we need the lower bound of the upper feasible region and the upper bound of the lower feasible region with respect to $d(\epsilon_1^{t'})$. For the middle infeasible region of Figure 3.1, we would have an upper bound $u = -3$ and a lower bound $l = 3$. According to the definitions in [11], the set of rectangles between the upper and lower bound is

$$\begin{aligned}\big\{([x_1, x_1 + \delta x(l,u)] \times [x_2, x_2 + \delta y(l,u)]) \mid \forall x_2 \in \{-\delta y(l,u), \dots, 59\} :\\ x_1 + \delta x(l,u) \geq 0 \wedge x_1 \leq 59 :\\ x_1 = x_2 - u - 1 - \delta x(l,u)\big\},\end{aligned}$$

where

$$\begin{aligned}\delta x(l,u) &= \left\lceil \frac{l - u - 1}{2} \right\rceil - 1,\\ \delta y(l,u) &= \left\lfloor \frac{l - u - 1}{2} \right\rfloor.\end{aligned}$$

The set of rectangles to describe the middle infeasible region of Figure 3.1 then is

$$\big\{([x, x+2] \times [y, y+2]) \mid \forall y \in \{-2, \dots, 59\} : x + 2 \geq 0 \wedge x \leq 59 : x = y\big\}.$$

In this way we get 62 rectangles to describe this region, instead of 60 vertical line segments.

We now generalize the strategy in Example 3.2.1. For every PESP constraint the infeasible region is described by the following set of vertical line segments.

Definition 3.2.2. Let ϵ_1, ϵ_2 be periodic events and let $c \in a(\epsilon_1, \epsilon_2)$ be a PESP constraint. Then

$$\begin{aligned}\zeta(\epsilon_1, \epsilon_2, c) = \big\{\{x\} \times [y_1, y_2] \mid x \in dom(d(\epsilon_1));\ y_1, y_2 \in dom(d(\epsilon_2));\\ y - x \in c\ \forall y \in [y_1, y_2];\ y - x \notin c \text{ for } y = y_1 - 1, y_2 + 1\big\},\end{aligned}$$

is the set of all infeasible vertical line segments imposed by constraint c between the events ϵ_1 and ϵ_2.

Then the following formula, adapted from [11], shows the encoding of a PESP constraint.

Definition 3.2.3. Let ϵ_1, ϵ_2 be periodic events and let $c \in a(\epsilon_1, \epsilon_2)$ be a PESP constraint. Then

$$\begin{aligned} encode\_ordered\_con : \mathcal{V} \times \mathcal{V} \times \mathcal{I}_T &\to \mathcal{L}(\Sigma_{SAT}) \\ (\epsilon_1, \epsilon_2, c) &\mapsto \bigwedge_{\{x\}\times[y_1,y_2]\in\zeta(\epsilon_1,\epsilon_2,c)} (\neg q_{d(\epsilon_1),x} \vee q_{d(\epsilon_1),x-1} \vee \neg q_{d(\epsilon_2),y_2} \vee q_{d(\epsilon_2),y_1-1}) \end{aligned}$$

is the order encoding function of a PESP constraint.

Given a railway timetabling periodic event network $\mathcal{N} = (\mathcal{V}, T, a)$, the propositional formula which encodes all PESP constraints is

$$\Psi^{\mathcal{N}}_{ordered} := \bigwedge_{t,t',i,i'} \bigwedge_{c\in a(\epsilon^t_i,\epsilon^{t'}_{i'})} encode\_ordered\_con(\epsilon^t_i, \epsilon^{t'}_{i'}, c),$$

where $t, t' = 1, \dots, N$, $i = 1, \dots, s_t - 1$ and $i' = 1, \dots, s_{t'} - 1$. This formula is in CNF.

3.2.3. Encoding PESP

Taking together the formulas for variables and constraints, the encoding of the PESP is given as follows.

Definition 3.2.4. Let $\mathcal{N} = (\mathcal{V}, T, a)$ be the periodic event network. Then

$$\Omega^{\mathcal{N}}_{ordered} \wedge \Psi^{\mathcal{N}}_{ordered}$$

is the order encoding of $\mathcal{N}$.

4 Incorporating Multiple Route Options

The PESP model presented in Section 2.2 assumes the tracks each train takes on its route as fixed. Now we create an extension to the SAT reduction of the PESP model explained in Chapter 3, where we incorporate multiple route options. By considering a larger part of the solution space in this way, we hope to make it easier to find a feasible timetable. In this chapter we first explain the idea of our method to incorporate multiple route options for each train in the railway timetabling problem. Then we give the formal definition of our method, by defining the concept of an Open-ended PESP and thereafter explaining the details of the Open-ended PESP in case of cyclic railway timetabling.

4.1. The idea of the method

We start off with an example to show the base idea of our method.

Example 4.1.1. Let us reconsider the railway network from Example 1.4.1 with three stations, station 1, 2 and 3 respectively. There are two routes from station 1 to station 2, say route α and route β and from station 2 to station 3 there are two routes as well, say route γ and route δ. A train is standing still at the second platform of station 2. Figure 1.4 is a representation of this network and situation. The problem of scheduling the second train, which is supposed to go from station 1 to station 3 via route β and δ, is unsatisfiable. In fact, there is no feasible departure time for this train in the domain $\{0, \dots, 59\}$.

Now let us say there are two route options for this second train, where the first route option corresponds to taking route β and δ and the second option corresponds to taking route α and γ. We extend the domain of the departure time of this train to $\{0, \dots, 119\}$, instead of $\{0, \dots, 59\}$, where a departure time in the set $\{0, \dots, 59\}$ corresponds to taking the first route option and a departure time in the set $\{60, \dots, 119\}$ corresponds to taking the second route option.

Generalizing this example, we extend the domain of each departure time from $\{0, \dots, 59\}$ to $\{0, \dots, 60n - 1\}$, where n is the number of route options for the stage corresponding to this departure. A departure time in the set $\{60(k-1), 60k-1\}$ corresponds to choosing the kth route option. Constraints are defined per route option, where each constraint between two departures is a restriction between two sets in the domains of their departure times. When route options of successive stages are not connected, we can make sure only feasible routes are created by excluding the combination of two sets in domains in a constraint.

Hence, by extending the domain of the departure corresponding to the number of route options for this stage, we model the geographical decision of incorporating multiple route options on the time axis. Another method that one could think of is the use of decision variables to indicate the route choice. Appendix B discusses why the use of decision variables is less efficient when using SAT.

In the next sections, the idea of our method to incorporate flexible track use is formalized.

4.2. Formulation of the Open-ended PESP

We introduce a generalization of the PESP, namely the Open-ended Periodic Event Scheduling Problem (OPESP). In this case all periodic events are open-ended, they could take place in several ways. There are multiple options for each periodic event, with each corresponding to different constraints, and there is a choice for each periodic event to meet the properties of one of the options. To formulate OPESP, we introduce the notion of an open-ended periodic event network (OPEN).

Definition 4.2.1. A *open-ended periodic event network* (OPEN) is a quadruple $\mathcal{N} = (\mathcal{V}, T, h, a)$, with $\mathcal{V}$ the set of periodic events, $T \in \mathbb{N}$ the period, $h : \mathcal{V} \to \mathbb{N}$ a function, which assigns to each periodic event $i \in \mathcal{V}$ the number of options, and $a((i,k),(i',k')) \subseteq \mathcal{I}_T$ a set of intervals modulo T for each $i, i' \in \mathcal{V}$, which is the set of constraints corresponding to option k for event i and option k' for event i'.

An OPEN can be visualized by a $|\mathcal{V}|$-partite directed multigraph, where $\{(i,k) : i \in \mathcal{V}, k \in \{1, \dots, h(i)\}\}$ is the set of nodes, the sets $\{(i,k) : k \in \{1, \dots, h(i)\}\}$ for $i \in \mathcal{V}$ form a partition of the node set, and every interval $c \in a((i,k),(i',k'))$, $i, i' \in \mathcal{V}$, $k \in \{1, \dots, h(i)\}$, $k' \in \{1, \dots, h(i')\}$, denotes the labeling of a directed edge $((i,k),(i',k'))$.

Definition 4.2.2. A *schedule* is a function $d : \mathcal{V} \to \mathbb{N}$. A schedule d is *valid* with respect to a if and only if $d(i) \in \{0, \dots, h(i)T - 1\}$ for all $i \in \mathcal{V}$ and

$$\left.\begin{array}{l} d(i) \in [(k-1)T, kT-1] \\ d(i') \in [(k'-1)T, k'T-1] \end{array}\right\} \Rightarrow d(i') - d(i) \in \bigcap_{c \in a((i,k),(i',k'))} c$$

for all $i, i' \in \mathcal{V}$, $k \in \{0, \dots, h(i)\}$ and $k' \in \{0, \dots, h(i')\}$.

We define OPESP as follows.

Definition 4.2.3. Given an OPEN $\mathcal{N} = (\mathcal{V}, T, h, a)$, find a *valid schedule* $d : \mathcal{V} \to \mathbb{N}$ with respect to a.

Note that a PESP with PEN $\mathcal{N} = (\mathcal{V}, T, a)$ is equal to an OPESP with OPEN $\mathcal{N}' = (\mathcal{V}, T, h, a')$, where $h(i) = 1$ for all $i \in \mathcal{V}$ and $a'((i,1),(i',1)) = a(i,i')$.

4.3. Railway Timetabling as an OPESP

The input for the railway timetabling problem as an OPESP consists, just as for the PESP, of the infrastructure and the line planning. However, the line planning now only consists of the general route for each train. We define *stage points* in the same way as for the PESP, which are used to split the route of each train in small stages.

Each stage of each train is now only characterized by the stage point it departs from. Let $\mathcal{P} \subseteq \mathbb{Z}$ be the set of stage points and let N be the number of trains. Let $r_t \in \mathcal{P}^{s_t}$ be the route of train $t \in \{1, \dots, N\}$, where $s_t \in \mathbb{N}$ is the number of stage points the train passes, including its end point. We assume that the technical travel time does not depend substantially on the choice of the track, so let $y_i^t \in \mathbb{N}$ be the technical travel time of stage i of train t. As for the PESP formulation, let $z_i^t \in \{0, 1\}$ be the code of stage i of train t, where

$$z_i^t = \begin{cases} 1 & \text{if train } t \text{ has a stop between stage } i \text{ and stage } i+1, \\ 0 & \text{otherwise.} \end{cases}$$

For each stage there are multiple stage options, which are characterized by the stage point it departs from, as well as the track it is leaving from and the track it is arriving at. The infrastructure is used to decide on the track options at each stage point and the route options for each direction between these tracks. Some route options are always taken into account, some are not feasible when for example a train has a stop at a particular station and there is a track without an adjacent platform. Let $\mathcal{T}(p) \subseteq \mathbb{Z}$ be the set of tracks at stage point $p \in \mathcal{P}$. Let $o(p, p', q)$ be the number of route options for a stage from stage point p to stage point p', where $q = 1$ if a stop should be possible at the arrival track, and $q = 0$ if this is not needed. The possible combinations of departure and arrival tracks for a stage from stage point p to stage point p' with $q \in \{0, 1\}$ are then given by

$$\mathcal{O}(p, p', q) \in \left(\mathcal{T}(p) \times \mathcal{T}(p')\right)^{o(p,p',q)}.$$

For stage i of train t, let $o_i^t := o((r_t)_i, (r_t)_{i+1}, z_i^t)$ be the number of route options and let $\mathcal{O}_i^t := \mathcal{O}((r_t)_i, (r_t)_{i+1}, z_i^t)$ be the possible combinations of departure and arrival tracks.

The OPEN in the railway timetabling case, just as the PEN, has period $T = 60$ when determining the timetable in minutes. Each periodic event corresponds to the periodic departure of a train t from a stage point on its route, say at $(r_t)_i$, which gives us the set $\mathcal{V} = \{\epsilon_i^t : t = 1, \dots, N; i = 1, \dots, s_t - 1\}$. Each option for each departure ϵ_i^t corresponds to taking a certain route option, so the number of options is given by $h(\epsilon_i^t) = o_i^t$. Let $\epsilon_{ij}^t := (\epsilon_i^t, j) \in \mathcal{V} \times \{1, \dots, o_i^t\}$ be the departure of train t from stage point $(r_t)_i$, taking route option j for this stage.

The definition of the constraint function a will we discussed in the following sections. For a more extensive description of the types of constraints we refer to Section 2.2.1 and Section 2.2.2, as the same types are used here.

Example 4.3.1. Let us consider a similar railway network to Example 2.2.1 with three stations, station 1, 2 and 3 respectively. Instead of one platform at station 2, there are now two platforms with corresponding routes. So from station 1 to 2 there are two routes, say route α and route β, and from station 2 to station 3 there are two routes as well, say route γ and route δ. Figure 4.1 is a representation of this network.

Let $T = 60$ and let the line planning again consist of two trains per hour going from station 1 to station 3, where one train, say train t, has a stop at station 2 and one train, say train t', is not stopping in between. In this case the stations are the only stage points, but the track at station 2 is unknown. So we have $s_t, s_{t'} = 3$, $r_t = r_{t'} = (1, 2, 3)$, $z_1^t = 1$ and $z_2^t, z_1^{t'}, z_2^{t'} = 0$. Let the technical travel times be as in Example 2.2.1: $y_1^t = 11$, $y_1^{t'} = 10$, $y_2^t = 22$ and $y_2^{t'} = 20$. We have $o_i^t = o_i^{t'} = 2$ for $i = 1, 2$, $\mathcal{O}_1^t = \mathcal{O}_1^{t'} = (1\ 1; 1\ 2)$ and $\mathcal{O}_2^t = \mathcal{O}_2^{t'} = (1\ 1; 2\ 1)$.

We have four periodic events, $\mathcal{V} = \{\epsilon_1^t, \epsilon_2^t, \epsilon_1^{t'}, \epsilon_2^{t'}\}$. Here, ϵ_{11}^t and $\epsilon_{11}^{t'}$ correspond to taking route α, ϵ_{12}^t and $\epsilon_{12}^{t'}$ correspond to taking route β, ϵ_{21}^t and $\epsilon_{21}^{t'}$ correspond to taking route γ and ϵ_{22}^t and $\epsilon_{22}^{t'}$ correspond to taking route δ.

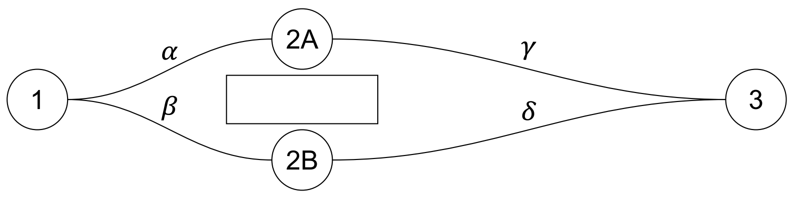


Figure 4.1: Example of a railway network

4.3.1. Constraints for stage options from the same train

Stage options from the same train only have to be connected when they correspond to consecutive stages. Two options of periodic events, say ϵ_{ij}^t and $\epsilon_{i'j'}^{t'}$, correspond to two consecutive stages of the same train if and only if $t = t'$ and, without loss of generality, $i' = i + 1$. There are three different constraints for consecutive stages, which will be discussed below.

D-constraints When the options of periodic events correspond to consecutive route options and there is no stop in between the corresponding stages, a *D-constraint* is added. Then the arrival track of the first stage option should be the same as the departure track of the second stage option.

So for all $t = 1, \dots, N$, $i = 1, \dots, s_t - 2$ and $j = 1, \dots, o_i^t$ we have

$$\begin{aligned}&\left(z_i^t = 0 \ \wedge\ ((\mathcal{O}_i^t)_j)_2 = ((\mathcal{O}_{i+1}^t)_{j'})_1\right)\\ \Rightarrow\ &[y_i^t + minSlackTime, y_i^t + maxSlackTime]_{60} \in a(\epsilon_{ij}^t, \epsilon_{(i+1)j'}^t).\end{aligned}$$

S-constraints When the options of periodic events correspond to consecutive route options and there is a stop in between the corresponding stages, an *S-constraint* is added. Again the arrival track of the first stage option should be the same as the departure track of the second stage option.

So for all $t = 1, \dots, N$, $i = 1, \dots, s_t - 2$ and $j = 1, \dots, o_i^t$ we have

$$\begin{aligned}&\left(z_i^t = 1 \ \wedge\ ((\mathcal{O}_i^t)_j)_2 = ((\mathcal{O}_{i+1}^t)_{j'})_1\right)\\ \Rightarrow\ &[y_i^t + minStopTime, y_i^t + maxStopTime]_{60} \in a(\epsilon_{ij}^t, \epsilon_{(i+1)j'}^t).\end{aligned}$$

I-constraints Two stage options, corresponding to consecutive stages of the same train, of which the arrival track of the first stage option is different then the departure track of the second stage option, cannot be chosen at the same time as this would give an infeasible route. For these incompatible stages every combination is infeasible, which is stated in an *I-constraint*.
So for all $t = 1, \dots, N$, $i = 1, \dots, s_t - 2$ and $j = 1, \dots, o_i^t$ we have

$$((\mathcal{O}_i^t)_j)_2 \neq ((\mathcal{O}_{i+1}^t)_{j'})_1 \quad \Rightarrow \quad \{\emptyset\} \in a(\epsilon_{ij}^t, \epsilon_{(i+1)j'}^t).$$

4.3.2. Constraints for stage options from different trains

Let ϵ_{ij}^t and $\epsilon_{i'j'}^{t'}$ again be two options of periodic events, where $t \neq t'$.

UU-constraints When two stage options of different trains have the same stage point and track as departure point, an *UU-constraint* is added.

So for all $t, t' = 1, \dots, N$, $t \neq t'$, $i = 1, \dots, s_t - 1$, $i' = 1, \dots, s_{t'} - 1$, $j = 1, \dots, o_i^t$ and $j' = 1, \dots, o_{i'}^{t'}$, we have

$$\big((r_t)_i = (r_{t'})_{i'} \ \wedge \ ((\mathcal{O}_i^t)_j)_1 = ((\mathcal{O}_{i'}^{t'})_{j'})_1\big) \quad \Rightarrow \quad [3, 57]_{60} \in a(\epsilon_{ij}^t, \epsilon_{i'j'}^{t'}).$$

II-constraints When two stage options of different trains arrive at the same stage point and track, an *II-constraint* is added.

So for all $t, t' = 1, \dots, N$, $t \neq t'$, $i = 1, \dots, s_t - 1$ and $i' = 1, \dots, s_{t'} - 1$, $j = 1, \dots, o_i^t$ and $j' = 1, \dots, o_{i'}^{t'}$, we have

$$\begin{aligned} &\big((r_t)_{i+1} = (r_{t'})_{i'+1} \ \wedge \ ((\mathcal{O}_i^t)_j)_2 = ((\mathcal{O}_{i'}^{t'})_{j'})_2\big) \\ \Rightarrow \ &[3 + y_i^t - y_{i'}^{t'}, 57 + y_i^t - y_{i'}^{t'}]_{60} \in a(\epsilon_{ij}^t, \epsilon_{i'j'}^{t'}). \end{aligned}$$

UI-constraints An *UI-constraint* connects the options of periodic events corresponding to two stage options of different trains where the first stage option departs from the same stage point and track as where the second stage option arrives.

So for all $t, t' = 1, \dots, N$, $t \neq t'$, $i = 1, \dots, s_t$ and $i' = 1, \dots, s_{t'} - 1$, $j = 1, \dots, o_i^t$ and $j' = 1, \dots, o_{i'}^{t'}$, we have

$$\big((r_t)_i = (r_{t'})_{i'+1} \ \wedge \ ((\mathcal{O}_i^t)_j)_1 = ((\mathcal{O}_{i'}^{t'})_{j'})_2\big) \quad \Rightarrow \quad [3 - y_{i'}^{t'}, 59 - y_{i'}^{t'}]_{60} \in a(\epsilon_{ij}^t, \epsilon_{i'j'}^{t'}).$$

IU-constraints An *IU-constraint* is used for the options of periodic events corresponding to two stage options of different trains where the first stage option arrives at the same stage point and track as where the second stage option departs.

So for all $t, t' = 1, \dots, N$, $t \neq t'$, $i = 1, \dots, s_t - 1$ and $i' = 1, \dots, s_{t'}$, $j = 1, \dots, o_i^t$ and $j' = 1, \dots, o_{i'}^{t'}$, we have

$$\big((r_t)_{i+1} = (r_{t'})_{i'} \ \wedge \ ((\mathcal{O}_i^t)_j)_2 = ((\mathcal{O}_{i'}^{t'})_{j'})_1\big) \quad \Rightarrow \quad [1 + y_i^t, 57 + y_i^t]_{60} \in a(\epsilon_{ij}^t, \epsilon_{i'j'}^{t'}).$$

FB-constraints Two options of periodic events that correspond to stage options of different trains that take the same single track part in opposite direction, are connected by an *FB-constraint*.

So for all $t, t' = 1, \dots, N$, $t \neq t'$, $i = 1, \dots, s_t - 1$ and $i' = 1, \dots, s_{t'} - 1$, $j = 1, \dots, o_i^t$ and $j' = 1, \dots, o_{i'}^{t'}$, we have

$$\left.\begin{matrix} \big((r_t)_i = (r_{t'})_{i'+1} \ \wedge \ (r_t)_{i+1} = (r_{t'})_{i'} \ \wedge \\ ((\mathcal{O}_i^t)_j)_1 = ((\mathcal{O}_{i'}^{t'})_{j'})_2 \ \wedge \ ((\mathcal{O}_i^t)_j)_2 = ((\mathcal{O}_{i'}^{t'})_{j'})_1\big) \end{matrix}\right\} \Rightarrow \ [y_i^t, 60 - y_{i'}^{t'}]_{60} \in a(\epsilon_{ij}^t, \epsilon_{i'j'}^{t'}).$$

F-constraints Two different trains taking the same route should be (almost) equally distributed over the period of the timetable, say with a margin m. Let n be the number of trains taking this same route. An *F-constraint* connects the options of periodic events corresponding to the first stage options of these trains to get the right distribution.

So for all $t, t' = 1, \dots, N$, $t \neq t'$, if

$$\frac{k}{n}60 + m < \frac{k+1}{n}60 - m \ \Leftrightarrow \ 2m < \frac{60}{n},$$

the requirement consists of $n - 1$ disjunct intervals and we have

$$\begin{aligned} & s_t = s_{t'} \ \wedge \ (r_t)_i = (r_{t'})_i \ \forall i = 1, \dots, s_t \\ \Rightarrow & \begin{cases} \left[\frac{1}{n}60 - m, \frac{n-1}{n}60 + m\right]_{60} \in a(\epsilon_{1j}^t, \epsilon_{1j'}^{t'}), \\ \left[\frac{k}{n}60 - m, \frac{k+n-1}{n}60 + m\right]_{60} \in a(\epsilon_{1j}^t, \epsilon_{1j'}^{t'}) \ \forall k = 2, \dots, n-1, \end{cases} \end{aligned}$$

for all $j = 1, \dots, o_i^t$ and $j' = 1, \dots, o_{i'}^{t'}$.

Example 4.3.2. Let $\mathcal{V} = \{\epsilon_1^t, \epsilon_2^t, \epsilon_1^{t'}, \epsilon_2^{t'}\}$ be the set of periodic events corresponding to the network of Example 4.3.1, where $r_t, r_{t'} = (1, 2, 3)$, $z_1^t = 1$, $z_1^{t'} = 0$, $y_1^t = 11$, $y_1^{t'} = 10$, $y_2^t = 22$, $y_2^{t'} = 20$, $\mathcal{O}_1^t = \mathcal{O}_1^{t'} = (1\ 1; 1\ 2)$ and $\mathcal{O}_2^t = \mathcal{O}_2^{t'} = (1\ 1; 2\ 1)$. Let $minSlackTime = 0$, $maxSlackTime = 1$, $minStopTime = 1$ and $maxStopTime = 5$.

As $z_1^t = 1$, $((\mathcal{O}_1^t)_1)_2 = ((\mathcal{O}_2^t)_1)_1$ and $((\mathcal{O}_1^t)_2)_2 = ((\mathcal{O}_2^t)_2)_1$, there are *S-constraints* between the options ϵ_{11}^t and ϵ_{21}^t and between ϵ_{12}^t and ϵ_{22}^t. As $z_1^{t'} = 0$, $((\mathcal{O}_1^{t'})_1)_2 = ((\mathcal{O}_2^{t'})_1)_1$ and $((\mathcal{O}_1^{t'})_2)_2 = ((\mathcal{O}_2^{t'})_2)_1$, there are *D-constraints* between the options $\epsilon_{11}^{t'}$ and $\epsilon_{21}^{t'}$ and between $\epsilon_{12}^{t'}$ and $\epsilon_{22}^{t'}$.

There are also options that cannot be chosen at the same time, as

$$\begin{aligned} &((\mathcal{O}_1^t)_1)_2 \neq ((\mathcal{O}_2^t)_2)_1, && ((\mathcal{O}_1^t)_2)_2 \neq ((\mathcal{O}_2^t)_1)_1, \\ &((\mathcal{O}_1^{t'})_1)_2 \neq ((\mathcal{O}_2^{t'})_2)_1 \quad \text{and} && ((\mathcal{O}_1^{t'})_2)_2 \neq ((\mathcal{O}_2^{t'})_1)_1. \end{aligned}$$

Therefore we have *I-constraints* between the options ϵ_{11}^t and ϵ_{22}^t, between ϵ_{12}^t and ϵ_{21}^t, between $\epsilon_{11}^{t'}$ and $\epsilon_{22}^{t'}$ and between $\epsilon_{12}^{t'}$ and $\epsilon_{21}^{t'}$.

There are departures from the same station as $(r_t)_i = (r_{t'})_i$ for $i = 1, 2$. We have

$$\begin{aligned} &((\mathcal{O}_1^t)_1)_1 = ((\mathcal{O}_1^{t'})_1)_1, && ((\mathcal{O}_1^t)_1)_1 = ((\mathcal{O}_1^{t'})_2)_1, \\ &((\mathcal{O}_1^t)_2)_1 = ((\mathcal{O}_1^{t'})_1)_1, && ((\mathcal{O}_1^t)_2)_1 = ((\mathcal{O}_1^{t'})_2)_1, \\ &((\mathcal{O}_2^t)_1)_1 = ((\mathcal{O}_2^{t'})_1)_1 \quad \text{and} && ((\mathcal{O}_2^t)_2)_1 = ((\mathcal{O}_2^{t'})_2)_1. \end{aligned}$$

Therefore there are *UU-constraints* between the options ϵ_{11}^t and $\epsilon_{11}^{t'}$, between ϵ_{11}^t and $\epsilon_{12}^{t'}$, between ϵ_{12}^t and $\epsilon_{11}^{t'}$, between ϵ_{12}^t and $\epsilon_{12}^{t'}$, between ϵ_{21}^t and $\epsilon_{21}^{t'}$ and between ϵ_{22}^t and $\epsilon_{22}^{t'}$.

Furthermore there are arrivals at the same station as $(r_t)_i = (r_{t'})_i$ for $i = 2, 3$. We have

$$\begin{aligned}
&((\mathcal{O}_1^t)_1)_2 = ((\mathcal{O}_1^{t'})_1)_2, && ((\mathcal{O}_1^t)_2)_2 = ((\mathcal{O}_1^{t'})_2)_1,\\
&((\mathcal{O}_2^t)_1)_2 = ((\mathcal{O}_2^{t'})_1)_2, && ((\mathcal{O}_2^t)_1)_2 = ((\mathcal{O}_2^{t'})_2)_2,\\
&((\mathcal{O}_2^t)_2)_2 = ((\mathcal{O}_2^{t'})_1)_2 \qquad \text{and} && ((\mathcal{O}_2^t)_2)_2 = ((\mathcal{O}_2^{t'})_2)_2.
\end{aligned}$$

Therefore there are *II-constraints* between the options ϵ_{11}^t and $\epsilon_{11}^{t'}$, between ϵ_{12}^t and $\epsilon_{12}^{t'}$, between ϵ_{21}^t and $\epsilon_{21}^{t'}$, between ϵ_{21}^t and $\epsilon_{22}^{t'}$, between ϵ_{22}^t and $\epsilon_{21}^{t'}$ and between ϵ_{22}^t and $\epsilon_{22}^{t'}$.

Summarizing, the constraint function a is defined as follows

$$\begin{aligned}
a(\epsilon_{11}^t, \epsilon_{21}^t) = a(\epsilon_{12}^t, \epsilon_{22}^t) &= \{[y_1^t + minStopTime, y_1^t + maxStopTime]_{60}\}\\
&= \{[12, 16]_{60}\}\\
a(\epsilon_{11}^t, \epsilon_{22}^t) = a(\epsilon_{12}^t, \epsilon_{21}^t) &= \{\emptyset\}\\
a(\epsilon_{11}^t, \epsilon_{11}^{t'}) = a(\epsilon_{12}^t, \epsilon_{12}^{t'}) &= \{[3, 57]_{60}, [3 + y_1^t - y_1^{t'}, 57 + y_1^t - y_1^{t'}]_{60}\}\\
&= \{[3, 57]_{60}, [4, 58]_{60}\}\\
a(\epsilon_{11}^t, \epsilon_{12}^{t'}) = a(\epsilon_{12}^t, \epsilon_{11}^{t'}) &= \{[3, 57]_{60}\}\\
a(\epsilon_{21}^t, \epsilon_{21}^{t'}) = a(\epsilon_{22}^t, \epsilon_{22}^{t'}) &= \{[3, 57]_{60}, [3 + y_2^t - y_2^{t'}, 57 + y_2^t - y_2^{t'}]_{60}\}\\
&= \{[3, 57]_{60}, [5, 59]_{60}\}\\
a(\epsilon_{21}^t, \epsilon_{22}^{t'}) = a(\epsilon_{22}^t, \epsilon_{21}^{t'}) &= \{[3 + y_2^t - y_2^{t'}, 57 + y_2^t - y_2^{t'}]_{60}\} = \{[5, 59]_{60}\}\\
a(\epsilon_{11}^{t'}, \epsilon_{21}^{t'}) = a(\epsilon_{12}^{t'}, \epsilon_{22}^{t'}) &= \{[y_1^{t'} + minSlackTime, y_1^{t'} + maxSlackTime]_{60}\}\\
&= \{[10, 11]_{60}\}\\
a(\epsilon_{11}^{t'}, \epsilon_{22}^{t'}) = a(\epsilon_{12}^{t'}, \epsilon_{21}^{t'}) &= \{\emptyset\}\\
a(\zeta, \eta) &= \emptyset \qquad \text{for } (\zeta, \eta) \notin \{(\epsilon_{11}^t, \epsilon_{21}^t), (\epsilon_{12}^t, \epsilon_{22}^t), (\epsilon_{11}^t, \epsilon_{22}^t), (\epsilon_{12}^t, \epsilon_{21}^t),\\
&(\epsilon_{11}^t, \epsilon_{11}^{t'}), (\epsilon_{12}^t, \epsilon_{12}^{t'}), (\epsilon_{11}^t, \epsilon_{12}^{t'}), (\epsilon_{12}^t, \epsilon_{11}^{t'}), (\epsilon_{21}^t, \epsilon_{21}^{t'}), (\epsilon_{22}^t, \epsilon_{22}^{t'}),\\
&(\epsilon_{21}^t, \epsilon_{22}^{t'}), (\epsilon_{22}^t, \epsilon_{21}^{t'}), (\epsilon_{11}^{t'}, \epsilon_{21}^{t'}), (\epsilon_{12}^{t'}, \epsilon_{22}^{t'}), (\epsilon_{11}^{t'}, \epsilon_{22}^{t'}), (\epsilon_{12}^{t'}, \epsilon_{21}^{t'})\}.
\end{aligned}$$

This flexible periodic event network $\mathcal{N} = (\mathcal{V}, T, h, a)$ is visualized in Figure 4.2. In Figure 4.3, one can see the feasible and infeasible regions for the constraints between the options of the periodic events ϵ_1^t and ϵ_2^t. In Figure 4.4, one can see the feasible and infeasible regions for the constraints between the options of the periodic events ϵ_1^t and $\epsilon_1^{t'}$.

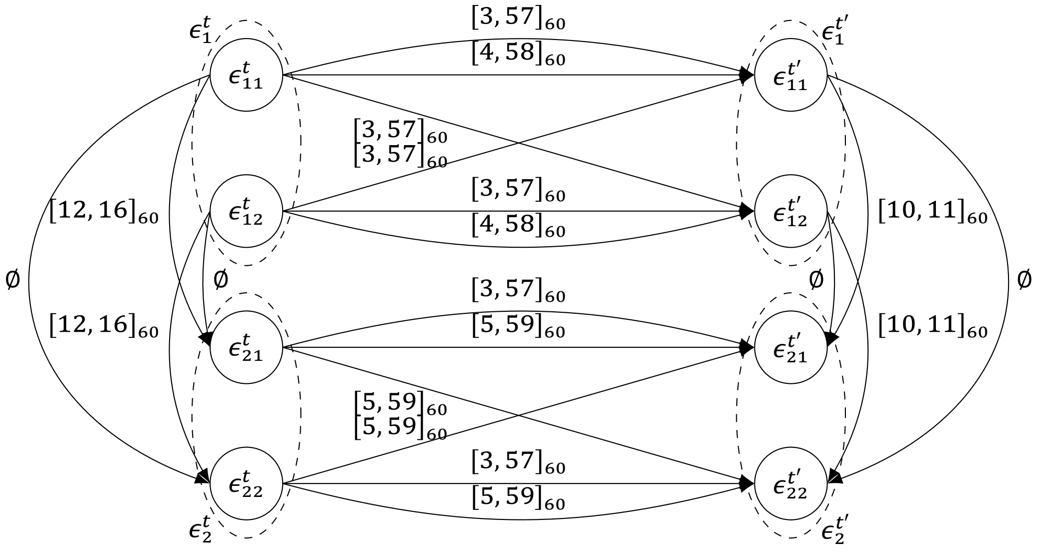


Figure 4.2: Flexible Periodic Event Network of Example 4.3.2

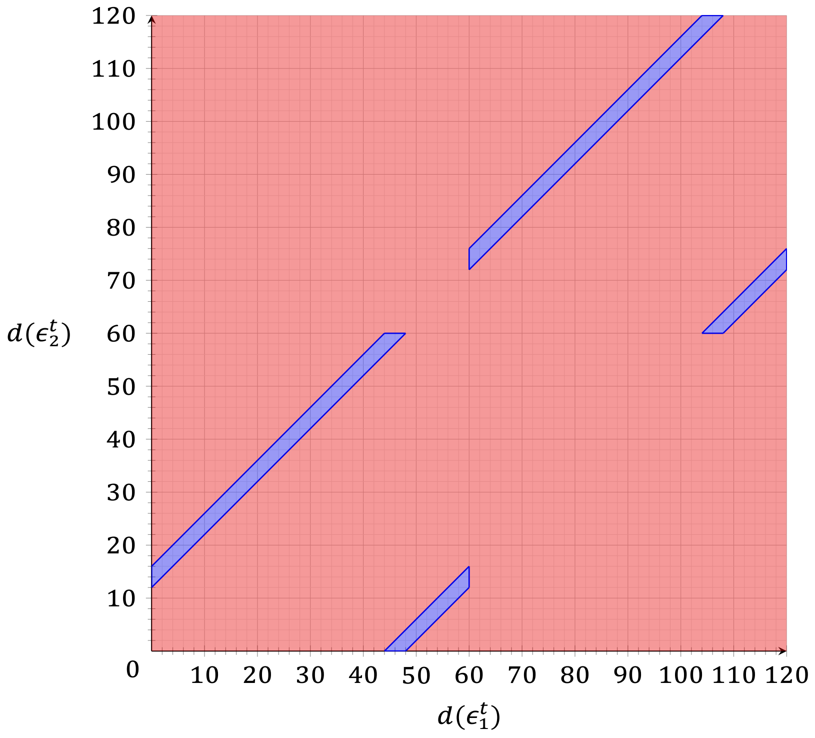


Figure 4.3: Feasible (in blue) and infeasible regions (in red) for the constraints between the options of the periodic events ϵ_1^t and ϵ_2^t from Example 4.3.2. Dividing the chart into four squares, the square in the lower left corner corresponds to the constraints between ϵ_{11}^t and ϵ_{21}^t. There the difference should be between 12 and 16 minutes. The square in the upper left corner corresponds to the constraints between ϵ_{11}^t and ϵ_{22}^t, an infeasible combination. The square in the lower right corner also corresponds to an infeasible combination, between ϵ_{12}^t and ϵ_{21}^t. The square in the upper right corner corresponds to the constraints between ϵ_{12}^t and ϵ_{22}^t, where the difference should again be between 12 and 16 minutes.

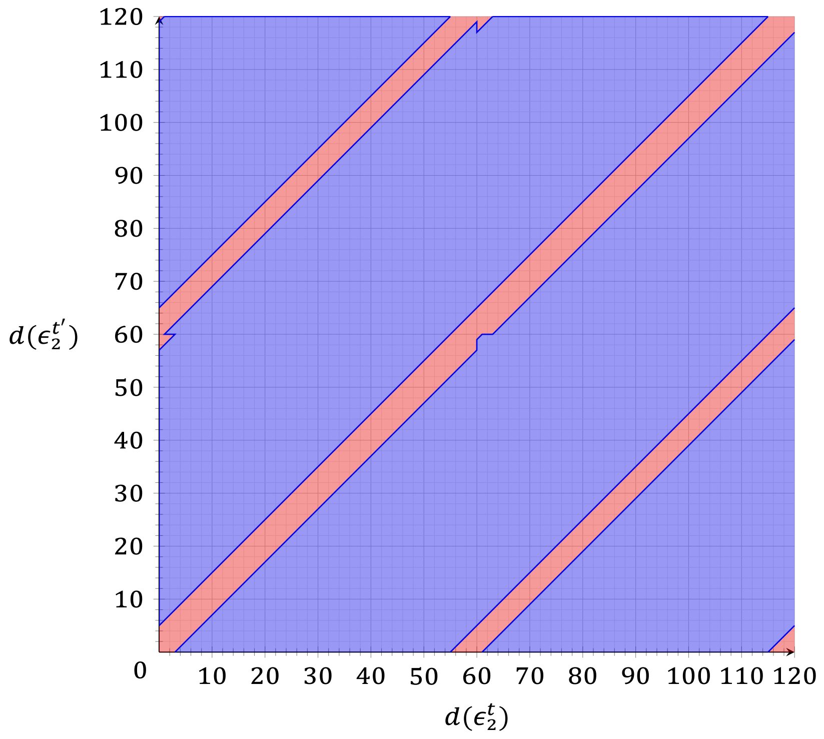


Figure 4.4: Feasible (in blue) and infeasible regions (in red) for the constraints between the options of the periodic events ϵ_2^t and $\epsilon_2^{t'}$ from Example 4.3.2. Dividing the chart into four squares, the square in the lower left corner corresponds to the constraints between ϵ_{21}^t and $\epsilon_{21}^{t'}$. There the difference should be between 5 and 57 minutes. The square in the upper left corner corresponds to the constraints between ϵ_{21}^t and $\epsilon_{22}^{t'}$, where the difference should be between 5 and 59 minutes. The square in the lower right corner also corresponds to a difference between 5 and 59 minutes, between ϵ_{22}^t and $\epsilon_{21}^{t'}$. The square in the upper right corner corresponds to the constraints between ϵ_{22}^t and $\epsilon_{22}^{t'}$, where the difference should again be between 5 and 57 minutes.

4.4. Solving OPESP Railway Timetabling using SAT

To solve the OPESP in the railway timetabling case, we extend the SAT encoding from Chapter 3. The pseudocode of the algorithm is given in Algorithm 1. We will now discuss the algorithm in more detail.

Setup First the infrastructure is defined, where the stage points, the possible stages between them and possible routes for trains are given as input. Next, based on the routes of the trains that have to be scheduled in this specific case, the stage options for each train are created. For each stage, the number of stage options is known and the options are indexed. Based on these options and the stages that they correspond to, the constraints for every pair of stage options are determined as described in the previous section.

Adding options Next, an iterative phase is started where in each iteration, new options are added by $addNewOptions$ to the set $stageOptions$, which is used in the solving phase. In the first iteration, we try to solve a PESP version of the problem, where the route is fixed. This could be a random route or a preferred route for each train.

There are multiple ways to decide which new options are added. One option for the function $addNewOptions$ is to add the next stage option for each stage in each iteration. The pseudocode of this function is given in Algorithm 2. In fact, then we start with $x = 1$ and then iteratively we try to solve OPESP with OPEN $\mathcal{N} = (V, T, x, a)$ and do $x = x + 1$, until all options are added.

The function $addNewOptions$ could also be more specific. In case of an unsatisfiable set of stage options, the SAT solver could give us information about where the unsatisfiability could come from. This information could be used to choose promosing new options to add.

Solving phase Then the solving phase can start. We try to solve the problem with the stage options in the set $stageOptions$. It is determined how many stage options for each stage there are in this set, say n, and then a SAT variable with a domain of $60n$ values is created, as in Subsection 3.2.1. For each constraint, we check if this is a constraint between two stage options that are both in the set $stageOptions$. If this is the case, we have to take this constraint into account and SAT constraints are created to describe the infeasible region imposed by this constraint, as in Subsection 3.2.2.

When the SAT problem with these variables and constraints is unsatisfiable, new options are added and we try again, until a solution is found or until the maximum of iterations is reached.

Algorithm 1: Railway Timetabling

```
defineInfrastructure;
createStageOptions;
createConstraints;
Set $nOptions = 1$;
Set $solutionFound = false$;
while $Not\ solutionFound\ \&\ nOptions \leq maxOptions$ do
    addNewOptions($nOptions$) to $stageOptions$;
    for departure $\epsilon_i^t$ do
        Set $n$ = number of options for $\epsilon_i^t$ in $stageOptions$;
        add SAT constraints for variable with a domain of $60n$ values;
    end
    for constraint $c \in a(\epsilon_{ik}^t, \epsilon_{i'k'}^{t'})$ do
        if both $\epsilon_{ik}^t$ and $\epsilon_{i'k'}^{t'}$ in $stageOptions$ then
            add SAT constraints for infeasible region imposed by constraint $c$;
        end
    end
    tryToSolve;
    if $output = SATISFIABLE$ then
        $solutionFound = true$;
        giveSolution;
    else
        $nOptions = nOptions + 1$;
    end
end
```

Algorithm 2: addNewOptions($nOptions$)

```
for departure $\epsilon_i^t$ do
    for departure option $\epsilon_{ij}^t$ do
        if $j <= nOptions$ & Not($\epsilon_{ij}^t$ in $stageOptions$) then
            add $\epsilon_{ij}^t$ to $stageOptions$;
        end
    end
end
```

5 Results

In this chapter, we show the results obtained from experiments with our method to incorporate multiple route options. First, we introduce the test instance for which we conduct the experiments. Second, we present resulting timetables of the experiments. Lastly, we analyse the results quantitatively.

For the following experiments, we have used Algorithm 1 in Section 4.4, with Algorithm 2 for the function addNewOptions. These algorithms are coded in Java 8 Update 171 and the model is implemented in Eclipse IDE for Java Developers 2019-12 (4.14). Problem instances are solved with the open-source SAT solver MiniSAT 1.14. MiniSAT is a small, complete, and efficient SAT solver in the style of conflict-driven learning [29]. A laptop with Intel®Core™i7-4510U CPU @ 2.00 GHz Processor with 8,0 GB RAM memory is used to run the experiments.

5.1. Description of the Test Instance

As a test instance, a network based on a part of the Dutch railway network is used. This network is visualized in Figure 5.1. The list of abbreviations used is given in Appendix C. The network is a slightly simplified version of the part of the railway network between Amsterdam Sloterdijk, Enkhuizen and Den Helder, in the province Noord-Holland. There are connections to the rest of the Dutch railway network from Haarlem (Hlm), Amsterdam Erasmusgracht aansluiting (Aeg) and Overbrakerpolder aansluiting (Obpa). Most of the stage points in 5.1 are also stage points in the Dutch railway network, although some stage points are added to make it possible to create the constraints automatically. This is, for example, the case for Hn1, Utg1 and Ut2, but does not have influence on the resulting timetable.

The line planning in Table 5.1 is used to create a timetable. There are six different train routes, three intercity trains and three sprinter trains, which all have a frequency of 2. When taking all six train routes into account, there are 408 departures to schedule.

For each stage, only the options that correspond to ”driving on the right” are taken into account, where in case of an odd number of tracks, the middle track is taken into account for both ways. This largely corresponds to reality and keeps the number of options, and therefore the size of the problem, limited. In this network, this results in a maximum of four route options per departure. The order in which the route options are added is the same for each train at each stage point. In every iteration, an extra route option is added for each stage, until all options are added.

To create the constraints as in Section 4.3, the following values are used:

$$\begin{aligned} minSlackTime &= 0, \\ maxSlackTime &= 1, \\ minStopTime &= 1, \\ maxStopTime &= 3. \end{aligned}$$

Trains of the same line as specified in Table 5.1 must be in frequency with a margin of $m = 2$, attained by *F-constraints* as described in Section 4.3.

Index	Train type	Departure	Arrival	Stops	Frequency
1	IC	Obpa	Amr	Ass, Zd, Cas, Amr	2
2	IC	Amr	Obpa	Amr, Cas, Zd, Ass	2
3	IC	Obpa	Hdr	Ass, Zd, Cas, Hlo, Amr, Amrn, Hwd, Sgn, Ana, Hdrz, Hdr	2
4	IC	Hdr	Obpa	Hdr, Hdrz, Ana, Sgn, Hwd, Amrn, Amr, Hlo, Cas, Zd, Ass	2
5	IC	Obpa	Ekz	Ass, Hn, Hnk, Bkg, Bkf, Ekz	2
6	IC	Ekz	Obpa	Ekz, Bkf, Bkg, Hnk, Hn, Ass	2
7	SP	Obpa	Utg	Ass, Zd, Kz, Zzs, Wm, Kma, Utg	2
8	SP	Utg	Obpa	Utg, Kma, Wm, Zzs, Kz, Zd, Ass	2
9	SP	Obpa	Hn	Ass, Hw, Hlms, Hlm, Bll, Sptz, Sptn, Drh, Bv, Hk, Utg, Cas,Hlo, Amr, Amrn, Hwd, Obd, Hn	2
10	SP	Hn	Obpa	Hn, Obd, Hwd, Amrn, Amr, Hlo, Cas, Utg, Hk, Bv, Drh, Sptn, Sptz, Bll, Hlm, Hlms, Hw, Ass	2
11	SP	Aeg	Hnk	Ass, Zd, Zdk, Pmw, Pmr, Pmo, Hn, Hnk	2
12	SP	Hnk	Aeg	Hnk, Hn, Pmo, Pmr, Pmw, Zdk, Zd, Ass	2

Table 5.1: Line planning of the first experiment

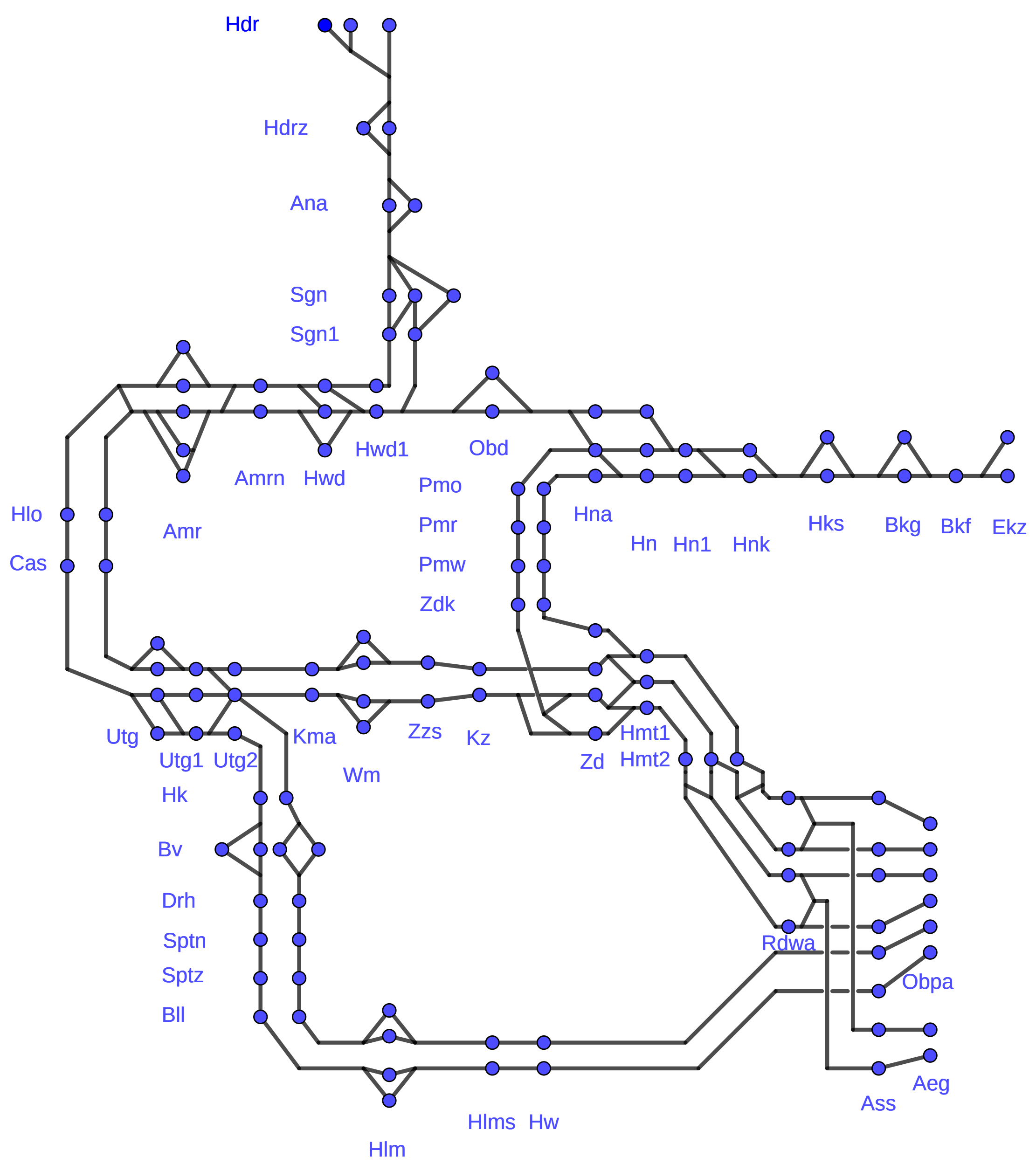


Figure 5.1: Overview of the network

5.2. Resulting Timetables

In this section, we will discuss the resulting timetables of the experiments. First, as the main experiment, the network and line planning from Section 5.1 are used as input. We will discuss the time space diagram of the resulting timetable between Amsterdam Sloterdijk and Alkmaar and the platform occupation chart for Amsterdam Sloterdijk. Second, another experiment is done by taking only lines 1 to 8 from Table 5.1 as input. The resulting timetable of the first experiment is compared to the resulting timetable of the second experiment by comparing the platform occupation charts for Amsterdam Sloterdijk.

5.2.1. Resulting timetable of the first experiment

For the first and main experiment, where the network and line planning from Section 5.1 are used as input, three iterations were needed to get the resulting timetable. The network with only one or two route options per stage, in the order that we defined them, is unsatisfiable. A feasible timetable with three route options per stage is found. The third route option is used around the stage points Amr, Utg and Hdr, and between the stage points Ass and Zd.

Figure 5.2 shows the time space diagram of the resulting timetable between Amsterdam Sloterdijk and Alkmaar. Trains 1H and 2H follow line 1 from Table 5.1, trains 1T and 2T line 2, trains 3H and 4H line 3, etcetera. Indeed, we see that train 1H, leaving Ass just after minute 10, is in frequency with train 2H, leaving Ass just before minute 40. Figure 5.2 also shows that trains in the same direction are mainly crossing at stations, where there are often more tracks. This is, for example, the case at stage point Wm between minutes 0 and 5. Trains that run in different directions tend to have the opportunity to cross between stations as well, since they typically use different tracks.

Admittedly, we see that between minute 5 and 25 mainly trains are scheduled to arrive at Ass, while between minute 25 and minute 45 mainly trains are scheduled to leave Ass. This uneven distribution may lead to long waiting times, which is unattractive to passengers. This example shows that timetables found by this algorithm are not necessarily attractive, as it finds *some* feasible timetable.

To get more insight in the use of the different tracks, the platform occupation chart for Amsterdam Sloterdijk is shown in Figure 5.3. Here, platforms 1 and 2 are used for connections from Obpa to Rdwa, platforms 3 and 4 are used for connections from Rdwa to Obpa, platform 5 for connections from Obpa to Hw, platform 6 for connections from Hw to Obpa, platform 7 for connections from Aeg to Rdwa and platform 8 for connections from Rdwa to Aeg. We see that platform 3 and 4 are both needed as train 8T is scheduled at the same time as trains 3T and 5T. Moreover, trains 6T and 1T are scheduled at the same time as well. Platform 1 and 2 need to be used both too, as train 5H on platform 1 is scheduled too close to train 3H on platform 2 to be scheduled on the same platform.

5.2.2. Resulting timetable of the second experiment

To investigate the difference compared to a problem for which less route options are needed, a second experiment is done. In this experiment only lines 1 to 8 from Table 5.1 are taken as input. Now, a feasible timetable is found using only two route options for each stage. The resulting platform occupation chart for Amsterdam Sloterdijk for this experiment is shown in Figure 5.4. In this case, platform 5, 6, 7 and 8 are not needed as the connections they provide are not used. It is possible to schedule all trains with a connection from/to Obpa to/from Rdwa on platform 2 and 3, instead of using platform 1 and 4 as well. On these tracks, the same number of trains are scheduled as on platform 1, 2, 3 and 4 in Figure 5.3. There could be two reasons why these trains are not scheduled on just two platforms, as we add extra options for each stage in every iteration and create *some* feasible timetable. First, trains might have to be scheduled on the same time at this stage point to make a feasible timetable possible for the whole network. Second, this extra option might just be used because it was possible, although it might not have been needed to add it here.

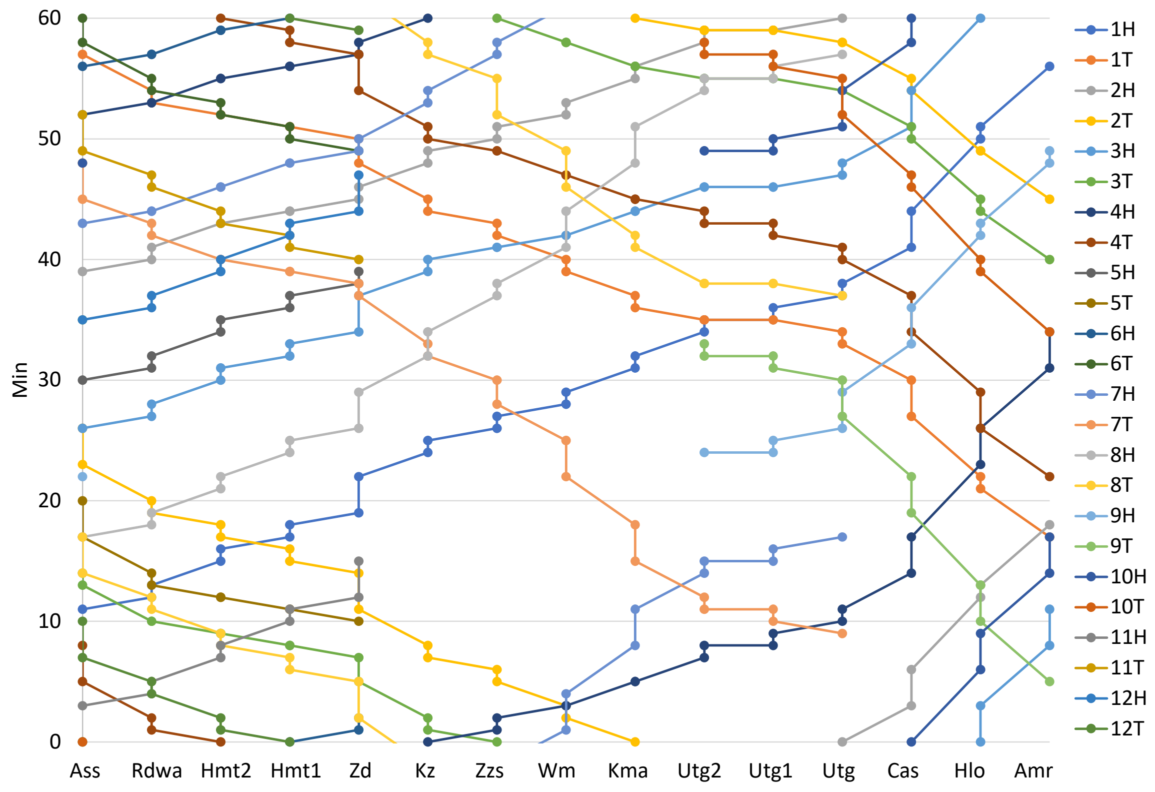


Figure 5.2: Time space diagram of the resulting timetable between Amsterdam Sloterdijk and Alkmaar

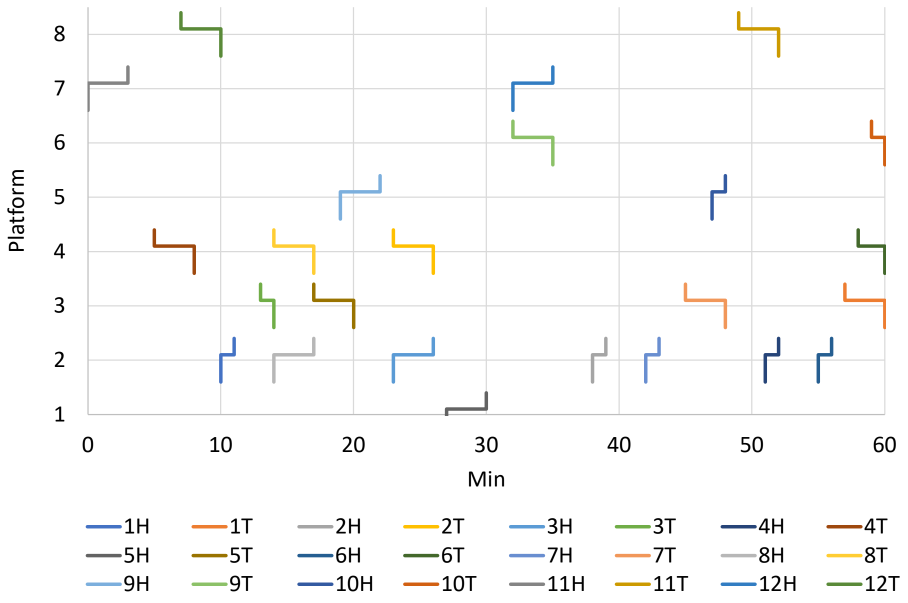


Figure 5.3: Platform occupation chart for Amsterdam Sloterdijk of the resulting timetable

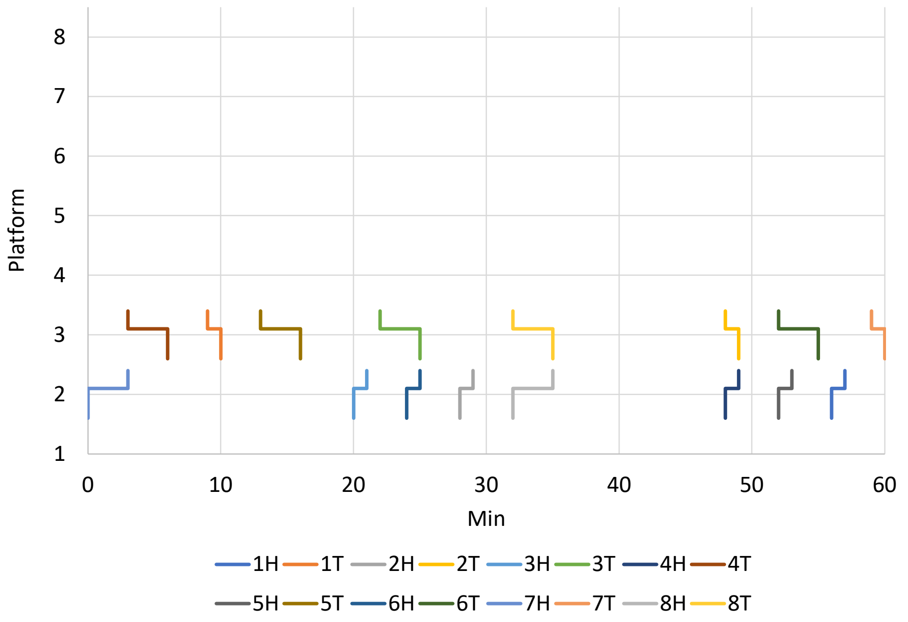


Figure 5.4: Platform occupation chart for Amsterdam Sloterdijk of the resulting reduced timetable

5.3. Quantitative results

We also used the network and line planning from Section 5.1 to analyse the number of SAT clauses needed and the CPU time to solve the problem. To solve this problem, three iterations were needed. By analysing these iterations separately, we get some insight in how this method would perform for bigger instances.

In Figure 5.5, the number of route options is plotted versus the number of SAT clauses. One can see that the number of clauses increases linearly with respect to the number of route options taken into account. When adding another route option, the domain of each variable is extended and more constraints are added, which results in a higher number of clauses. The reason why the number of clauses is not increasing faster is that *I-constraints* can be added in just one SAT clause. For the other constraint types, the number of constraints needed when an extra option is added, is about the same as the number of constraints when having just one route option. Therefore the number of clauses increases linearly with respect to the number of route options taken into account.

In total, less than forty seconds were needed to run the algorithm, including the setup and running the SAT solver three times. In Figure 5.6, the number of route options is plotted versus the CPU time the SAT solver needs to solve this specific problem. One can see the CPU solve time increases more than exponentially with respect to the number of route options taken into account. When taking only one route option into account, the solve time is almost 0 seconds. The problem is designated as trivial by SAT, because at some points no feasible routes can be created. This is due to the fact that the order in which the route options are added is the same for each train at each stage point. For a train going from Obd to Hn via Hna, for example, this is a problem. The first option for a stage between Hna and Hn is via the bottom track in Figure 5.1. But in our network, it is not possible to reach the starting point of this track at Hna from Obd. A second option is needed here to create a feasible route.

The CPU solve time increases more than exponentially, as extra options are added for each stage for which an extra option exists. The combination of route options therefore increases very fast, which results in a longer solve time. However, computation times are still relatively low, as taking three route options into account for every stage is nonetheless solved by the SAT solver in less than 30 seconds.

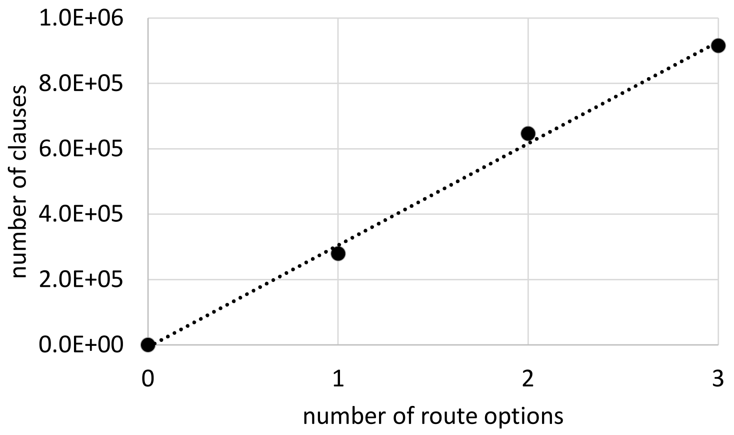


Figure 5.5: The number of SAT clauses increases linearly with respect to the number of route options taken into account

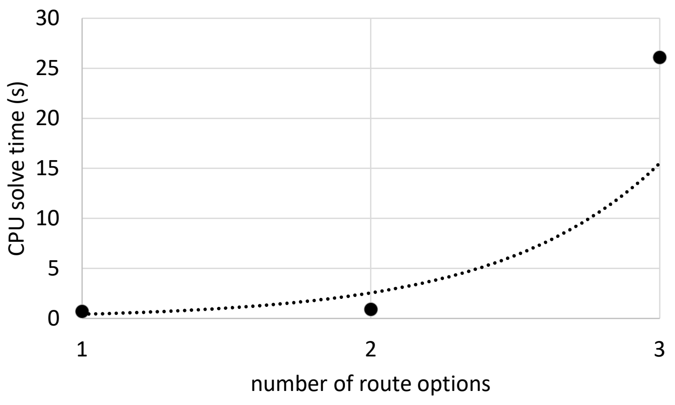


Figure 5.6: The CPU solve time increases more than exponentially with respect to the number of route options taken into account

6 Conclusions and Recommendations

In this chapter, we present the conclusions of our research and discuss the results by giving recommendations for future work and implementation.

6.1. Conclusions

As stated in Chapter 1, the solver CADANS reports unsatisfiability for the Dutch railway timetabling problem, although human planners are able to create a timetable with only small deviations to the original problem. CADANS is based on a PESP model with fixed detailed routes for each train. Incorporating multiple route options for each train in this model results in the consideration of a larger part of the solution space, which might make it possible to find a feasible timetable for the Dutch railway timetabling problem. Additional extensions to the PESP model are permitted by improvements in SAT-based PESP solving, which is currently the most efficient approach. The goal of this thesis was to **incorporate flexible track use in the SAT formulation of the cyclic railway timetabling problem**.

By introducing the Open-ended Periodic Event Scheduling Problem (OPESP), a method to incorporate flexible track use in the SAT model of the cyclic railway timetabling problem is presented. This method uses the time axis to model a geographical decision, by extending the domain of the departure for each stage of each train corresponding to the number of route options for this particular stage.

Using generic norms to create constraints automatically, this method is tested on a test instance based on a substantial part of the Dutch railway network, namely all train lines in the area between Amsterdam Sloterdijk, Enkhuizen and Den Helder. Our method can solve the cyclic railway timetabling problem for this network within reasonable calculation time. We have therefore shown that it is possible to incorporate flexible track use in the SAT formulation of the cyclic railway timetabling problem.

6.2. Recommendations

As this thesis presents the first method to incorporate flexible track use in the SAT formulation of the cyclic railway timetabling problem, there is much more to explore. We will first give some suggestions for improvements on the method itself. Afterwards, we present our suggestions for future work using this method and our recommendations for the implementation thereof.

6.2.1. Recommendations for improving the method

In this thesis our own, more generic norms are used to create constraints. These norms could be extended to create a situation that is closer to reality. For the Dutch infrastructure there exist detailed norms which could be used. Furthermore, only *F-constraints* are added now when trains have completely the same route. These constraints could be added for each part of the route that is the same. Lastly, connection constraints could be added. These extra constraints will make the problem slightly more difficult, but no substantial challenges are expected in integrating these features into our model.

The process of adding more route options can also be improved. First, currently in every iteration an extra route option is added wherever that is possible. This may lead to consideration of too many options that are not needed to create a feasible timetable. In case of unsatisfiability, SAT can indicate stage points where this unsatisfiability may originate. Around these stage points, extra route options could be added. Second, some commonsensical or other decision rules could be used to make better guesses for the first route option, instead of using the same order for every train. For example, differentiating between intercity trains and sprinter trains might already make a big difference. This would result in less iterations to find a feasible timetable, which might be preferable when considering a larger network as, for example, the entire Dutch railway network.

6.2.2. Recommendations for future research

The method presented in this thesis is tested on a substantial part of the Dutch railway network. However, it would be interesting to test the method on bigger parts of the network. It is recommended to create bigger test instances to verify whether the method still performs well.

Currently an open-source SAT solver is used to solve problem instances. As stated in [24], custom implementation of a SAT solver may be helpful for a particular application. It could be interesting to investigate the use of a specific SAT solver for the cyclic railway timetabling problem. In [29], the authors aim to give sufficient details to allow users of SAT-solvers “to make domain specific extensions of adaptions of current state-of-the-art SAT-techniques, to meet the needs of a particular application area” [p. 1].

The method presented in this thesis is used to solve the cyclic railway timetabling problem considered as a satisfiability problem. The SAT model is previously also used to find an optimal timetable. For example in [17], where an approach based on reinforcement learning, multi-agents and SAT is used. In future research, an attempt could be made to integrate the presented method in methods as discussed in [17] to create an optimal timetable.

6.2.3. Recommendations for implementation

Apart from these theoretical possibilities, the opportunities for NS are important. Our recommendation is to integrate this new method with the system DONS, that is currently used by NS, and the solver CADANS. The method presented in this thesis could be used to find the combination of routes of the trains that lead to a feasible timetable. These routes could then be used as input to the existing solver CADANS to create a timetable. As DONS, the system in which CADANS is used, is integrated with all current planning systems, the created timetable is immediately in the right format. In this case, all existing software could still be used. This is an advantage to the planning department, as they already know how these systems work. Moreover, existing optimisation tools could be used on a feasible timetable to create a better timetable.

A
Proof of Lemma 2.1.1

Lemma 2.1.1 is repeated here, after which the proof is given.

Lemma A.0.1. *Suppose that for some edge $(i,j) \in \mathcal{E}$ we want to impose the constraint*

$$d(j) - d(i) \in [l_1, u_1]_T \cup [l_2, u_2]_T \cup \ldots \cup [l_k, u_k]_T,$$

where the k intervals are disjoint and ordered:

$$0 \leq l_1 \leq u_1 < l_2 \leq u_2 < \ldots < l_k \leq u_k < l_1 + T.$$

Then this union of k intervals is equivalent to the intersection of k intervals modulo T given by the constraints

$$\begin{aligned}
[l_1, u_k]_T &\in a(i,j), \\
[l_2, u_1 + T]_T &\in a(i,j), \\
&\vdots \\
[l_k, u_{k-1} + T]_T &\in a(i,j).
\end{aligned}$$

Proof. We proof this lemma by induction. It is easy to see that the statement is true for $k = 2$, see Figure 2.1. Assume the statement is true for $k-1$, so we have

$$\begin{aligned}
& d(j) - d(i) \in [l_1, u_1]_T \cup [l_2, u_2]_T \cup \ldots \cup [l_{k-1}, u_{k-1}]_T \\
\Leftrightarrow\ & d(j) - d(i) \in [l_1, u_{k-1}]_T \cap [l_2, u_1 + T]_T \cap \ldots \cap [l_{k-1}, u_{k-2} + T]_T
\end{aligned}$$

for $0 \leq l_1 \leq u_1 < l_2 \leq u_2 < \ldots < l_{k-1} \leq u_{k-1} < l_1 + T$.
Then for k, we have

$$\begin{aligned}
& d(j) - d(i) \in [l_1, u_1]_T \cup [l_2, u_2]_T \cup \ldots \cup [l_k, u_k]_T \\
\Leftrightarrow\ & d(j) - d(i) \in \big([l_1, u_{k-1}]_T \cap [l_2, u_1 + T]_T \cap \ldots \cap [l_{k-1}, u_{k-2} + T]_T\big) \cup [l_k, u_k]_T \\
\Leftrightarrow\ & d(j) - d(i) \in \big([l_1, u_{k-1}]_T \cup [l_k, u_k]_T\big) \cap \big([l_2, u_1 + T]_T \cup [l_k, u_k]_T\big) \cap \ldots \\
& \qquad \cap \big([l_{k-1}, u_{k-2} + T]_T \cup [l_k, u_k]_T\big),
\end{aligned}$$

for $0 \leq l_1 \leq u_1 < l_2 \leq u_2 < \ldots < l_k \leq u_k < l_1 + T$, where the second equality follows from the induction hypothesis. We now have $k-1$ unions of two intervals. For the first union, we have $l_1 \leq u_{k-1} < l_k \leq u_k$. Therefore we can apply the base case of this lemma, so

$$[l_1, u_{k-1}]_T \cup [l_k, u_k]_T \ = \ [l_1, u_k]_T \cap [l_k, u_{k-1} + T]_T.$$

The remaining $k-2$ unions are in the following form

$$[l_i, u_{i-1} + T]_T \cup [l_k, u_k]_T \quad \text{for some } i = 2, \dots, k-1.$$

As $l_i < l_k \leq u_k < u_{i-1} + T$ for all $i = 2, \dots, k-1$, we have

$$[l_i, u_{i-1} + T]_T \cup [l_k, u_k]_T = [l_i, u_{i-1} + T]_T \quad \text{for all } i = 2, \dots, k-1.$$

Therefore we have

$$\begin{aligned} & d(j) - d(i) \in \big([l_1, u_{k-1}]_T \cup [l_k, u_k]_T\big) \cap \big([l_2, u_1 + T]_T \cup [l_k, u_k]_T\big) \cap \dots \\ & \qquad\qquad \cap \big([l_{k-1}, u_{k-2} + T]_T \cup [l_k, u_k]_T\big) \\ \Leftrightarrow\ & d(j) - d(i) \in [l_1, u_k]_T \cap [l_2, u_1 + T]_T \cap \dots \cap [l_k, u_{k-1} + T]_T. \end{aligned}$$

■

B Using Decision Variables to Incorporate Multiple Route Options

In this thesis, a method is presented to incorporate multiple route options in the SAT formulation of the cyclic railway timetabling problem. This method consists of extending the domain of the departure times. Another method that might come up is the use of decision variables that indicate the route choice for a departure. In this appendix, we show the difference in number of clauses for these two methods. Therefore we consider a small example.

Problem description

There are two stations, station A and station B. From station A to station B there are two different single-track-routes, α and β, with different lengths. Route α gives a travel time of 31 minutes and route β a travel time of 32 minutes. Train t has to go from station 1 to station 2, train t' from station 2 to station 1. A representation of this example can be seen in Figure B.1. In this case there are four different options: both trains take route α, both trains take route β, train t takes route α and train t' route β, or the other way around. As we consider a cyclic timetable for 60 minutes and both routes only have one track, it is not possible for both trains to take the same route. Therefore only the last two options are feasible.

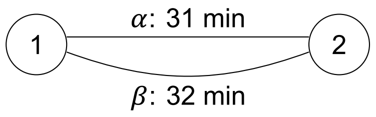


Figure B.1: Example of a railway network

Problem formulation

There are two ways to tackle this problem of choosing routes. The first way is to have a decision variable indicating the chosen route. The second way is to extend the domain of the departure time, according to the number of route options. We will elaborate on both ways by considering the *UI-constraint* that is needed as train t is leaving station 1 and train t' is arriving at this station.

Using a decision variable
In this case, when using a decision variable, the notation of Section 2.2 is used. We would have the following variables:

$d(\epsilon^t)$	departure time of train t,	$d(\epsilon^t) \in \{0, 59\}$,
r_t	route choice of train t,	$r_t \in \{0, 1\}$,
$d(\epsilon^{t'})$	departure time of train t',	$d(\epsilon^{t'}) \in \{0, 59\}$,
$r_{t'}$	route choice of train t',	$r_{t'} \in \{0, 1\}$.

Let $r_i = 0$ correspond to taking route α and $r_i = 1$ to taking route β, for $i = t, t'$. In SAT all these variables will separately be encoded to a propositional formula using order encoding.

The *UI-constraint* is

$$[3 - y_1^{t'}, 59 - y_1^{t'}]_{60} \in a(\epsilon^t, \epsilon^{t'}),$$

where $y_1^{t'}$ is the travel time of train t'. So we have

$$[32, 88]_{60} \in a(\epsilon^t, \epsilon^{t'}) \qquad \text{if } r_{t'} = 0,$$
$$[31, 87]_{60} \in a(\epsilon^t, \epsilon^{t'}) \qquad \text{if } r_{t'} = 1.$$

In SAT we will have the following, where $q_{x,i} \Leftrightarrow x \leq i$:

$$(q_{r_{t'},0} \wedge encode\_ordered\_con((d(\epsilon^t), d(\epsilon^{t'})), [32, 88]_{60}))$$
$$\vee (\neg q_{r_{t'},0} \wedge encode\_ordered\_con((d(\epsilon^t), d(\epsilon^{t'})), [31, 87]_{60})).$$

To use these constraints in SAT, they have to be in CNF (Conjunctive Normal Form). Let ζ be as in Definition 3.2.2. We have

$$encode\_ordered\_con((d(\epsilon^t), d(\epsilon^{t'})), [32, 88]_{60}) =$$
$$\bigwedge_{\{x\}\times[y_1,y_2]\in\zeta((d(\epsilon^t),d(\epsilon^{t'})),[32,88]_{60})} (\neg q_{d(\epsilon^t),x} \vee q_{d(\epsilon^t),x-1} \vee \neg q_{d(\epsilon^{t'}),y_2} \vee q_{d(\epsilon^{t'}),y_1-1})$$

and

$$encode\_ordered\_con((d(\epsilon^t), d(\epsilon^{t'})), [31, 87]_{60}) =$$
$$\bigwedge_{\{x\}\times[y_1,y_2]\in\zeta((d(\epsilon^t),d(\epsilon^{t'})),[31,87]_{60})} (\neg q_{d(\epsilon^t),x} \vee q_{d(\epsilon^t),x-1} \vee \neg q_{d(\epsilon^{t'}),y_2} \vee q_{d(\epsilon^{t'}),y_1-1}).$$

Say both constraints consist of n clauses. If we look at the whole formula again, we get something like

$$\left(q_{r_t,0} \wedge B_1 \wedge \ldots \wedge B_{62}\right) \vee \left(\neg q_{r_t,0} \wedge C_1 \wedge \ldots \wedge C_{62}\right),$$

where $\forall i \; B_i, C_i = (\neg q_{d(\epsilon^t),x} \vee q_{d(\epsilon^t),x-1} \vee \neg q_{d(\epsilon^{t'}),y_2} \vee q_{d(\epsilon^{t'}),y_1-1})$, for some x, y_1, y_2. This formula in CNF is

$$(q_{r_t,0} \vee \neg q_{r_t,0}) \wedge (\bigwedge_{i=1,\ldots,n} q_{r_t,0} \vee C_i) \wedge (\bigwedge_{i=1,\ldots,n} \neg q_{r_t,0} \vee B_i) \wedge (\bigwedge_{i,j=1,\ldots,n} B_i \vee C_j).$$

This is a formula with $1 + 2n + n^2$ clauses.

Extend the domain

In this case, we extend the domain of the departure time, according to the number of route options. Here the notation of Section 4.3. We have the following variables:

$d(\epsilon^t)$	departure time of train t,	$d(\epsilon^t) \in \{0, 119\}$,
$d(\epsilon^{t'})$	departure time of train t',	$d(\epsilon^{t'}) \in \{0, 119\}$.

In SAT all these variables will separately be encoded to a propositional formula using order encoding. Let ϵ_1^t correspond to train t taking route α, ϵ_2^t to train t taking route β, $\epsilon_1^{t'}$ to train t' taking route α and $\epsilon_2^{t'}$ to train t' taking route β.

Now there will be two separate *UI-constraints*:

$$[3 - y_1^{t'}, 59 - y_1^{t'}]_{60} \in a(\epsilon_1^t, \epsilon_1^{t'}),$$
$$[3 - y_1^{t'}, 59 - y_1^{t'}]_{60} \in a(\epsilon_1^t, \epsilon_2^{t'}),$$
$$[3 - y_1^{t'}, 59 - y_1^{t'}]_{60} \in a(\epsilon_2^t, \epsilon_1^{t'}),$$
$$[3 - y_1^{t'}, 59 - y_1^{t'}]_{60} \in a(\epsilon_2^t, \epsilon_2^{t'}),$$

where $y_1^{t'}$ is the travel time of train t'. So we have

$$[32, 88]_{60} \in a(\epsilon_1^t, \epsilon_1^{t'}),$$
$$[31, 87]_{60} \in a(\epsilon_1^t, \epsilon_2^{t'}),$$
$$[32, 88]_{60} \in a(\epsilon_2^t, \epsilon_1^{t'}),$$
$$[31, 87]_{60} \in a(\epsilon_2^t, \epsilon_2^{t'}).$$

In SAT we will encode all these constraints separately and take them together in one conjunction. The encoding per constraint will be the same as for the method using decision variables, so each of the constraints consist of n clauses. Therefore we will have $4n$ clauses using the method presented in this thesis, instead of the $1 + 2n + n^2$ clauses in the first method. This difference will even be bigger when there are stages with three or more route options.

C
Abbreviations of Stage Points

Table C.1 contains the abbreviations of the stage points that are used in this thesis.

Abbreviation	Stage point	Abbreviation	Stage point
Aeg	Amsterdam Erasmus-gracht aansluiting	Hnk	Hoorn-Kersenboogerd
		Hw	Halfweg
Amr	Alkmaar	Hwd	Heerhugowaard
Amrn	Alkmaar Noord	Hwd1	Heerhugowaard 1
Ana	Anna Paulowna	Kma	Krommenie-Assendelft
Ass	Amsterdam Sloterdijk	Kz	Koog aan de Zaan
Bkf	Bovenkarspel Flora	Nhk	Noordhollandse kanaalbrug
Bkg	Bovenkarspel-Grootebroek		
		Obd	Obdam
Bll	Bloemendaal	Obpa	Overbrakerpolder aansluiting
Bv	Beverwijk		
Cas	Castricum	Pmo	Purmerend Overwhere
Drh	Driehuis	Pmr	Purmerend
Ekz	Enkhuizen	Pmw	Purmerend Weidevenne
Hdr	Den Helder	Rdwa	Rdwa
Hdrz	Den Helder Zuid	Sgn	Schagen
Hk	Heemskerk	Sgn1	Schagen 1
Hks	Hoogkarspel	Sptn	Santpoort Noord
Hlm	Haarlem	Sptz	Santpoort Zuid
Hlms	Haarlem Spaarnwoude	Utg	Uitgeest
Hlo	Heiloo	Utg1	Uitgeest 1
Hmt1	Hemtunnel 1	Utg2	Uitgeest 2
Hmt2	Hemtunnel 2	Wm	Wormerveer
Hn	Hoorn	Zd	Zaandam
Hn1	Hoorn 1	Zdk	Zaandam Kogerveld
Hna	Hoorn aansluiting	Zzs	Zaandijk Zaanse Schans

Table C.1: Abbreviations of stage points

Bibliography

[1] Netherlands Railways. Annual Report 2019. Technical report, NS, 2019.

[2] NS. Halfjaarcijfers NS: Nettoverlies van €185 miljoen, afspraken over aanvullende steun voor ov-sector noodzakelijk, August 2020. URL https://nieuws.ns.nl/halfjaarcijfers-ns-nettoverlies-van--185-miljoen-afspraken-over-aanvullende-steun-voor-ov-sector-noodzakelijk/.

[3] J.-W. Goossens, S. Van Hoesel, and L. Kroon. A branch-and-cut approach for solving railway line-planning problems. *Transportation Science*, 38(3):379–393, 2004.

[4] E. Abbink, B. Van den Berg, L. Kroon, and M. Salomon. Allocation of railway rolling stock for passenger trains. *Transportation Science*, 38(1):33–41, 2004.

[5] E.J.W. Abbink, L. Albino, T. Dollevoet, D. Huisman, J. Roussado, and R.L. Saldanha. Solving large scale crew scheduling problems in practice. *Public Transport*, 3(2):149–164, 2011.

[6] C. Liebchen and L. Peeters. On cyclic timetabling and cycles in graphs. Technical report, Technische Universität Berlin, 2002.

[7] G.J. Polinder. Resolving infeasibilities in the PESP model of the Dutch railway timetabling problem. Master's thesis, Delft University of Technology, 2015.

[8] Leon Peeters. *Cyclic railway timetable optimization*. PhD thesis, Erasmus University Rotterdam, 2003.

[9] J.H.A. Van den Berg and M.A. Odijk. DONS: Computer aided design of regular service timetables. *WIT Transactions on The Built Environment*, 7, 1994.

[10] A. Schrijver and A. Steenbeek. Dienstregelingontwikkeling voor Railned: rapport CADANS 1.0. *Report, CWI, Amsterdam*, 1994.

[11] P. Großmann. Polynomial Reduction from PESP to SAT. Technical report, Technische Universität Dresden, 2011. URL https : / / iccl . inf . tu - dresden.de/w/images/1/1a/Report11-05.pdf.

[12] P. Serafini and W. Ukovich. A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematics*, 2(4):550–581, 1989.

[13] S.A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.

[14] M. Kümmling, P. Großmann, K. Nachtigall, J. Opitz, and R. Weiß. A state-of-the-art realization of cyclic railway timetable computation. *Public Transport*, 7(3): 281–293, 2015.

[15] P. Gattermann, P. Großmann, K. Nachtigall, and A. Schöbel. Integrating passengers' routes in periodic timetabling: a SAT approach. In *16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[16] G.P. Matos, L. Albino, R.L. Saldanha, and E.M. Morgado. Optimising Cyclic Timetables with a SAT Approach. In *EPIA Conference on Artificial Intelligence*, pages 343–354. Springer, 2017.

[17] G.P. Matos, L.M. Albino, R.L. Saldanha, and E.M. Morgado. Solving periodic timetabling problems with SAT and machine learning. In *Conference on Advanced Systems in Public Transport*, 2018.

[18] R.M. Lusby, J. Larsen, M. Ehrgott, and D. Ryan. Railway track allocation: models and methods. *OR spectrum*, 33(4):843–883, 2011.

[19] R. Weiß, M. Kümmling, and J. Opitz. On Optimally Allocating Tracks in Complex Railway Stations. In *Operations Research Proceedings 2015*, pages 287–292. Springer, 2017.

[20] M.E.H. Petering, M. Heydar, and D.R. Bergmann. Mixed-integer programming for railway capacity analysis and cyclic, combined train timetabling and platforming. *Transportation Science*, 50(3):892–909, 2016.

[21] P. Großmann, S. Hölldobler, N. Manthey, K. Nachtigall, J. Opitz, and P. Steinke. Solving periodic event scheduling problems with SAT. In *International conference on industrial, engineering and other applications of applied intelligent systems*, pages 166–175. Springer, 2012.

[22] J. Franco and J. Martin. A History of Satisfiability. In A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of satisfiability*, chapter 1, pages 3–74. IOS press, 2009.

[23] E.A. Lee, J. Roychowdhury, and S.A. Seshia. Fundamental Algorithms for System Modeling, Analysis, and Optimization: Boolean Satisfiability (SAT) Solving. *University of California, Berkeley*, 2011.

[24] L. Zhang and S. Malik. The quest for efficient boolean satisfiability solvers. In *International Conference on Computer Aided Verification*, pages 17–36. Springer, 2002.

[25] M.W. Moskewicz, C.F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th annual Design Automation Conference*, pages 530–535, 2001.

[26] H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical logic*. Springer Science & Business Media, 2013.

[27] B. Chambers, P. Manolios, and D. Vroon. Faster SAT solving with better CNF generation. In *2009 Design, Automation & Test in Europe Conference & Exhibition*, pages 1590–1595. IEEE, 2009.

[28] N. Tamura, A. Taga, S. Kitagawa, and M. Banbara. Compiling finite linear CSP into SAT. *Constraints*, 14(2):254–272, 2009.

[29] N. Eén and N. Sörensson. An extensible SAT-solver. In *International conference on theory and applications of satisfiability testing*, pages 502–518. Springer, 2003.