Early Conflict Detection

Conflict detection in the Crew Schedule of Netherlands Railways
Bachelor Thesis Econometrics & Management Science

02-07-2010

Written by:
Stephen Kil
312179sk

Supervisor:
Dr. D. Huisman

Abstract

The purpose of this thesis is to design and test an algorithm to predicts conflicts early – up to 2 hours and 45 minutes before they occur. The proposed algorithm tries to bound the delay at the end station using two dispatching strategies. The 'fixed order strategy' will not allow trains to pass each other and thus will create the upper bound. The lower bound is constructed through the 'first opportunity strategy' where a delayed trains will pass other trains at the first opportunity. When a conflict is predicted by both bounds, the algorithm will assume it has occured. When only the upper bound predicts a conflict, the algorithm will postpone the overtaking of trains in the lower bound untill both bounds will again predict the same conflict. If, in the lower bound, overtaking can be postponed less than 60%, the conflict is assumed to occur in the near future.

Using this approach 95% of the conflicts can be predicted prior to them occuring. Those conflicts not predicted can generally be solved without rescheduling. If we force the algorithm to predict conflicts up to 30 minutes in advance this percentage drops to between 52% and 78%, depending on the dispatching strategy used. When increasing this timeframe to 60 minutes the algorithm can only predict about 25% of all conflicts, but the accuracy increases. Note that not all delays occur 30 or 60 minutes prior to the conflict.

The increasing accuracy of the algorithm suggests that the earlier a train is delayed the better the outcome will be. Because for dispatchers it is generally the other way around, this algorithm can successfully used to aid dispatchers in their decisions.

Contents

1 Introduction

Netherlands Railways (NS) is the principal passenger railway operator in the Netherlands. Currently over one million passengers travel by train every day. To provide a high level of service to its passengers, it is of great importance for the NS to adequately schedule rolling stock and crew.

To maintain a high level of service when unforeseen events occur, minimizing the total delay of passengers, rescheduling is required and has to be done accurately. These events include, but are not limited to, broken down rolling stock, damaged crossovers and accidents involving other traffic. As a result of these disruptions both rolling stock and crew will arrive late for their next task. If the rolling stock and crew is not reallocated, all tasks associated with the delayed crew or rolling stock will be delayed as well.

Reallocation of tasks is a time critical process. Most of these conflicts, where crew or rolling stock will arrive late, are known only limited time in advance and need to be solved before the new tasks are supposed to start. More important only limited information is available at the time of rescheduling. For example, the actual delay at start of the next task is still unknown. Without this information it is hard to make decisions which tasks to reallocate or not. While rolling stock can be changed quite easily, albeit some work to find the right rolling stock, the crew might complain about unnecessary changes to their schedule. To avoid these complaints NS does not want to reschedule unnecessary. At the same time, to avoid a drop in service level, the next tasks should start on time if possible.

Currently these decissions, which tasks to reallocate, are made by hand. A controller reallocates to what he thinks to be the best solution. While in general this works quite well, the accuracy of the solution clearly depends on the experience of the controller. Some tasks might never get rescheduled, while they should, where others do, while they should not. As a result of these mistakes, the severity of a problem might increase.

In the past years NS has developed an algorithm, as described in Potthoff et al. (2010), to reschedule crew automatically when part of the infrastructure is blocked. This will allow NS to react to major disruptions faster according to Huisman and Potthoff (2009).

When part of the infrastructure is unavailable there is sure to be a problem, where for delays only it is not. For a similar algorithm to be implemented for these delays, a more precise conflict detection has to be developed.

This thesis will focus on developing and testing an algorithm to accurately predict both the delays and conflicts as early as possible.

This thesis will focus on developing and testing an algorithm to accurately predict

both the delays and conflicts. Part of the algorithm has already been descibed in Kil and Naber (2010), but this implementation has never been tested in a simulated enviroment.

In Section 2 we will provide some background information on the Dutch railway network. Section 3 will describe the algorithm as it is now and in Section 4 we will provide a simulation model to check the algorithm. Section 5 will provide some in depth information on how the simulation generates arrival times and dispatching order and in Section 6 we will do a general overview on the data. The results will be presented in Section 7. Section 8 will contain some conclusions.

2 Background Information

Providing a high level of service to its passengers NS has to keep close control on its train operation. This section provides some background information on the idea behind the timetable and how NS strives to maintain it.

2.1 Dutch Railway Network

The Dutch railway network is one of the busiest railway networks in the world. Currently there are over a dozen railway operators active on the network all having their own trains. This results a complex schedule of several thousand trains on the network. The Dutch Government keeps close control over all these companies and their schedules through ProRail. This company is responsible for maintaining, exploiting and expanding the railway infrastructure.

2.1.1 Line System

Having such an amount of trains on the network can be quite confusing to passengers. To simplify the timetable many railway operators, including NS, use a cyclic timetable. Such a timetable has a fixed cycle of trains every 15, 30 or 60 minutes for a line. A line is defined by Huisman et al. (2005) as *'a direct railway connetion between two end stations that is operated with a certain frequency and with a certain train type'*. Figure 1 shows an example of such a line system.

Figure 1: Example of the Dutch Line System around Amsterdam

NS currently operates using two types of trains for passenger transport. Intercity trains only stops at larger stations on the network – i.e. Rotterdam – and will pass

all others. Therefore, these trains are most suitable for long distance passenger transport. Sprinters, regional trains, stop at all stations they pass and thus are more suitable for short distance – regional – transportation.

Choosing the length of a line is a difficult decision and very important to both the railway operator and the passengers. The longer a line is, the easier it is for passengers to travel long distances – less transfers. However, a drawback of long lines is that delays will spread over a wider area and rolling stock management is inefficient. For shorter lines rolling stock management is easier and delays will be contained. Passengers will however be required to make more transfers to reach their destination and might choose for alternate transportation.

Once the length of all lines have been determined, the timetable can be constructed.

2.1.2 Rolling Stock Schedule

While the timetable is most likely the only schedule the passengers will ever see, railway operators also have to build a rolling stock schedule. This rolling stock schedule defines the distribution of the rolling stock over the lines and railway network. The process of building the rolling stock schedule is generally devided into several phases.

Huisman et al. (2005) distinguish four planning phases; That is strategic, tactical, operational and short-term planning. All long term decisions, like the acquisition of new rolling stock, are considered to be *strategic*. Once a year a railway operator has to assign the rolling stock to the different lines of the network. This phase is called the *tactical* planning phase. Finding the rolling stock schedule with low operational costs and a high level of service for customes is generally done in the *operational* planning phase.

Surely, each train has to have at least one rolling stock unit, but in many cases this will not be enough. The amount of units required depends on the expected number of passengers for that train and line. For a railway operator it is thus important to know what the demand will be. Too little seats will eventually force passengers to take alternate transportation, but too much seats will cost too much money. Combining rolling stock units can be done to match the demand for a train. However, the right rolling stock units have to be available at the station or shunting yard. Not all rolling stock units can be combined, they need to be of the same type – but may be of different subtypes, like i.e. different years. Furthermore, when rolling stock units are combined they all share the same destination. So at that station all these units need to get a new destination – they do not necessarily get the same destination. Also rolling stock units will need to be serviced every once in a while. The rolling stock schedule thus has to allow some units to be unavailable at times – if not, the schedule will become infeasible even if only one rolling stock unit is unavailable.

Once the rolling stock schedule has been constructed, a railway operator can use it to carry out the timetable throughout the year. However, due to unforeseen disturbances, the schedule might have to be adjusted some days. These adjustments are considered to be part of the *short-term* planning. We will discuss these adjustments further in Section 2.2.

2.1.3 Crew Schedule

When the timetable and rolling stock schedule is complete, a railway operator is left with a set of tasks which have to be performed, for which a crew schedule has to be constructed. According to Hartog et al. (2009) rostering crew can be done in several ways, but most railway operators use a cyclic crew schedule. Creating this crew schedule is not as easy as it sounds at first, as there are several important constraints which have to be met in order for the schedule to be feasible.

Starting, a crew member has its own home depot, where his duty[1] has to start and end within a reasonable timeframe. This means that at each home depot at most the same number of duties can start as there are crew members available there. If a crew members first assignment is not at his home depot, he will have to be transported to the starting location first. The same goes if the last assignment will not end at the home depot. All non-required crew members will be seen as passengers, as they are not working[2].

Trains require to have at least one driver and one conductor, but some trains might require more than one conductor.

Furthermore, crew members should not be working more than 5.5 hours without a break. A break should be at least 30 minutes and always be at a relief location where crew has access to a cafeteria. Breaks are to preferred to be in the middle of a crew members duty.

Also, when a crew member is changing tasks, some time might be required for him to get to the right departure platform. A task will always end at a relief location, but a train might continue further. So it is possible for the crew members next task to be on the same train. If he will continue on the same train, he is already at the correct platform and thus no extra time is required. However it might be possible he has to get from platform 1 to 14, at the other end of the station, which requires extra time to walk to that platform. Furthermore, it might be possible for a crew member to continue on the same rolling stock but in reversed direction. The crew member is then already at the right platform, but he will require some time to walk to the other end of the train.

Once the crew schedule, in which all the above constraints are met, is found and

[1] A duty the set of tasks a crew member has to perform, plus the time it takes to travel from and to his home depot.

[2] Note that this is different from a break. A break has to be at a relief location and not while on a train.

all duties are fulfilled, a railway operator can start transporting passengers.

2.2 Daily Management

In 2007 NS introduced a new 60 minute cyclic timetable, which is still being used – minus a few adjustments. By this timetable, NS was able to able to set a record high punctuality of 87% in 2007 – Figure 2 shows the punctuality of the last 10 years at 3 minutes bound. However a good timetable alone is not enough. A railway operator has to keep an eye on possible conflicts in crew or rolling stock schedules.

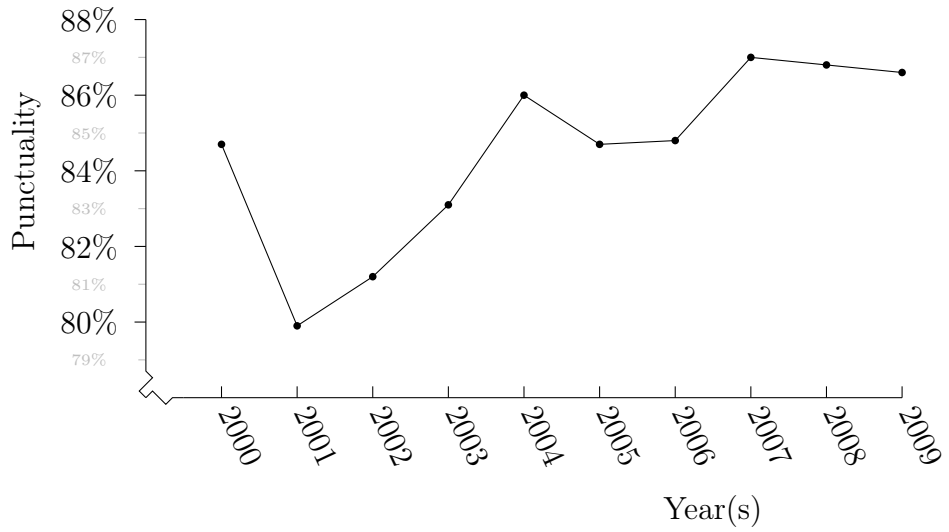

Figure 2: 3 Minute punctuality data for NS from 2000 to 2009

2.2.1 Dispatching

The order in which trains depart from a station has been generated when the timetable was constructed. However this departure order was constructed to be valid if all trains are on time, or at least close to be on time. When a delayed train enters a station, the best dispatching order might be different from the one in the schedule. NS monitors the trains at 5 locations in The Netherlands. At these locations, dispatchers are closely monitoring all delayes of the trains. Each dispatcher is responsible for a portion of the Dutch railway network. By changing the train dispatching order, they try to minimize delays and conflicts.

2.2.2 Conflicts & Rescheduling

At some point, when delays are large enough, conflicts are unavoidable. A conflict will occur if one of the following 4 events occur:

1. **Rolling stock or crew will miss its connection;**
   Occurs when: Rolling stock or a crew member is scheduled to depart from the next station, but due to some delay is not yet ready.
2. **Break will be too short;**
   Occurs when: A crew member has a scheduled break which, due to delays, will be shorter than 30 minutes.
3. **Crew member has to work more than 5.5 hours straight;**
   Occurs when: A crew member is working longer than 5.5 hours without any kind of break. Mostly this is due to this crew member has missed a connection earlier.
4. **Crew member has to work more than 1 hour overtime;**
   Occurs when: A crew member arrives at his home depot more than 1 hour later than scheduled.

When a conflict occurs, a dispatcher[3] has several options which can be used to avoid further delays in next tasks. At most relief locations – the location where crew tasks end – some crew members are on 'standby', which means they can be used to fulfill the tasks now conflicted. However, these 'standby' crew members have to be back at the relief location before their 'standby' duties end.

When no such option can be found, a dispatcher can reschedule tasks, by assigning them to different crew members. This rescheduling is a time critical process, as it has several time constraints. The most obvious is that the conflict has to be resolved before it actually occurs, but that is not the only constraint. If it involves multiple crew members, the unforeseening has to be complete before any of these crew members start their next task or leave the relief location. Also, when reassigning tasks, a rescheduler has to take into account the extra time the crew member has to work to fulfill these tasks, because no crew member may work more than 5.5 hours straight – conflict (3) – or more than 1 hour overtime – conflict (4). Finally, not all crew members can fulfill all tasks, as some have no experience driving a certain type of rolling stock or have no knowledge about that part of the railway network.

If the dispatcher is unable to find the required crew members to reassign tasks to, there are two more options a dispatcher has. Both these options have negative influence for travellers so should be avoided if possible. The first option is to ignore the conflict, this will automatically delay the next task untill the crew and rolling stock is available. The second option is to cancel the next task, which results in a cancelled train. While both these options solve the conflict at hand, the should be used with care as they might create a new conflict in the future.

In general it is true that the sooner a conflict is detected, the sooner dispatchers can solve it and the more options there are available.

[3]These decisions are made by had by the dispatcher. Some of these decisions might be computerized in the future. Budai et al. (2007) describes the *Rolling Stock Balancing Problem* which can be used to reschedule rolling stock and Potthoff et al. (2010) a tool to reschedule crew.

3 Algorithm

Recently some research has been done on detecting conflicts in the crew schedule. According Kil and Naber (2010) the propagation of delays can be written as a dynamic programming model. The idea is to create an upper and lower bound for the delay and predict conflicts with both of them. First, in Section 3.1, we will provide the assumptions which they used as basis for this algorithm and in Section 3.2 we will describe it further.

3.1 Assumptions

For the algorithm to work as described in Section 3.2, we have to make several assumptions about the railway network. Each station will be refered to as a node and the connection between two consecutive stations will be called an edge.

3.1.1 Node(s)

1. For all nodes the ability for trains to pass is given.

3.1.2 Edge(s)

2. An edge consists of two tracks – one in each direction.
3. For safety a minimum time between trains must be adhered at all time, this time is given.

3.1.3 Train(s)

4. No train will be delayed unexpected.
5. The train's scheduled driving time contains a safety margin of 5 percent.
6. The trains scheduled dwell times contain a safety margin of 5 percent.
7. All trains will depart as scheduled – no trains are cancelled.
8. A train will always reach its intended destination – this destination will never be changed.

3.1.4 Crew

9. A crew member requires at least 3 minutes to be ready for his next task if this task is not on the same train he is on now.

3.1.5 Dispatching Strategies

10. If not otherwise stated, trains will depart 'first in, first out'.

3.2 Basic Algorithm

Detecting conflicts on its own is not a problem, but merely checking if the crew arrival time is later than the scheduled departure time. The real challenge is estimating the delays at the relief locations – the location(s) where a crew member's task will end. There are many decisions which affect a train's delay. As mentioned in Section 2.2 a dispatcher can use the dispatching order to try and minimize the delays.

Because these dispatching strategies vary between dispatchers and may even vary in time, it is hard to include these directly. However, two dispatching strategies are consistent at all time. The first is the 'fixed order strategy', where no trains – unless scheduled – are allowed to pass each other, and the second is 'first opportunity passing', where a delayed train will pass at the first opportunity available.

3.2.1 Upper Bound

The 'fixed order strategy' is the worst case scenario as a delayed train will be stuck behind all possible trains on the track. If we estimate the delay in this worst case scenario, we get an upper bound for the current train's arrival times.

If we define the set of stations S as all stations that delayed train (DT) will cross along its route, we can write the prediction for the current delay at station s (cd_s) as equation (1).

$$cd_s = max_{t \in T} \left\{ \begin{array}{l} cd_{s_{-1}} - x_d * (bt_{DT,s} - at_{DT,s}) - x_r * \left(at_{DT,s} - bt_{DT,s_{-1}}\right), \\ I_{\left(bt_{DT,s_{-1}} + cd_{s_{-1}} > bt_{t,s_{-1}}\right)} * (bt_{t,s} - bt_{DT,s}) \end{array} \right\} \quad (1)$$
$$\forall s \in S$$

At every station there are two options, either there is no train blocking DT or there is. For every other train $I_{(bt_{DT,s_{-1}} + cd_{s_{-1}} > bt_{t,s_{-1}})}$ is 1 if this other train is in front of DT. In equation (1) $at_{t,s}$ is the arrival time and $bt_{t,s}$ the departure time for t at station s. The first argument of the $max()$ in equation (1) will use the running time buffer x_r and the dwell time buffer x_d to decrease DTs delay. The second argument is the extra delay DT gains when a train, which is in front of DT, departs from s later than DT itself – note that this is 0 if a train is not in front of DT, as $I_{(bt_{DT,s_{-1}} + cd_{s_{-1}} > bt_{t,s_{-1}})}$ will be 0. The final delay at station s is equal to the maximum of all delays.

3.2.2 Lower Bound

The lower bound will use 'first opportunity passing' which will force DT to pass all other trains at the first station possible. Knowing this, we do not have to check for other trains at stations where passing is possible as it will always pass other trains here and thus does not gain any extra delay. If we introduce O as the set of stations where passing is possible and allowed[4] we can use the a slightly altered version of equation (1).

$$cd_s = max_{t \in T} \left\{ \begin{array}{l} cd_{s_{-1}} - x_d * (bt_{DT,s} - at_{DT,s}) - x_r * \left(at_{DT,s} - bt_{DT,s_{-1}}\right), \\ I_{\left(bt_{DT,s_{-1}} + cd_{s_{-1}} > bt_{t,s_{-1}}\right)} * (bt_{t,s} - bt_{DT,s}) \end{array} \right\} \quad (2)$$
$$\forall s \in S \backslash O$$

$$cd_s = cd_{s_{-1}} - x_d * (bt_{DT,s} - at_{DT,s}) - x_r * (at_{DT,s} - bt_{DT,s_{-1}}) \quad (3)$$
$$\forall s \in O$$

The use of equation (2) is exactly the same as equation (1), though it will not be executed for all stations in O. If a station is in O equation (3) is used, which only decreases the delay by the driving time and dwell time buffer. This effectively puts the DT in front of the other trains currently at station s.

3.3 Method 1

For 'Method 1' prediction, the upper and lower bound will be constructed by equation (1) and equations (2-3), respectively.

The algorithm will start by taking a 'snapshot' of reality, which include the current location for all trains, their delays and their schedule. The data for stations will not be altered in any way, so passing is allowed for all stations where passing is possible. For both, the upper and lower bound delay the algorithm will check which conflicts occur.

If a conflict is predicted with both the upper and lower bound delay, 'Method 1' prediction will assume this conflict to occur in the near future. The same holds if no conflict is predicted with neither the upper nor lower bound delay, the algorithm then will assume no conflict will occur. When a conflict is predicted by only one of the two bounds – note that this is always the upper bound – the algorithm will be forced into 'Method 2' prediction for some extra information.

[4]When constructing the lower bound in 'Method 1' – Section 3.3 – prediction overtaking is allowed for all stations where passing is possible, but when using 'Method 2' – Section 3.4 – prediction this will change

3.4 Method 2

'Method 2' prediction will only be executed if required, when the 'Method 1' prediction is inconclusive, as it is fairly time consuming. This method will alter the available data in such way that the lower bound will increase graduately in each iteration.

Each iteration passing will be disallowed at the station where the passing occured now, this station will thus be removed from O – note that passing will not be allowed there at any future iteration. The amount of iterations depends on the amount of passing points along the track. The more passing points there are, the more accurate this method should be, but also the longer it takes to come to a conclusion.

Because we remove the station from the set O, which was defined as all stations along the route at which overtaking is possible and allowed, we can use the same equations (2-3) as in the lower bound situation. Using this new lower bound, the algorithm will check for both bounds if a conflict will occur. The algorithm will continue to remove stations as long as the result of both bounds is still inconclusive or when all passing points are removed. Note that if all passing points are removed the estimated delay is the same as in the upper bound and a conflict has to occur in both bounds – if not, 'Method 2' prediction should not be iterating at all.

When passing can be postponed more than 60%, meaning that the result is inconclusive more than 60%, of the time 'Method 2' prediction will assume the conflict will not happen as a dispatcher has enough time to change the dispatching order. In all other situations, 'Method 2' prediction will predict the conflict to occur in the near future.

4 Simulation Model

Validating the model proposed by Kil and Naber (2010) – Section 3 – can be done by simulation. According to Veelenturf (2008) a similar[5] simulation model can be written as a *mixed integer linear programming problem* and does require some assumptions. While our model will be based on this simulation model, some adaptations will have to be made in order to fit our needs. As a result not all assumptions of the original simulation model remain valid. So in Section 4.1 we will present the required assumptions to make the new simulation model. Next, in Section 4.2, we will provide the model and the implementation.

4.1 Assumptions

The following assumptions about the railway network, rolling stock and crew are required to build the basic simulation model. Some of these assumptions simplify the real situation a little, where others simply state reality. Additionally, some of these assumptions were also used when creating the algorithm in Section 3.

4.1.1 Railway Network

I Like Veelenturf (2008), we will assume the railway network consists of a set of nodes and edges. A node, also refered to as a *measure point*, will be either a station or a junction. The edges represent the lines connecting all nodes.

4.1.2 Node(s)

II All nodes have infinite tracks available to store rolling stock.

III As result of assumption (II) trains are able to pass each other in the nodes.

IV When a node is a relief location, a location where crew can switch tasks, the node can accomodate an infinite number of crew members.

V When a node is a relief location, an infinite number of reserve crew members are available.

4.1.3 Edge(s)

VI An edge, or line, consists of two tracks - one in each direction.

[5] Veelenturf (2008) describes a model used for simulating the effect of dispatching rules on rolling stock and ignores conflicts. We will be focussing on conflicts in the crew schedule and thus remove the rolling stock from the simulation.

VII For safety a minimum time between trains must be adhered at all time, this will be called the headway time. No second train may pass a certain point if the headway time has not passed yet. This headway time might vary per edge.

4.1.4 Train(s)

VIII A train may never leave a node before the scheduled departure time, except when it does not stop at the corresponding station.

IX A train will always depart from the first station as scheduled.

X A train cannot depart a node if not all crew members and rolling stock units are available at the node.

XI The minimum technical driving time is the minimum time it takes for a train to drive from the current node to the next. A train can never reach the next node faster than the minimum technical driving time.

XII When a train reverses direction, even when keeping the current rolling stock unit(s), this will be seen as a new train and not just a new number.

XIII The dwell time of a train is the time it takes to pick up passengers. When a train does not stop at a node, the dwell time is 0 minutes.

4.1.5 Crew

XIV A crew member's task will only end at a relief location.

XV When a crew member is late for his next task, all remaining tasks are assigned to a new crew member. The crew member's duty will thus end after the current task.

XVI When a crew member has to switch trains a certain amount of time is required for him to get to the right platform. Also by assumption (XII) the same time is taken into account when a crew member reverses direction with the train.

4.2 Model

There are many techniques we can use to simulate railway operation, all having their own advantages and disadvantages. Railway operation is a complex process, but can be easily decomposed as a series of events. Using these events an event based simulation technique can be used. Event based simulation is, as its name suggest, based on the principle of a series of events succeeding each other in time – see Figure 3. The events in the simulation are dynamic, meaning that new events can be generated as the current ones are being triggered.

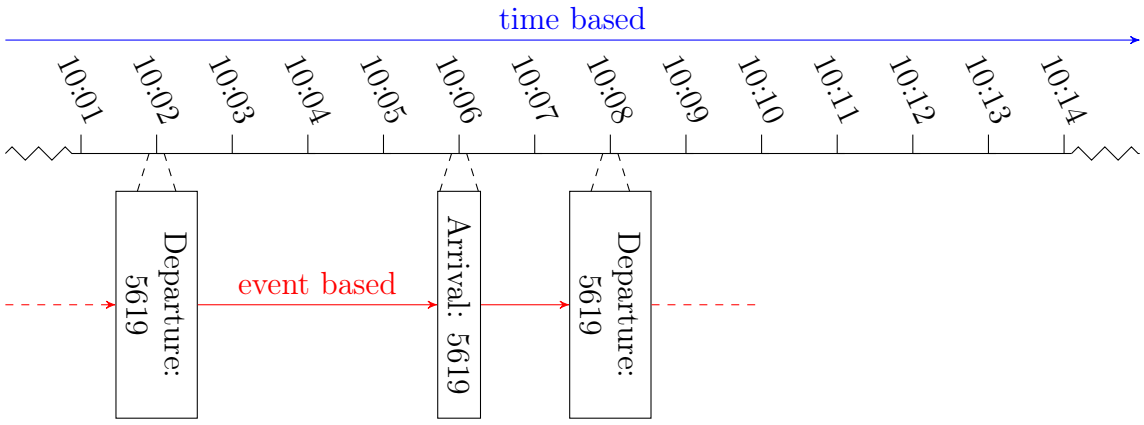

Figure 3: Time Based vs. Event Based Simulation

The advantage of event based simulation is the speed at which it can be executed. As in Figure 3 there are only 3 events from 10:01 till 10:14. So an event based simulation can do this whole timespan in 3 iterations, where for example a 'real-time' simulation does 14 iterations – if set to a minute stepsize. This advantage is at the same time also its downside. Delays are not updated in real-time, but rather at some key events, nor can the algorithm be excuted at all times.

4.2.1 Events

Decomposing the railway operation will result in 2 basic events: arrival events and departure events. For example, an arrival event is triggered when a train arrives at a station and a departure event is triggered when a train leaves again. Besides trains, crew members and rolling stock have the same 2 events. Our simulation will not include the rolling stock events, as we are only interested in the conflicts in the crew schedule.

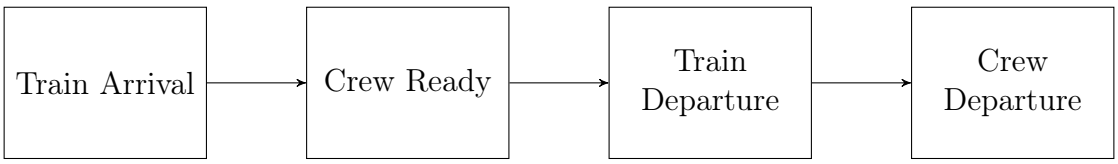

Figure 4: Order in which events are triggered for all trains

As a result the simulation model will contain 4 events in total. At any given station, except for the first and last station a train will pass, the following 4 events will occur in the order seen in Figure 4. The *crew ready* event is a variation of an arrival event and will include the extra time required to get to the right platform. Note that the train and crew departure events will always be triggered at the same time and thus there is an option of integrating them. However we choose, for ease of reading, not to do this.

The train arrival and train departure event will always be about the same train – i.e. the 530 intercity service – but the crew arrival and departure events can be about different crew members. As crew can change trains at relief locations, it is possible for a train to continue having a different crew member operating it then before.

4.2.2 Event: Train Arrival

When a train arrives at a station it is assigned a platform to pick up its passengers and it will release its crew, but due to assumption II the platform assignment can be left out, as there will always be at least one platform free.

If a train is transporting a crew member which has to switch trains at the current station, the train has to release its crew member. To do this, the current event will generate a *crew ready event*. This event might be triggered straight away – the crew member is available at the same time the train arrives – but it might also be triggered at a later time. The time between the train arrival and the crew ready event is the time required to get to the right platform.

4.2.3 Event: Crew Ready

As soon as this event is triggered, a crew member is available again and can be put onto its next task without conflicts. This event will be automatically generated when a train arrives, but the time at which it executes depends on several factors.

Getting to the platform: When a crew member's current task ends and its next task is not on the same train as the current, he will have to walk to the right platform. We assumed – assumption 9 in Section 3.1 – that it will take at least 3 minutes to walk to the right platform. So if the crew member has to walk, this event will be triggered 3 minutes later then the train arrival event which generated this event.

Break: If a crew member has a break at the current station, it has to have a free period of at least 30 minutes. This will result in this event being triggered 30 minutes after the train arrives. It is assumed that if a crew member has a break, it can use the last 3 minutes of its break to walk to the right platform.

If none of the above conditions are true – i.e. the current task is the last one or the crew member will continue on the same train – the crew ready event is triggered at the same time as the train arrival event.

4.2.4 Event: Train Departure

This event, together with the crew departure event, is one of the most important events in this simulation. During this event the simulation will check whether or not

the train may depart, according to the dispatching rules – see Section 5.2. But also it will generate the arrival times at the next station – see Section 5.1 – and trigger the forecast of delays.

When this event is triggered, the first it will do, is check whether it has permission to depart. Section 5.2 provides several dispatching strategies which can be used to check departure permission. If the current train does not have departure permission, the current departure event is cancelled and postponed to a later time.

If a train does get permission it will generate two new events, an arrival event and a departure event from the next station. The time required to get to the next station is calculated as in Section 5.1 and the arrival event is executed at this time.

The departure event is scheduled to be executed as soon as the dwell time is over or, due to assumption (VIII), at the scheduled time to depart. If a train is delayed the dwell time is decreased by the safety margin.

After generating these two events, the prediction algorithm will be run. This algorithm will take a 'snap-shot' of the current state of the simulation and calculates the delays. This means all delays currently in the simulation will be used in the algorithm, but as said in Section 3 only the delay of the current train will be changed and forecasted. All forecasted conflicts will be added to a list and stored for later use.

When both these events are generated and the prediction algorithm has run, the departure event will generate a crew departe event.

4.2.5 Event: Crew Departure

This event will always be triggered at the same time the train departure event is triggered and will check whether or not the crew member is available. When a crew member is available this event will automatically finish, nothing will be done.

When a crew member is not available however, this event will start logging the conflict detected. First it will check at which train the required crew member is now, and what its current delay is. Next it will check whether this conflict already exists in the list containing forecasted conflicts and, if found, it will combine the current conflict result with the forecasted one. Finally, it will create a new crew member containing all remaining tasks of the crew member still traveling. Creating this new crew member suggests a dispatcher has solved the conflict.

4.3 Mixed Integer Linear Programming Problem

As we mentioned earlier this simulation can be written as a mixed integer programming problem. However this problem will only provide us with the solution on how the trains should have run after the day has passed. While this has its use, it is not the one we intend. We want to predict the conflicts before they happen, with the

same limited information a dispatcher have, rather than find the optimal solution at the end of the day.

Though, for sake of completion we decided to provide and describe the mixed integer programming problem of our simulation model. The model and its description can be found in appendix A.

5 Method

In Section 4 we described the simulation model and mentioned the generation of arrival times, as well as the order in which trains depart the station, but never explained either one of them. This section will provide a more in depth explanation of how we simulate both.

5.1 Realized Arrival Times

Driving from one station to another will require at least a fixed amount of time $m_{t,s,s+1}$ – the minimal technical driving time. In the optimal situation all trains will run for exactly that amount of time. However due to unforeseen circumstances trains might require a little longer to reach the next station. Veelenturf (2008) suggests that modelling these disturbances can be done by an exponential distribution.

Generating realized arrival time $v_{t,s-1,s}$ can thus be done by drawing $\xi_{t,s,s+1}$ from a exponential distribution, which can be combined with the minimal technical driving $m_{t,s,s+1}$ time – equation (4) – to estimate the arrival at the next station – a value representing contraint (9).

Using assumption (5) in Section 3.1 we can rewrite the minimum technical driving time as a percentage of the scheduled running time. Defining this percentage as x – which is assumed to be equal for the complete schedule – and the scheduled running time for train t from station $s-1$ to s as $n_{t,s-1,s}$, the minimum technical driving time can be written as $n_{t,s-1,s}/x$ – equation (5).

$$v_{t,s-1,s} = m_{t,s-1,s} + \xi_{t,s-1,s} \tag{4}$$

$$= \frac{n_{t,s-1,s}}{1+x} + \xi_{t,s-1,s} \qquad \begin{array}{l} \xi \sim \exp(\mu) \\ \mu = y\left(\frac{n_{t,s-1,s}}{1+x}\right) \end{array} \tag{5}$$

Using equation (5) the realized arrival time can be later then the scheduled arrival time and thus the train can be delayed. These delays can be used to check the algorithm proposed in Section 3. However on average the trains should arrive on time and thus the expected value – equation (6) – should be less than or equal to the scheduled arrival time.

$$\begin{aligned} E\left[v_{t,s-1,s}\right] &= E\left[\frac{n_{t,s-1,s}}{1+x} + \xi_{t,s-1,s}\right] \\ &= \frac{n_{t,s-1,s}}{1+x} + E\left[\xi_{t,s-1,s}\right] \\ &= \frac{n_{t,s-1,s}}{1+x} + y\left(\frac{n_{t,s-1,s}}{1+x}\right) \\ &= \frac{n_{t,s-1,s}(1+y)}{1+x} \end{aligned} \tag{6}$$

Equation (6) shows that for x is y the expected value will be $n_{t,s-1,s}$ as both will cancel eachother out. In general the expected value will be less than or equal to $n_{t,s-1,s}$ as long as $x \geq y$. According to Veelenturf (2008) if x is y about 63.2 per cent of the trains will arrive on time. However, in reality, for NS between 80% and 90% of the trains arrive on time. This suggests that y should be a fraction lower than x.

By simulating values for equation (5) we can show that if we choose $y = \frac{x}{2}$ about 87% of the trains will arrive on time. Note that, while this is more consistent with reality, it will also affect the size of the delays. Lowering the value for y has to be done carefull.

5.2 Realized Departure Ordering

As we discussed in Section 2, dispatchers decide the order in which trains leave a station by themselves. They use a set of rules to decide in which order a train should leave the station. Some of these rules are learned, where others are purely based on previous experience. Modelling these choices we will introduce several rules which can be used to dispatch trains. All these rules will adhere to the passing points we have gathered, if overtaking is not allowed at a station the order will be fixed to 'first in, first out'.

5.2.1 Fixed Order

Dispatching Rule (I): No trains will be allowed to pass if it was not scheduled.

To create the upper bound, the algorithm assumes a 'first in, first out' dispatching strategy. Because trains cannot pass each other on the edge(s) – assumption (VI) – this will fix the train order completely.

As said in Section 3, one of the assumptions is that other delays will not increase but only decline. This could be true, depending on the value for y in Section 5.1, but it is unlikely. Therefore this dispatching strategy is good to see whether or not the upper bound solution is close enough for an accurate prediction of these conflicts.

It is important to know that the algorithm has an exception build in for scheduled overtakes. When a train is scheduled to pass another train at a certain station, it will always do so if the two trains are at that station at the same time. This exception will also be build in into the dispatching strategy in exactly the same way.

5.2.2 Conflict Minimization

Dispatching Rule (II): Try to avoid conflicts as much as possible, even if this means delaying more trains.

The order in which trains depart from a station in the original schedule, are so that no tasks will conflict. So if no trains are delayed, reordering is not required. However if a delayed train is involved, the concept changes.

When a delayed train is involved in the dispatching order, there are several situations possible. For example, not allowing the delayed train to pass a slower train will increase its delay. Also allowing it to pass will not necessarily mean the delay will decrease – it might be caught behind a different train.

The idea behind the 'conflict minimization' strategy is to avoid conflicts as much as possible. This might, for example, increase the total amount of trains delayed, but also decrease the average delay of all trains together. One might say that the same total delay is now distributed over a larger amount of trains. Minimizing conflicts, trains are allowed to pass if and only if the delay might decrease enough, so that no conflict occurs. If a conflict occurs regardless of the order, the delayed train will always depart later than a non-delayed train.

Deciding the order in which the trains will depart from station s, using this idea, requires multiple steps. The first step is to check at which station $s+n$ the current crew members task will end. Then, we can predict the arrival time of that crew member at that station. Let $a^*_{t,s}$ be the scheduled arrival, $d^*_{t,s}$ the scheduled departure and $d_{t,s}$ the real departure time of train t at station s. We can then write the predicted arrival time $a^p_{c,s+n}$ of crew member c at station $s+n$ as equation (7).

$$a^p_{c,s+n} = d_{t,s} + \left(\frac{a^*_{t,s+n} - d^*_{t,s}}{1+x} \right) \tag{7}$$

The current departure time $d_{t,s}$ combined with the minimum technical driving time $\frac{a^*_{t,s+n} - d^*_{t,s}}{1+x}$ to station $s+n$ will give the minimum delay possible at the end station[6]. We can now use this approximation for the delay at the end station to see if a conflict might occur.

Assuming that this train is the only delayed train involved in the decision process, this train is allowed to pass if no conflict occurs at station $s+n$ using $a^p_{c,s+n}$ as arrival time. If, again using the predicted arrival time, a conflict will still occur, the train is not allowed to pass at this station – and most likely no station in the future.

The real challange is deciding the order if more than 1 delayed train is involved in the process. We now have to calculate the predicted arrival time for all trains involved in the process. Assuming that only 1 train can avoid a conflict, that train will leave the station first. When more than 1 train can avoid its conflict, the trains will leave the station in order of train type. This means that intercity services will always depart before the slower regional – Sprinter – trains.

[6]This is an approximation, as the train might have to stop at stations along the way to pick up passengers. This dwell time might have a different buffer than the driving time, but will be ignored by this rule.

5.2.3 Binomial Distribution

Dispatching Rule (III): Randomly order trains using a binomial distribution.

Though dispatching is usually done quite effectively, it remains a human decision process. This implies that errors can be made during the process. This dispatching rule is based on the idea that dispatchers are completely incapable of assessing the situation correctly and thus flip a coin to decide which train goes first. This dispatching strategy should work 'bad' at best.

Flipping a coin is a binomial process, so this dispatching rule will decide the order by drawing values from this binomial distribution. The chance on succes p can be varied if neccesary, but a $50 - 50$ chance will be used for now.

5.2.4 Fixed Order & Binomial

Dispatching Rule (IV): No trains will be allowed to pass unscheduled, unless a random event occurs.

Dispatching rule (5.2.1) will fix the order in which trains depart completely. However there might be random events that change the order regardless of the dispatching strategy chosen. This dispatching rule assumes that the order will be fixed 50% of the times and the other 50% will be random – binomial, dispatching rule (5.2.3).

Note that even if the random event occurs, the outcome of the binomial process might be the same as if it has not occured.

5.2.5 Remaining Travel Time & Binomial

Dispatching Rule (V): Try to avoid conflicts at all times, but only if the situation is assessed correct.

Assuming that dispatchers are completely incapable of assessing a situation is rather harsh. Better is it to assume this will only be true in some cases, only part of the time. This dispatching strategy will assume that approximately 50% of the situations is assed correct, the other 50% is again a random process. In situations which are assed correct, trains will be dispatched using the 'Conflict Minimization' dispatching rule – rule (5.2.2). The other situations will be a random binomial process like dispatching rule (5.2.3).

6 Data

In this section we want to take a close look at the data provided to us by NS. The data consists of a crew schedule for a portion of the network on an unspecified Friday in 2009 and the corresponding timetable for the same month. We further had to gather information about that part of the railway network, specifically the passing points on the track. No data about rolling stock is available, but is not required to run the intended simulation.

6.1 Line & Stations

All duties in the crew schedule take place on the track from The Hague Central Station (GVC) to Groningen Central Station (GN). We asssumed that every edge only consists of two tracks, one in each direction, and for most part this is consistent with reality. We also assumed that passing on these edges is impossible, so we will only include stations in the list of passing points. This list will be the same for both the algorithm and the simulation. Of the 38 stations, there are 13 where overtaking is an option.

6.2 Timetable

NS provided us with the timetable for the whole country, but we will be using parts of this data. From all the data we have selected the 283 trains running on the track between GVC and GN. A little more than half of these trains are intercity services, the others are regional – Sprinter – trains.

For all the 283 trains we selected the arrival and departure data for a Friday, as the crew schedule was said to be for that day only. The timetable specifies the arrival and departure for all stations[7] a train will cross, even if it will not stop at that station – the arrival and departure will then equal.

We also had to select all trains that, according to the timetable, would pass each other during a day. These trains and the station will be added to the exception list, that if no trains are delayed their dispatching order at that station will be correct. This has little or no effect on the dispatching strategy used if one of these trains is delayed, as they then will either not cross paths at all or at another station.

6.3 Crew Schedule

The crew schedule specifies duties for 72 crew members, which in total have to fulfill 688 tasks. An example of a duty can be found in Table 1. In the simulation we

[7]There are some exceptions for intercity services, but we were able to generate their arrival and departure times using the known travel times of other trains.

will use the combination of base, which is the crew members home depot, and the duty number, which is unique, as the actual name for a crew member. Each crew member has 6 to 16 tasks and his duty is at most 4.5 hours long.

Task	Index	From	To	Start	End	Break	Train	Passenger
1	509	Amf	Ut	19:54	20:09	*	1768	
2	545	Ut	Amf	20:51	21:05		1775	
3	575	Amf	Zl	21:43	22:38		5677	
4	617	Zl	Amf	22:49	23:44		5684	
5	648	Amf	Ut	23:45	00:07		5684	
6	1196	Ut	Amf	00:21	00:42		589	*

Table 1: Example of duty Amf 1, as provided by NS

The first task starts at 6:05AM and the last tasks ends at 1:07AM the next morning – note that NS sees this as the same day. Of all crew members, 69 have breaks in their duty. As we mentioned earlier, these breaks are required to be at least 30 minutes. Some crew members have to be transported to the next task, or home depot, which is noted as a task in the schedule – see task 1196 in Table 1.

Furthermore, the time between tasks where crew has to switch trains ranges from 3 minutes to 1:37 hours and is on average 29 minutes. This implies that the assumption that the time required to get to the right platform is 3 minutes is valid for this dataset. This also implies that there is on average a relatively large time between tasks, so the amount of conflicts should be within reasonable numbers – given that we use a good dispatching strategy.

7 Results

In this section we will discuss the results of the simulation model presented in the previous section. We will begin with a case description and continue with the different simulations and their results.

7.1 Case Description

The main idea of these simulations is to validate the algorithm in Section 3. For that purpose we introduced several dispatching strategies and methods to generate arrival times. From here on forward we will be doing several simulations using the dataset in Section 6, to see if the algorithm predicted the conflicts correctly.

WILL THE ALGORITHM PREDICT THE CONFLICTS CORRECTLY?

The first couple of simulations will not restrict the prediction algorithm to a timeframe. If you do not restrict the algorithm to a timeframe, it will eventually predict almost all conflicts, even though some are only predicted seconds in advance. Note that the algorithm is only able to add conflicts, not remove them. Once a conflict is predicted, it may not be changed. Hereby we will be able to see which predicted conflicts will eventually be fulfilled and which conflicts will not.

HOW EARLY CAN WE PREDICT THEM?

Besides which conflicts are correctly predicted, we are also interested in how early we can predict these conflicts. To answer that, we will restrict the prediction algorithm to a timeframe. The algorithm is not allowed to predict conflicts if the next task is less than 30 or 60 minutes away. We can use the information gathered to see how many conflicts we can actually predict 'early'.

7.1.1 How to read the results

All the figures in the following sections are the average of 100 simulations. The tables will specify the total amount of conflicts occured in the simulation, the total predicted conflicts and the amount of conflicts that were not predicted. Note that the sum of the predicted conflicts and the not predicted conflicts is not necessarily the same as the number of conflicts in the simulation. The algorithm might predict conflicts that not occur at all or ignore conflicts which do arise.

The predicted conflicts are devided in two groups, the amount of conflicts predicted by method 1 and by method 2 – see Section 3 for more information on both methods. If a conflict was predicted and it has occured in the simulation it is con-

sidered 'fulfilled', which is shown beneath the total number of conflicts predicted by either method. All conflicts left, those predicted but never occured, can be found next to 'not fulfilled'.

The roman numbers I through V correspond with the dispatching strategies in Section 5.2.

7.2 No Limit

Not restricting the algorithm should result in very few 'open' conflicts, but also in predictions which a shorter time in advance.

	I	II	III	IV	V
Total Conflicts	16.28	15.44	36.40	29.88	27.08
Total Predicted Conflicts	15.36	16.32	38.16	29.56	28.12
Predicted Conflicts (M. 1)	8.52	8.24	24.52	17.84	16.68
Fulfilled	8.16	7.96	21.84	16.52	15.52
Not Fulfilled	0.36	0.28	2.68	1.32	1.16
Predicted Conflicts (M. 2)	6.84	8.08	13.64	11.72	11.44
Fulfilled	6.64	6.72	11.88	10.44	9.72
Not Fulfilled	0.20	1.36	1.76	1.28	1.72
Not Predicted Conflicts	1.48	0.76	2.68	2.92	1.84

Table 2: Results of simulation without any restrictions.

As seen in Table 2 the algorithm predicts between 94% and 107% of the conflicts that occur. With dispatching strategies I and IV the algorithm will predict slightly less conflicts than actually occur. Though when dispatching using strategies II, III and V the algorithm will predict slightly more conflicts than occur.

7.2.1 Dispatching Strategies I & IV

The algorithm is the most accurate for dispatching strategies I & IV.

The reason for the algorithm to predict slightly less conflicts than occur when using dispatching strategy I or IV, can be explained by the type of simulation. Every departure event, the simulation will instruct the algorithm to check whether a conflict will occur for the current delay. After the algorithm has run, the train will be allowed to leave the station and thus an arrival time will be generated. The next time the delay will be updated is at the arrival event for the next station, but at that time the conflict has already occured.

If one would not be interested in an 'early' prediction, this will not be a problem in reality. As in reality the delay is updated real-time and the algorithm can thus

also update real-time, if necessary. Still, these conflicts are not that interesting, as in many cases the crew member is only seconds late; If he was to walk his next task a little faster, the conflict could be avoided. For both dispatching strategies this accounts for about 9% of the conflicts simulated.

What is interesting is the fact that of the conflicts predicted by method 1 about 95% is fulfilled. As dispatching strategy I is nearly the same as the upper bound of the algorithm, one would expect all conflicts predicted to occur. However, consider the example in Figure 5, where two trains will travel from station A to C and further on the same track. Also assume that the 529 intercity service is scheduled to pass the 5629 regional train at station C – the exception mentioned in Section 6.

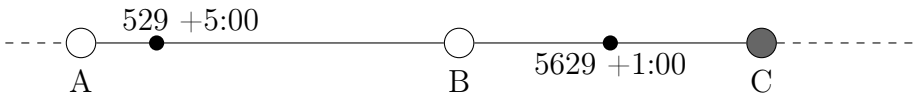

Figure 5: Example: Situation as the algorithm sees the simulation.

The simulation will instruct the algorithm to predict the conflicts and passes all current delays to the algorithm. The algorithm will then use all the current delays to forecast the final delay and conflicts. However, it will not update the delay for 5629 as mentioned in Section 3. Due to the 5 minute delay of the 529 intercity service, it will not arrive at C before the 5629 Sprinter has already left C. According to the algorithm the 529 will then be caught behind the 5629 at least until the next station.

However, what the algorithm does not include is that the 5629 might be delayed more than 1 minute before it departs from station C and thus allowing the 529 intercity service to catch up. If the 529 intercity service arrives at station C before the 5629 leaves, it will be able to pass it. As the 529 intercity service is now in front of the regional train, it might be able to decrease its delay enough to avoid a conflict.

What is also interesting is the fact that, though the amount of conflicts predicted by method 2 is about 5% lower at strategy IV, the amount of open conflicts increased from 2% to 10%. Due to the more irregular ordering of trains, this method has more trouble forecasting the correct delay and thus the correct conflicts. Method 2 assumes that if one can postpone the overtaking without creating a conflict by less than 60%, a conflict will occur. When the order is fixed, the conflict will always occur except if the above example is true. However for strategy IV there is now a 25% chance, the order of trains will change for the better and thus 1 in 4 conflicts might be avoided.

7.2.2 Dispatching Strategies II & V

The algorithm is capable of explaining up to 96% of the conflicts for dispatching strategies II and V.

When dispatching according to strategy II or V the amount of conflicts will be slightly less than when using strategy I or IV, however – not shown in the table – the percentage of trains delayed more than 3 minutes is larger. This is conform with our expectations of what this dispatching strategy does, however it avoids far less conflicts than expected.

For both strategies conflicts predicted through method 1 are quite accurate. For strategy II and IV respectively only 2% and 6% of the conflicts predicted via method 1 are incorrect. However, the conflicts predicted by method 2 are fairly inaccurate as about 16% of these conflicts are predicted incorrect. This actually a side-effect of the used parameters when predicting for method 2.

Take for example a delayed intercity service which is caught behind a slower, regional train. If the intercity service is capable of avoiding the conflict if it passes the regional train at the next one of the next two passing points, but otherwise not, the algorithm is forced into method 2 prediction. When postponing the overtaking will avoid the conflict for less than 60%, the algorithm assumes the conflict will occur. However, the dispatching strategy will almost always force the intercity service to pass the regional train at the first opportunity and thus avoiding the conflict.

At the same time the complete algorithm, method 1 and 2 combined, is the most accurate for both these strategies, with only 5% left unexplained. Further more, the average time a conflict is predicted prior to it occuring is about 1 hour and slightly higher than with any other strategies.

7.2.3 Dispatching Strategy III

When predicting conflicts using dispatching strategy III, the algorithm predicts too much conflicts.

Dispatching strategy III has by far the worst result if you only look at the amount of conflicts occured and predicted. The algorithm predicts about 7% more conflicts than actually occuring – 5% at strategy II. A little over 10% of the conflicts predicted by method 1 and 13% of the conflicts predicted by method 2 are not valid. Furthermore, 7% of the conflicts were not predicted at all.

Due to the dispatching order being random, the average delay per train increases. Hereby, the average delay of the next task will also increase. This will automatically solve some of the predicted conflicts, as there now is enough time for a crew member to get on to its new (delayed) task.

7.3 30 Minute Timeframe

Predicting conflicts up to seconds prior to occuring is not the intended use for the algorithm. It has been developed to predict the conflicts ‘early’, so at least some

time in advance. At this point we restricted the timeframe of the algorithm to 30 minutes. All conflicts have to be predicted 30 minutes prior to the next task.

	I	II	III	IV	V
Total Conflicts	16,44	15.44	35.90	30.10	26.92
Total Predicted Conflicts	12.88	9.64	18.68	17.04	14.92
Predicted Conflicts (M. 1)	9.72	6.60	11.84	11.40	9.76
Fulfilled	9.50	6.57	11.40	11.28	9.50
Not Fulfilled	0.22	0.03	0.44	0.12	0.26
Predicted Conflicts (M. 2)	3.16	3.04	7.20	5.64	5.16
Fulfilled	3.10	2.80	6.64	5.18	4.50
Not Fulfilled	0.06	0.24	0.56	0.56	0.66
Not Predicted Conflicts	3.48	6.08	17.86	13.64	12.90

Table 3: Results of simulation with 30 minute timeframe.

Its quite easy to see that the amount of conflicts predicted more than 30 minutes before they happen is less than if there is no restriction at all. Still, between 52% and 78% of the conflicts can be predicted more than 30 minutes in advance. We also have to take in mind that there might be tasks that are less than 30 minutes apart, or trains that run for less than 30 minutes. The algorithm now will never run for these trains, while conflicts might still occur, and will not predict conflicts at all.

7.3.1 Predicted via Method 1

While the accuracy of the algorithm increased, the amount of conflicts predicted decreased by 25% to 60%.

With exception of dispatching strategy 1, the amount of conflicts predicted by method 1 decreased by 25% to 60%.

When predicting using method 1, both bounds have to be the same. The closer we are to the actual conflict occuring, the closer the bounds will be as the delay can decrease less and less. So when nearly at the conflict, the higher the chance both bounds will be the same. Restricting the algorithm to a fixed timeframe results in less conflicts predicted via method 1, as you restrict the algorithm to how 'close' it can be to the actual conflict.

7.3.2 Predicted via Method 2

The algorithm can now only predict delays that occur 30 minutes prior to a possible conflict.

Most conflicts predicted via method 2 are generally more than 30 minutes in advance. Though, same as with method 1, the algorithm will not make any predictions

for trains running less than 30 minutes due to the time restriction. About 50% of the conflicts predicted via method 2 are less than 30 minutes in advance and those are removed by this timeframe.

However what is more interesting is that, besides that less conflicts are predicted, of all predicted conflicts more are predicted correctly. Where in the unrestricted simulation at most 16% of the predicted conflicts were incorrect, now at most 10% will not occur – the same holds for 'Method 1' predicted conflicts. So while less conflicts are actually predicted, the accuracy of the algorithm is higher.

7.4 60 Minute Timeframe

Restricting the algorithm further and making the timeframe 60 minutes stresses the algorithm even more. Note that there now are a relatively large amount of trains, mostly regional trains, for which the algorithm will not run at all, as their travel time is shorter than the timeframe. These tasks will still conflict, but not be predicted. Also trains travelling longer are not necessarily delayed before the timeframe has passed.

	I	II	III	IV	V
Total Conflicts	16,88	15.68	36.88	28.64	27.20
Total Predicted Conflicts	3.48	3.92	8.88	7.00	6.36
Predicted Conflicts (M. 1)	1.36	1.92	5.28	4.04	3.56
Fulfilled	1.36	1.92	5.28	4.04	3.52
Not Fulfilled	0.00	0.00	0.00	0.00	0.04
Predicted Conflicts (M. 2)	2.12	2.00	3.60	2.96	2.80
Fulfilled	2.08	1.84	3.08	2.64	2.48
Not Fulfilled	0.04	0.16	0.52	0.32	0.32
Not Predicted Conflicts	13.44	11.92	28.52	21.96	21.20

Table 4: Results of simulation with 60 minute timeframe.

Using a 60 minute timeframe, the predictive capabilities of the algorithm drops to at most 25%. Though it is interesting to see that almost all conflicts predicted via method 1 – with exception of when using dispatching strategy V – did actually occur in the simulation. Furthermore, what is also interesting that where previously the conflicts predicted via method 2 dropped by 50% it now declines far less.

THE PREDICTIVE CAPABILITIES OF THE ALGORITHM DECREASED TO ABOUT 25%, BUT IS INCREASINGLY ACCURATE.

Still the fact remains that where we could, for dispatching strategy II, predicted 9.64 of the 15.44 conflicts – 62% – we can now predict only 3.92 – 25%. While 1 hour might a long time in advance, the main purpose of the algorithm is to be able to predict conflicts 'early'. The average time between when the conflict was predicted

and when it occured increased from 1 hour in the non restricted simulation to a little less than 2 hours in this simulation. While it is true that this is mainly because we filtered all the lower values, it does show that the algorithm is capable of predicting conflicts hours in advance – using dispatching rule I and II some conflicts could even be predicted correctly up to 2 hours and 45 minute in advance.

8 Conclusion

The main focus for NS is to provide a high level of service for all its customers. For this purpose NS introduced a new timetable in 2007. However having a good timetable alone is not enough. During the day trains might be delayed and conflicts in the crew schedule might occur. If a crew member is delayed hours before he has to switch trains, a dispatcher might not be able to assess the situation correct. Usually a dispatcher will postpone his decision until he has no other choice; The closer a crew member is to switching trains, the easier it is for a dispatcher to see whether a conflict will occur or not. While this prevents unnecessary rescheduling, it might also decrease the dispatchers rescheduling possibilities.

The purpose of this thesis is to provide an algorithmic approach for predicting conflicts 'early' and hereby aid the dispatchers in their decisions. The algorithm we presented tries to estimate how the delay will progress in the future. Using this estimation a conflict can be predicted correctly up to 2 hours and 45 before it occured.

When the algorithm is not restricted, it is capable of predicting up to 95% of all conflicts. The conflicts which could not predicted occured by mere seconds; For example a crew member arriving at its next task 5 seconds late. These type of conflicts generally do not require rescheduling, as they can simply be solved by the crew member hurrying to the platform.

However, some of the predictions are only minutes before the actual conflict occurs. Removing these conflicts, using a restricted simulation with a 30 minute and 60 minute timeframe, we were able to see what percentage of the conflicts can be predicted further ahead. In the 30 minute timeframe simulation, between 52% and 78% can be predicted correct and in the 60 minute timeframe simulation the algorithm could predict about 25% of the occured conflicts. Furthermore, the algorithm was increasingly accurate when restricting the simulation. Note that, while this is a sharp drop in predictive capabilities, not all delays occur up to 60 minutes prior to the conflict occuring.

The increasing accuracy of the algorithm suggests that the earlier a train is delayed the better the outcome will be. Because for dispatchers it is generally the other way around, this algorithm can successfully used to aid dispatchers in their decisions.

9 References

Budai, G., Maróti, G., Dekker, R., Huisman, D., & Kroon, L. (2007, May). Rescheduling in railways: the Rolling Stock Ballancing Problem. *Econometric Institute Report*, *EI2007*(27).

Hartog, A., Huisman, D., Abbink, E. J., & Kroon, L. G. (2009). Decision support for crew rostering at ns. *Public Transportation*(1), 121-133.

Huisman, D., Kroon, L. G., Lentink, R. M., & Vromans, M. J. C. M. (2005). Operations research in passenger railway transportation. *Statistica Neerlandica*, *59*(4), 467-497.

Huisman, D., & Potthoff, D. (2009, April). NS kan steeds sneller reageren op verstoringen. *EconomieOpinie.nl*.

Kil, S., & Naber, S. (2010). *Early Conflict Detection.* (Study Project, Netherlands Railways)

Nowak, I. (2005). *Relaxation and decomposition methods for mixed integer nonlinear programming.* Basel, Switzerland: Birkhauser.

Potthoff, D., Huisman, D., & Desaulniers, G. (2010, May). Column generation with dynamic duty selection for railway crew rescheduling. *Transportation Science*, DOI: 10.1287/trsc.1100.0322.

Veelenturf, L. (2008). *Dispatching Strategies for Disruption Management at Netherlands Railways.* (Bachelor's Thesis, Erasmus University Rotterdam)

A Mixed Integer LP Simulation Model

The simulation can be written as a mixed integer linear programming problem having a objective function and will adhere serveral restrictions. Both the restrictions and the objective function will contain several symbols, representing indices, sets, parameters and decision variables. Before we will formulate the simulation model, we will introduce these symbols and explain their meaning.

A.1 Character Definition(s)

Indeces

index	description
s	measure point
t	train
c	crew
r	rolling stock

Sets

set	description
S	The set of measuring points ($s \in S$)
S_t	The set of measure points which are on the route of train t ($s \in S_t$)
T	The set of trains ($t \in T$)
C	The set of employees ($c \in C$)
R	The set of rolling stock units ($r \in R$)

Parameters

parameters	description
$p_{t,s}$	Planned departure time of train t at measure point i
$m_{t,s,s+1}$	Minimal driving time of train t from measure point s to $s+1$
$\xi_{t,s,s+1}$	A certain disruption in the driving time from of train t from measure point s to $s+1$
$w_{t,s}$	Dwell time of train t at measure point s
$o_{c,s}$	Required transition time for crew member c at measure point s
$b_{r,s}$	The shunting time for rolling stock unit r at station s
$g_{c,t,s}$	Will become 1 if a crew member c is planned to depart with train t at station s
$f_{r,t,s}$	Will be 1 if rolling stock unit r is planned to depart with train t at station s
$q_{t,t^*,s}$	When train t will depart before $t*$ at measure point s, $q_{t,t^*,s}$ will be 1
h_s	The headway time at measure point s

Decision Variables

variable	description
$d_{t,s}$	The realised departure time of train t at measure point s
$a_{t,s}$	The arrival time of train t at measure point s
$a_{c,s}$	The arrival time of crew member c at measure point s
$a_{r,s}$	The arrival time of rolling stock unit r at measure point s

Using the above definition for parameters and decision variables the model is lineair, but the order in which the trains depart is fixed. Choosing the order of the trains such that the first arriving train at a station will be the first to leave again, this definition will be largely consistent with the assumptions in Section 3.1. The only exception to this rule is when the train is ordered to pass explicit.

Another option is to make $q_{t,t^*,s}$ an decision variable. This will result in a non-lineair model – which can be linearized (Section A.3) – but might result in a better minimal solution.

A.2 Simulation Model

Using the previous defined symbols we can write the simulation model as seen below. The objective function (8) minimizes the total arrival times, which will result in the best solution given the disturbances.

$$min \sum_{t \in T} \sum_{s \in S_t} a_{t,s} \tag{8}$$

$$a_{t,s} \geq d_{t,s-1} + m_{t,s-1,s} + \xi_{t,s-1,s} \qquad \begin{matrix} \forall t \in T \\ \forall s \in S_t \end{matrix} \tag{9}$$

$$d_{t,s} \geq (a_{c,s} + o_{c,s}) g_{c,t,s} \qquad \begin{matrix} \forall t \in T \\ \forall s \in S_t \\ \forall c \in C \end{matrix} \tag{10}$$

$$d_{t,s} \geq (a_{r,s} + b_{r,s}) f_{r,t,i} \qquad \begin{matrix} \forall t \in T \\ \forall s \in S_t \\ \forall r \in R \end{matrix} \tag{11}$$

$$d_{t,s} \geq a_{t,s} + w_{t,s} \qquad \begin{matrix} \forall t \in T \\ \forall s \in S_t \end{matrix} \tag{12}$$

$$d_{t,s} \geq p_{t,s} \qquad \begin{matrix} \forall t \in T \\ \forall s \in S_t \end{matrix} \tag{13}$$

$$d_{t*,s} \geq (d_{t,s} + h_s) q_{t,t*,s} \qquad \begin{matrix} \forall t \in T \\ \forall t* \in T \\ \forall s \in S_t \end{matrix} \tag{14}$$

$$a_{r,s} \geq f_{r,t,s-1} a_{t,s} \qquad \begin{matrix} \forall t \in T \\ \forall s \in S_t \\ \forall r \in R \end{matrix} \tag{15}$$

$$a_{c,s} \geq g_{c,t,s-1} a_{t,s} \qquad \begin{matrix} \forall t \in T \\ \forall s \in S_t \\ \forall c \in C \end{matrix} \tag{16}$$

$$a_{t^*,s} \geq a_{t,s} q_{t,t^*,s-1} \qquad \begin{matrix} \forall t \in T \\ \forall t* \in T \\ \forall s \in S_t \end{matrix} \tag{17}$$

$$1 = q_{t,t*,s} + q_{t*,t,s} \qquad \begin{matrix} \forall t \in T \\ \forall t* \in T \\ \forall s \in S_t \end{matrix} \tag{18}$$

$$q_{t,t*,s} \in \{0, 1\} \qquad \begin{matrix} \forall t \in T \\ \forall t* \in T \\ \forall s \in S_t \end{matrix} \tag{19}$$

Using this optimization problem, when minimizing the arrival times, the depar-

ture times will automatically be minimized too. However, note that reverse is not true, minimizing the departure times will not automatically minimize all arrival times as the last station does not have a departure time. The departure times will only be minimized because by constraint (9) the arrival time at station s has to be later than or equal too the sum of the departure time at the previous station $s-1$, the minimum technical driving time and a disturbance.

Constraint (10) and (11) make sure that the crew and rolling stock have to be available before a train can leave. Both crew members and rolling stock will not be ready at the same time they arrive at a certain station. Crew members will have to walk to the right platform and rolling stock will have to be shunted into position. This is reflected in both the constraints as $o_{c,s}$ and $b_{r,s}$ – see Section A.1.

Constraint (12) includes the dwell time in the selection of a departure time, where no train can leave before its dwell time has passed. Also, for passenger convenience, no train can leave before its scheduled departure time, which is what constraint (13) will take care of. For safety, no train can leave before the headway time has passed, constraint (14) will make sure that two consecutive trains are at least h_s minutes apart.

The next two constraints (15-16) will make sure that the arrival time of either rolling stock or crew is equal to later than the train transporting them will arrive at the station. This automatically implies that, when the transporting train is delayed, the departing train requiring either the rolling stock or crew will be delayed as well.

Previously, in Section 4.1, we assumed that only one track per direction was available – assumption (VI) – and that as result no trains can pass while on an edge. Constraint (17) restricts the train order while on the track – an edge, thus prevent trains from passing while on an edge. The arrival time of all other trains t^* have to be later than the arrival time of the current train t if it departed from the previous station before t^* did.

The last two constraints (18-19) are only required if $q_{t,t^*,s}$ is a decision variable. When releasing the fixed order – fixed $q_{t,t^*,s}$ – of trains, several problems arise when these constraints are not included. First, $q_{t,t^*,s}$ has to be binary and thus has to be an element of $\{0,1\}$ – constraint (19). More importantly, both trains t and t^* cannot be leaving at the same time or both before the other. When for example both $q_{t,t^*,s}$ and $q_{t^*,t,s}$ are 1, train t will depart before t^* and t^* will depart before t. Constraint (19) will restrict the sum of both $q_{t,t^*,s}$ and $q_{t^*,t,s}$ to be equal 1. Because both are binary, only one can have value 1 and the other will have to be 0.

A.3 Dynamic Departure Order

We want to use the simulation model to check the conflict model presented in Section 3. This conflict model allows, when constructing the lower bound, the order of trains to be altered, if two trains are close together, to avoid the delay from getting too large. Simulating the change of train order, will require $q_{t,t^*,s}$ to be a decision

variable.

Changing $q_{t,t^*,s}$ from parameter to decision variable has a rather large impact on the model. First of all, there is more freedom of choice, which might result in better solutions. However this will also make the model nonlinear in constraint (14). Nonlinear programming problems are rather hard to solve, so it best to avoid them or rewrite them as linear programming problems using relaxation or reformulation.

For this problem a 'Big-M' reformulation can be used best, as described in Nowak (2005). A 'Big-M' reformulation involves rewriting the nonlinear constraint using a large arbitrary number M – hence 'Big-M'. Rewriting equation (14) will result in constraint (20).

$$d_{t*,s} \geq (d_{t,s} + h_s) - M(1 - q_{t,t*,s}) \qquad \begin{array}{l} \forall t \in T \\ \forall t* \in T \\ \forall s \in S_t \end{array} \tag{20}$$

This formulation will effectively force linearity onto this constraint, while keeping the original functionality. When $q_{t,t^*,s}$ equals 1, the departure time of train t^* has to be later than the departure time of the previous train t and the headway time. If $q_{t,t^*,s}$ becomes 0 the most right part of inequality (20) will be far negative, depending on M.

Note that while using the 'Big-M' method reformulates a nonlinear problem to a linear problem, one has to be careful choosing a value for M. The value assigned M should be as small as possible. Therefore the term 'Big-M' is rather misleading, as one should be choosing a 'sufficient-large-M'. As result of a large M the relaxations will become weaker and thus will increase computation time.

B Figures and Tables

B.1 List of Figures

B.2 List of Tables

C Acronyms & Symbols

C.1 Mathematical Symbols

cd_s	the Current Delay at Station s
s	the Current Station

C.2 Used Acronyms

NS	Netherlands Railways - in Dutch "Nederlandse Spoorwegen"
GN	Groningen Central Station
GVC	The Hague Central Station
DT	Delayed Train