Routing Trains through Railway Stations: Planning, Robustness and Rescheduling

Tessa Matser

June 10, 2014
## Contents

1 Introduction 3

2 Problem Statement 5
   2.1 Current situation/Planning Process ........................................ 5
   2.2 Train routing problem ............................................................... 5
   2.3 Formalization and Notation .......................................................... 6
   2.4 Complexity issues ..................................................................... 8
   2.5 Possible routes .......................................................................... 10
      2.5.1 Determining possible routes ................................................... 12
      2.5.2 Conflicting train pattern pairs ................................................. 13

3 Searching for robust routings 15
   3.1 Routing methods from literature .................................................... 15
      3.1.1 Conflict graphs ................................................................... 15
      3.1.2 Graph colouring .................................................................... 19
      3.1.3 Multicommodity flow ............................................................. 20
      3.1.4 Set packing for time interval track sections ................................ 24
      3.1.5 Local search ......................................................................... 24
   3.2 Routing model: example case ........................................................ 25
   3.3 Basic routing model ................................................................. 26
   3.4 Routing with maximal buffer times ............................................ 30
      3.4.1 Choosing penalty function ...................................................... 31
   3.5 Routing with small differences in time ........................................ 32

4 Measures for robustness 37
   4.1 Deterministic stability measures .................................................. 37
      4.1.1 Buffer times between consecutive events .................................. 38
      4.1.2 Dependencies between all pairs of events ................................ 39
   4.2 Probabilistic Stability Measure: Robustness based on resulting delays 40
      4.2.1 Objective function ................................................................ 41
      4.2.2 Trains from/to the same direction ........................................... 41

5 Rescheduling strategies 43
5.1 Assumptions for rescheduling ........................................ 44
5.2 Delay Propagation strategy ......................................... 45
  5.2.1 Description of algorithm .................................... 47
5.3 Delayed order strategy .............................................. 48
5.4 Re-optimizing strategy .............................................. 50
  5.4.1 Model ............................................................ 50

6 Routing by Stochastic Optimization .................................. 57
  6.1 Implementation details ............................................. 59

7 Experimental results ...................................................... 62
  7.1 Comparing different routing plans: deterministic measures .... 62
  7.2 Distribution of delays ........................................... 64
  7.3 Influence of different kinds of conflicts ....................... 65
  7.4 Routing plans combined with rescheduling strategies ........ 65
  7.5 Influence of the distribution of initial delay ................. 67
  7.6 Influence of assumptions in reoptimization strategy ......... 67
  7.7 Quality of SOM plans under wrong assumptions ............. 68

8 Conclusions and discussion ............................................. 69

A Solving ILP in CPLEX .................................................. 71

Bibliography ................................................................. 71
Chapter 1

Introduction

The Dutch railway tracks are used very intensively. With about 50,000 train km/track km each year, the Dutch tracks belong to the busiest in Europe (Nederlandse Spoorwegen [2013]). This means that the times and routes of the different trains have to be matched carefully in order to prevent conflicts. Railway stations are often the bottlenecks in the network because many trains come together there. Small delays can easily lead to larger disruptions due to the interaction between different trains within the station. In order to prevent delay propagation, the platforms and routes of the trains travelling through the railway station are therefore important.

In the passenger railway planning process, several steps have to be taken. First, a tentative timetable is generated, consisting of the arrival and departure times of the trains at the different railway stations. It is based on the rough layout of the railway network without taking into account the detailed layout of the stations. In the second step, the feasibility of the tentative timetable is checked within the stations by working out the details. Each train is assigned to a platform and to a route leading to and from this platform. The Train Routing Problem refers to the part of the train scheduling problem that concerns routing trains through railway stations.

In most Dutch railway stations, certain platforms are chosen to serve as a standard platform for a certain direction. This is easy for the customers, as they can remember at which platform they have to be. Mostly, the shortest route between the entry/exit point and this platform is chosen to be the standard. A large part of the planning of the routes is done by hand by dispatchers. They are supported by the decision support system STATIONS, which searches for a feasible routing based on the tentative timetable.

The STATIONS module is based on a weighted node packing formulation developed by Zwaneveld et al. [1996], Zwaneveld [1997], Zwaneveld et al. [2001]. Later, an extension to improve the objective function was proposed by Kroon and Maróti [2008]. However, this extension of the model has not yet been included in the STATIONS module, used by the Netherlands Railways.
Several other approaches have been proposed in the literature to solve the Train Routing Problem. A review by Lusby et al. [2011] summarized many of them. Because of the difference in requirements, layout and occupancy of the tracks in different countries, not all methods are useful for or relevant to the Dutch situation. However, ideas from these methods can be used.

Punctuality is an important quality measure for the service of the NS. It is strongly influenced by the robustness of the schedule. We call a scheme robust if it avoids the propagation of delay as much as possible. Train delays can have many different causes that are inevitable from a planner’s perspective, such as technical problems. However, many punctual trains are delayed due to the interaction with other delayed trains. This means that if we can avoid this interaction, then the amount of delay will be reduced. Different measures for robustness are given by Herrmann [2006]. By calculating the robustness of different schedules, the quality of the plans can be compared.

The first aim of this thesis is to develop algorithms for the Train Routing Problem with robustness objectives. To this end, a basic routing model was developed that was extended in different ways. The second objective is to find a good measure for robustness of routings.

In case delays occur that would lead to conflicts, they have to be solved by real time rescheduling. Currently, this is done by hand by dispatchers. Decision support systems could help them in order to make the best decisions. Moreover, a realistic rescheduling strategy is needed in order to model the performance of routing plans in practice. The final objective of this thesis is to develop rescheduling strategies such that the quality of the routings can be tested.
Chapter 2

Problem Statement

2.1 Current situation/Planning Process

The process of timetable generation roughly consists of two stages. In the first stage, a network timetable is generated. In this stage, the individual stations are considered as nodes in the railway network and a detailed route of the trains through the station is not specified. The planners are in this stage supported by the module CADANS[]. In the second stage, the timetable is worked out in detail at the individual nodes of the network. Then it is checked if the original timetable is feasible at all the stations and a routing of the trains through the railway station is generated. In this stage, the planners are supported by the module STATIONS.

[Meer uitleg CADANS, STATIONS, Dagplan...]

2.2 Train routing problem

The train routing problem aims at defining a detailed routing plan for a given railway station whose timetable has already been defined. We consider the problem as it was defined in Zwaneveld [1997].

The layout of a railway station is assumed to be known, as is the set of trains that have to pass through the station. For these trains, the planned arrival and departure times and directions are fixed. The aim is thus to find a suitable route for each of the trains from the arrival to the departure direction via one of the platforms.

Several conditions are imposed on such a routing plan. Conditions on individual trains limit the number of possible routes and platforms for a certain train. Simultaneous use of platforms and tracks is avoided by conditions on combinations of trains. These conditions together ensure that no pair of trains is scheduled on the same platform at
the same time and that no conflicting use of tracks will occur if all trains are punctual. The first aim of this thesis is to develop methods for finding suitable routing plans. This topic is treated in chapter 3.

The quality of the routing plan can be expressed by an objective function. Short routes should be favoured above long routes and some platforms could be preferred, for example because they are closer to the entrance of the railway station. Moreover, a good routing plan should be able to cope with small disturbances. Long buffer times between trains that use the same platform or track sections could be preferred above short buffer times. In chapter 4, different measures for robustness of the routing plan are discussed.

In order to determine the performance of a routing plan in practice, it should be known how rescheduling decisions are made in real-time situations. The final purpose of this thesis is to develop and compare different rescheduling strategies. This is covered in chapter 5.

2.3 Formalization and Notation

To formalize the problem, let $T$ be the set of trains that have to be routed through the railway station. We assume that the schedule is cyclic with a cycle length of 60 minutes. This means that we only need to consider the arrivals and departures of trains that occur in one hour. Unless stated otherwise, we denote by $T$ the set of trains that occur in one hour. The railway station consists of a large set of track sections $S$ that cover all infrastructure elements that can be used by the trains. We assume that each track section can be used by at most one train at the same time. Let $D \subseteq S$ be the set of entry and exit points of the station (all corresponding to a direction of travel). The pair of entering and leaving point for a train $t$ is denoted by $D(t) = (D(t)_a, D(t)_d) \in D^2$.

Let $B \subseteq S$ be the set of platform tracks. A platform track may be adjacent to a platform, but may also by-pass all platforms. For each train $t \in T$ let the corresponding arrival event be denoted by $t_a$ and the departure event by $t_d$. The pair of planned arrival and departure time (in minutes) be given by $\tau(t) = (\tau(t)_a, \tau(t)_d) \in \mathbb{N}^2$. The train is planned to stop at the platform at the arrival time and is planned to leave at the departure time. For a train that does not stop at the station, arrival and departure time are equal and refer to the time at which the train is planned to pass the platform track.

A route consists of consecutive track sections. Let $R_a$ be the set of inbound routes leading from an entry point of the station to a platform track. Let $R_d$ be the set of outbound routes leading from a platform track to a station exit point. Section 2.5 describes in more detail how these routes can be found. For $r \in R_a$ or $r \in R_d$ we denote by $D(r) \in D$ the entry or exit point of the route.

A pattern for a train $t$ refers to a feasible combination of a platform and in- and outbound
route. A pattern consisting of a platform $b \in B$, inbound route $r_a$ and outbound route $r_d$, is feasible for a train $t$ if the inbound route starts from the arrival direction of the train $(D(r_a) = D(t)_a)$, the outbound route leads to the departure direction of the train $(D(r_d) = D(t)_d)$, the inbound route leads to the platform $b$ and the outbound route starts from $b$. The set of all feasible patterns for train $t$ is denoted by $\mathcal{P}_t$.

A routing plan is given by a function $\phi : T \to \cup_{t \in T} \mathcal{P}_t$ such that $\phi(t) \in \mathcal{P}_t$ for each $t \in T$. We can consider $\phi$ to be a set of train-pattern pairs $(t, P)$ with $P = \phi(t) \in \mathcal{P}_t$ that consists of exactly one pair for each train.

Each train claims its platform, inbound and outbound route for a certain period of time. Because each element of the infrastructure can only be claimed by one train at a time, some pairs of train-pattern pairs are incompatible. A feasible routing should not contain incompatible train-pattern pairs. Two patterns are incompatible if they plan to claim the same track section (platform or part of the route) for a non-disjoint period of time. Section 2.5.2 provides more details on incompatible patterns.

It is often assumed that the objective function has a specific form. The preference for a certain train-pattern pair can be expressed by a penalty $p_{t,P}$ for train $t$ and pattern $P$. A penalty for a combination of two train pattern pairs $q_{(t,P),(t',P')}$ expresses the disadvantage of the combination of pattern $P$ for train $t$ and pattern $P'$ for train $t'$. The sum of the penalties then expresses the quality of the routing plan.

If it is not possible to find a plan of compatible patterns for all trains, then we could allow some small deviations from the planned times. We can impose an additional shift $s$ in minutes on the planned arrival and/or departure time. This could, of course, influence the rest of the network, so the shifts cannot be imposed independently of the other stations. However, it could be useful feedback to the larger planning system that a certain time shift for a train would improve the situation at a certain railway station.

[deze letters voor vertraging nog niet helemaal goed gekozen... Ze komen uit Kroon et al. [2007]]

The random vector variables $\Delta_a$, $\Delta_h$, $\Delta_d$ describe the primary disturbances on the arrival, halting and departure times. Let $\delta_a : T \to \mathbb{R}^+$ be a realisation of $\Delta_a$ that gives the difference between the planned and realised initial arrival times (the time that the train is ready to enter the station area plus the time it would take to reach the platform without having any conflict with other trains). A realisation of $\Delta_h$ is given by $\delta_h : T \to \mathbb{R}^+$. This halting disturbance gives for every train the additional halting time above the minimal halting time. By a realisation $\delta_d : T \to \mathbb{R}^+$ of $\Delta_d$, we describe the delay that occurs on the moment of departure or during the outbound route. In $\Delta_h$ and $\Delta_d$, we do not take into account the time that a train has to wait before the track section of the outgoing track is free. This is not part of the primary delay and this influence is described separately. Let $\Delta = (\Delta_a, \Delta_h, \Delta_d)$ describe the total primary disturbances. We use $\delta$ for a realization of $\Delta$.

The sets of arrival and departure events are denoted by $E_a$ and $E_d$. For each event $e$, we
denote by $D_{\phi,\delta}(e)$ the realised delay. This depends on the routing scheme $\phi$, the initial delays $\delta$ and the rescheduling strategy.

### 2.4 Complexity issues

The problem of finding a feasible routing plan can be considered an instance of the maximum independent set problem. An independent (or stable) set of a graph $G = (V, E)$ is a subset $U \subseteq V$ of the nodes of the graph, such that $\{v, v\}' \not\subseteq U$ for all edges $\{v, v\}' \in E$. A maximum independent set is an independent set of the largest possible cardinality. The problem of finding a maximum independent set is known to be NP-hard. The decision version of the maximum independent set problem asks for a given $K \in \mathbb{N}$ if there is an independent set of size at least $K$. This decision problem is known to be NP-complete.

![Figure 2.1: Example of conflict graph from Caimi [2009].](image)

The train routing problem can be described as a maximum independent set problem as follows. We define a conflict graph $G = (V, E)$ with all train-pattern pairs $(t, P)$ as nodes: $V = \{(t, P)|t \in T, P \in \mathcal{P}_t\}$. There are edges between all train-pattern pairs that concern the same train. These edges express that one train can only be assigned to one pattern. Moreover, there are edges between all incompatible train-pattern pairs. Then a feasible routing would correspond to an independent subset of $G$ of cardinality $|T|$. This independent set would then contain the train-pattern pairs that are taken. If a feasible routing exists, then $|T|$ is the cardinality of the maximum independent set, because there is an edge between all pairs of patterns of the same train. In figure 2.1, an example of a conflict graph is given for three trains. Each row represents the route alternatives for one train. The cliques for each train are drawn separately for clarity reasons.

The problem of deciding whether there is an independent set of size at least $K$ is NP-complete. This does not immediately imply that the train routing problem is NP-complete as well, because the independent set problem could be more general. One could hope for an efficient (or polynomial) algorithm for a restricted version of the independent set problem for specific instances that correspond to instances of the train...
routing problem. However, in Kroon et al. [1997] it was proved that if we allow three possible routes for each train and arbitrary pairs of incompatible routes for different trains, then the question whether there exists a feasible routing where each train is assigned to a route is still NP-complete. This was proved by a reduction from 3-SAT.

**Restricted version of independent set decision problem:** Let \( K \in \mathbb{N} \) (corresponding to the number of trains) and let \( V_j \) be a disjoint set of nodes with \( |V_j| = 3 \) for every \( j \in \{1, \ldots, K\} \) (corresponding to the possible routes for a fixed train). Let \( V = \bigcup V_j \) and let \( E \) be a set of edges on \( V \), such that \( \{u, v\} \in E \), for \( \{u, v\} \subseteq V_j, u \neq v \) for every \( j \) (every train has at most one route). Decide whether there exists an independent set of size \( K \) in the graph \((V, E)\).

**Lemma 1.** The restricted version of the independent set decision problem is NP-complete.

**Proof.** The problem is certainly in NP. For a subset \( U \subseteq V \) of size \( K \), it can be checked in time \( O(K^2) \) if it is independent, by checking for every pair of elements of \( U \) whether it is in \( E \).

We give a polynomial time reduction from 3-SAT to show that the problem is NP-complete. Let \( \psi = \psi(x_1, \ldots, x_n) = (l_{1,1} \lor l_{1,2} \lor l_{1,3}) \land \ldots \land (l_{K,1} \lor l_{K,2} \lor l_{K,3}) \) be a formula in conjunctive normal form where every clause \( i \) consists of three literals \( l_{i,1}, l_{i,2}, l_{i,3} \), where \( l_{i,j} = x_{f(i,j)} \) or \( l_{i,j} = \neg x_{f(i,j)} \), with \( f(i, j) \in \{1, \ldots, n\} \).

Now we can define a graph, depending on \( \psi \). Define a set of \( 3 \times K \) nodes \( V = \bigcup_{i \in \{1, \ldots, K\}} V_i \), with \( V_i = \{v_{i,1}, v_{i,2}, v_{i,3}\} \), for \( i \in \{1, \ldots, K\} \). Let the set of edges \( E \) be given by the pairs \( \{v_{i,j}, v_{i',j'}\} \), for which \( l_{i,j} = \neg l_{i',j'} \) or \( l_{i,j} = l_{i',j'} \) and all pairs \( \{v_{i,j}, v_{i',j'}\} \). Then it holds that the formula \( \psi \) has a satisfiable assignment if and only if the graph \( G = (V, E) \) has an independent set of size \( K \).

Suppose a satisfiable assignment of \( \psi \) is given. Then for every clause \( i \), there is at least one literal \( l_{i,j} \) that is assigned to the value TRUE. For every set \( V_i \), take one element \( v_{i,j} \), such that \( l_{i,j} \) is assigned to the value TRUE. This gives an independent set \( U \). For an edge \( \{v_{i,j}, v_{i',j'}\} \), with \( l_{i,j} = \neg l_{i',j'} \) or \( l_{i,j} = l_{i',j'} \), we have that \( l_{i,j} \) and \( l_{i',j'} \) cannot both be assigned to the value TRUE, so \( \{v_{i,j}, v_{i',j'}\} \not\subseteq U \). For an edge \( \{v_{i,j}, v_{i,j'}\} \), at most one of the nodes is chosen to be part of \( U \). So \( \{v_{i,j}, v_{i,j'}\} \not\subseteq U \). So \( U \) is indeed an independent set.

Suppose an independent subset \( U \) of \( V \) of size \( K \) is given. Then we know that \( U \) consists of one element from each set \( V_i \). Now define an assignment of \( \psi \) such that \( l_{i,j} \) is assigned to the value TRUE if \( v_{i,j} \in U \). Note that this is possible because if \( l_{i,j} \) is the negation of \( l_{i',j'} \), then there is an edge between \( v_{i,j} \) and \( v_{i',j'} \), so not both \( v_{i,j} \in U \) and \( v_{i',j'} \in U \). By choosing an arbitrary assignment for the other variables, we obtain a satisfying assignment because in each clause, there is at least one literal that is assigned to the value TRUE.

So there is a polynomial time reduction from the NP-complete problem 3-SAT to the restricted version of independent set decision problem, proving that the latter problem
is NP-complete.

By allowing arbitrary combinations of incompatible routes, we still have a quite general problem where many instances do not arise from a natural railway station with a common timetable. However, in Kroon et al. [1997] it was shown that each instance of the problem can be described by a railway problem in the following way:

Define $K$ trains and for each train a different platform where every train has three possible inbound routes leading to the train’s platform, and one possible outbound route. Suppose that all trains have a different entering point and a different leaving point. Suppose that all pairs of inbound routes for different trains have to cross each other. Some by cross-overs and others by fly-overs. A cross-over is present if the pair of routes is connected by an edge. A fly over is present if this is not the case.

This description assumes that the railway station grows linearly with the number of trains. In Herrmann [2006] it was stated that there is a polynomial algorithm if we assume the railway station to be of constant size.

Indeed if we assume that the railway station has constant size, then the number of train patterns is bounded by a number $n$. If we moreover assume that the times are discrete ($m$ moments in time, for example $m = 60$ minutes in an hour), then we have a finite number $nm$ of possible combinations of a route and a time. We assume that it is not possible that two trains take the same route at the same time, so we can now consider the problem as an assignment of the trains to the time, pattern pairs (where we can also assign no train to a time pattern pair). Then the number of possible assignments is bounded by $(|T| + 1)^{nm}$, which is polynomial in the number of trains. The practical value of this observations is limited because the number of possibilities is of course very large. In Herrmann [2006] it is stated that there is an algorithm in time $O(|T|^{I+1} \log |T|)$, where $I$ is the number of relevant track sections. [p. 41 of Herrmann. How?]

2.5 Possible routes

In the description of the train routing problem, we assumed that the possible routes for each train are known. In this section, it is described how the set of possible routes can be found.

A formal description of the railway station is given by a double node vertex graph of the track topology... [Description][find article Montigel, 1992, Representation of track topologies with double vertex graphs]

**Definition 2** (Double Vertex Graph). Let $V$ be a set of nodes, $E \subseteq V \times V$ a set of edges and $\circ : V \to V$ a mapping that pairs the set of nodes: $\circ(v) \neq v$ and $\circ(\circ(v)) = v$. Then $D = (V, E, \circ)$ is a called a double vertex graph. By $v^\circ$ we denote $\circ(v)$. 

10
Figure 2.2: Example of a railway network (top) with its corresponding conventional graph representation (lower left) and double vertex graph (lower right) from Caimi [2009].

[question: how are crossings described by a double vertex graph?]

A route in the double vertex graph visits the pairs of vertices always consecutively. This allows us to forbid trains to make sharp curves. A path from $v_1$ to $v_n$ is now described as follows:

$$(v_1, v_1^o, v_2, v_2^o, v_3, \ldots, v_{n-1}, v_n, v_n^o)$$

Where $(v_1^o, v_2), (v_2^o, v_3), \ldots, (v_{n-1}^o, v_n) \in E$.

In order to determine the set of different routes $R_t$ for a train $t$, we can use different levels of detail of the track sections. In the most detailed case, we take into account every individual switch. This will lead to a lot of different routes for each train, and thus a larger problem to solve. It would be interesting to see which level of detail is needed such that no essential routing possibilities are overlooked, no infeasible solutions will be proposed and the resulting problem is as small as possible.

In Herrmann [2006] an algorithm is given to find all paths. In general there can be an exponential number of paths, because the number of paths can be doubled by each switch (example).

Figure 2.3: ... routes from left to right.

A lot of routes that in theory can be taken, will never be used, because there is another
route that is always better.

**Definition 3 (Dominated route).** A route \( r_1 \in R_t \) is called dominated if there exists a route \( r_2 \in R_t \) with the same platform, entry and exit point and the property that for the set of routes \( r \in R \) that has a conflict with \( r_2 \) is a proper subset of the set of routes that has a conflict with \( r_1 \).

[Question:] if \( r_1 \) is dominated by \( r_2 \) is \( r_1 \) always shorter than \( r_2 \)? What assumptions are needed to prove this?

[conflicting routes described in next section. Is this a problem?]

In order to reduce the number of routes, we can remove routes that are dominated by a route that is not longer. In Zwaneveld et al. [2001] a preprocessing step is included to exclude dominated routes. In Caimi [2009] a method using a conflict matrix is given to reduce the number of routes. The last exploits the characteristics of the railway infrastructure, which gives an advantage for the computation time. However, the approach of Caimi [2009] does not guarantee to maintain feasibility of the problem, because not only dominated routes are removed (see policy 6.4 in Caimi [2009]). [more explanation]

In Kroon and Marótí [2008] aggregated routes are proposed, that are characterized by the platform track and by the destination. Then for each two aggregated routes, we can decide whether or not they can be implemented without having a conflict. A problem here is that it is possible that for three trains every two can be combined, but the three of them can not. So this simplification possibly leads to infeasible solutions. Another approach (taken in Cornelsen and Di Stefano [2007]) to simplify this situation is to fix one detailed route for every combination of platform and destination. Then the problem becomes a track assignment problem. Now we can be sure that we will not find infeasible solutions, but possibly we overlook possible solutions. This method is not suitable for complex stations, where several alternative routes are possible, because then essential possibilities are excluded.

In Borndörfer et al. [2011] aggregation methods are given. However, in this paper a larger area containing several railway stations is considered. Perhaps it is possible to take a similar approach for the area of one railway station. It is not yet clear if this is a useful method in the case of one station.

The set \( R_t \) of possible routes for train \( t \) can thus be determined by all paths in the Double Vertex Graph of the railway station from the entering point to the leaving point of train \( t \), that are not dominated by a shorter route. (Perhaps also exclude dominated routes that have the same length...)

### 2.5.1 Determining possible routes

[Stond eerst verderop]
Figure 2.4: Not all dominated routes are excluded in this case. \( D \) is a relevant point because \( ADG \) and \( BDF \) only share this point. \( BDG \) dominates \( BEG \), but the set of relevant points is not a subset, because \( D \) is in \( BDG \) and not in \( BEG \).

In order to find all possible routes through the stations, it is useful to determine the crucial points in the network. To start with, we let \( A \) be the set of all entrance and exit points of the railway station, platforms and all crossings and switches. If two routes between an exit or entrance point and a platform share part of the network, then they certainly also share one of these crucial points in \( A \). We say that a point \( x \) in the network implies another point \( y \) if \( x \) it is not reachable from any of the entrance points and exit points without passing \( y \) or if \( x \) is not reachable from any of the platforms without passing \( y \). In other words: each route between an entrance or exit point and a platform that contains \( x \) does also contain \( y \). Note that it is possible that \( x \) and \( y \) imply each other. For the remainder of the method, it is useful to reduce the number of crucial points. So we would like to find a minimal set \( B \subseteq A \) such that all elements of \( A \) imply an element of \( B \). We make sure that all entrance, exit point and platforms are still in \( B \). We can further reduce the number of crucial points by merging some points from \( B \). For the three points in the set \( k \), we have that every train crossing one of them, will also cross a second point from \( k \). So two trains using an element from \( k \) will always share at least one of the three points in \( k \). However, none of the points imply each other. We therefore consider \( k \) as one point. For Uitgeest this is done by hand, leading to the points that are labeled by a letter.

For each combination of a platform and one of the three directions we need to determine the possible routes. In the case of Uitgeest, they were determined by hand. In general, we can use the following strategy. From each inbound and outbound point, we make a tree of possible routes from that point to a platform. Then for every route \( r \), we determine the set of relevant points that occurs in the route. If the set of relevant points of route \( r \) is a proper subset of the relevant point of a route \( r' \) between the same points, then we exclude \( r' \) from the set of routes. (Do we exclude all dominated routes in this way? What do we need to know about the relevant points?)

### 2.5.2 Conflicting train pattern pairs

By defining conflicting pairs, we avoid that two punctual trains need to claim the same part of infrastructure at the same time. Until now, we have assumed that we know which train pattern pairs are conflicting. This is however not trivial and it depends
on the assumptions. In order to determine the conflicting routes, we need information about the time it takes to travel the paths and the track sections that are used by the routes. Different levels of detail are possible.

Two routes are conflicting if they share part of the track sections. If we have information about the relevant track sections of a train, then we can check if two routes are conflicting by comparing the lists of relevant track sections. Two train pattern pairs are conflicting if they claim an element of the infrastructure for overlapping time intervals. Therefore we have to define for what period of time the elements of the infrastructure are claimed by a train pattern pair.

In the most simple case, we just assume that all infrastructure elements in the inbound route are blocked for a constant period of time around the arrival time of the train and similarly for the outbound route. In a much more detailed case, we take into account the maximal speed of the train through the different track sections and study conflicts for all different track sections separately. In Zwaneveld [1997] driving time calculations are given.

By safety assumptions, a train can only enter or leave the station if the hole inbound or outbound route is free. This procedure is called the route locking and sectional release system. As soon as a train arrives at a certain point outside the railway station, it then claims the hole route. The different sections are released one by one as the train passes them. Any track section can be claimed by at most one train at the same time. Under these assumptions, it was shown in Zwaneveld et al. [2001] that the only relevant track sections are those that contain a switch, an intersection of tracks or correspond to an entering point, leaving point or platform. This means that if there are no conflicts at these relevant track sections, then there are no conflicts at any of the track sections.

In most cases we will assume that the claim times for all track sections in the route are equal, instead of assuming that they are released one by one. For an arrival we assume that all track sections in the inbound route are claimed from a constant time before the arrival until the arrival time. For a departure we assume that all track sections in the outbound route are claimed from the departure time until a certain number of minutes after the departure time.
Chapter 3

Searching for robust routings

3.1 Routing methods from literature

In this section, several known approaches to create routing plans are described. Moreover, it is discussed how robustness objectives could be imposed for these methods.

3.1.1 Conflict graphs

Most of the known solution approaches for the train routing problem use in some way the concept of a conflict graph. This view was introduced by Zwaneveld and al. in Zwaneveld et al. [1996]. Then the problem boils down to an independent set problem, as was explained in section 2.4. The nodes in the graph were given by the set of all train-pattern pairs \( \{(t, P)|t \in \mathcal{P}_t, t \in T\} \). An edge between two nodes expresses that the two routes are conflicting, so can not both be chosen. The resulting independent set problem is often solved using integer linear programming.

We thus need to describe the problem as a linear program and introduce variables. For a pair \( (t, P) \), let the binary variable \( x_{t,P} = 1 \), if pattern \( P \) is assigned to train \( t \). Now we can impose conditions on combinations of nodes that have a conflict.

- Each train can only be assigned to one pattern: \( \sum_{P \in \mathcal{P}_t} x_{t,P} = 1 \) (or \( \sum_{P \in \mathcal{P}_t} x_{t,P} \leq 1 \) if we do not require all trains to be routed).
- Conflicting patterns are excluded: \( x_{t,P} + x_{t',P'} \leq 1 \) for conflicting pairs of patterns.

A feasible solution would now consist of an assignment of the variables \( x_{t,P} \in \{0, 1\} \) for all \( t \in T, P \in \mathcal{P}_t \) that satisfies the above conditions.
Another possibility is to introduce variables for each combination of a train and an inbound or outbound route: $x_{t,r}$ where $r \in R_a$ or $r \in R_d$ and the direction of route $r$ and train $t$ corresponds. We then also need to impose conditions to guarantee that the inbound and outbound route of a train share a platform. It depends on the station which approach is more efficient. In the case where there is only one route between a combination of entering point and platform, the number of patterns for each train is at most as big as the number of platforms. Each of these patterns consists of two routes and therefore the total number of routes is twice as big. In the situations where there are several routes from an entry or exit point, it can be better to split the patterns in an inbound and outbound part. If there are $k$ routes between every entry or exit point and platform track, then the total number of patterns for each train is $k^2 \times \#\text{platforms}$, where the number of single routes is $2 \times k \times \#\text{platforms}$. So as soon as $k$ is larger than 2, splitting the patterns leads to less variables.

Besides these conditions, the quality of the schedule should be as high as possible. Therefore we can define an objective function that should be minimised. An objective function can include the preference for individual combinations of trains and routes:

$$\sum_{t \in T, P \in \mathcal{P}_t} p_{t,P} x_{t,P}$$  \hspace{1cm} (3.1)$$

where $p_{t,P}$ is the penalty for pattern $P \in \mathcal{P}_t$ for train $t$. The objective function can also include penalties for certain combinations of different routes:

$$\sum_{t \in T, P \in \mathcal{P}_t, t' \in T, P' \in \mathcal{P}_{t'}} q_{t,P,t',P'} x_{t,P} x_{t',P'}$$  \hspace{1cm} (3.2)$$

where $q_{t,P,t',P'}$ is the penalty for the combined assignment of pattern $P \in \mathcal{P}_t$ to train $t$ and $P' \in \mathcal{P}_{t'}$ to train $t'$. The penalty $q_{t,P,t',P'}$ only needs to be defined for compatible pairs $(t, P)$ and $(t', P')$. In this last function we can impose robustness measures. This quadratic penalty was proposed by Caprara and al. in Caprara et al. [2010] and Caprara et al. [2011] and was also used in Kroon and Maróti [2008] and Dewilde et al. [2014].

**ILP**

[Dit stuk over ILP en SDP zou misschien beter ergens anders kunnen staan...]

Let $G = (V, E)$ be the conflict graph corresponding to the problem and for $t \in T$, let $V_t$ be the set of nodes corresponding to one train. Objective function 3.1 induces an integer linear program:

$$\min_{x_i \in \{0,1\}} \left\{ \sum_{i \in V} p_i x_i : x_i + x_j \leq 1 \ (\ (i, j) \in E), \sum_{i \in V_t} x_i = 1 \ (t \in T) \right\}$$  \hspace{1cm} (3.3)$$
The appendix explains how ILPs are solved using CPLEX with a branch and cut method. In this solution approach, bounds for the solution are calculated by relaxing the integer constraints. It is therefore useful that the solution space for non integer solutions is as small as possible. This can be reached by strengthening the conditions by grouping the conflicting routes in maximal cliques $C$, to obtain constraints $\sum_{i \in C} x_i \leq 1$ as in the following program.

$$\min_{x_i \in \{0,1\}} \{ \sum_{i \in V} p_i x_i : \sum_{i \in C} x_i \leq 1 (C \text{ maximal clique in } G), \sum_{i \in V_i} x_i = 1 (t \in T) \} \quad (3.4)$$

By relaxing the integer constraints, we obtain the following linear program:

$$\min_{x_i \in \mathbb{R}} \{ \sum_{i \in V} p_i x_i : \sum_{i \in C} x_i \leq 1 (C \text{ clique in } G), \sum_{i \in V_i} x_i = 1 (t \in T), x_i \geq 0 (i \in V) \} \quad (3.5)$$

Grouping several pairs into one clique makes the solution space of non integer solutions smaller. In general, this makes it easier to find the optimal integral solution. However, the problem of finding a maximal clique is NP-complete and the number of cliques can be very big. Therefore it can be difficult to find all maximal cliques from the pairs of conflicting nodes. However, many cliques can be found already in an earlier stage instead of deducing them from the conflicting pairs. The objectives concerning combinations of routes are in quadratic form and should be linearised. Strategies are discussed in Kroon and Maróti [2008] and Caprara et al. [2011].

SDP

Instead of the usual linear programming relaxation, the problem can also be formulated as a semidefinite program. Suppose $x_1, \ldots, x_n$ are the 0-1 decision variables that determine the assignment of routes to the trains. Suppose the set $E \subseteq [n]^2$ gives all pairs of events that are not allowed. Let $y = (1, x_1, \ldots, x_n)^T$. Define the $(n+1) \times (n+1)$ matrix

$$Y = yy^T = \begin{pmatrix} 1 & x_1 & \cdots & x_n \\ x_1 & x_1^2 & \cdots & x_1 x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_n & x_n x_1 & \cdots & x_n^2 \end{pmatrix}$$

Then $Y$ is a positive semidefinite matrix because it is given by its Gram representation (or Cholesky decomposition). Moreover we know that $Y_{00} = 1$, $Y_{ii} = Y_{ii}$ (because the variables are 0-1 valued) and $Y_{ij} = 0$ for all $(i,j) \in E$. These conditions can all be expressed by the trace inner product. If we now relax the integer constraints, we obtain the semidefinite program:

$$\max_{Y \in S^{n+1}} \{ \sum_{i \in V} Y_{ii} : Y \succeq 0, Y_{00} = 1, Y_{ii} = Y_{ii}(i \in V), Y_{ij} = 0(\{i, j\} \in E) \} \quad (3.6)$$

17
The value of equation 3.6 is known as the theta number $\theta(G)$ Lovász [1979] of the graph $G = ([n], E)$. It is an upper bound for the size of the maximal independent set. In our case we insist that all trains are assigned to a route, so we have the additional constraints $\sum_{i \in V_t} Y_{ii} = 1$ for all $t \in T$, where $V_t$ is the set of nodes corresponding to train $t$. The following program corresponds to objective function 3.1.

$$\min_{Y \in S^{n+1}} \left\{ \sum_{i \in V} p_i Y_{ii} : Y \succeq 0, Y_{00} = 1, Y_{0i} = Y_{ii} (i \in V), Y_{ij} = 0 (\{i, j\} \in E), \sum_{i \in V_t} Y_{ii} = 1 (t \in T) \right\}$$

(3.7)

The objective function is a linear combination of the diagonal elements in the matrix. A solution to this program can be approximated in polynomial time and it turns out that this relaxation gives a bound that is at least as strong as the LP relaxation with all clique constraints.

**Lemma 4.** Let $(m_1, \ldots, m_n) = (Y_{01}, \ldots, Y_{0n}) \in \mathbb{R}^n$ be part of a solution $Y$ to 3.7. Then $\sum_{i \in C} m_i \leq 1$, for all cliques $C$ in $G$.

**Proof.** Suppose $C = \{i_1, \ldots, i_k\}$ forms a clique in the conflict graph. That is $C^2 \subseteq E$. Let $M$ be a positive semidefinite matrix that suffices the conditions from 3.7. If we consider the principal submatrix of $M$ indexed by the rows and columns $1, i_1, \ldots, i_k$, we obtain a positive semidefinite matrix $M'$.

$$M' = \begin{pmatrix}
1 & m_{i_1} & \cdots & m_{i_k} \\
m_{i_1} & m_{i_1} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
m_{i_k} & 0 & \cdots & m_{i_k}
\end{pmatrix}$$

Let $v = (-1, 1, 1, \ldots, 1)^T \in \mathbb{R}^{k+1}$. Then $v^T M' v = 1 - \sum_{j \in [k]} m_{ij} \geq 0$. The last inequality follows from the fact that $M'$ is positive semidefinite. So we can conclude $\sum_{i \in C} m_i \leq 1$. So all clique constraints are imposed. 

**Corollary 5.** The sdp formulation 3.7 gives a bound that is at least as strong as the lp formulation 3.5, with all maximal clique constraints.

[so far only the linear case is introduced. Explain how penalties for pairs can be handled, objective function 3.2]

**Robustness**

By Dewilde et al. [2014], Caprara et al. [2010] and Kroon and Maróti [2008], robustness objectives are imposed using the penalties $q(t, P)[(v, P)]$ from equation 3.2. In Kroon and Maróti [2008] and Caprara et al. [2010], the value of the penalties is not specified. In Dewilde et al. [2014], these are chosen to be $1/B$, where $B$ is the buffer time between two events that use common track sections. If $B \geq B^{\text{max}}$, no penalty is imposed. A
constant penalty is chosen for $B = 0$. In Schlechte and Borndörfer [2010], the problem is formulated as a maximization problem. There the objective value for a pair with buffer time $B$ is chosen to be $\sqrt{B}$, with a certain maximum $\sqrt{B}^{\text{max}}$.

Besides the buffer time between two trains using the same track section, we could take into account a certain importance factor for the punctuality of the train, for example because it has a lot of connections to other trains. It is interesting how these penalties should be chosen such that the resulting schedule is as robust as possible.

Another way to impose robustness, that was not found in one of the papers, is to replace each train pattern pair by a set of possibilities that all refer to the same route, but where the occupation time of the different resources varies within this set. For example, we can introduce two different nodes for a pattern $P$: $\{(t, P_0), (t, P_1)\}$, where $P_1$ claims the resources for a longer period of time, than strictly needed. Then the set of nodes that conflicts with $(t, P_0)$ is a subset of the nodes that conflict with $(t, P_1)$. By the value of the penalties $p_{t,P}$ of the individual routes, we can then express our preference for the route that claims the infrastructure for a longer period of time and thus, in practice will have less conflicts in case of a small deviation from the schedule. The set of alternatives could contain more than two possibilities, but then the problem becomes larger and more difficult to solve.

A disadvantage, compared to the quadratic penalty function, is that the more robust choice can only be chosen if the pattern does not conflict with all other trains, where in practice it is already an advantage if there is more time between two conflicting routes. It is for example possible that by changing the platform of a certain train, we prevent a small buffer time during its arrival, but that we can not prevent a small buffer time during the departure. Then the more robust alternative is not recognized because not all small buffer times are prevented. It would then be better to split the pattern into an inbound part and an outbound part as was suggested in section 3.1.1. By the quadratic objective function, we can express the influence of all conflicting pairs.

### Time deviations

Small deviations in arrival of departure time can be allowed by copying the complete node set for different times (for example one minute earlier and one minute later) and also include constraints between the nodes in different time shifts. In Fuchsberger [2007] this is described as pulsed train scheduling. In Dewilde et al. [2014], time deviations were considered using a tabu search algorithm.

#### 3.1.2 Graph colouring

In Cornelsen and Di Stefano [2007] the track assignment problem is stated as a node colouring problem. Here, it is assumed that for each combination of an entering/leaving
point and a platform, there is only one route. Then we only need one node for one train and by colouring the node, we can assign a certain platform to the train. Nodes are connected if their corresponding trains have overlapping halting periods. The problem is now to find a colouring of the nodes, such that connected nodes have different colours. We can restrict the possible platforms for a certain train and we can impose additional requirements for combinations of events (for example because two assignments of trains to a platform result in conflicting routes).

### 3.1.3 Multicommodity flow

For the conflict graph approach, introduced in section 3.1.1, we can either consider complete patterns with inbound route and outbound route or we can consider these routes separately. This leads to variables $x_{t,r}$ for each in- or outbound route $r$ or variables $x_{t,P}$ for each pattern $P \in \mathcal{P}$. For the approach proposed in Caimi [2009] and Fuchsberger [2007], even smaller parts of the route are considered. The routes are split into track sections and variables are introduced for each combination of a train and a relevant track section. This method emphasizes the layout of the tracks. Resource trees are rooted trees that represent the alternative routing possibilities for one train. An example from Caimi [2009] is given in figure 3.1.

![Figure 3.1: Resource trees from Caimi [2009] [make own pictures]](image)
The root node of the tree represents the entrance point of the train \( t \) and for each node in the tree the children represent the possible sequels of the route. Multiple different nodes in the tree can represent the same element of the infrastructure with a different route leading to this point. For example \( u_5 \) and \( u_{15} \) represent the same node in the track section. Let \( E_t \) be the set of edges in the resource tree for train \( t \). Let \( S_t \) be the set of edges from the source node. Let \( L_t \subseteq E_t \) be the set of edges leading to a leave at the bottom of the tree. Each leave of the resource tree uniquely corresponds to a possible route, so \( |L_t| \) is equal to the number of possible routes for train \( t \).

Now the route of a train can be defined by an integer flow of value 1, from the root node to one of the leaves, where all edges have capacity 1. For each edge \( e \in E_t \) in the tree, a variable \( x_e \) is introduced that takes value in \( \{0, 1\} \) and defines the flow through this edge. Each route thus induces an assignment of these variables. We have \( x_e = 1 \) if and only if edge \( e \) belongs to the route that is taken. Now the following conditions are imposed on the trees:

\[
\sum_{e \in S_t} x_e \leq 1: \text{ each train starts with at most one of the routes.} \tag{3.8}
\]

\[
\sum_{e \in L_t} x_e \leq 1: \text{ each train ends in at most one of the leaves.} \tag{3.9}
\]

\[
x_{(u,v)} = \sum_{(v,w) \in E} x_{v,w} \text{ for all } (u,v) \in E_t \setminus L_t: \text{ flow conservation in node } v. \tag{3.10}
\]

The flow conservation constraint can be compared to the constraint that the inbound route should lead to the same platform as the outbound route if we have variables \( x_{t,r} \) for \( r \) an in- or outbound route.

In order to impose conditions for conflicting routes, several trees for different trains are combined. For each relevant track section (represented by nodes in the tree), conditions are imposed on the simultaneous occupation of this section by two trains. The occupation intervals of the different track sections should therefore be known. Instead of a constraint for every pairs of conflicting routes (3.10), (maximal) cliques can be considered. In Caimi [2009] an algorithm is suggested to create conflict cliques for each track section. This results in a condition for conflict cliques \( C \):

\[
\sum_{v \in C, (w,v) \in E} x_{(w,v)} \leq 1: \text{ for all conflict cliques } C \text{ on a relevant track section.} \tag{3.11}
\]

**Relation to conflict graph formulation**

[consequent gebruiken: node packing/independent set/stable set/conflict graph]

The leaves of the trees uniquely correspond to a route for the train. Therefore the
variables \( x_e \) for \( e \in L_t \) correspond to the values \( x_{t,r} \) in the node packing problem, where \( r \) is the route corresponding to edge \( e \). For a node \( v \in V_t \), suppose \( L_t(v) \subseteq L_t \) is the set of edges, that correspond to a route containing node \( v \).

\[
L_t(v) = \{ e' \in L_t \mid \text{the path in the resource tree from the source to } e \text{ contains node } v \}
\]

Because of flow conservation, we know that node \( v \) is used if and only if there is an edge \( e \in L_t(v) \) that is used. So instead of having conditions on all edges, we can impose all conditions on the edges in \( L_t \). We formulate equivalent conditions as follows:

\[
\sum_{e \in L_t} x_e \leq 1: \text{each train has at most one leave (and thus a route)} \tag{3.12}
\]

\[
\sum_{v \in C, e' \in L_t(v)} x_{e'} \leq 1: \text{for all conflict cliques } C \text{ on a relevant track section } s. \tag{3.13}
\]

This formulation now corresponds directly to the node packing formulation. So the resource tree formulation of the problem can be translated into the conflict graph formulation. Objective functions can also be imposed on the leaves, in the same way as in the node packing problem. An advantage of the approach with the resource tree, is that conflict cliques can be found relatively easily as described in Caimi [2009]. However, the cliques defined in this way are not necessarily maximal in our new description, as we can see from the following example.

**Example 6.** Suppose we consider three train routes: the red, blue, and green route in figure 3.2.

![Figure 3.2](image)

Figure 3.2: Three conflicting routes that do not share a common point.

Now every pair of two routes is conflicting, but there is not one point where all pairs of conflicts take place. This would not result in a clique in the resource tree formulation.

The consequence is that maximal cliques in the original multicommodity flow problem do not give us all maximal cliques in the equivalent node packing formulation. The three routes form a conflict clique, but from the cliques of edges in the resource trees, we only identify pairs of conflicting routes.
Time deviations

In the resource tree approach, it is also possible to allow a discrete deviation from the planned arrival and departure times for a train \( t \). In order to allow this, the resource tree for a train \( t \) is copied for different time deviations. These time deviations can be seen as different routing options and can all be connected to one source node for \( t \). In the same way as in the original approach, conflict cliques can be defined and similar conditions can be imposed.

Robustness

Robustness can be imposed in a way similar to the quadratic method for conflict graphs. The edges \( e \in L_t \) uniquely determine the route that is chosen. So for every pair of edges connected to a leave, a penalty factor \( q_{e,e'} \) can be defined for the simultaneous use of the edges \( e \in L_t, e' \in L_{t'} \). This would lead to the following quadratic objective function.

\[
\sum_{t \in T, e \in L_t} \sum_{t' \in T, e' \in L_{t'}} q_{e,e'} x_{e} x_{e'}
\]

The penalty can be determined by first defining penalties \( q_{e,e'} \) for all pairs of edges \( e \in E_t \) and \( e' \in E_{t'} \) and then defining the penalty on \( e_0 \in L_t \) and \( e_1 \in L_{t'} \) as:

\[
q'_{e_0,e_1} = \max\{q_{e,e'} \mid e_0 \in L_t(e), e_1 \in L_{t'}(e')\}
\]

Resource graph instead of tree

Instead of using a tree, where a track section can occur several times, we could also the double vertex graph of the relevant part of the station, where each track section occurs only once. Then again a flow of value 1 from the entrance to the exit point of the station defines a route through the station. A disadvantage would be that it is not possible to take into account the exact time that a train arrives at a certain resource, because this depends on the route. However, in a lot of situations, this would not change the solution space too much.

Another disadvantage is that it is more difficult to define a good objective function in this case. In the case of resource trees, the penalty factors can be associated with the edges in \( L_t \) and thereby with a variable that determines the complete route. If penalties are defined for all different track sections of the route, it is possible that the same conflict gets a penalty multiple times. It is for example possible that two train routes \( (t,r) \) and \( (t',r') \) both use the consecutive edges \( e \) and \( e' \). Then a penalty would be given for both conflicts, where the probability and impact of the conflict would be nearly the same if they only share edge \( e \). This problem could be solved by introducing variables for a
complete route, but then the advantage of the reduction of the number of nodes would be lost.

### 3.1.4 Set packing for time interval track sections

In the set packing approach from Lusby et al. [2013], a set of space-time resources is defined. The time is discrete: it is divided in a finite set $I$ of intervals (for example intervals of 15 seconds). Moreover there is a finite set $S$ of track sections. Now let $P = I \times S$ be the set of time interval track sections (tints). Each possible route $r \in R_t$, for a certain train $t$ uses a certain subset $P_r \subseteq P$ giving both the time and space resources.

A feasible solution to the routing problem is now defined as an assignment of a route to each train, such that each tints in $R$ is used at most once. As an objective function each route can also have a certain preference or penalty.

Note that it is possible that a solution without conflicts is not feasible in this formulation if two trains both claim a certain tints, but only use it for a disjunct interval of the time.

**Robustness**

Also in this situation, a penalty can be defined for combinations of routes using a quadratic objective function. The method with alternative routes that claim the infrastructure for a longer period of time is also applicable here. If this more robust route obtains a lower penalty than the route that claims the infrastructure for a shorter time, there will be a preference for the route that claims more tints.

### 3.1.5 Local search

In chapter 5 of Herrmann [2006] a local search method is proposed using different objective functions. An advantage of local search is that more advanced objective functions can be studied because we only need to be able to evaluate the objective function in reasonable time for different solutions. A disadvantage is that no guarantee can be given that assures that the resulting solution has a certain quality compared to the optimal solution. In chapter 3.5 of Herrmann [2006], a fixed point iteration method is used to find several feasible solutions. Then a random-restart local search method tries to improve upon these initial solutions. Neighbourhoods are defined for every neighbourhood-size $k$, where at most $k$ routes of $k$ different trains are replaced by another route. Four different objective functions are described that can be evaluated for a given solution. Moreover, for each solution, critical nodes can be determined that contribute most to the objective value. Depending on the value of $k$, not all solutions from the neighbourhood are studied. For larger $k$, only a part of it is explored. It is reasonable that the routes of
the \( k \) most critical trains are reconsidered. Depending on the objective function, new routes are chosen for the \( k \) most critical trains. From example 7, we can conclude that for \( k = 4 \), there are local optima that are not globally optimal.

**Example 7** (A 4-Optimal Solution that is not an optimal solution). Suppose there are trains at 0, 8, 15, 23, 30, 38, 45, 53 and we have to divide them into two groups such that within the group trains can conflict each other, but between the groups there is no interaction. A solution with minimal time slots would be one group of 0, 15, 30, 45 and one group of 8, 23, 38, 53 such that the time between two trains is always 15 minutes. However, suppose we start with a solution 0, 15, 23, 38 and 8, 30, 45, 53. Then the 4 most critical trains according to the simple time slot definition are 15, 23, 45, 53. However, there is no assignment where we only change these 4 trains that has a better objective function with respect to simple time slots. We can conclude that this is a 4-Optimal Solution. In order to reach a better solution we should change the route of two trains with the largest time slots: 30 and 37 and two of the critical ones.

### 3.2 Routing model: example case

The last section described several approaches to obtain a routing plan. Some possibilities to impose robustness objectives were discussed. Based on the ideas from literature, a new routing model has been developed. The basic model was extended in different ways to impose robustness objectives. The remaining part of this chapter describes the model and its extensions in detail.

The railway station Uitgeest serves as an example case. This station was chosen because it is relatively small but still interesting. A small stations was preferred, to avoid too many implementation problems. Uitgeest is relatively interesting because there are five different train series passing by or stopping at Uitgeest to and from three different directions of travel.

![Figure 3.3: Double vertex graph of railway station Uitgeest and its relevant track sections.](image)

The station consists of five platforms (1a, 1b, 3, 4a, 4b) and two additional tracks without
a platform (2, 5). Each hour, 18 trains have to pass through Uitgeest. For some com-
binations of a platform and a direction different inbound or outbound routes are possible.
Planned times for arrival and departure of all trains are given. The routing problem is
to assign an inbound and outbound route to each train such that the resulting schedule
is as robust as possible.

3.3 Basic routing model

The conflict graph approach from section 3.1.1 serves as a starting point for the basic
routing model. In this model, we have chosen to introduce binary variables $x_{t,r}$ for all
trains $t \in T$ and all arrival or departure routes $r \in R_a$ or $r \in R_d$. In contrast to the
conflict graph approach from Section 3.1.1, the complete patterns $P$ are split up into an
inbound route and an outbound route. This results in a smaller number of variables,
especially if the number of routes from or to every platform is large. An assignment of
routes to the different trains should suffice the following conditions:

1. Every train is assigned to one inbound and one outbound route.
2. The entrance point of the inbound route of each train should correspond to the
   inbound point of the train and the exit point of the outbound route of each train
   should correspond to the outbound point of the train.
3. The platform of the inbound route and outbound route should be equal.
4. Trains that stop should be assigned to routes that correspond to a platform track
   next to a platform (in the case of Uitgeest: not 2 or 5).
5. Each platform track can be claimed by at most one halting train at the same time.
6. If the platform track is claimed by a stopping train, it can not be used in the route
   of a driving train.
7. Each (relevant) track section can be claimed by at most one train at a time that
   is driving.

Conditions 4, 5 and 6 together express that each track section can be claimed by at
most one train at the same time. This constraint is split up into three cases because the
platform claim period depends on both the arrival and the departure time, whereas the
claim time of the in- and outbound route only depends on one of these times.

In order to solve this problem, it was described as an Integer Linear Program. The
conditions can be translated into the ILP constraints $A_1, A_2, A_3, A_4, A_5, A_6$ and $A_7$
on the variables $x_{t,r}$.

In constraints $A_5, A_6$ and $A_7$ certain pairs of trains and routes were selected based on
overlapping claim times or track sections. The occurrence of overlapping claim times
is determined beforehand for all pairs of trains. The claim times are determined based
One in- and one outbound route for each train

\[
\begin{align*}
\text{forall } t \in T : \\
\sum_{r \in R_a} x_{t,r} &= 1; \\
\sum_{r \in R_d} x_{t,r} &= 1;
\end{align*}
\]

**Constraint A1:** For each train, the sum over all variables indicating an arrival route is equal to 1. This corresponds to the fact that each train can only be assigned to one arrival route. Similar for the departure route.

**Direction of train and route corresponds**

\[
\begin{align*}
\text{forall } t \in T : \\
\text{forall } r \in R_a : D(t)_a &\neq D(r) : \\
&x_{t,r} = 0; \\
\text{forall } r \in R_d : D(t)_d &\neq D(r) : \\
&x_{t,r} = 0;
\end{align*}
\]

**Constraint A2:** This constraint limits the set of routes that is suitable for a specific train because of the direction of travel.

**Arrival and departure same platform**

\[
\begin{align*}
\text{forall } t \in T : \\
\text{forall } r_a \in R_a : \\
\text{sum } r_d \in R_d : \text{station track of } r_d \text{ and } r_a \text{ is different} \\
&x_{t,r_d} \\
+x_{t,r_a} \\
\leq 1;
\end{align*}
\]

**Constraint A3:** The arrival and departure routes should correspond: they should use the same platform.

on the arrival or departure time. The claim intervals are given by a start and an end, both natural numbers smaller than 60. If the claim period contains the transition to the next hour, the minute corresponding to the end of the claim period is smaller than the minute corresponding to the begin of the claim period.

Suppose two pairs of begin and end of the claim times are given: \((\sigma_1, \sigma_2)\) and \((\tau_1, \tau_2)\), where \(\sigma_1, \sigma_2, \tau_1, \tau_2 \in \{0, \ldots, 59\}\) and \(\sigma_1 \neq \sigma_2, \tau_1 \neq \tau_2\). We assume that the claim periods are smaller than one hour. Otherwise there would be a conflict with the same train in the next hour. The periods (as open intervals) have overlap in a cyclic schedule if the
Platform for stopping train

\textbf{Constraint A4:} A halting train should not be assigned to a track that by-passes all platforms.

Platforms claimed by at most one stopping train

\textbf{Constraint A5:} redundant (follows from A6 and A7). This constraint can still be useful because it gives additional constraint cliques.

Platforms not claimed by both stopping and driving train

\textbf{Constraint A6:} Some platform tracks are used in the route of a train that do not stop at this platform. It should be avoided that a platform track is claimed by a halting and a driving train at the same time.

formula from (3.14) holds:

\[
\begin{align*}
& (\sigma_1 \leq \tau_1 \land \tau_1 < \sigma_2) \\
& \lor (\tau_1 \leq \sigma_1 \land \sigma_1 < \tau_2) \\
& \lor (\sigma_1 < \tau_2 \land \tau_2 \leq \sigma_2) \\
& \lor (\tau_1 < \sigma_2 \land \sigma_2 \leq \tau_2) \\
& \lor (\sigma_2 < \sigma_1 \land (\tau_1 < \sigma_2 \lor \tau_2 > \sigma_1)) \\
& \lor (\tau_2 < \tau_1 \land (\sigma_1 < \tau_2 \lor \sigma_2 > \tau_1)) \\
& \lor (\tau_1 > \tau_2 \land \sigma_1 > \sigma_2)
\end{align*}
\]
Overlapping routes not used at the same time

forall \{t_1, t_2\} \subseteq T: t_1 \neq t_2, v_1, v_2 \in \{a, d\} (arrival or departure):

\( t_1 \) for \( v_1 \) and \( t_2 \) for \( v_2 \) have overlapping route claim time.

forall \( r_1 \in R_{v_1} \):

\[ \sum_{r_2 \in R_{v_2}: r_2 \text{ has track section overlap with } r_1} x_{t_2,r_2} + x_{t_1,r_1} \leq 1; \]

Constraint A7: Routes that share track sections should not be claimed by two trains at the same time.

Same trains take same routes

forall \{t_1, t_2\} \subseteq T: t_1 \neq t_2 \text{ same train series}:

forall \( r \in R \):

\[ x_{t_1,r} = x_{t_2,r}; \]

Constraint A8: Optional constraint to make sure that a train series always departs from the same platform and takes the same routes.

[explanation...]

For constraint A7, overlap of track sections between routes is determined by comparing the relevant track sections. The list of relevant track sections is also used to determine if a platform track of a stop is part of a route (constraint A6).

Note that the constraints can all be expressed using independent set constraints on a graph. Some of the constraints limit the number of nodes, by setting \( x_{t,r} \) to 0 directly. Other constraints are defined by larger sums, but could also be described by conflicting pairs. By summation over larger sets of variables, we obtain larger cliques and thereby a stronger relaxation (see appendix). The problem of finding maximal cliques in a graph in general is NP-complete. Moreover, the number of maximal cliques can be very large, so adding constraints for all maximal cliques leads to a very large number of constraints. There should thus be made a consideration between the utility and effort of finding maximal cliques. In this model, the constraints are formulated to directly describe large cliques that naturally arise, without actively searching for larger cliques.

The objective function (see objective function 1) in the basic routing model is based on the penalties for individual combinations of trains and routes. This penalty can be based on the length of the route.

Objective function

\[ \text{minimise} \quad \sum_{t \in T} \sum_{r \in R} P_{t,r} x_{t,r}; \]

Objective function 1: Linear
3.4 Routing with maximal buffer times

Constraints $A1$ to $A7$ guarantee that if all trains drive on time, then no train will have to wait before entering or leaving the station. However, it is inevitable that trains will enter or leave the station with a certain delay. If the routes of two trains share a track section and are used in a short period of time, this can affect the robustness of the schedule. Delay of the first train can cause a delay of the second train as well. Critically planned use of the infrastructure is therefore not preferred. In this extension of the basic routing model, an objective function is defined that penalties small buffer times.

For each pair of routes that share a track section, we penalise the use of both of them in a short period of time. This is done using a penalty factor $q_{t,r}(t',r')$ for each relevant combination of two train-route pairs, similarly to equation (3.2).

The goal is now to find a routing where the sum of all penalties for pairs of events is as small as possible. It turns out that this can be solved by adding some variables and constraints to the simple routing model and changing the objective function.

In order to obtain the total penalty, we need to sum over the penalty factors $q_{t,r}(t',r')$ for which both $t$ takes route $r$ and $t'$ uses route $r'$. This leads to the following sum:

$$
\sum_{t \in T, r \in R, t' \in T, r' \in R} q_{t,r}(t',r')x_{t,r}x_{t',r'}
$$

The product $x_{t,r}x_{t',r'}$ equals 1 if and only if both factors equal 1. Because we use linear optimization methods, we need to linearise the product $q_{t_1,r_1}(t_2,r_2)x_{t_1,r_1}x_{t_2,r_2}$. Therefore binary variables $z_{(t_1,r_1),(t_2,r_2)}$ are introduced for all relevant combinations $(t_1, r_1), (t_2, r_2)$ and constraint $A9$ ensures that they equal $x_{t_1,r_1}x_{t_2,r_2}$.

**Variables** $z_{(t_1,r_1),(t_2,r_2)}$ represent product $x_{t_1,r_1}x_{t_2,r_2}$

**forall** Relevant combinations $(t_1, r_1), (t_2, r_2)$:

$$
x_{t_1,r_1} + x_{t_2,r_2} - z_{(t_1,r_1),(t_2,r_2)} \leq 1;
\quad z_{(t_1,r_1),(t_2,r_2)} \geq 0;
$$

**Constraint A9:** This constraint ensures that if both $x_{t_1,r_1} = 1$ and $x_{t_2,r_2} = 1$, then $z_{(t_1,r_1),(t_2,r_2)} = 1$. However, it does not ensure that $z_{(t_1,r_1),(t_2,r_2)} = 0$ if $x_{t_1,r_1}x_{t_2,r_2} = 0$. This does not influence the value of the objective function, because this is a minimisation.

Objective function 2 takes into account the factors $q_{(t_1,r_1),(t_2,r_2)}$, as an extension of objective function 1.

Using CPLEX, this system could be built and solved to optimality for Uitgeest within 6 seconds.
3.4.1 Choosing penalty function

The penalty factors $q(t_1, r_1)(t_2, r_2)$ are defined as follows. First of all, a constant is fixed that expresses the number of minutes buffer time for which no penalty is needed. For example this constant is set to be $M = 7$ (minutes). A combination $(t_1, r_1), (t_2, r_2)$ is called a relevant combination if $r_1$ and $r_2$ have common track sections and there is less than $M$ minutes of buffer time between the planned claim times (where $(t_1, r_1)$ takes place before $(t_2, r_2)$). The buffer time in a cyclic, feasible plan is calculated as:

\[(\text{begin of claim time } (t_2, r_2) - \text{end of claim time } (t_1, r_1)) \mod 60 \in [0, 60).\]

If this buffer time is smaller than $M$, then the corresponding penalty is $(M - \text{buffer time})^2$.

Figure 3.4: Example of buffer time between two trains using the same platform $p$. In blue the claim times of the platform during the entry and exit. In grey the dwell time.

The penalty function should reflect the expected performance of the plan when it is used in real time. The penalty for a combination of two train-route pairs should be chosen in such a way that it represents the probability and impact of the possible problems this combination can cause in real time. We say that a problem occurs in real time if two trains intend to claim the same track section at the same moment or if they intend to use a track section in the reversed order. In both cases an operational decision has to be made to decide which train has priority, which train has to wait and eventually alternative routes can be assigned.

An important aspect that should be taken into account is the dependency between certain events. One should be aware of the fact that some problems that most of the time occur together and have the same impact, both obtain a penalty. It should be avoided that certain possible problems obtain unfairly a double penalty.

The probability of the occurrence of problems in real time depends on the distribution of the relevant delays and on the buffer time. Suppose $e_1$ and $e_2$ are two events (arrival or departure) that have common track sections and suppose $e_1$ is planned before $e_2$. A problem occurs as soon as the end of the claim time of $e_1$ is later than the beginning of
the claim time of $e_2$. The probability that a conflict occurs for a certain track section is thus the probability that the end of the claim time of $e_1$ is later than the beginning of the claim time of $e_2$. This depends strongly on the buffer time between the events.

The impact of a problem depends on the claim times of the shared infrastructure and on the rescheduling strategy. It can be expressed by the resulting additional delay. This additional delay can become larger if the required claim times are long. Usually, the claim time of a platform where the train halts is larger than the claim times of other track sections. Therefore, a conflict at a platform can have more impact than a conflict at another track section. The platform is part of the inbound and part of the outbound route. If a conflict at a platform occurs, then this counts as a conflict in the both the arrival and the departure event, so this could be counted multiple times.

If the dwell time of the first train is large, then the probability of a large departure delay is small, so a conflict in outbound route or platform with a train that is planned later, does not have a large probability. If the dwell time of the second train is big, then a delay in arrival will in most cases not lead to a delay in departure, so a delay has less impact. However, if the first train is late, then changing the order of the trains can have big impact, because the first train can not enter.

Conclusion?

3.5 Routing with small differences in time

It is possible that a small difference in planned arrival or departure time leads to a more robust solution within the station. This chosen difference will be called a shift. An extension of the model was made, where a small set of shifts is allowed for each arrival and departure. Not all deviations are supported by the larger model, where we take into account the whole network. We can thus see the proposed time deviations as a feedback from the routing model to the larger planning model, which should always be tested on feasibility in the larger network. New variables were introduced: for each combination of train $t$, route $r$ and possible shift $s$, the binary variable $x_{t,r,s}$ expresses whether this combination is used. By this, the model became quite large. By allowing three possible shifts (-1, 0, 1 minute) for each event, it was not possible to solve it without additional conditions. With the condition that each occurrence of the same train should have the same route and shift (B8), the model became solvable. It can be useful to deduce from the larger network model which shifts are possible for each train. Adding these conditions would reduce the number of possibilities and therefore favour the solvability.

The buffer time between two events in this case also depends on the shift. Only for relevant combinations $(t_1, r_1, s_1), (t_2, r_2, s_2)$ a variable $z_{(t_1, r_1, s_1)(t_2, r_2, s_2)}$ is introduced. Otherwise the number of variables would be too large.

The conditions from the original model were adapted by adding the shift to the entrance
and exit times.

[verdeling van constraints over pagina’s gaat niet helemaal goed]

\begin{align*}
\text{ONE IN- AND ONE OUTBOUND ROUTE FOR EACH TRAIN} \\
\text{forall } t \in T : \\
& \quad \sum_{r \in R_a} \sum_{s \in S} x_{t,r,s} = 1; \\
& \quad \sum_{r \in R_d} \sum_{s \in S} x_{t,r,s} = 1; \\
\end{align*}

Constraint B1

\begin{align*}
\text{DIRECTION OF TRAIN AND ROUTE CORRESPONDS} \\
\text{forall } t \in T, v \in \{a, d\} \text{ (arrival or departure)} : \\
& \quad \text{forall } r \in R_v : D(t)_v \neq D(r) : \\
& \quad \quad \text{forall } s \in S : \\
& \quad \quad \quad x_{t,r,s} = 0; \\
\end{align*}

Constraint B2

\begin{align*}
\text{ARRIVAL AND DEPARTURE SAME PLATFORM} \\
\text{forall } t \in T : \\
& \quad \text{forall } r_a \in R_a : \\
& \quad \quad \text{sum } r_d \in R_d : \text{ station track of } r_d \text{ and } r_a \text{ is different} \\
& \quad \quad \quad \text{sum } s_d \in S \\
& \quad \quad \quad \quad x_{t,r_d,s_d} \\
& \quad \quad \quad \text{sum } s_a \in S \\
& \quad \quad \quad \quad x_{t,r_a,s_a} \\
& \quad \quad \quad \leq 1; \\
\end{align*}

Constraint B3

\begin{align*}
\text{PLATFORM FOR STOPPING TRAIN} \\
\text{forall } t \in T : \text{ } t \text{ is a stopping train} : \\
& \quad \text{forall } r \in R : \text{ route } r \text{ does not lead to a platform} : \\
& \quad \quad \text{forall } s \in S : \\
& \quad \quad \quad x_{t,r,s} = 0; \\
\end{align*}

Constraint B4

Constraints B1, B2, B3, B4, B5, B6, B7, B11 and B9 will always be assumed. Moreover there is an optional constraint B12.

Constraint B5 can not be expressed using independent set constraints. The time that a train occupies the platform depends on both the arrival shift and the departure shift. So the occupation of the platform by two trains at the same time does depend on four variables. Therefore we obtain conditions like $x_{t_1,r_{1i},s_{1i}} + x_{t_1,r_{1o},s_{1o}} + x_{t_2,r_{2i},s_{2i}} +$
Platforms claimed by at most one stopping train

forall \{t_1, t_2\} \subseteq T: t_1 \neq t_2,

\begin{align*}
s_{1a}, s_{1d}, s_{2a}, s_{2d} \in S: t_1 \text{ with arrival and departure shifts } s_{1a} \text{ and } s_{1d} \text{ and } t_2 \text{ with given arrival shifts have overlapping platform claim times:} \\
\forall \text{ platforms } p: \\
\sum r \in R_a: r \text{ stops at platform } p \\
&= x_{t_1, r, s_{1a}} \\
+ \\
\sum r \in R_d: r \text{ stops at platform } p \\
&= x_{t_1, r, s_{1d}} \\
+ \\
\sum r \in R_a: r \text{ stops at platform } p \\
&= x_{t_2, r, s_{2a}} \\
+ \\
\sum r \in R_d: r \text{ stops at platform } p \\
&= x_{t_2, r, s_{2d}} \\
\leq 3; 
\end{align*}

Constraint B5 In this case, the shifts also determine whether \(t_1\) and \(t_2\) have overlapping stopping times. This means that we do need the decision variable for the arrival as well as for the departure of \(t_1\) and \(t_2\). Not all four cases are allowed together, so we need that the sum is \(\leq 3\).

\(x_{t_2, r_{2a}, s_{2o}} < 4\) if train \(t_1\) using certain routes and shifts combined with train \(t_2\) using certain routes and shifts leads to a conflict. Not all four events are allowed together. This is however not an independent set constraint. If we define the variables for a combination of train with inbound route, outbound route, shift on arrival and shift on departure, then it can still be expressed by independent set constraints. However, then the number of variables would be much bigger.
Platforms not claimed by both stopping and driving train
\[
\text{forall } (t_1, t_2) \in T^2: t_1 \neq t_2, v \in \{a, d\} \text{ (arrival or departure), } s_{1a}, s_{1d}, s_2 \in S:\n\]
platform claim time of \(t_1\) with shifts \(s_{1a}\) and \(s_{1d}\) overlaps route \(v\) claim time of \(t_2\):
\[
\text{forall platforms } p:\n\]
\[
\text{sum } r_{1a} \in R_a: \text{\(r_{1a}\) stops at platform } p\n\]
\[
+ \text{sum } r_{1d} \in R_d: \text{\(r_{1d}\) stops at platform } p\n\]
\[
+ \text{sum } r_2 \in R_v: \text{\(r_2\) uses platform } p\n\]
\[
\leq 2; \tag{1}
\]
Constraint B6

Overlapping routes not used at the same time
\[
\text{forall } \{t_1, t_2\} \subseteq T: t_1 \neq t_2, v_1, v_2 \in \{a, d\} \text{ (arrival or departure): }\n\]
\[
\text{forall } r_{1} \in R_{v_1}, s_1 \in S:\n\]
\[
\text{sum } r_{2} \in R_{v_2}: \text{\(r_2\) has infra overlap with } r_{1}, s_2 \in S: t_1 \text{ for } v_1 \text{ with shift } s_1\n\]
and \(t_2\) for \(v_2\) with shift \(s_2\) have overlapping route claim time.
\[
\text{\(x_{t_1,r_2,s_2}\)}
\]
\[
+ \text{\(x_{t_1,r_1,s_1}\)}
\]
\[
\leq 1; \tag{2}
\]
Constraint B7

Trains in same series take same routes and have same shift
\[
\text{forall } \{t_1, t_2\} \subseteq T: t_1 \neq t_2 \text{ same train series: }\n\]
\[
\text{forall } r \in R, s \in S:\n\]
\[
\text{\(x_{t_1,r,s}\) }= x_{t_2,r,s}; \tag{3}
\]
Constraint B8: Optional

Variables \(z(t_1,r_1,s_1)(t_2,r_2,s_2)\) represent product \(x_{t_1,r_1,s_1}x_{t_2,r_2,s_2}\)
\[
\text{forall Relevant combinations } (t_1, r_1, s_1), (t_2, r_2, s_2):\n\]
\[
x_{t_1,r_1,s_1} + x_{t_2,r_2,s_2} - z(t_1,r_1,s_1)(t_2,r_2,s_2) \leq 1;
\]
\[
z(t_1,r_1)(t_2,r_2) \geq 0; \tag{4}
\]
Constraint B9
**Shift of Arrival and Departure Correspond**

forall \( t \in T, s_1, s_2 \in S \):

- Let \( \tau(t)_d = \tau(t)_d \) if \( \tau(t)_a \leq \tau(t)_d \), and \( \tau(t)_d = \tau(t)_d + 60 \) otherwise.
- if \( t \) is a stopping train and \( \tau(t)_a + s_1 + \text{minimal stopping time} > \tau(t)_d + s_2 \)
- or \( t \) is not stopping and \( s_1 \neq s_2 \) then

\[
\sum_{r \in R_a} x_{t,r,s_1} + \sum_{r \in R_d} x_{t,r,s_2} \leq 1
\]

**Constraint B11:** For a stopping train, there should be a minimal halting time. For a train that does not stop the shift of the arrival and departure should be equal.

---

**Trains in Same Series Have Same Shift**

forall \( t_1, t_2 \in T \) in same train series, \( s \in S \):

\[
\begin{align*}
\sum_{r \in R_a} x_{t_1,r,s} &= \sum_{r \in R_a} x_{t_2,r,s} \\
\sum_{r \in R_d} x_{t_1,r,s} &= \sum_{r \in R_d} x_{t_2,r,s}
\end{align*}
\]

**Constraint B12:** Optional
Chapter 4

Measures for robustness

An important aspect of the quality of the schedule is the extend delays propagate from one train to another. A routing is said to be robust if it avoids delay propagation.

We distinguish primary disturbances from secondary disturbances. The primary disturbances ($\Delta_a, \Delta_b, \Delta_d$), described in section 2.3, are not caused by the interaction between trains at the current railway station and are not influenced by the choice of the train routing. The secondary disturbances are caused by propagation of the primary disturbances. So we could say that a routing is robust if the primary disturbances result in a secondary disturbance that is as low as possible.

This section describes how robustness of a schedule can be measured. We can distinguish deterministic stability measures from probabilistic stability measures.

4.1 Deterministic stability measures

Deterministic stability measures can be calculated directly from a candidate routing. They do not depend on the distribution of the primary delays. In general, deterministic stability measures are relatively easy to calculate, but less informative about the performance of the schedule than probabilistic stability measures. In the literature, several deterministic stability measures for train routings are defined. Most of them rely on time intervals between blocking events. The idea is that the time slot between two planned events with common track sections is a measure for the robustness because it refers to the available buffer time. If the buffer times are large, trains can be delayed, without influencing other trains.
4.1.1 Buffer times between consecutive events

There are a lot of variations between the different time slots that can be measured. An arrival or a departure is referred to as an event. One could decide the time slot between two consecutive events or between every pair of events that shares track sections. The track sections can be emphasized by deciding for each track section how much time there is between consecutive occupations. Moreover it is important how the different buffer times in a schedule should be combined into one measure.

In Herrmann [2006], the definition of the simple time slot of an event is given as the time interval in which the event can take place without conflicting any other train, under the assumption that all other trains run on time. The simple time slot of event $e$ can be divided in two parts: the part before the actual scheduled time and the part after the scheduled time. In Herrmann [2006] it is suggested that the length of the total time slot gives us information about the stability of this train. However, it could be important to make the difference between the length of the time slot before the scheduled time and after the scheduled time. The first period gives an indication of the dependency of event $e$ from earlier trains. The second is a measure of the dependency of later events from event $e$. Both should be sufficiently large to prevent propagation of delays. In Herrmann [2006] a method is given in order to determine the simple time slot for an event. A schedule is there called deterministically stable if a desired minimal time slot for each train is provided. So only the smallest time slot is taken into account.

The measure of robustness that is suggested in Caimi [2009] is the amount of delay that can be tolerated without changing anything in the plan. That is: the maximal time deviation that any train may have, while still taking the scheduled route, without changing the order in which the different trains cross the track section and while all other trains run as scheduled. This in fact corresponds to the minimal left or right part of a simple time slot. A drawback of this measure of robustness is that only the most limiting time interval is taken into account. It could be useful to also consider other time intervals.

The method in Schlechte and Borndörfer [2010] does not focus on the events, but on the track sections. It is suggested, that we should consider the intervals between the occupation of each track section. As was stated in Section 3.1.1, $\sqrt{B}$ expresses the robustness of an event pair with buffer time $B < B_{\text{max}}$. By summing over all track sections and their consecutive event pairs, one obtains the robustness of the schedule. A problem of focussing on the track sections is that two consecutive track sections are often both occupied by the same trains, so the penalties or objectives of these time slots count double if one track section is divided in two. It is not reasonable to impose a penalty for both sections, because the delay of a train on consecutive track sections is highly dependent or even equal. Moreover, if a track section is divided in more parts, then this results in a higher sum, which is undesirable.
4.1.2 Dependencies between all pairs of events

For the measures from Caimi [2009], Herrmann [2006], Schlechte and Borndörfer [2010] the buffer time between consecutive events was considered. However, it depends on the routes of other events if a pair of events is consecutive. This makes it difficult to use these measures in the LP model. In the objective functions from Kroon and Maróti [2008], Dewilde et al. [2014], Caprara et al. [2010], not only consecutive events were taken into account, but every pair of events with common track sections and a small time difference. Depending on the time interval, a penalty factor is defined. Then, the penalties on combinations of routes do not depend on the whole schedule, but only on the two involving events. It is therefore suitable for optimization, as is described in Section 3.1.1.

Defining a penalty between every pair with common track sections reflects the fact that the delay of a train can have impact on all following events on that track section and not only at the next train. A drawback of this method is that it suggests that delay can only propagate between trains that share a track section. Example 8 illustrates why this could be a disadvantage.

**Example 8.** Suppose four different trains have to be assigned to a route (see figure 4.1). For the red train there are two possibilities: a and b.

![Figure 4.1: The red train has two possibilities.](image)

*We could consider the situation with route a as less robust than the situation with route b. The red train has a common track section with either the blue or the yellow train. Because the blue train can also be influenced by the green train, the event time is less reliable. However, these situations would obtain the same objective value.*

A method that takes into account these details in a more direct way, determines for a given routing for every pair of trains how much delay one train can have before it influences the other. It is described in Delorme et al. [2009]. It is assumed that no rescheduling takes place: the order of trains using the different track sections is preserved. A graph is defined where each node corresponds to an event. Directed edges are defined between pairs of events that share a track section. The length of the edge is the minimal time that is needed between the occurrence of the events. Then the length of shortest
path from the node of event $e$ to the node of event $e'$ (if it exists) describes the dependency of the punctuality of event $e'$ from the punctuality of event $e$. If there is no path from $e$ to $e'$, then a delay of event $e$ will have no consequences for event $e'$. If the path has length $k$, then a delay of $d$ minutes of event $e$ will lead to a delay of $\max(0, d - k)$ minutes on event $e'$ if no rescheduling takes place. We can now define penalties based on these shortest path lengths. The (weighted) sum of these penalties is now a measure of stability. In Delorme et al. [2009] this method is used to determine the resulting delay from certain initial delays. It is difficult to directly optimize over this measure.

### 4.2 Probabilistic Stability Measure: Robustness based on resulting delays

In probabilistic stability measures the distribution of the primary disturbances is taken into account. The goal of this section is to describe the dependency of secondary delay from primary delay for a certain schedule. For a given station, the train routing has no influence on the primary disturbance. The dependency of the secondary delay from the primary delay therefore expresses the quality of the train routing.

Different probabilistic stability measures were described by Herrmann [2006]. He described the expected number of conflicts, the schedule failure probability and determines critical trains, that are expected to cause the largest number of conflicts. Based on simulations, knock-on delays were determined by Dewilde et al. [2014] and the total amount of delay was counted by Caprara et al. [2010].

Because the amount of delay does reflect the quality of the performance best, it was chosen to simulate the rescheduling process. Different strategies are discussed in Chapter 5. The differences between the performance of the routings are better distinguished if we only consider the knock-on delays. In the average delay, the primary delay will play a large role.

We say that an arrival event has **knock-on delay** if the arrival is later than the planned arrival plus the primary arrival delay. We say that a departure event has knock-on delay if the departure takes place later than the earliest possible departure given the planned departure time, primary arrival delay, primary halting delay and primary departure delay of this train.

If an event obtains knock on delay, the last event that uses a necessary track section is said to be **responsible** for this knock on delay. An exception is made for departures with knock on delay that are delayed by the arrival of the same train that has knock-on delay. Then no other event is called responsible for this departure delay.
4.2.1 Objective function

If we considered all stations at the same time, a possible objective to minimise would be the average arrival delay of the trains. However, because only one station at a time is considered, the departure delays are also important. These departure delays give in a way a prediction for the primary arrival delay at the next station and thereby a prediction for the influence on the network. If information is available about the track to the next station, it is possible to transform the departure delays into a better predictor for the delay at the next station. Because there can be trains that stand still for a long period of time, the arrival delay does not always have impact on the departure delay. It was therefore decided that both the arrival and departure delay are counted and in the objective function they are weighted equally.

For some rescheduling strategies it is possible to change the routes of trains. A small penalty is imposed if only the route changes. A larger penalty is imposed if also the platform changes. This penalty can also depend on the distance from the planned platform. We could also take into account a weight for the importance of the punctuality of the event.

Suppose a routing plan $\phi$ is given and suppose this plan is performed with certain deviations according to a rescheduling strategy. Let $s(e)$ be the knock-on delay of event $e$. Let $b_0(e) = 1$ if the route of event $e$ is different in the execution and the planning (and $b_0(e) = 0$ otherwise). Let $b_1(e) = 1$ if event $e$ relates to a halting train and the platform of event $e$ is different in the execution and the planning (and $b_1(e) = 0$ otherwise). Let $b_2(e) = 1$ if moreover, the planned platform and the realised platform are far from each other for the passenger. Let $c_0$, $c_1$ and $c_2$ be constants. Then the performance of this routing plan and rescheduling strategy for the given primary delays can then be expressed by:

$$\sum_{e \in E} s(e) + c_0 b_0(e) + c_1 b_1(e) + c_2 b_2(e)$$ (4.1)

Because it is not completely clear how the different objectives contribute to the performance, they will be shown separately for the different tests in Chapter 7. Moreover, the number of events with knock-on delay will be presented.

4.2.2 Trains from/to the same direction

Trains that come from the same direction and trains leaving in the same direction inevitable have a common track section in their inbound or outbound route. Therefore, these events can have inevitable conflicts. Additional common track sections in the inbound or outbound route for trains from or to the same direction will not further influence the robustness. Primary arrival delays are chosen such that there is no conflict on the entrance point between the arrival events from the same direction. In most cases, the delays are chosen such that the trains from the same direction arrive in the planned...
order.
Chapter 5

Rescheduling strategies

In order to know how the routing schemes would perform in practice, the real-time rescheduling of trains has been simulated. For a realistic simulation, information is needed about the rules dispatchers use to handle a delayed scheme. In case of a small delay, we can assume that the route of the trains and the order in which the different trains pass a certain track section in the network do not deviate from the plan. This means that if a train is delayed, then subsequent trains may have to wait. In case of larger delays, it is more realistic to assume that the order of the trains and the platforms should be changed. It would be useful to know how this is decided and in which cases the order and routes are preserved. However, it is very difficult to incorporate operational traffic control decisions, as these decisions are made on the spot and depend on the situation.

By Dewilde et al. [2014], priority rules based on the real practice were applied to solve conflicts in the rescheduling simulation. However, these priority rules are not available in the Dutch situation. Caprara et al. [2010] use different rescheduling strategies. One of them is a simple strategy where the order is preserved, one is based on a backup platform for each event and one of the rescheduling strategies is a full rescheduling strategy that allows any kind of re-scheduling action. The problem with a full rescheduling strategy were information about future delays is used, is that it is unrealistic to assume that all information is known.

For this section three different strategies were implemented. The methods are first and foremost meant to measure the quality of the routing schemes. For real rescheduling purposes, the assumptions of the first two methods are too restrictive and would probably not perform better than the manual dispatching. Ideas from the last method could also be used for systems supporting the decisions during the rescheduling.

The first rescheduling strategy uses the rule that for events that share relevant track sections, the planned order is retained. By Caprara et al. [2010] this method is called *Delay Propagation*. For the second method, the first event that is ready to take place
is handled first. The resulting order will be called the _Delayed Order_ or _first-come first-served_ strategy.

The first two methods have a simple rule to decide the order of the trains that does not use any knowledge about the schedule or delay of future trains. The last method is much more involved and is based on a _re-optimizing strategy_ that can use both knowledge about the planning and an estimation of the delay of future trains. We assume that at each point in time, a certain estimation of the future delays is known. Given this estimation, the routes and order of the trains in the near future are optimised with respect to objective function (4.1).

## 5.1 Assumptions for rescheduling

For the schedule that is made beforehand, it is assumed that if all trains are on time, no conflicts take place. If the trains arrive or leave with a certain delay, then it is possible that different trains intend to claim the same track section at the same time. Therefore, the rescheduling strategy is allowed to impose additional waiting time either at the platform, or at the point where the train enters the station. We can see this additional waiting time as a non-negative shift, as introduced in Section 3.5 for the planning phase. The shift of an event corresponds to its knock-on delay, described in Section 4.2. An important difference between the rescheduling and planning stages concerns the fixed order on the track. In the planning stage, the order of arriving trains can, in principle, be changed if the complete network allows this. For real time rescheduling, arriving trains have a certain order on the track and cannot overtake.

In principle, routes of trains can differ from the original plan, but this should be avoided. The first two strategies do not allow the use of alternative routes. Moreover, the use of positive shifts (resulting in knock-on delays) should also be avoided. Objective function 4.1 involves both factors.

In the planning stage we assumed that there should be a minimal number of minutes between a pair of events. A similar assumption is made for the rescheduling. It is possible to make the assumptions less strict. However, because we assume there are no conflicts if all trains run on time, we can not make the assumptions stricter than in the planning phase.

**By RT(\(e\)) we denote the realised time event \(e\) takes place.** Given a routing plan, we determine for each pair of events how much time there is needed between the occurrence of the first and the second event. The order in which events take place can not always directly be determined from the event times. It can happen that the event times are equal, but one of the trains arrives earlier at the place of the conflict. This can be explained by the fact that the claim period of a track section during an arrival event is in general before the actual arrival and the claim period of a departure is after the actual departure. We denote the dependency by a function \(w : E \times E \to \mathbb{R}^{\geq 0} \cup \{-1\}\).
Then $w(e, e')$ denotes the number of minutes that is needed between $\text{RT}(e)$ and $\text{RT}(e')$ if $e$ takes place before $e'$, or $-1$ if $e$ and $e'$ do not have common track sections.

In the simulation of the rescheduling, it will be assumed that the trains can wait just before they enter the station and while halting at the station. The additional time it takes to stop before entering the station and accelerating again will be ignored.

Example 9 illustrates what kind of decisions should be made by the rescheduling strategy.

![Diagram showing different possibilities for rescheduling](image)

Figure 5.1: Example of different possibilities for rescheduling

**Example 9.** Suppose the red train in Figure 5.1 is stopping at the station and is ready to leave. The blue train is ready to enter the station. The platform tracks of the two trains are different and their routes cross. The red train has a slight delay such that the route claim time of the two trains would overlap if we did not let one of them wait. We can choose to let one of the trains wait. We call this a shift. If we impose a shift of one minute on the red train, the route claim time will become the open interval $(10, 11)$ for the blue train: $(9, 10)$. We can also impose a shift of one minute on the blue train. Then the route claim times will be $(10, 11)$ for the blue train and $(9, 10)$ for the red train. It could also be possible to choose another route for one of the trains.

### 5.2 Delay Propagation strategy

The method described here, is based on the definition of robustness by Delorme et al. [2009]. It is changed a bit such that it can also take into account disturbances in the schedule caused during the halting period of the train and importance factors for different trains.

For the delay propagation strategy we then have $\text{RT}(e) \geq \text{RT}(e') + w(e', e)$ for all events $e, e'$ for which $e$ is planned before $e'$ and $w(e', e) \geq 0$. The time of event $e'$, $\text{RT}(e')$ can
again depend on other events. So $e$ can possibly depend on all events that are planned earlier. We can determine the times of the events in the order in which they are planned. Start with the first event planned on a day and handle all events in their planned order. Then the realised time for an arrival event $t_a$ is:

$$RT(t_a) = \max\{\tau(t_a) + \delta_a(t) \cup \{RT(e) + w(e, t_a) \mid e \in E \text{ planned before } t_a, w(e, t_a) \geq 0\}\}$$

(5.1)

The realised time for a departure event $t_d$ is:

$$RT(t_d) = \max\{\tau(t_d) \cup \{RT(e) + w(e, t_d) \mid e \in E \text{ planned before } t_d, w(e, t_d) \geq 0\}\} + \delta_d(t)$$

(5.2)

Here, $w(t_a, t_d) = \text{minimal halting time} + \delta_h(t)$. These times can be calculated as the longest paths in a graph. Suppose every event (arrival or departure of a train) is given by a node in a graph. Between the nodes we have directed edges from event $e$ to event $e'$ if event $e$ is planned before event $e'$ and the paths have a track section in common. The length of this edge is equal to $w(e, e')$ if $e'$ is an arrival and equal to $w(i, t_d) + \delta_d(t)$ for a departure event. Moreover, we have a source node, and from this node, there is an edge to all arrival events $t_a \in E_a$, with their simulated arrival time $\tau(t_a) + \delta_a(t_a)$ as the length of the edge. From the source node we have an edge to all departure nodes $t_d \in E_d$ with length $\tau(t_d) + \delta_d(t)$. From an arrival event $t_a$ and a departure event $t_d$ both concerning train $t$, we have an edge of length $\text{minimal halting time} + \delta_h(t)$, the simulated halting time. Now the longest path from the zero node to a certain node, gives the simulated time of the event. Note that the longest path problem for acyclic graphs can be solved in linear time. In this case the graph is acyclic because there are only edges from an event to a later event. Let $n$ be the number of events. Then the number of edges is bounded by $n^2$, so all event times can be calculated in time $O(n^2)$. 

46
5.2.1 Description of algorithm

[eventueel nog toevoegen: welk event is verantwoordelijk voor de knock-on delay]

<table>
<thead>
<tr>
<th>DELAY PROPAGATION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Set $E$ of arrival and departure events for all trains in $T$</td>
</tr>
<tr>
<td>Planned times for all events: $\tau: E \rightarrow \mathbb{N}$,</td>
</tr>
<tr>
<td>A strict order $&lt;$ on $E$ that represents the planned order (if $\tau(e) &lt; \tau(e')$ then $e &lt; e'$)</td>
</tr>
<tr>
<td>Initial delays on the arrival time of each train: $\delta_a : T \rightarrow \mathbb{R}^{\geq 0}$,</td>
</tr>
<tr>
<td>Initial delays on halting time: $\delta_h : T \rightarrow \mathbb{R}^{\geq 0}$,</td>
</tr>
<tr>
<td>Initial delays on the departure time: $\delta_d : T \rightarrow \mathbb{R}^{\geq 0}$,</td>
</tr>
<tr>
<td>Dependency function $w : E \times E \rightarrow \mathbb{R}$.</td>
</tr>
<tr>
<td><strong>Output:</strong> Realised event times for each arrival and departure event</td>
</tr>
</tbody>
</table>

**Initialise:**
For the arrival of train $t$: $RT(t_a) = \tau(t_a) + \delta_a(t)$
For the departure of train $t$: $RT(t_d) = \tau(t_d) + \delta_d(t)$

**foreach** $e \in E$ (**handle events according to order <**) **do**
  **foreach** $e' \in E$ with $e' > e$ **do**
    if $e = t_a$ and $e' = t_d$ for a certain $t \in T$ **then**
      $RT(t_d) = \max(\tau(t_d), RT(t_a) + \text{minimal halting time} + \delta_h(t) + \delta_d(t))$
    else
      $RT(e') = \max(RT(e'), RT(e) + w(e, e') + \delta_d(t))$
  **end**
  **end**
**Return** $RT$.

**Algorithm 23:** Delay propagation strategy for real-time dispatching

In the implementation, the dependency function $w$ is only defined for one basic hour of trains. This leads to some small exceptions in the code when a train arrives and departs in a different hour.

The order $<$ determines the order in which the trains can use the elements of the infrastructure. If $\tau(e) = \tau(e')$ and $e$ is an arrival and $e'$ is a departure, then we assume $\tau(e) < \tau(e')$.

The knock-on delay for an arrival or departure event can now be determined as:

$$s(t_a) = RT(t_a) - (\tau(t_a) + \delta_a(t))$$
$$s(t_d) = RT(t_d) - \max(\tau(t_a) + \text{minimal halting time} + \delta_h(t) + \delta_d(t), \tau(t_d) + \delta_d(t))$$

47
5.3 Delayed order strategy

For this strategy, it is assumed that the events take place in the order in which they are ready to take place. For the decision of the order of the events, the departure delay $\delta_d(t)$ is not taken into account, because this delay occurs after the decision of the order is taken. First it is decided which event is the next event that can take place. Therefore we take into account that departures can only take place when the arrival has already taken place, a route can only be taken if all track sections are free and an arrival from a certain direction can only take place if the train is the first one on the track. Then the
consequences for all future events are calculated.

Delayed Order($G$)

**Input:** Set $E$ of arrival and departure events for all trains in $T$

Planned times for all events: $\tau : E \rightarrow \mathbb{N}$,

Initial delays on the arrival time of each train: $\delta_a : T \rightarrow \mathbb{R}_{\geq 0}$,

Initial delays on halting time: $\delta_h : T \rightarrow \mathbb{R}_{\geq 0}$,

Initial delays on the departure time: $\delta_d : T \rightarrow \mathbb{R}_{\geq 0}$,

Dependency function $w : E \times E \rightarrow \mathbb{R}$,

Set of events and trains where route of event has overlap with halting of train.

**Output:** Realized event times $RT : E \rightarrow \mathbb{R}$

Initialise: For the arrival of train $t$: $RT(t_a) = \tau(t_a) + \delta_a(t)$;

For the departure of train $t$: $RT(t_d) = \tau(t_d) + \delta_d(t)$;

Subset of departures, for which arrival has taken place: $F_d = \emptyset \subseteq T$;

Subset of arrivals that are first in row from a direction: $F_a \subseteq E_a$;

Subset of events that have not taken place yet: $future = E$;

Subset of events for which route is free from halting train: $routefree = E$;

while There is an event in $future \cap routefree \cap (F_d \cup F_a)$ do

$e =$ first event in $future \cap routefree \cap (F_d \cup F_a)$ according to the current $RT$;

Remove $e$ from $future$;

Update $routefree$;

if $e = t_a$, the arrival of a train $t$ then

add departure $t_d$ to $F_d$;

determine new first arrival $e'$ from direction of $e$ and replace $e$ by $e'$ in $F_a$;

end

foreach $e' \in future$ do

if $e = t_a$ and $e' = t_d$ for a certain $t \in T$ then

$RT(t_d) = \max(RT(t_d), RT(t_a) + \text{minimal halting time} + \delta_h(t) + \delta_d(t))$

else

$RT(e') = \max(RT(e'), RT(e) + w(e, e') + \delta_d(t))$

end

end

end

if $future = \emptyset$ then

return $RT$;

else

return $Error$;

end

Algorithm 24: Delayed order strategy for real-time dispatching

It can happen that the algorithm ends with non-empty $future$ set. This means that not all events take place. For example, if two trains use each others stopping platform in their outbound route and by accident they are both halting at the station at the same time, then it is not possible that one of them leaves the station without changing the
planned routes.

An important disadvantage of this rescheduling strategy, is that there is no possibility to take into account future trains. This makes this strategy unsuitable for larger delays and larger halting periods. Suppose two trains are planned on the same platform. The first train does only stop for one minute and the second train stops for twenty minutes. If it happens that the first train has a delay such that the second one arrives earlier, then according to this strategy the fast train will have to wait for twenty minutes, before it can enter the station. It is not reasonable to assume that the slow train will be allowed to enter the station first and that the fast train will not be assigned to an alternative route.

5.4 Re-optimizing strategy

The two rescheduling algorithms described in the last two sections both use very simple rules to decide in which order the events take place. This results in fast algorithms. However, both are unsuitable to give a realistic picture of the real situation when larger delays take place.

Therefore a more involved method was developed where at each moment of time, an optimised solution is chosen within the possibilities and with respect to the knowledge that is present at that time. It should be clear what are the possibilities at a certain moment of time, at which moment in time certain decisions have to be made and from which moment they can not be changed any more. Moreover, the present knowledge at each moment should be modelled.

5.4.1 Model

Each minute (or relevant moment in time), the main part of the re-optimising model (RE-OPTIMISING: MAIN) considers a certain period in the future up to a horizon (for example, we consider the coming twenty minutes). The set of trains that occur in this period is called the set of current trains. For this set of trains, an estimation of their delays, \( \hat{\delta}_a \) and \( \hat{\delta}_h \) is available. Moreover, for trains that are already allowed to depart (that means: the track sections in the departure route are claimed for this train), the delay during the departure \( \delta_d \) is known. For each minute, the main function calls the function OPTIMISE CURRENT SITUATION. This part of the model searches for a solution for the current set of trains with current delay estimations. This is done with respect to objective function (4.1) under the assumption that the estimations of the delays are
Re-optimising: main

**Input**: Set $E$ of arrival and departure events for all trains in $T$,
Set $R$ of all routes with their corresponding relevant sections,
Planned times for all events: $\tau : E \to \mathbb{N}$,
Planned routes for all events: $\phi : E \to R$,
A strict order $<$ on $E_a$ that gives for each incoming direction the order of the trains,
For each moment in time an estimation of the initial arrival delays $\delta_a : T \to \mathbb{R}^{\geq 0}$,
For each moment in time an estimation of the halting delays $\delta_h : T \to \mathbb{R}^{\geq 0}$,
Departure delays (known from the moment of departure) $\delta_d : T \to \mathbb{R}^{\geq 0}$,
Objective function (concerning additional delay, route deviations)

**Output**: Realised times and routes for each arrival and departure event

```plaintext
foreach minute do
  Fix last chosen inroutes for trains $t$ that arrive within 5 minutes;
  Fix last chosen outroutes for trains $t$ that depart within 1 minute;
  Update delay estimations $\delta_a$, $\delta_h$ and $\delta_d$ for this minute;
  Optimise current situation for current trains, current delay estimation, routing plan and fixed shifts and routes;
  Update set of trains that have left (claim time of departure has ended);
  Fix shift $s_a(t)$ for trains $t$ for which the arrival starts this moment or earlier;
  Fix shift $s_d(t)$ for trains $t$ for which the departure starts this moment or earlier;
end

Return Realised routes and shifts for each train;
```

This algorithm refers to the time on several places. The routes for the arrival are fixed 5 minutes before arrival. It is chosen that the real delays are used to determine if it is 5 minutes before arrival, although these can be unknown on that moment of time. Moreover, the last chosen shift is taken. This means that the routes are fixed when $\tau(t_a) + \delta_a(t) + s_a(t) < \text{time} + 5$. The routes for the departure are fixed when $\tau(t_d) + \max(0, \tau(t_a) + \delta_a(t) + \text{minimal halting time} + \delta_h(t) - \tau(t_d)) + s_a(t) < \text{time} + 1$.

The shifts are fixed on the moment the operational control claims the track sections of the routes for the train. We assume that $\delta_a$ is known when the claim period of the arrival route starts and $\delta_h$ is known in case of an departure. The delay during departure $\delta_d$ is only known from the moment the tracks for departure are claimed.

The goal of the module Optimise current situation is to choose certain additional waiting times for the different trains and rerouting decisions such that the amount of resulting delay and the number of deviations from the routing plan is as small as possible, as is expressed in objective function (4.1). These additional waiting times in the real time rescheduling can be compared to the shifts in the planning phase from Section 3.5. Although the goal of real time rescheduling is different from the creation of a routing plan beforehand, the conditions here are similar to the conditions that were imposed in Section 3.5. Note that in this case, the shifts can only be non negative, because they
correspond to waiting times. In this model, additional conditions are imposed that fix the decisions that were made on an earlier moment of time. Another difference from the model in section 3.5 is that the rescheduling should be done based on an existing plan and not too many deviations should be made. This is incorporated in the objective function.

**Optimise current situation**

**Input**: Set $T$ of current trains (trains that have not left up to horizon)
Set $E$ of arrival and departure events for all trains in $T$
Planned routes for all events $\phi : E \rightarrow R$
Planned times for all events of the current trains: $\tau : E \rightarrow \mathbb{N}$,
Current delay estimations $\hat{\delta_a}, \hat{\delta_h}, \hat{\delta_d}$
Information about fixed routes and fixed shifts
A strict order $<_{\text{a}}$ on $E_a$ that determines for each incoming direction the order of the trains

**Objective function** (concerning additional delay, route deviations)

**Output**: Feasible plan with shifts and routes for all current trains

Introduce binary variables $x_{t,r,s}$ for all trains $t$, routes $r$ and possible shifts $s$.

Optimize objective function under constraints $B_1, B_2, B_3, B_4, B_6, B_7, C_{10}, C_{11}, C_{13}, C_{14}$;

Return routes and shifts for each current train;

In this case it is chosen not to use condition $B_5$, because it is redundant and it leads to a lot of additional constraints on the variables. This model turned out to be faster without this constraint. Constraint $C_{10}$ was used to fix the order of trains from the same direction because they can not pass each other on the track. Constraint $C_{11}$ was used instead of $B_{11}$. The cyclic time from the original model is not used in $C_{11}$ and in the rescheduling we do allow trains to stop even if they were not planned to stop, where in the planning no stops can be added. Constraint $C_{13}$ and $C_{14}$ expres that certain routes and shifts are already fixed in an earlier stage.

**Fixed order**

forall $t_1, t_2 \in T$ (current trains), $v_1, v_2 \in \{a, d\}$:

if It is fixed that event $v_1$ of $t_1$ takes place before $v_2$ of $t_2$ then

forall $s_1, s_2 \in S$:

if $\tau(t_1)_{v_1} + D_{v_1}(t_1) + s_1 > \tau(t_2)_{v_2} + D_{v_2}(t_2) + s_2$ then

$\sum_{r_1 \in R_{v_1}} x_{t_1, r_1, s_1} + \sum_{r_2 \in R_{v_2}} x_{t_2, r_2, s_2} \leq 1$;

end

end

**Constraint C_{10}:** order can be fixed for trains from same direction

In this model, the begin and end of claim times depend on the delays. From the estimated arrival delay $\hat{\delta_a}$ and estimated halting delay $\hat{\delta_h}$, we can derive an estimation for the

52
**Shift of arrival and departure correspond**

\[
\forall t \in T, s_1, s_2 \in S: \\
\begin{align*}
&\text{if } t \text{ is a stopping train and the platform claim time is smaller than} \\
&\quad \text{minimal stopping time } + \hat{\delta}_h(t) \\
&\quad \text{or } t \text{ is not stopping and the arrival is later than the departure then} \\
&\quad \sum_{r \in R_a} x_{t,r,s_1} + \sum_{r \in R_d} x_{t,r,s_2} \leq 1
\end{align*}
\]

**Constraint C11:** For a stopping train, there should be a minimal halting time. For a train that does not stop the departure cannot be earlier than the arrival.

**No route deviations from fixed plan**

\[
\forall t \in T \ (\text{current trains}), r \in R: \\
\begin{align*}
&\text{if Route } r \text{ is fixed for train } t \text{ then} \\
&\quad \sum_{s \in S} x_{t,r,s} = 1
\end{align*}
\]

**Constraint C13:** Routes in plan are fixed because decision is already made in an earlier stage.

**Fixed shifts**

\[
\forall t \in T \ (\text{current trains}), v \in \{a, d\}: \\
\begin{align*}
&\text{if } v \text{ shift for train } t \text{ is fixed to be } s \text{ then} \\
&\quad \sum_{r \in R_v} x_{t,r,s} = 1
\end{align*}
\]

**Constraint C14:** Shifts for certain trains are fixed because the events have already taken place.

resulting delay on departure:

\[
\hat{\delta}_o(t) = \max(0, \tau(t_a) + \delta_a(t) + \text{minimal halting time} + \hat{\delta}_h(t) - \tau(t_d)) \tag{5.3}
\]

In this estimation, \(\hat{\delta}_d(t)\) is not taken into account, because this delay occurs on the moment of departure, so is only known at the moment it is decided the train can depart.

The claiming interval for the arrival route of train \(t\) is as follows:

\[
(\tau(t_a) + \delta_a(t) + s_a - 1, \tau(t_a) + \delta_a(t) + s_a) \tag{5.4}
\]

The claim time for the departure route:

\[
(\tau(t_d) + \hat{\delta}_o(t) + s_d, \tau(t_d) + \hat{\delta}_o(t) + s_d + \delta_d(t) + 1) \tag{5.5}
\]
The claim time for the platform:

\[(\tau(t_a) + \delta_a(t) + s_a - 1, \tau(t_d) + \delta_o(t) + s_d + \delta_d(t) + 1)\]  (5.6)

Note that \(\delta_a\) and \(\delta_o\) do not make the claim time longer but only shift this time. The delay \(\delta_d\) does not change the begin of the claim time, but only makes the claim time longer. This corresponds to the interpretation that \(\delta_d\) occurs on the moment the departure route is claimed for the train.

The definition of overlapping time intervals from equation (3.14) should be adapted for this model, because in this model there is a linear time scale, where the planning model is cyclic. In this case it becomes very simple. Suppose two pairs of begin and end of the claim times are given: \((\sigma_1, \sigma_2)\) and \((\tau_1, \tau_2)\). The periods (as open intervals) have overlap in a cyclic schedule if the formula from (5.7) holds:

\[\begin{align*}
(\sigma_1 & \leq \tau_1 \land \tau_1 < \sigma_2) \\
\lor (\tau_1 & < \sigma_1 \land \sigma_1 < \tau_2) \\
\lor (\sigma_1 & < \tau_2 \land \tau_2 \leq \sigma_2) \\
\lor (\tau_1 & < \sigma_2 \land \sigma_2 \leq \tau_2)
\end{align*}\]  (5.7)

The number of possible shifts for each train should be finite and the complexity of the problem increases when we increase the number of possible shifts. [For Uitgeest example: how does the calculation time increase?] Therefore, if we use this method in real time rescheduling, we could either make the number of possible shifts as high as possible for the given complexity (for example 6 possible shifts each hour), or we can determine the possible shifts depending on the relevant points in time, determined by expected events that take place. This is not done here. Suitability for real time rescheduling should further be investigated.

Note further that we could consider not to optimize every minute, but for example only every 5 minutes or only in cases a decision has to be made (a route fixed or order chosen). In the first case it is possible that we do not use the latest information to decide, which could lead to worse solutions. However, if the estimations are already quite good, then based on this information, a good decision can be made. If we optimize at all points in time where a decision has to be made, then we always use the latest information and are thus able to make a good decision. The problem here is that in the minutes where we do not optimize, certain information about the delay of arriving trains can be updated. Because we do not re-optimize in these minutes, this information can not be used to fine tune the event shifts. The event shifts do not need to be fixed before the event happens. It is an outcome of the other decisions. However, the order of the trains do have to be decided before the first of them takes place. So in the minutes where we do not re-optimize, we should keep track of what happens and what shifts are the outcome of the other decisions. If we already fix the shifts before we have the latest information, it is possible that infeasible situations occur. If we fix the shifts after the events happened, it
is possible that we use new information to change the order of events that have already taken place.

It is essential that the shifts of trains with track section overlap in the route are not fixed on the same moment. Then it is possible that the additional delay upon departure leads to an infeasible situation. Because the claim times are always at least one minute, it is not possible that the start of the claim times of two events with overlapping track sections takes place within the same minute. With this algorithm it is not possible to take time steps larger than one minute. Another algorithm was made that can use information about $\delta_a$ at all times and can cope with larger time steps, but can not take into account departure and halting delays.

Knowledge

For every minute and train we can define the estimated arrival delay of the train that is known at that moment. For our tests, we assumed that initially, all arrival delays were assumed to be 0. A constant number of minutes (for example 10) before the planned arrival time, a first estimation of the arrival time is given. A positive or negative error $e(t)$ is added to the real delay to obtain the non-negative estimation. A constant number of minutes (for example 2) before the real initial arrival time, the arrival delay $\delta_a$ is assumed to be known.

\[
\hat{\delta}_a(t) = \begin{cases} 
0 & \text{if time} < \tau(t_a) + c_1 \\
\max(0, \delta_a(t) + e(t)) & \text{if } \tau(t_a) + c_1 \leq \text{time} < \tau(t_a) + \delta_a(t) + c_2 \\
\delta_a(t) & \text{if } \tau(t_a) + \delta_a(t) + c_2 \leq \text{time}
\end{cases} \tag{5.8}
\]

The delay upon halting, $\delta_h$, is assumed to be known from the moment the train has arrived.

\[
\hat{\delta}_h(t) = \begin{cases} 
0 & \text{if time} \leq \tau(t_a) + \delta_a(t) + s_a(t) - 1 \\
\delta_h(t) & \text{if } \tau(t_a) + \delta_a(t) + s_a(t) - 1 < \text{time}
\end{cases} \tag{5.9}
\]

The delay $\delta_d(t)$ is assumed to be known from the moment of departure:

\[
\hat{\delta}_d(t) = \begin{cases} 
0 & \text{if time} \leq \tau(t_d) + \delta_a(t) + s_d(t) \\
\delta_d(t) & \text{if } \tau(t_d) + \delta_a(t) + s_d(t) < \text{time}
\end{cases} \tag{5.10}
\]

Rescheduling possibilities

We assumed that from a certain constant number of minutes before the arrival or departure it is not allowed to change route and platform. It can make a lot of difference if
this final decision has to be made before or after the moment we are certain about the arrival delay.

**Objective function**

For the objective function we sum over the penalties for all combinations \((t, r, s)\) that are chosen. We sum over all shifts, \(c_0\) is added to the objective value if \(r\) different from planned route of \(t\). We add \(c_1\) if \(t\) is stopping and \(r\) and \(\phi(t)\) use different platforms. We add \(c_2\) if moreover, the platform of \(\phi(t)\) is far from the platform of \(r\). This is expressed in objective function (4.1).

**Remarks**

A weak point of this method, is that the certainty of the estimations is not used. A possibility is to let the penalty for a certain train \(t\) be dependent on the certainty of the estimation for \(t\). However, the certainty of the delay does not directly correspond to the certainty of a conflict, but also on the amount of the overlap.

A second model was made, where more than one possible sequence of input delays is possible and the average performance over these two models is taken as objective.
Chapter 6

Routing by Stochastic Optimization

The method described in this section is inspired by an article on stochastic optimization used for cyclic railway timetables by Kroon et al. [2007]. The Stochastic Optimization Model the is proposed in this chapter uses a sample average approximation method, and thereby combines a timetabling model with a rescheduling simulation model.

As was explained in Section 4.2, a possible objective function is the expected total delay of all events:

$$
E_{\delta \in \Delta} \left[ \sum_{e \in E_a \cup E_d} D_{\phi, \delta}(e) \right]
$$

Here, the set $E_a$ is the set of arrival events and $E_d$ is the set of departure events.

The goal is thus to find a routing $\phi$ that minimises this expectation. However, since the expected delay is difficult to compute and minimise directly, an approximation is used. To this end, we take a random sample of initial delay vectors $\delta_1, \delta_2, \ldots, \delta_R$ as input, either obtained by real data or by simulation. The expected total delay is then approximated using a certain rescheduling strategy by $\frac{1}{R} \sum_{r=1}^{R} \sum_{e \in E_a \cup E_d} D_{x, \delta_r}(e)$. It turns out that we can find a routing $\phi$ that minimises this function using the ILP described in the next section. By taking the number of realizations $R$ large enough, $x_R$ approximates a good and robust routing.

[In what sense does it approximate the “best” routing?]

The trick in this model, is that routing and rescheduling is done simultaneously. In a way, the rescheduling agent is assumed to be omniscient. All disturbances are generated prior to solving the model. Given the initial delay vectors $\delta_1, \delta_2, \ldots, \delta_R$ for $R$ hours of trains, a plan should be made for multiple hours at the same time. The techniques are similar to those used in the routing model with shifts from section 3.5. However,
here the time is not assumed to be cyclic: multiple hours occur in the model. The shifts represent rescheduling decisions instead of changes in the planning and are thus assumed to be non-negative. The goal is to find one routing plan that applies for all hours, where rescheduling decisions are made for the hours individually. It can be demanded that in each hour the same routes are taken or that rerouting is possible for individual hours. In each hour a different additional delay can be given to make it feasible for the different delays.

Figure 6.1: Example where information about the delay of the red train is useful for the decision of the order of the blue and green train. Suppose the blue and green train both arrive at 12:05 at the entry of the station. Assume all trains claim their inbound route for one minute, the platform for another minute and the outbound route for one minute. Without knowledge about the delay of the red train, one could decide to let the green train enter first, because this is the planned order. However, with this knowledge, it would be better to let the blue train enter first, because then the brown train can enter earlier.

A drawback of this method is that the knowledge in the rescheduling is not realistic. All delays of the future trains are assumed to be known. This means that in the rescheduling, complete knowledge of all disturbances in the future can be used at any point of time. If we do not allow trains to take different routes, this is not a very strong assumption.

In figure 6.1, an example is given where knowledge about future events can help to take decisions about the order. In the example, we only use information of trains that enter in the near future. It is reasonable to assume that we have a good estimation of the initial delay of these trains. Examples where we use delay information of trains that are planned much later are quite artificial. Even if the rescheduling strategy behaves a bit different by this knowledge and therefore gives a slightly optimistic estimation of the delay, it is not clear how this would influence the resulting routing \( x_R \). It would slightly favour situations in which the omniscient rescheduling strategy has some advantage, but this does not seem to make a large difference, because situations in which two trains have to wait for each other are still not favoured above situations in which there is no conflict.

We could also add the option that in the rescheduling different routes can be chosen. We then let the algorithm find a standard routing and a deviation for each hour. The objective function then penalises a deviation from the standard routing and minimises the (weighted) sum of this deviation penalty and the total delay. In this case, there seem to be more problems with the omniscience of the rescheduling strategy, because trains
can have long planned dwell times.

**Example 10.** Suppose that a train has a planned stopping time of 20 minutes on platform 4A and that it is leaving in the Z direction. Suppose 5 minutes before its planned leaving time, another train enters the station from direction B. Suppose this train has a delay of 6 minutes, then one of the trains will have to wait. However, in the situation where we know beforehand that this will happen, the reschedule strategy can choose to let the first train arrive at another platform to avoid this situation. It is however unlikely that this will be done in real situations, because we do not exactly know the arrival time of the second train 15 minutes before its planned arrival time and the conflict only occurs if the arrival time of the second train is within a short time of the departure of the first train.

It however still seems that the amount of cases where rerouting would take place based on unrealistic future knowledge is small. Especially if we only allow deviation from the route without deviating from the planned platform, the omniscience does not seem to have much influence on the chosen routing.

The number of cases where this rescheduling method underestimates the delay can be calculated afterwards, by applying a rescheduling strategy on the found routing, assuming the same initial delays and a more realistic availability of knowledge.

### 6.1 Implementation details

We introduce a set $T$ of all trains over several hours and a set $T_0$ of the trains in one basic hour. Each train in $T$ corresponds to one element of $T_0$ which is the train in the basic hour.

The delay $\delta_a(t)$ of each train $t \in T$ is added to its arrival time and a minimal departure delay is added to the departure time. We assume that a stopping train always has to stop at least a minimal stop time and a train that does not stop can not leave before it arrives. The minimal departure delay is calculated from the initial delays as:

$$\max(0, \delta_a(t) + \text{minimalstoptime} + \delta_h(t) - \text{plannedstoptime}) + \delta_d(t)$$

This delay is added to the departure time.

Because part of this model involves real time rescheduling, we have to allow trains from $T$ to wait before they enter or leave the station. Therefore, each event (arrival and departure) is assumed to have a certain non negative shift. This is the waiting time that is needed in order to avoid conflicts. This shift is similar to the shifts from section 3.5, but here represent real time rescheduling, where in section 3.5 they represented a deviation of the planned event times.

[C11 staat nu in eerder hoofdstuk]
Ensure value of variables \(dr\) and \(db\)

For all \(t \in T\) (real time trains), \(t_0 \in T_0\) (basic hour), such that \(t_0\) corresponds to \(t\):

For all \(r \in R_a\):

\[
\text{dr}(t, a) \geq x_{t_0, r} - \sum_{s \in S} x_{t, r, s}
\]

For all \(r \in R_d\):

\[
\text{dr}(t, d) \geq x_{t_0, r} - \sum_{s \in S} x_{t, r, s}
\]

For all platforms \(p\):

\[
\text{db}(t) \geq \\
\sum_{r \in R_a, r \text{ uses platform } p} x_{t_0, r} - \\
\sum_{r \in R_a, r \text{ uses platform } p} \sum_{s \in S} x_{t, r, s}
\]

Constraint D15

Fixed order of trains from same direction

For all \(t_1, t_2 \in T\) arriving on same track (relevant combination):

If \(\tau(t_1)_a < \tau(t_2)_a\) then

For all \(s_1, s_2 \in S\):

If \(\tau(t_1) + s_1 > \tau(t_2) + s_2\) then

\[
\sum_{r_1 \in R_a} x_{t_1, r_1, s_1} + \sum_{r_2 \in R_a} x_{t_2, r_2, s_2} \leq 1
\]

End

End

Constraint D10

The order of the trains on one track is given by their initial arrival time. This can not be changed in the rescheduling by the shifts.

No route deviations for \(T\) from \(T_0\)

For all \(t \in T, t_0 \in T_0\) such that \(t_0\) corresponds to \(t\):

For all \(r \in R\):

\[
x_{t_0, r} = \sum_{s \in S} x_{t, r, s}
\]

Constraint D13: Optional

Binary variables \(x_{t, r, s}\) are introduced for the trains \(t \in T\) and variables \(x_{t_0, r}\) for \(t_0 \in T_0\). The \(x_{t_0, r}\) variables determine the planned routes and the variables \(x_{t, r, s}\) determine the route with (non negative) shift \(s\) that is taken in a certain hour. The binary variables \(\text{dr}(t, v)\) (\(t \in T, v \in \{a, d\}\)) and \(\text{db}(t)\) denote whether the route and the platform of the train deviate from the planned route and platform. Condition D15 ensures the value of these variables.

The trains coming from the same direction are assumed to have a certain order on the track. It is not possible that trains overtake on the same track. This is translated into the condition D10.
Objective function 4

\[
\text{minimise} \quad \sum_{t \in T, r \in R, s \in S} x_{t,r,s} + s + c_0 \sum_{t \in T} d_r(t, v) + c_1 \sum_{t \in T} d_b(t)
\]

We use the constraints \(B_1, B_2, B_3, B_4, B_6\) and \(B_7\) from section 3.5 on the set \(T\). Note that in this case the trains from \(T\) are not cyclic. This means that the definition of overlap in time is different. Because the number of trains in much larger if we consider multiple hours, the constraints are adapted a bit such that only relevant combinations of trains are considered in the constraints that handle combinations of trains (\(B_6, B_7, ..\)). In order to make the routing feasible, we should also state constraints \(A_1, A_2, A_3, A_4, A_6\) and \(A_7\) for the set \(T_0\). The timescale of the trains in \(T_0\) is cyclic, so we use the cyclic definition of overlap. Constraint \(B_5\) is not needed and in this case does not speed up the calculation. Moreover, a constraint \(C_9\), similar to \(B_12\) is used, that states that the arrival and departure shift of a train should correspond.

Constraint \(D_{13}\) is optional. This condition expresses that in the rescheduling no route deviations are made.

The problem now becomes to find values for \(x_{0t,r}\) and \(x_{t,r,s}\) that minimise objective function 4 and satisfy the conditions.
Chapter 7

Experimental results

Different methods to obtain a routing plan were described in Chapter 3 and Chapter 6. The aim of this chapter is to determine the quality of these plans by means of the deterministic and stochastic measures of robustness that were covered in Chapter 4. For the stochastic measures, a rescheduling strategy is required. The different strategies from Chapter 5 were used.

[Remark: data in the first version of chapter are obtained from small tests and will later be replaced by new data]

7.1 Comparing different routing plans: deterministic measures

In order to compare the different routing plans, the simple time slots, as described in Section 4.1.1, were determined for each routing. Let \((e_1, e_2)\) be a pair of events for different trains that occur in the basic hour. Suppose the planned routes for these events share a track section. It was determined how many minutes of tolerance are present between the planned claim time of \(e_1\) and the next planned claim time of \(e_2\). We consider the occurrence of \(e_1\) before \(e_2\). If \(e_2\) occurs earlier in the hour than \(e_1\), 60 minutes of time is added to the event time of \(e_2\). It is assumed that the trains claim the whole in- or outbound route for a certain interval that does not depend on the route. This means that, for each event, the planned claim time is an interval that is known independently of the chosen route. Moreover, it is assumed that the routing is feasible, so if \(e_1\) and \(e_2\) share infrastructure, then the claim times should be disjoint. The interval claim times are given by open intervals. This means that the least possible tolerance is 0 minutes.

Table 7.1 compares the tolerance intervals for three different routing plans \((U, E6\) and \(F6)\). For each amount of tolerance it shows how often this tolerance occurs in the routing
Table 7.1: Number of event pairs with small buffer times and common track sections. For the routing plans U, E6 and Z6, the number of event pairs with certain buffer times are given. Conflicts in entrance point, exit point and platform are presented separately.

<table>
<thead>
<tr>
<th>routing</th>
<th>Amount of buffertime (min)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td># pairs with same entrance point</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>all</td>
<td># pairs with same exit point</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>U</td>
<td># pairs with overlapping track section</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>U</td>
<td># pairs with same platform</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>E6</td>
<td># pairs with overlapping track section</td>
<td>0</td>
<td>2</td>
<td>8</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>42</td>
</tr>
<tr>
<td>E6</td>
<td># pairs with same platform</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>Z6</td>
<td># pairs with overlapping track section</td>
<td>0</td>
<td>2</td>
<td>8</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>42</td>
</tr>
<tr>
<td>Z6</td>
<td># pairs with same platform</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>SOM1</td>
<td># pairs with overlapping track section</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>39</td>
</tr>
<tr>
<td>SOM1</td>
<td># pairs with same platform</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
</tbody>
</table>

For the routing plans U, E6 and Z6, the number of event pairs with certain buffer times are given. Conflicts in entrance point, exit point and platform are presented separately.

Figure 7.1: Occupation of a platform by a stopping train $t_1$ and a train $t_2$ passing by. Blue: occupation time of inbound route. Green: occupation time of outbound route. All four combinations are counted.

A special case of track section overlap occurs when two trains share the inbound or outbound direction. This kind of overlap is not specific to the routing plan and is thus inevitable. In the first two rows of Table 7.1, the occurrence of overlap of entrance or exit point is shown. It can thus be concluded that the two occurrences of 1 minute buffer time for E6 and F6 are inevitable.

The use of the same platform is another special case of track section overlap that is distinguished. If trains $t_1$ and $t_2$ use the same platform to stop, then both the arrival and departure of $t_1$ share track sections with the departure and arrival of $t_2$. All four combinations are pairs of events that share track sections and are thus counted, where only the time between the departure of $t_1$ and arrival of $t_2$ is relevant for the quality of the routing. An example of two trains using the same platform is given in Figure 7.1.

Comparing the results for the three different routing plans, it can be seen that routing plan F6 has the least occurrences of small buffer times. This suggests that routing plan...
$F_6$ would perform best. However, there is only a small difference between $E_6$ and $F_6$.

### 7.2 Distribution of delays

We distinguish initial delays and resulting delays. As was stated in section..., three different kinds of initial delays are taken into account. Delay upon arrival, delay upon the minimal halting time and delay when leaving. It is likely that the initial arrival delay is known (or at least expected) some time before the arrival.

The delay on the minimal halting time can also be expected, because this depends on the number of passengers that want to leave and enter the train through each door. The halting period in a way also depends on the arrival time. If a train is late, then it is likely that some passengers for the next train already arrived such that there are more passengers who want to enter the train. Moreover it depends on the length of the train (if the train is too short for the amount of passengers, it takes longer to let all passengers enter the train). The halting time also depends on the delay of other trains (if another train has a delay, then the passengers might have missed their next train and therefore more or less passengers might want to take this train). Moreover, it can depend on factors concerning the train driver, railway guard or there might be problems with the train itself.

The delay during departure is delay that occurs on the moment the train is allowed to leave. This can be caused by problems during the process of closing the doors.

Although there are some small dependencies between these kinds of delays, for the testing we assume they are independent. We denote the delay as a triple $\Delta = (\Delta_a, \Delta_h, \Delta_d)$, for arrival, halting and departure delay. For the tests, random delays are taken according to a certain distribution.

The exponential distribution is often taken for the arrival delay. By $E_m^\lambda$ we denote that $m$ trains (chosen randomly) obtain a delay that is chosen according to an exponential distribution with rate parameter $\lambda$. For $n$ the number of trains, $E_{n/2}^1$ is the distribution where half of the trains has a delay that is distributed according to $\exp(1)$.

For the halting and departure delay, it is often assumed that only one amount of delay can occur with a certain probability. This is denoted by $P_m^{(d)}$, where $d$ is the amount of delay. For example $P_{n/2}^{0.5}$ is the distribution where half of the trains obtains a delay of 0.5 minute.

It is assumed that from each direction, the trains arrive in their planned order. So after choosing randomly the initial delays, some additional delay is imposed to achieve this and to make sure that there is at least a minimal follow-up time (for example one minute) between two consecutive trains from the same direction.
7.3 Influence of different kinds of conflicts

See section about penalty function.

Different kinds of conflicts:

1. Same platform
2. Route of one train uses platform of other train
3. Common track sections
4. Same entry point
5. Same exit point

We assumed that the platform is a track section that is part of the route, so 1 implies 2 and 2 implies 3. Moreover the entry point is a track section, so 4 and 5 both imply 3. The question is, how we weight these different possibilities. The fourth and fifth are inevitable from our perspective. We can not change the direction of the trains. However, we can have higher weight for cases where besides 4 or 5 another conflict occurs. We assumed that the hole inbound route is claimed for a certain period of time. This means that additional conflicts in the route do not make any difference. However if two routes have the same platform, then this can make the delay larger, because if we do not reschedule, the second train also has to wait before the platform is free. This becomes worse if the stopping time is long. If we assume that the order of the events can be changed, then conflicting routes are not that bad. If we assume that a train claims the inbound route for one minute and that the first train can enter first, then the second train has to wait at most 1 minute (assuming no other trains play a role). Having fixed one of the routes, there is only two minutes of time in which the other can arrive such that there is a conflict. If we assume that the order of conflicting events is maintained, then also the the three kinds of conflicts are similar. If we assume that orders can be changed, then the third possibility is

7.4 Routing plans combined with rescheduling strategies

In this section, different combinations of routing plans and rescheduling strategies are compared. Again, the plans for railway station Uitgeest were used. For the sake of simplicity, only incoming initial delay was taken into account and for each test the same initial delays were used. It was assumed that all trains coming from the same direction arrive in their planned order with at least one minute of time between arrivals. The result is that the arrival events from the same entering point did not conflict if no additional shifts were imposed.

All delays were given with a precision of 30 seconds. By rounding all delays to a hole minute, the events would fit unrealistically well and the capacity of the tracks would be
used very efficiently, because all claim periods are open intervals starting and ending in a hole minute. Shifts were also allowed to be multiples of 30 seconds.

<table>
<thead>
<tr>
<th>Plan</th>
<th>Strategy</th>
<th>Arrival shifts</th>
<th>Departure shifts</th>
<th># Platform deviations</th>
<th>Route deviations (% of trains)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>U Delay prop</td>
<td>9</td>
<td>58.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>E6 Delay prop</td>
<td>13.5</td>
<td>27</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Z6 Delay prop</td>
<td>13.5</td>
<td>27</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>U Delayed order</td>
<td>9</td>
<td>11.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>E6 Delayed order</td>
<td>2.5</td>
<td>9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>Z6 Delayed order</td>
<td>2.5</td>
<td>9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>U Rescheduling route cons.</td>
<td>2.5</td>
<td>13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>U Rescheduling route var.</td>
<td>0</td>
<td>4.5</td>
<td>13</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 7.2: (Old) Results for different combinations of plans and rescheduling strategies. Input: 20 hours of delay where the arrival order of two trains from the same direction is preserved and there is at least one minute of time between two trains from the same direction.

<table>
<thead>
<tr>
<th>Plan</th>
<th>Strategy</th>
<th>Arrival shifts</th>
<th>Departure shifts</th>
<th># Platform deviations</th>
<th>Route deviations (% of trains)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>U Delay prop</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>E6 Delay prop</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Z6 Delay prop</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>U Delayed order</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>E6 Delayed order</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>Z6 Delayed order</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>U Rescheduling route cons.</td>
<td>3</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>U Rescheduling route var.</td>
<td>1.95</td>
<td>6.9</td>
<td>7.2%</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 7.3: (Has to be extended) Results for different combinations of plans and rescheduling strategies. Input: 10 times 20 hours of delay where the arrival order of two trains from the same direction is preserved and there is at least one minute of time between two trains arriving from the same direction.

In Table 7.2 the results are shown for three different routing plans and three different real-time strategies. It can be seen that the delay propagation strategy resulted in larger delays than the delayed order strategy. As expected, the rescheduling strategy led to the least amount of delay, even if no rerouting was allowed (line 10). From line 11, we can see that allowing rerouting can considerably reduce the amount of delay.

The impact of the routing plan can be seen in lines 1-6. It turns out that plan E6 and Z6 perform in a very similar way and that plan U leads to more delays in combination with both the delay propagation strategy and the delayed order strategy.
7.5 Influence of the distribution of initial delay

Table 7.4: Total number of shifts for 20 hours delayed order strategy. Mean of 100 tests. Also given: percentage of initial indelays greater than 5.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>% &gt; 5</th>
<th>U</th>
<th>Z6</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\exp(n/2), P_{n/4}^{(0.5)}, P_{n/4}^{(0.5)})</td>
<td>0.7</td>
<td>mean: (6.8, 15.3) var: (8.3, 10.6)</td>
<td>(1.8, 8.7) (10.6, 16.4)</td>
</tr>
<tr>
<td>(\exp(n), P_{n/4}^{(0.5)}, P_{n/4}^{(0.5)})</td>
<td>4.4</td>
<td>mean: (10.2, 17.0) var: (14.2, 22.2)</td>
<td>(14.5, 23.1) (207, 215)</td>
</tr>
<tr>
<td>(\exp(n/5), P_{n/4}^{(0.5)}, P_{n/4}^{(0.5)})</td>
<td>9.6</td>
<td>mean: (20.0, 24.8) var: (48.7, 55.1)</td>
<td>(72.5, 77.6) (3258, 3147)</td>
</tr>
</tbody>
</table>

Table 7.4 provides the testing results for two different routing plans. Different distributions of initial arrival delay were used for the different rows. In the second column, the percentage of initial arrival delay is given that is greater than 5 minutes. Furthermore, the mean and variance of the resulting shifts for entrance and exit are given. Interestingly, the routing plan Z6 seems to perform better for smaller delays and the routing plan U seems to perform better for larger delays. The good performance of Z6 for small delays can be explained by the objective function that penalties buffer times of less than 6 minutes. The bad performance of Z6 for large delays is more striking and can be explained by the occurrence of small and large waiting times in the schedule. In Z6 these different trains were combined on one station track, because this leads to fewer small buffer times. However, when a large delay occurs, it is possible that the train that has a very long dwell time (for example 24 minutes) overtakes the train with the short dwell time. Because no alternative routes were allowed, the train with the short dwell time then has to wait until the other train has left the platform. In the case of two trains with a long dwell time, this does not occur so quickly because the departure of a train with a very long dwell time is not often delayed.

7.6 Influence of assumptions in reoptimization strategy

For the rescheduling we should also assume a certain availability of information. We try different assumptions: Availability of all information at all time, Not allowing rerouting /
Allowing rerouting

-testen hoe erg het is om grotere stappen te nemen in rescheduling model.
-testen hoe belangrijk het moment van informatie is.

67
Plan: E6, realistic rescheduling

<table>
<thead>
<tr>
<th>Steps</th>
<th>Knowledge</th>
<th>Arrival shifts</th>
<th>Departure shifts</th>
<th># Platform deviations</th>
<th># Route deviations</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 min</td>
<td>sure before fixed</td>
<td>0</td>
<td>23</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>2 min</td>
<td>fixed before sure</td>
<td>1.5</td>
<td>25</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>0.5 min</td>
<td>fixed before sure</td>
<td>0.5</td>
<td>24</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>0.5 min</td>
<td>sure before fixed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>all knowledge available</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.5: Results for different assumptions in rescheduling. Input: 20 hours of delay where the arrival order of two trains from the same direction is preserved and there is at least one minute of time between two trains from the same direction.

7.7 Quality of SOM plans under wrong assumptions

One of the difficult point in testing the routings found by the stochastic optimization model, is that the assumptions while creating a routing are very similar to the assumptions while testing it. In a way the routing found is made to be optimal for the tests. This means that realistic assumptions in the creation of a routing are in this case very important for its quality in real time. In order to see the influence of the distribution of the delays, we create and test the model with different distributions of delays.

<table>
<thead>
<tr>
<th>Distribution in SOM</th>
<th>Distribution in testing</th>
<th>Arrival shifts</th>
<th>Departure shifts</th>
<th># Platform deviations</th>
<th># Route deviations</th>
</tr>
</thead>
</table>

Table 7.6

-hoe presteert SOM met vertragingen op dezelfde manier getrokken als in optimalisatie en hoe presteert het bij andere vertragingen?
Chapter 8

Conclusions and discussion

The purpose of this thesis was to develop algorithms to obtain robust routing plans for railway stations. Furthermore, real-time rescheduling strategies had to be developed in order to model the performance of the routing plans in practice.

Different methods were implemented to generate a routing plan for a given railway station. In the approach from Section 3.4, robustness objectives were imposed by taking into account the buffer times between events. This method was based on earlier studies by Kroon and Maróti [2008], Caprara et al. [2010] and Dewilde et al. [2014]. In Chapter 6, a Stochastic Optimization Model was proposed that in a way combines the generation of a routing plan with a model for real-time rescheduling for given delays. This approach has not been found in the literature. The idea came from the Stochastic Optimization Model for the generation of cyclic railway timetables by Kroon et al. [2007].

[Description of performance of the different methods for generating routing plans.]

Three different real-time rescheduling strategies were developed and used to test the routing plans. Two simple strategies choose an order for the events based on the planned order or by a first-come-first-served strategy. The more involved re-optimising strategy makes operational decisions within the current possibilities using the available delay information and a rolling horizon. All strategies could handle initial delays upon arrival, halting time and departure. Moreover, for the re-optimising strategy different assumptions about the availability of information were possible.

Even if no deviations from the planned routes were allowed, the re-optimising strategy turned out to perform much better than the two simple strategies. By allowing some deviations from the planned routes, the amount of knock-on delay could be further reduced. The two simple strategies did not perform well in combination with large initial delays. For the delay propagation method, larger initial delays immediately led to very large knock-on delays. For the first-come-first-served strategy, it turned out that large initial delays for a schedule with long dwell times can lead to large knock on delays.
This is especially the case when trains with a short dwell time are planned on the same platform track as trains with a long dwell time.

A drawback of the performed tests, is that the methods were only tested on one relatively small railway station. It is not known if the presented routing algorithms are efficient enough to be able to cope with much larger stations. Moreover, the rescheduling algorithms should be tested on other stations in order to draw more general conclusions. Therefore it is needed that the data about the infrastructure and timetable of the railway stations is available in a way such that it can be easily integrated with the algorithms. In the performed tests, it was assumed that the claim times of the complete routes during arrival or departure were equal for all routes. For larger stations, this assumption is not reasonable. Therefore, data should be available about the claim times of the different track sections of the different routes.

The Stochastic Optimisation model seemed to be promising, but the current implementation could only be applied to 90 hours of initial delays at the same time. This sample is probably not enough to represent the hole distribution of initial delays. It should therefore be tested on a computer with more memory or the implementation should be made more efficient.

The amount of knock-on delay during arrival and departure served as a measure for the quality of the performance. It would be good to also consider the influence of the disturbances out of the station. For example, the consequences of an intercity service leaving the station just behind a regional train should be noticed. Therefore the interaction between trains on the track after the station should be taken into account. For both the generation of a routing plan and for the rescheduling strategies, this could lead to a useful extension.
Appendix A

Solving ILP in CPLEX

[Description of CPLEX branch and bound]

Integer and combinatorial optimization - Nemhauser, George Lann en Wolsey, Laurence A

Theory of linear and integer programming - Schrijver, Alexander
Bibliography


L.G. Kroon, E. Romeijn, and P.J. Zwaneveld. Routing trains through railway stations:


