USEFULNESS OF OFF-LINE DELAY MANAGEMENT

- SIMULATION OF A REAL TRAIN TIMETABLE AND OFF-LINE DELAY MANAGEMENT DECISIONS –

BACHELOR THESIS

ERASMUS UNIVERSITEIT ROTTERDAM

30TH JUNE 2013

AUTHOR: WANG WEI GUO
344694

SUPERVISOR: DR. D. HUISMAN

Abstract

Every day, all over the world, some people get delays when they travel by train. A delay of a few minutes is often no problem, but big delays are really annoying. A big delay can be caused when passengers change trains and miss a connecting train, because of a small delay of the feeder train. The connecting train could wait for the delayed train and give the passengers the opportunity to change trains as planned. However, the connecting train now also has a delay. The problem we are dealing with here is called delay management. In this thesis we will check whether off-line delay management might be useful in real life. We will use a simulation of a real life timetable and generate delays. We make the 'waiting' decisions with an off-line delay management model for the next hour of the timetable each time a delay occurs. Finally, we evaluate the total delay passengers have.

CONTENTS

1 Introduction 3
2 Problem description 4
2.1 The problem 4
2.2 The objective 4
3 Simulation 5
3.1 Terms 5
3.2 Assumptions 5
3.3 Generating delays 6
3.4 Input 7
3.5 Output 7
3.6 Steps of the simulation 7
3.7 Example of the simulation 9
4 Optimization model 11
4.1 Defining the sets 11
4.2 Defining the parameters 11
4.3 Defining the variables 11
4.4 The model 12
5 Evaluation of the solutions 13
5.1 Method 13
5.1.1 The nodes 13
5.1.2 The arcs 13
5.2 An example 14
5.3 The evaluation 14
6 Results 15
6.1 Data 15
6.1.1 The data in more detail 15
6.1.2 The small dataset 15
6.1.2 The big dataset 15
6.2 Implementation 15
6.3 Results small dataset 16

6.3.1 The generated delays 16
6.3.2 The optimization model 16
6.3.3 The decision made by the model 19
6.3.4 The total overall delay 19
6.3.5 The 100 simulation runs compared to the 10 simulation runs 20
6.4 Results of the big dataset 21
6.4.1 The generated delays 22
6.4.2 The optimization model 22
6.4.3 The decisions made by the model 24
6.4.4 The total overall delay 25
6.4.5 The big dataset run compared to the small dataset run 26
6.5 Sensitivity Analysis 27
7 Conclusion 29
8 Future research 31
Appendix 32
References 33

1 Introduction

Every day, all over the world, millions of people take the train instead of driving a car. People mainly choose to take the train because of the traffic jams one tends to get stuck in, when they take the car in the early morning and at five or six o'clock when everybody is going back home. However, the trains are not always on time due to, for example, an accident or at rush hours. Most of the times the delays are very minor and trains only depart with a delay of a few minutes. But inevitably there might be major delays every day. Thus, it can occur that one travels about as long by train, as by car when stuck in a traffic jam. This can be very frustrating and keeping the passengers satisfied is an important task for the railway company. When they have bad service, less people will take the train and in turn this will result in big losses.

Some passengers also have to change trains when they travel to reach their destination. When the feeder train has a delay, there is a chance that they miss the connecting train. They often have to wait a long time for the next train. To keep the passengers on the delayed train satisfied we should let the connecting train wait. However, this might also cause other passengers to miss their connection. We will check how we can minimize the total delay of all passengers. The major delays are irrelevant for our thesis as with major delays passengers can often take the next connecting train in the same direction. Therefore, we will consider small delays up to 18 minutes.

These situations also occur in the Netherlands. Here approximately 1.2 million people travel by train. About 25% of the passengers have to change at a certain point to reach their destination. When a train has a delay, some passengers might miss their connection. We have some data of the main railway operator of the Netherlands, called Nederlandse Spoorwegen, or shortly NS. The data are from the Randstad, which is considered the busiest part of the network.

In this thesis we try to minimize the delay passengers have by applying optimal off-line delay management decisions in an on-line setting. This means we minimize the delay of the passengers at a short time interval. This thesis is divided in the following chapters: in chapter 2 we look at the problem in more detail. In chapter 3 we discuss the simulation which will generate the delays to get a real life situation. Within the simulation the wait and depart decisions of delayed trains will be optimized with the model discussed in chapter 4. Next we evaluate the solutions. This will be done with a shortest path algorithm described in chapter 5. We discuss the results in chapter 6 and we give a conclusion in chapter 7. Finally, we give some advice for future research in chapter 8.

2 Problem description

In this chapter we discuss the problem we are going to deal with. First we describe the problem, and then we are setting the objective.

2.1 The problem

The problem is the fact that if a feeder train has a delay of as small as even 5 minutes, a connecting train can be missed because of this delay. The passengers that missed the connecting train will not be pleased with this, because they might have to wait for half an hour for the next train. If there is a great number of passengers that missed the connecting train, there will be a lot of complaints and thus the service will be considered bad.

To solve this problem, we could let the connecting train wait, so the passengers will not miss their train. But this could result in other passengers missing their connection on the next train and of course the passengers in the connecting train will have an 'unnecessary' delay. So we have to decide whether to let the connecting train wait or let it depart as scheduled. This problem is called delay management.

2.2 The objective

The objective is to minimize the total delay of all passengers. There is a lot of research on off-line delay management, see Dollevoet et al. (2011), Dollevoet et al. (2012a), Dollevoet et al. (2012b) and Dollevoet et al. (2012c), among others. Off-line delay management means we look at the delay problem on, for example a whole day. This is of course very unlikely in reality. We simply do not know the delays beforehand. Normally we know the delays when they occur. So we will simulate an on-line setting and solve a sequence of off-line delay management problems. The simulation generates delays randomly and therefore the delays will not be the same every run. We use an off-line integer programming model to calculate the minimal overall delay every time a delay occurs. This model will decide which connecting trains should wait. After that, we evaluate how much delay the passengers had. We will use a shortest path algorithm for that, with the new timetable in which some trains wait for connecting trains. This is the timetable that is executed. We then can compare these arrival times to the arrival times of the original timetable. This way we can compute the overall delay. For more details of the methods see chapter 3 for the simulation, chapter 4 for the integer programming model and chapter 5 for the evaluation.

3 Simulation

In this chapter we will discuss the structure of the simulation. A simulation is an approximation of the reality and therefore we hope to realize results that can be used in real life. We will simulate the given timetable of the trains with delays and decide whether trains should wait for the delayed trains or not. This is decided by the optimization model discussed in chapter 4. When the timetable is done, we evaluate the decisions made and check the combined delay of all passengers. This will be done with a shortest path algorithm discussed in chapter 5. First of all, we will define the terms used in this thesis. Then we will discuss the assumptions that have to be made in order to validate the methods. Next we will discuss how we generate delays. Then we will discuss the input of the simulation and the output of the simulation. After that, we will discuss the steps of the simulation. Finally, we will give a small example of how the simulation works.

3.1 Terms

First we will define the terms we will use in this thesis and will recur often. We define:

- An event. An event is a departure or arrival of a train at a station at a certain time. Thus, every arrival and departure of every train is considered as a different event.
- An activity. An activity is a link between events. It can be a train that drives from a station to another, a train that waits or passengers that change trains at a station.

3.2 Assumptions

To validate the simulation we need to make some assumptions. We assume that:

- The number of delays that occur in the timetable is always about 5% of the events. This is of course in reality not the case but we wanted to take a general day. NS confirmed that 5% of the trains have a delay. Thus, taking an average of 5% is a good approximation of the reality.
- A generated delay can only be from 6 to 18 minutes. This is of course not the case in reality, but we only consider small delays and the delays of 1 to 5 minutes will be too small to make a difference. The slack times, which are the time that can be spared by doing an activity as fast as possible, will shorten the delays to 0 really fast. For example, when a train has 4 minutes of delay and arriving at the station has a slack time of 2 minutes, it will only have 2 minutes of delay after arrival. In addition, arriving at the next station with also 2 minutes of slack time will result in a delay of 0 minutes. This will probably not let the optimization model in chapter 4 take any actions and thus the delay would be useless for our research. The average slack time is about 1 minute and has a corresponding standard deviation of 1 minute.

  Furthermore, a bigger delay will not be within the range of our problem, because after a too big delay the passengers can simply take the next train in the same direction. A lot of trains drive every 15 minutes or every 30 minutes. Thus, by taking 18 minutes as a

maximum is to make sure the connecting trains that drive every 30 minutes can have some considerable delay, without the passengers being able to take the next connecting train. Also we do not want trains that drive every 15 minutes to take too much delay such that the passengers can take even one of the next two trains in the same direction. Therefore, we think that delays of 6 to 18 minutes will be a good assumption for our research.

- We know the number of passengers who will travel during the whole timetable and their routes. We need this assumption because one of the parameters for the model is the number of passengers that change trains and one is the number of passengers that finish their travel at a particular event. For more details on this see the model in chapter 4.
- Passengers take their fastest routes when a delay occurs. In the simulation we use the shortest path algorithm after all the delays have occurred. This means that the passengers actually know which trains will have delays and plan accordingly. This is of course a bad assumption but it makes the problem a bit easier.
- In the next period all vehicles run on time. This is not a good assumption as in reality most of the times the next train on the same route has also a delay although it is smaller than the previous one.
- The passengers who change trains can change immediately. In reality when a lot of passengers change it will take a few minutes before all passengers have changed.
- Connecting trains that wait for the passengers who change trains, cannot cause problems within the station and the order of the trains that drive on a track. A train that waits can block another train that arrives at a station or he can block a another train when it drives with a delay as some trains, the intercities, only stop at the big stations and some trains stop at all stations. For research on delay management that considers capacities of stations, we refer to Dollevoet et al. (2012c)

3.3 Generating delays

From past data, NS confirmed that about 5% of the trains have a delay. Therefore, we simulate delays for about 5% of the events. If a departure event has a delay, it means that the train got a delay when it waited at a station. If an arrival event has a delay, it means that the train got a delay when it drove from a station to a station. To simulate whether an event has a delay, we draw a pseudorandom value from the standard uniform distribution on the open interval (0,1). When this value is greater than 0.95 that event has a delay. In other words, the event has a delay if the drawn number is within 5%. When an event has a delay we will randomly pick an integer number from the discrete uniform distribution on the interval [6,18] and take that as the number of minutes the train is delayed.

With the setup of generating delays like this it is possible that two consecutive events will both have a delay. Of course the probability is very small, but when this occurs a train could get 36

minutes of delay or even more when there are move consecutive events with delays. As this could happen in reality we will not try to avoid this.

3.4 Input

The inputs we need for the simulation are as follows:

- A timetable for the trains.
- The slack times of all activities
- The number of passengers that end their journey at all the arriving events
- The number of passenger that change at all the changing activities

3.5 Output

We will discuss what we want as output from the simulation. We want:

- The executed timetable for the trains.
- The decisions made in each optimization.
- The number of events in the hours that were optimized.
- The values of the objective function of each optimization.
- The number of passengers that has a delay.
- The number of passengers that will not have a connection anymore (this occurs when the delay causes passengers to miss the last connecting train).
- The total delay of all passengers.

3.6 Steps of the simulation

We use a discrete event simulation, which means that we order the events on time of occurrence and we begin at the first event of the timetable. Next we go to step 1 which is described as follows:

Step 1: check whether this event will have a delay and generate one if it has, as described in the section 3.3. If a delay occurs, go to step 2. If no delay occurs, go to step 4.

Step 2: Define and set t_e to the time of the event that has a delay. We optimize the model which will decide whether trains should wait for the delayed trains or not. We will use the data of the upcoming hour. This will be enough because the delays are at most 18 minutes. In other words, we take the events that occur between t_e to t_e +60. The parameters for the whole timetable are given as input, so we can take the parameters of the needed hour for the optimization. After the optimization, we will have new departures for the trains in that hour as we let some trains wait or they simply have a delay. Next we go to step 3.

Step 3: Because we have used only an hour for the optimization, only the events of that hour will change. Thus, we store the new times of those events. We also store the number of events in the hour that was optimized and the optimal value of the objective function. Next we can also compute and store the number of trains waiting and number of trains not waiting. The train

waits if the actual departure time is greater than the delay it has and of course the train does not wait if the delay is equal to the actual departure time. Next go to step 4.

Step 4: If there is a next event. Go to that event and return to step 1.
If there is no next event, we have run through the whole timetable. Now we can compare the fastest routes computed with the executed timetable to the ones computed with the original timetable. By doing this we can get the number of passengers that have a delay and thus, the overall total delay of all passengers. We will use a shortest path algorithm to get the fastest route from origin to destination for each passenger. For more details see chapter 5. This algorithm also allows us to get the number of passengers that cannot get to their destinations anymore because of the delays.

Going trough steps 1-4 will give us one run of the simulation. Therefore, if we want more runs we will have to go the first event again and start with step 1. The pseudo code of the simulation can be seen in Figure 1 which will clarify the simulation.

```
Sort the events

Go to the first event

Step1:

If a delay occurs (i.e. if U > 0.95 with U drawn from a standard
      uniform distribution on the open interval (0,1))


      get a delay(get a random integer from the discrete
      uniform distribution on the interval [6,18]

      Step2: Set t_e to time of the event at which delay occurs

            optimize the optimization model by taking the
            parameters needed from the events between t_e and
            t_e+60

      Step3: compute and store the nr of trains waiting and
            not waiting. Store the number of events in this hour
            and the output of the model for the events between t_e
            and t_e+60

Endif

if there is a next event

      go to next event and go to step 1

endif
```

```
Step4: Compute fastest routes with shortest path algorithm

       compute the passengers delayed, passengers that can not
        arrive at destination and total delay of all passengers
```

Figure 1: pseudo code of the simulation.

3.7 Example of the simulation

We will give a small example of how the simulation works. An illustration is given in figure 2. This timetable contains only seven events and the time goes from 0 to 120 minutes. First, we start at event 1 and go to step 1 and check whether there is a delay. There is no delay so we proceed to step 4. As there is a next event, we go to event 2. The second event also occurs without delay, so we proceed to the third event. It follows from step 1 that there is a delay. Therefore, we generate a delay and set t_e to 15. Then we go to step 2, where we optimize the model with the events that occur between t = 15 and t = 75. These are events 3, 4 and 5. Thus, this optimization has three events in the hour that is optimized. In step 3 we compute and store the output as described above. We proceed to step 4 and therefore the next event. At step 1 there is a delay again. Thus, we generate a delay and set t_e to 40. Next we optimize the model with the events from t = 40 and t = 100 as described in step 2. This optimization takes events 4, 5 and 6 as input. Thus, this optimization has also three events in the hour that is optimized. Then we compute and store the output as described in step 3. After that, at step 4 we see that there is a next event. From this point on, we follow the same steps as at the first and second events until we reach step 4 at event 7. At this point there is no next event. So, we compute the fastest routes with the shortest path algorithm of the original and the executed timetable. Next we compare them to get the total delay of all passengers, the number of passengers with a delay and the number of passengers that cannot get home anymore. Note that the decisions made at event 3 can be overwritten. At event 4 we will have new information and thus it could be optimal to let, for example the train of event 5 not wait anymore.

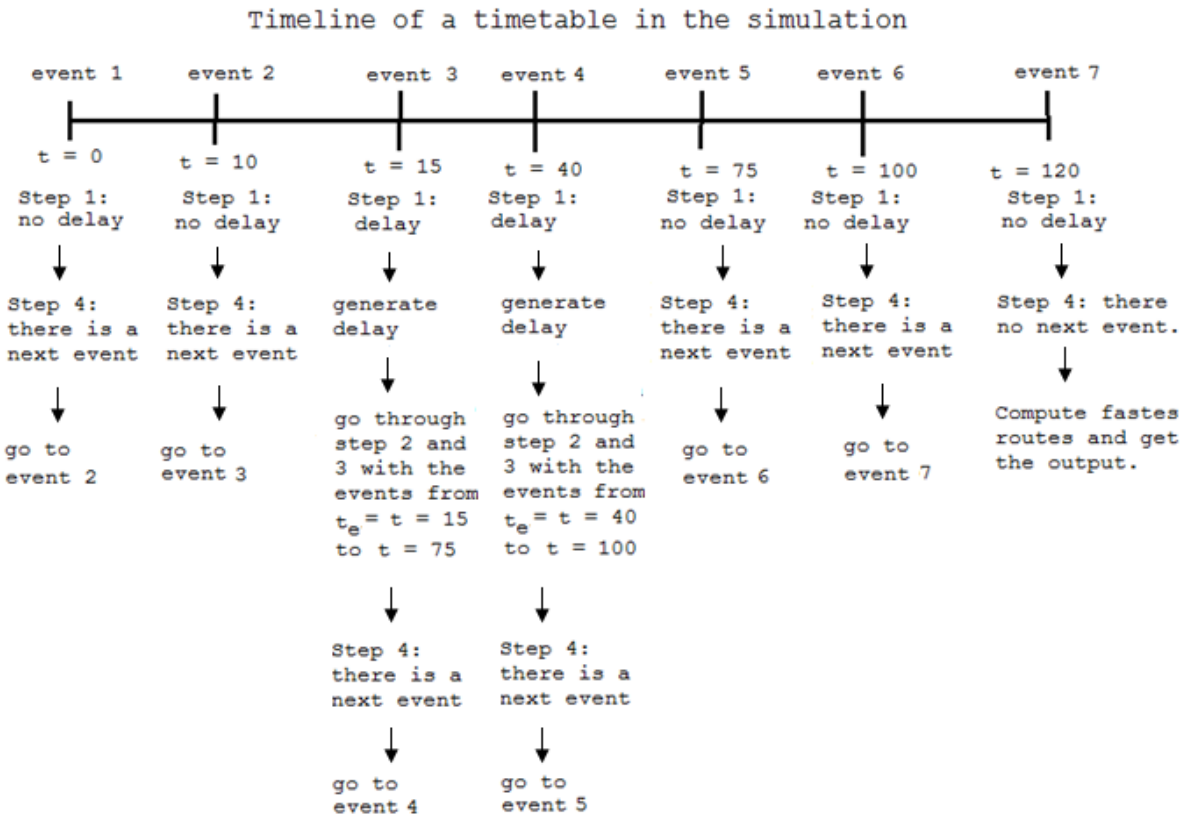

Figure 2: Illustration of the time interval of a timetable in the simulation.

4 Optimization model

In this chapter we discuss the classic off-line delay management model which is mentioned in section 3.2 of Dollevoet et al. (2011) and originated from Schöbel (2007). We use this model to minimize the delay of all the passengers for the upcoming hour every time a delay occurs. The model will decide whether trains should wait for a connecting train or not. Actually, the model of Schöbel (2007) looks at the paths of the passengers and the model of Dollevoet et al. (2011) looks at the connections. However, the optimization is the same, since both models aim to minimize the delay. In this thesis we use a combination of both models. We use the slack times in the constraints as in the model of Schöbel (2007) and we use the connections formulation of Dollevoet et al. (2011). First we define the sets and the parameters. Next we define the variables and finally we discuss the model.

4.1 Defining the sets

First we define the set of events as E = E_{dep} U E_{arr} . An event is defined in section 3.1. E_{dep} and E_{arr} denote the set of all events where a train departs and arrives at a station, respectively. Next we define the set of activities as A = $\mathrm{A}_{\mathrm{change}}$ U $\mathrm{A}_{\mathrm{drive}}$ U A_{wait}. An activity is also defined in section 3.1. $\mathrm{A}_{\mathrm{drive}}$ denotes the set of activities where a train drives, $\mathrm{A}_{\mathrm{wait}}$ the set of activities where the train waits and $\mathrm{A}_{\mathrm{change}}$ the set of activities where passengers change trains.

4.2 Defining the parameters

Now we have the sets we can define the parameters needed. We define:

T = the cycle time of the trains. This is the cost parameter and this value is the number of minutes the passengers wait when they miss a connection.

d_e= delay of event e

s_a= slack time of activity a

w_e = the number of passengers that plan to end their journey with event e

w_a = the number of passenger that plan to use changing activity a

4.3 Defining the variables

The model of course also has some variables. We define the following variables:

x_e = actual delay of event e (it can be that the train waits for a connecting train)

$$z_a = \begin{cases} 0 & if\ connection\ a\ is\ maintained \\ 1 & otherwise \end{cases}$$

4.4 The model

Now we define the model we used as follows:

$$\min \quad \sum_{e \in E_{arr}} w_e \; x_e \; + \sum_{a \in \mathrm{A_{change}}} w_a * \mathrm{T} * z_a \tag{1}$$

$s.t.$

$$x_e \; \geq \; d_e \qquad for\ all\ e \in E \tag{2}$$

$$x_i \; - \; x_j \leq \; s_a \qquad for\ all\ a = (i,j) \in \; \mathrm{A_{drive}} \cup \; \mathrm{A}_{wait} \tag{3}$$

$$-M * z_a + x_i \; - \; x_j \leq \; s_a \qquad for\ all\ a = (i,j) \in \; \mathrm{A_{change}} \tag{4}$$

$$x_e \; \epsilon \; \mathbb{N} \qquad for\ all\ e \in E \tag{5}$$

$$z_a \; \epsilon \; \{0,1\} \qquad for\ all\ a \in \; \mathrm{A_{change}} \tag{6}$$

The objective function is given by (1) where the total delay is minimized. We assume that if a passenger misses his or her connection that he or she has to wait 20 minutes. Thus, T has a value equal to 20 in our model. In the original model T is equal to the cycle time, but because not every train has the same cycle time we use a value that probably will give us the minimum delay. Dollevoet et al. (2011) varied the value of T from 0 to 60 with a slightly different model. The value that gave the minimum objective function was 20. Therefore, we choose 20 as we think this will also give us the minimum delay. In section 6.5 we provide a little sensitivity analysis on the value of T. Next we make sure that the actual delay is always greater than the delay the event has with constraint (2). With constraint (3) we make sure the delays are becoming smaller with the slack times. In constraint (4) we check whether connections that passenger take are being maintained. If the connection is not maintained z_a will be 1, so M will make sure the constraint is still valid. If the constraint would not be valid the model would not have a solution. M has to be sufficiently large. We just take M as a really large number, like 10000. This number is big enough as the variables will not be equal to 10000 as a train will not have that much of a delay. Lastly (5) and (6) define that x_e is a natural number and that z_a is binary respectively.

5 Evaluation of the solutions

In this chapter we will discuss the methods that evaluate the solutions of the model discussed in chapter 4. After the timetable is all run through, we have decided whether we let trains wait or we let them depart as soon as possible. To evaluate whether the decisions are good or not, we have to check the overall delay of all passengers. In other words, we are going to compare the fastest routes of the passengers without any delays with the new fastest routes with the delayed trains. After that we will have the difference between the arrival times of the two situations and thus we can calculate the overall delay of all the passengers. First we will discuss the method. Then we will give a small example. Finally we discuss how to evaluate the decisions.

5.1 Method

To compute fastest routes we use a shortest path algorithm. We use the one of Bellman-Ford but any shortest path algorithm will do. To use the algorithm we will have to define a graph on which the algorithm operates.

5.1.1 The nodes

First we define two types of nodes. The first type of nodes contains information about the origin-destination-time pairs. An origin-destination-time pair, or ODT-pair for short, is a pair of stations which are, as the name says, the origin and the destination of a passenger with a starting time. These nodes are the starting and ending nodes for the algorithm. So actually, we want the shortest route from a starting ODT-pair to the same ending ODT-pair. The second type of nodes is defined as an event. So the node has a train, a station, a time and is a departure or an arrival of a train.
An example of the graph is given in figure 3. We have drawn the first type of nodes as hexagonal nodes and the second type of nodes is drawn as a circle.

5.1.2 The arcs

Next we define the arcs as the possibility to go from a node to another. We define the weight of the arcs as the time that is needed to go from one node to the other. A starting node has an arc to all the nodes with the same station label and a time label that is greater than or equal to the time label of the ODT-pair. An ending node has only incoming arcs and will have an arc coming from every node that has the same station label and a time label greater than the ODT-pair time label. The time needed to go from a node to the ending node will have a time of 0. For the other nodes, an arc is possible if the time label of the node from which the arc departs plus the weight of the arc is smaller or equal to the time label of the node in which the arc arrives. Also an arc is only possible, if there is a connection between the station labels of the nodes given in the timetable or the station labels are the same. To get a better view of how the graph should look like see figure 3.

5.2 An example

In figure 3 we can see that the starting and ending nodes are drawn as hexagonal nodes. We see the shortest path from ODT-pair, ams-zl 21.00, is to go with train 1 which departs at 21.00 from Amsterdam. Next it will stop in Utrecht for 3 minutes and finally it will arrive in Zwolle at 21.30. Notice that there is a path that takes train 2 from the same ODT-pair starting node. This is actually also a possible path, although it is not the optimal one. This path is actually the same as taking a train later, as all the arrivals and departure of the train are exactly half an hour later. We can see that the shortest path for ODT-pair, ams-rtd 21.30, is to change trains in Utrecht. The path for this ODT-pair is: take train 2 which departs at 21.30 from Amsterdam. When it arrives in Utrecht at 21.45, change to train 3 which departs at 21.50. This train will go to Rotterdam directly and the destination of this ODT-pair is reached at 22.10.

5.3 The evaluation

With this method we compute the shortest route of both the original timetable and the executed timetable. After computing the routes we will have the time of arrival and we can check how much they differ. We can now compute the number of passengers that have a delay and the total delay all the passengers have combined. Unfortunately, because of delays at the end of the timetable some passengers will miss their connection and will not be able to get to their destination. We can also compute the number of passengers that have this problem.

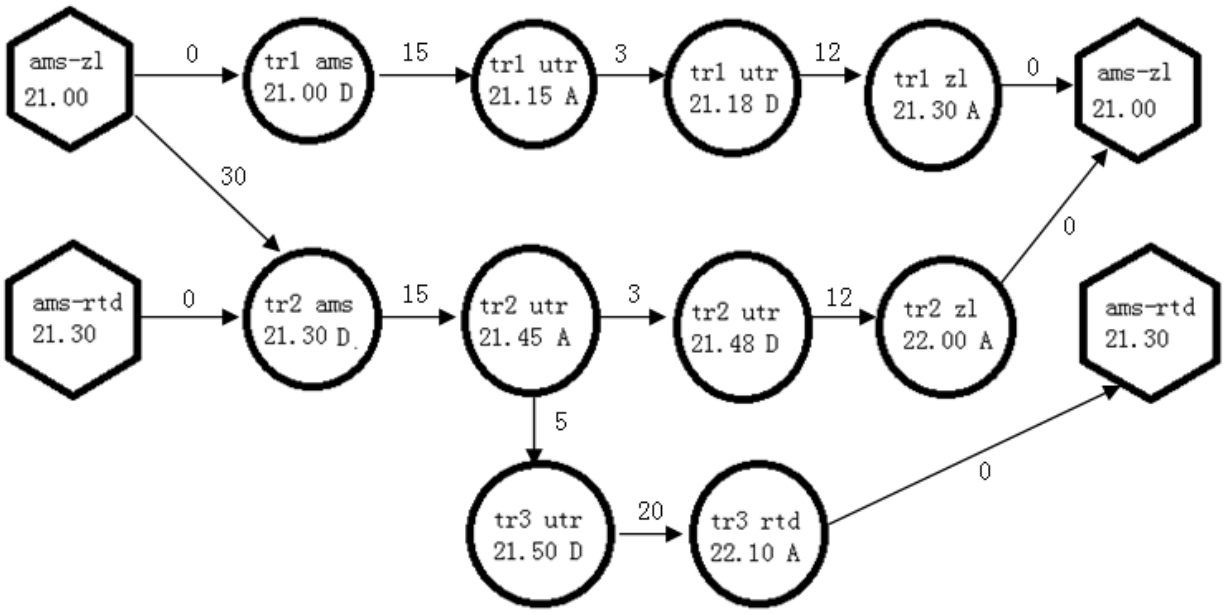

Figure 3: Small example of the graph representation of the timetable

6 Results

In this chapter the results will be discussed. Before we do that, we start by discussing the data. Next we will discuss the implementation of the simulation. And after that we discuss results of the simulations of the small dataset. Next we discuss the results of the big dataset. Finally we will do a little sensitivity analysis.

6.1 Data

We have data of the Randstad of the Netherlands, which is considered one of the busiest parts of the Dutch railways. The data of the trains starts from 20.30 and end about 01.00 for the small dataset. The trains in the big dataset do not end until 02.00. We have a dataset of only intercities and one with all the trains. We use the smaller dataset of intercities to check whether the methods are useful for a smaller dataset with only intercities and to check whether the program works properly. It is easier to check with a small dataset and it has a smaller running time. We then use the bigger dataset with all the trains to check whether the methods are useful for real life data.

6.1.1 The data in more detail

The given data have the routes of several trains, these routes include a train number, a departure time and arrival time and the corresponding stations. It also has the slack time of the activities. Further, the activities are defined as 'DRIVE' which is a drive from station to station or as 'DWELL' which is waiting at a station.

Further we have a list of ODT- pairs which are origin-destination-time pairs. These pairs contain an origin station, a destination station and a time at which a passenger wants to travel from origin to destination. It also has the number of passengers who want to travel from the same origin to the same destination at the same time.

6.1.2 The small dataset

The small dataset contains 457 activities of which 288 are 'DRIVE' and 169 are 'DWELL'. It contains 576 events. Furthermore, it contains 119 different trains. The 4288 passengers that will travel during the timetable, have 700 different ODT-pairs.

6.1.2 The big dataset

The big dataset contains 3741 activities of which 2066 are 'DRIVE' and 1675 are 'DWELL'. This dataset contains 4132 events. This timetable contains 391 different trains. The 9822 passengers that will travel during the timetable, have 3849 different ODT-pairs. So the big dataset is much bigger than the small dataset.

6.2 Implementation

We used Matlab 2010a to program the simulation in. We chose Matlab as it supports vector and matrix calculations very well and we have large data. But Matlab has not a great build in optimization function to calculate the model for us. Thus, we connected Gurobi with Matlab. Gurobi is one of the fastest optimization solvers, so it solves the model in no time.

The data did not have all the input needed for the simulation. We had to compute the number of passengers that end their journey at all the events, the number of passengers that change at an activity and thus also the changing activities themselves and the slack times. We could compute these with the same shortest path algorithm in chapter 5 using the original timetable. The computer we used is a laptop with 3 GB RAM and a AMD Athlon(tm) II P340 Dual-Core Processor with 2.20 GHz. All the runs are done with this computer.

6.3 Results small dataset

Now we will discuss the results of the small dataset with 10 simulation runs. The whole duration of the program was about 2 and half hours. We start by discussing the delays. Next we discuss the value of the objective function of the optimization model and the number of events in the hour it optimized. Then we will discuss the decisions made by the model followed by the total delays, the total passengers with delays and the total of passengers that cannot get home anymore. Finally, we will show the average results of 100 simulation runs, which lasted 12 and half hours, and compare them with the results of the 10 runs.

6.3.1 The generated delays

The delays we generated in the simulation are shown in the appendix in Table a.1. The average number of delays per run is 27, which is to be expected as we have a total of 576 events which can have a delay and 5% of that number is 28.8. Thus, we have an about average number of delays.

We made a scatter plot of the time of occurrence of the generated delays, in order to see whether the delays were spread out evenly or if they were all fairly close to each other. If delays occur very fast after each other they can build up to very big delays and thus the overall delay will be worse than when the delays are more spread out. The plot can be seen in figure 4. On the x-axis, we can see the time of occurrence in minutes. The data we used goes from 1290 to 1526 minutes after midnight. On the y-axis, we see the number of the simulation run and thus the dots on that height correspond to that run. We can see that runs 1 and 2 generated a lot of delays that occurred really fast after each other. We also see that runs 6, 7, 8 and 10 have a good spread of the delays.

Because we generate delays per event we have more chance of delay in the early and middle parts of the timetable. This is because most trains ride within these parts of the timetable. The last part is in the middle of the night. Therefore, there are less trains driving at that part. We conclude that these runs are about average.

6.3.2 The optimization model

In the simulation the optimization model is called and we want to know if the model had good results. We can see the average maximum and minimum of the value of the objective function of the 10 runs in table 1. The overall average value is thus 237.5, the maximum is 1372 and the minimum is 4 minutes. Thus, the value of the objective function had pretty diverse values. Of course, it all depends on the number of events there were in the hour the model optimized. The number of events that were in the hours the model optimized is given in table 2.

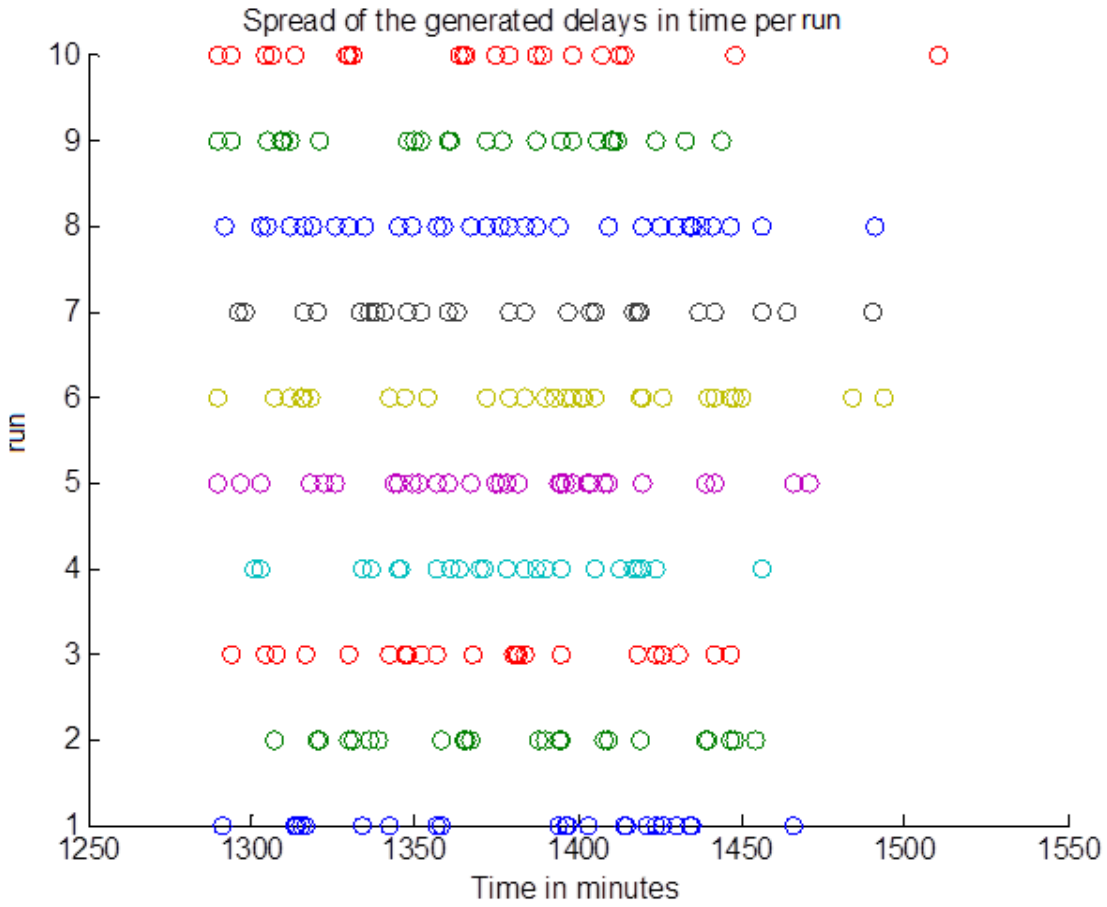

Figure 4: The time of occurrence of the generated delays per run.

Objective function	Average value	Max value	Min value
Run			
1	292.86	836	54
2	190.92	546	11
3	188.6	443	10
4	195.38	481	18
5	311.08	916	10
6	288.10	1372	4
7	183.40	636	14
8	235.88	1236	7
9	172.92	663	8
10	315.90	1172	63
Average/maximum/ minimum of all runs:	237.50	1372	4

Table 1: Table of the average, maximum and minimum value of the objective function of the optimization model of the simulation of 10 runs with the small dataset.

Events in the optimized hour	Average number	Max number	Min number
Run			
1	150.33	222	12
2	162.96	220	24
3	173.57	222	32
4	174.00	220	22
5	170.69	224	10
6	150.28	224	6
7	162.69	222	4
8	160.25	222	4
9	181.69	224	38
10	189.64	224	28
Average/maximum/ minimum of all runs:	167.61	224	4

Table 2: Table of the average, maximum and minimum number of events in the hour the optimization model optimized in the simulation of 10 runs with the small dataset.

Trains waiting	Average number	Max number	Min number
Run			
1	1.93	5	1
2	2.38	5	1
3	2.50	6	1
4	2.14	4	1
5	2.04	6	1
6	1.70	4	1
7	2.00	4	1
8	2.04	4	1
9	2.20	5	1
10	2.41	5	1
Average/maximum/ minimum of all runs:	2.13	6	1

Table 3: Table of the average, maximum and minimum number of times trains wait on a connecting train in the simulation of 10 runs with the small dataset.

We can see that the overall average of events in the hours is 167.6 which is a lot. The total number of events is 576, thus the events in the hour the model optimizes is about 29% of all the events. This means a lot of events occur at about the same time as we discussed in section 6.3.1. The overall maximum is 224. We can see in the table that the maximum is about the same, so it does not change much per run. The minimum deviates a lot with an overall minimum of 4. This, of course, is mostly the case when there are not many events in an hour, like at the end of the timetable.

Trains not waiting	Average number	Max number	Min number
Run			
1	149.21	221	12
2	161.44	220	24
3	171.83	221	32
4	172.27	218	21
5	169.09	221	10
6	149.10	222	6
7	161.24	220	4
8	158.66	222	3
9	180.00	221	38
10	187.77	221	28
Average/maximum/ minimum of all runs:	166.06	222	3

Table 4: Table of the average, maximum and minimum number of times trains not waiting in the simulation of 10 runs with the small dataset.

6.3.3 The decision made by the model

Of all the events that were used by the model, the train at that event could wait if there is a train that wants to connect or just depart as soon as possible, because there is no connecting train or because it was the best option to minimize the overall delay. The average, maximum and minimum of trains waiting and departing as scheduled are given in tables 3 and 4 respectively. We can see the average number of trains waiting is about 2.1. The maximum and minimum numbers are 6 and 1 respectively. The average number of trains not waiting is about 166.1 and is at max 222 and at lowest 3.

The number of trains that do not wait is always a big number. This is not surprising as one train will wait if there is a delay, so the rest can depart as scheduled. It cannot be that there are more trains waiting than departing as scheduled, as this will cause too much overall delay.

6.3.4 The total overall delay

After running the shortest path algorithm with both the original and the executed timetable, we could compute the total delay, the number of passengers that have a delay and the number of passengers that cannot get home anymore. The results of the 10 runs are given in table 5. We can see that the average total delay is 5257 minutes, which is the sum of delays of about an average of 361.2 passengers. Thus, the average delay per passenger is about 14 and half minutes. This is a good result, as when a delay occurs it is inevitable that the passengers in that train get a delay. Unfortunately, the results also state that an average of 17.1 passengers could not get home because of the delayed trains as they missed their connection. This can occur if there is just too much delay and the connecting train decides not to wait. The reason for this result is that the optimization model assumes the passenger waits 20 minutes for the next train. But in reality when these passengers miss their train there will be no next train to take.

	Total Delay	Nr of passengers with delay	Nr of passenger that cannot get home
Run			
1	2161	230	0
2	7267	388	34
3	5355	271	0
4	4016	403	9
5	5617	414	54
6	5327	315	26
7	5611	391	15
8	6045	530	0
9	5168	260	19
10	6003	410	14
Average of all runs:	5257	361.2	17.1

Table 5: Table of the total delay, number of passengers with a delay and the number of passengers that cannot get home anymore of the simulation of 10 runs with the small dataset.

6.3.5 The 100 simulation runs compared to the 10 simulation runs

We can see in table 6 that the average results of the 100 runs and of the 10 runs. First, we give the results of the 100 runs briefly. The average value of the objective function is 207.213 minutes and has a maximum and minimum of 2362 and 4 respectively. The average number of events in the hour that was optimized with the model, is 162.91 and have a maximum and minimum of respectively 224 and 4. The average, maximum and minimum number of waiting trains are 2.01, 8 and 1, respectively. The average numbers of trains not waiting is 161.45 and the maximum and minimum are respectively 222 and 2. The average total delay is 5093.7 minutes. The average number of passengers with a delay is 348.21 and the average number of passengers not getting home is 27.32.

We then compare the results of both runs. We see that the values are fairly the same. Most values differ only a little bit. The 100 runs had a bit lower results than the 10 runs except for the maximum value of the objective function, the average total delay, the average number of passengers with a delay and the average number of passengers not getting home. These average value of the objective function of the 100 runs is fairly the same as the 10 runs, thus the maximum value was just a one bad run. The average total delay and the average number of passengers with a delay are a bit lower for the 100 runs. But if we compute the average delay per passengers, it is about 14 and a half minutes for both runs. Therefore, the average total delay and the average number of passengers with a delay are relatively similar. One run just had a bit more luck. The 100 runs has a bigger average number of passengers not getting home because of bad luck. The maximum number of passengers not getting home was more often a big number in comparison to the 10 runs. Therefore the average is also higher.

Overall we can say that the two runs are not very different from each other as the average values all are fairly the same. We conclude that a run of 10 is about the same as a run of 100. So taking 10 runs of the simulation is sufficient.

Results of the small dataset with	100 runs	10 runs
Average value of the objective function	207.213	237.5
Maximum value of the objective function	2362	1372
Minimum value of the objective function	4	4
Average waiting trains	2.01	2.1
Maximum waiting trains	8	6
Minimum waiting trains	1	1
Average not waiting trains	161.45	166
Maximum not waiting trains	222	222
Minimum not waiting trains	2	3
Average number of events in the hour the optimization model optimized	162.91	167.6
Maximum number of events in the hour the optimization model optimized	224	224
Minimum number of events in the hour the optimization model optimized	4	4
Average total delay	5093.7	5257
Average number of passengers with delay	348.21	361.2
Average number of passengers not getting home	27.32	17.1

Table 6: Table of the average, maximum and minimum results of the simulation of 100 runs and of 10 runs with the small dataset. It contains the value of the objective function of the model, the number of waiting trains, the number of trains not waiting, the number of events in the hour the optimization model optimized, the total delay of all passengers, the number of passengers with a delay and the number of passengers that do not get home anymore.

6.4 Results of the big dataset

Now we discuss the results of the big dataset. We just saw in section 6.3.5 that 10 runs is about the same as 100 runs, so we will only take 10 runs for the big dataset. This is a good thing as one run of the simulation takes about 14 hours. This is caused by the Bellman-Ford algorithm which has to compute the shortest route for all ODT-pairs and while computing the route it has to go through all the arcs in the data. First we discuss the delays shortly. Next we discuss the value of the objective function of the optimization model and the number of events it had has input. After that we discuss the decisions made by the model followed by the total delays, the total passengers with delays and the total of passengers that cannot get home anymore. Finally, we compare the results shortly with the results from the simulation of 10 runs with the small dataset.

Run	Number of delays	Average minutes of delay generated	Average time of occurrence
1	202	12.11	1389.8
2	192	12.11	1394.6
3	192	12.11	1387.7
4	205	12.37	1384.3
5	205	12.37	1390.9
6	232	12.08	1389.6
7	232	12.08	1383.7
8	195	12.01	1385.4
9	195	12.01	1393.0
10	195	12.01	1392.3
Average of all runs:	204.5	12.13	1389.0

Table 7: Table of the number of delays, average minutes of delay generated and average time of occurrence in minutes of the delays generated during the simulation of 10 runs with the big dataset.

6.4.1 The generated delays

The number of delays, average minutes of delays generated and the average time of occurrence of the 10 runs of the simulation can be seen in Table 7. The average number of delays per run is about 204.5. We have a total of 4132 events which can have a delay and 5% of that number is 206.6. Thus, the number of delays is about average. We cannot show all the delays as that would be too many. Also a scatter plot will not be very useful as the dots overlap each other too much. Thus, we give the average minutes of delay generated and the average time at which the delay occurs. The average delay is about 12.13. This is to be expected, as the expected outcome of a random integer number from the discrete uniform distribution on the interval [6, 18] is 12. Furthermore, the average time at which the delay occurs is 1389 minutes past midnight, which is about 23.00. The average time of the events from the data is 1389. Thus, we can say the delays are spread out pretty evenly.

6.4.2 The optimization model

We can see in table 8 that we have very diverse average and maximum values of the objective functions for the optimization model. The minimum is almost the same for all of the runs. One run had a minimum of 4 and the rest had 5 or 6. So the overall minimum is 4. The average value of the runs goes from 71.75 to 131.27, which is very diverse. The overall average is 96.45. The maximum value also deviates a lot. It goes from 408 to 1829 and the overall maximum is, thus 1829. The values also deviate for the small data, thus it does not say very much as the model has different input every time.

Objective function	Average value	Max value	Min value
Run			
1	73.47	900	6
2	71.75	408	6
3	129.76	836	6
4	89.44	500	6
5	131.27	747	4
6	87.14	1078	6
7	104.84	1052	5
8	92.66	924	5
9	75.01	1829	5
10	86.13	569	6
Average/maximum/ minimum of all runs:	96.45	1829	4

Table 8: Table of the average, maximum and minimum value of the objective function of the optimization model of the simulation of 10 runs with the big dataset.

Events in the optimized hour	Average number	Max number	Min number
Run			
1	1017.20	1364	18
2	1006.55	1364	22
3	1065.91	1364	6
4	1067.60	1364	8
5	1016.33	1364	12
6	1035.60	1358	4
7	1071.42	1364	12
8	1038.21	1358	8
9	994.83	1358	4
10	1028.50	1358	4
Average/maximum/ minimum of all runs:	1036.12	1364	4

Table 9: Table of the average, maximum and minimum number of events in the hour the optimization model optimized in the simulation of 10 runs with the big dataset.

The number of events in the hour the optimization model optimized is given in table 9. The average and maximum of the number of events in the hour optimized is steady. The overall average and maximum of the runs have values of 1036.12 and 1364, respectively. The minimum however deviates from 4 to 22. This all depends on the time when the delay is generated. A delay at the end of the timetable, will give a few events for the model as there are no events anymore. The overall minimum is 4.

Trains waiting	Average number	Max number	Min number
Run			
1	6.49	15	1
2	6.33	15	1
3	7.07	17	1
4	6.85	16	1
5	6.86	16	1
6	6.92	17	1
7	7.00	17	1
8	7.10	17	1
9	6.33	15	1
10	6.49	14	1
Average/maximum/ minimum of all runs:	6.77	17	1

Table 10: Table of the average, maximum and minimum number of times trains wait on a connecting train in the simulation of 10 runs with the big dataset.

Trains not waiting	Average number	Max number	Min number
Run			
1	1011.10	1361	17
2	1000,56	1359	22
3	1059.09	1361	5
4	1061,16	1359	8
5	1009.78	1359	11
6	1029.09	1352	4
7	1064.75	1361	11
8	1031.67	1352	8
9	988.85	1352	4
10	1022.31	1351	4
Average/maximum/ minimum of all runs:	1029.70	1361	4

Table 11: Table of the average, maximum and minimum number of times trains not waiting in the simulation of 10 runs with the big dataset.

6.4.3 The decisions made by the model

Of all the events taken as input in the model, the decision to wait for a feeder train or not has been made. The average, maximum and minimum number of decisions to wait is very constant and the overall average, maximum and minimum of these are 6.77, 17 and 1, respectively. The average and maximum number of decisions to not wait is also fairly the same for all runs.

	Total Delay	Number of passengers with delay	Number of passenger that cannot get home
Run			
1	25090	1317	128
2	20785	1184	67
3	22747	1595	19
4	18785	1147	60
5	19247	1275	44
6	18265	1233	66
7	19269	1364	112
8	20066	1303	77
9	17262	931	124
10	18394	1091	92
Average of all runs:	19991	1244	78.9

Table 12: Table of the total delay, number of passengers with a delay and the number of passengers that cannot get home anymore of the simulation of 10 runs with the big dataset.

The average number deviates a little, but are all close to the overall average of 1029.70. The maximum deviates a little with an overall maximum of 1361. The minimum deviates a lot. It goes from 4 to 22 and thus, has an overall minimum of 4.

6.4.4 The total overall delay

With the shortest path algorithm, we could compute the total delay, the number of passengers that have a delay and the number of passengers that cannot get home anymore using the original and the executed timetable. We can see in Table 12, that the total delay deviates a bit. It goes from 17262 to 25090 and gives an overall average of 19991 minutes. The number of passengers that had delay also deviates and goes from 931 to 1595 with an average of 1244 passengers. Thus, the average delay per passengers is about 16 minutes. The average number of passengers not getting home anymore is 78.9, but the number deviates from 19 to 128. This is caused by the delays generated. If a lot of delays are generated at the end of the timetable, it causes many passengers to miss their connection. Thus, the number of passengers not getting home anymore is high if it happens a lot and low if it not.

Overall a result of an average of 16 minutes of delay per passengers is very good but unfortunately the average of 78.9 passengers not getting home is quit big. This is probably because there are more events in the end of the timetable. Thus, there is more chance of a delay at the end of the timetable.

Results of the	Small dataset	Big dataset
Average value of the objective function	237.50	96.45
Maximum value of the objective function	1372	1829
Minimum value of the objective function	4	4
Average waiting trains	2.10	6.77
Maximum waiting trains	6	17
Minimum waiting trains	1	1
Average not waiting trains	166.0	1029.7
Maximum not waiting trains	222	1361
Minimum not waiting trains	3	4
Average number of events in the hour the optimization model optimized	167.60	1036.12
Maximum number of events in the hour the optimization model optimized	224	1364
Minimum number of events in the hour the optimization model optimized	4	4
Average total delay	5257	19991
Average number of passengers with delay	361.2	1244.0
Average number of passengers not getting home	17.1	78.9

Table 13: Table of the average, maximum and minimum results of the simulation of the 10 runs with the big dataset and the small dataset. It contains the value of the objective function of the model, the number of waiting trains, the number of trains not waiting, the number of events in the hour the optimization model optimized, the total delay of all passengers, the number of passengers with a delay and the number of passengers that do not get home anymore.

6.4.5 The big dataset run compared to the small dataset run

In this section we are going to compare the runs with the big dataset with the runs with the small dataset. The average results of both datasets are given in table 13. We see that the average value of the objective function with the big dataset is smaller than with the small dataset. The value for the small dataset is 237.50 and for the big dataset 96.45. This is probably caused by the number of events in the big dataset. The big dataset has more events. Therefore, getting multiple delays after each other, will affect the average value of the objective function less as the model optimizes more often. This can also be seen from the maximum value. The maximum value for the big dataset is a lot bigger with a value of 1829 compared to the small one with a value of 1372. Thus, having a maximum value that big, does not affect the average value much, when we run the bigger dataset. The minimum is the same for both datasets.

The number of trains waiting is also greater for the big dataset. This also is not surprising as there are more decisions that have to be taken in the big dataset. The average, maximum and minimum of the number of trains waiting in the big dataset are 6.77, 17 and 1, respectively. The

average, maximum and minimum of the small dataset are 2.10, 6 and 1 respectively. Thus, the minimum waiting trains is the same for both datasets. We also see that the average, maximum and minimum number of trains not waiting and number of events in the hour the optimization model optimizes are both much greater for the big dataset. The reason is, of course, that the big dataset has more trains within an hour and thus also more events. The average, maximum and minimum number of trains not waiting in the big dataset are 1029.7, 1361 and 4, respectively. The average, maximum and minimum for the small dataset are 166.0, 222 and 3, respectively. Thus, the minimum is again not very different. The average, maximum and minimum number of events in the hour of the optimization for the big dataset are 1036.12, 1364 and 4 respectively. The average, maximum and minimum number of events of the small dataset are 167.60, 224 and 4 respectively. If we compare them we see that the minimum number of events is the same. This probably happened when the model optimized at the end of the timetable as there are not many events at the end for either datasets.

The average total delay is of course also bigger for the big dataset compared to the small dataset, as there are more delays generated and there are more passengers. The average total delay of the big dataset is 19991 and of the small dataset 5257. These average total delays are divided among an average of 1244 passengers for the big dataset and 361.2 passengers for the small dataset. Thus, the average delay per passenger for the big dataset is about 16 minutes and for the small dataset 14 and half minutes. Thus, the average delay per passenger is not that different. It only differs 1 and a half minute. Unfortunately the number of passengers not getting home for the big dataset is a lot bigger than for the small dataset. The numbers are 78.9 and 17.1 respectively. The percentage of the passengers that did not get home is given by dividing the number of passenger not getting home by the total number of passengers. This gives us 0.8% of the passengers did not get home with the big dataset and 0.4% with the small dataset. Thus, overall there are more passengers not getting home in the big dataset than in the small dataset. This is probably because there are more events at the end of the timetable of the big dataset. So there is a higher probability that a delay occurs at the end of the timetable.

We conclude that the average delay per passengers is about the same for both datasets. The big dataset has an average delay of 16 minutes per passenger and the small dataset 14 and half minutes. Unfortunately with the big data set about 0.8% of the passengers could not get home anymore and with the small data set about 0.4%. Thus, the big dataset has a higher percentage of passengers not getting home anymore. This is probably because the big dataset has more events at the end of the timetable. Thus, the probability of getting a delay at the end of the timetable is higher.

6.5 Sensitivity Analysis

From the results, we saw that making the assumption that passengers wait 20 minutes might not be the best choice. We think a lot of passengers do not get home anymore, because of the assumption. We think a bigger value of T might get better results. Therefore, we varied the T from the optimization model in section 4.4. We generated the delays as normally in the simulation and optimized the model with three different values for T. Thus, we had the same delays for every value of T. We take the values 20, 25 and 30 for T and run the simulation 10

times with the small dataset. The results are given in table 14. Taking T bigger will automatically result in a greater than or equal value of the objective function, as the T is a cost parameter. Thus, an average value of the objective function that is increasing when T increases is not surprising. The minimum and maximum value has remained the same for all values of T. The average number of trains waiting and trains not waiting does change a little bit. The difference is 0.01 which is too small to make a difference. The minimum and maximum number of trains waiting and number of trains not waiting do not change as T changes. We see that the total delay, the number of passengers with a delay and the passengers not getting home also do not change as T changes. So, we conclude that changing T does not influence the number of passengers not getting home and also not the total delay and the number of passengers with a delay.

Model with	T = 20	T =25	T = 30
Average value of the objective function	195.07	196.23	197.31
Maximum value of the objective function	1163	1163	1163
Minimum value of the objective function	6	6	6
Average waiting trains	2.04	2.04	2.05
Maximum waiting trains	6	6	6
Minimum waiting trains	1	1	1
Average not waiting trains	162.91	162.91	162.90
Maximum not waiting trains	222	222	222
Minimum not waiting trains	3	3	3
Average total delay	4319	4319	4319
Average number of passengers with delay	294.4	294.4	294.4
Average number of passengers not getting home	23	23	23

Table 14: Table of the results from 10 runs of the simulation with the small dataset for T = 20, T = 25 and T = 30 with the same delays. It shows the average, maximum and minimum value of the objective function, number of trains waiting, number of trains not waiting, the average total delay, the number of passengers with a delay and the number of passengers not getting home

7 Conclusion

We made a simulation of the real world with a given timetable from NS. Of course a simulation which is really the same as the real world is not possible. Therefore, we made some assumptions. In the simulation we used an off-line delay management method to try and minimize the overall delay of all passengers. For this we used an optimization model that decides whether the trains should wait for a delayed train or they just depart as scheduled. After all the decisions are made, we have the executed timetable which we can compare to the original one. We used a shortest path algorithm to compute the arrival times of the passengers. With these arrival times we can now compute the overall delay passengers had, the number of passengers with delays and the number of passengers that could not get home. Finally, we checked whether changing the T from the optimization model in 4.4 would give better results.

From the results of the simulation of 10 runs with the small dataset, we can see that every optimization from the model results in about two trains waiting for a feeder train and the rest will depart as scheduled. This is not a very surprising result as when there are too much trains waiting, the delay will just be bigger. The total delay all passengers had was 5257 minutes. This was divided among about 361 passengers and that is about 14 and half minute of average delay which is not a bad result. Unfortunately, about 17 passengers could not get home. This is because the optimization model thinks these passengers just wait for 20 minutes. When we compare these results with the results of the 100 runs they did not differ that much. So, we can conclude that simulating with 10 runs is not different from 100 runs. Therefore, simulating with 10 runs is sufficient.

From the results of the simulation of 10 runs with the big dataset, we see that the every optimization results in about seven trains waiting for a feeder train and the rest does not wait. Again this is not very surprising, as when there are too much trains waiting the delay will just be bigger. The total delay the passengers had was 19991 minutes. This was divided among 1244 passengers. Thus, the average delay of a passenger is about 16 minutes which is a good result. Unfortunately, about 78.9 passengers could not get home anymore. Again this is because the optimization model thinks these passengers just wait for 20 minutes. If we compare the results of the two datasets we conclude that the average delay per passengers is about the same. The big dataset has an average delay of 16 minutes per passenger and the small dataset 14 and a half minute. Unfortunately, the number of passengers not getting home anymore is a lot more for the big dataset than for the small dataset. The numbers of passengers not getting home are 78.9 and 17.1, respectively. The big dataset has a lot more events. This might cause that more passengers cannot get home anymore. We generate delays per event. Therefore, the big dataset has a higher probability of getting delays at the end of the timetable.

Because we think that giving T the value of 20 in the optimization model from section 4.4 might be a bad assumption, we did a little sensitivity analysis. We varied the value of T to 25 and 30 and checked whether this would give us different results than with a value of 20. For these runs we used the same delays. The results were the almost the same for all values of T. Thus, we conclude that changing the value of T will not change the overall results.

Overall we think an average of 16 minutes of delay per passenger is a good result as we used real life data. Unfortunately about 78.9 passengers could not get home anymore. But maybe if we use a better optimization model or running the simulation with a daytime schedule, so the passengers that cannot get home will have a next train, we could use these methods in practice to minimize delay.

8 Future research

Given the results and conclusion, we could probably use the methods with a bigger dataset. The computation times will of course be a lot greater. Also taking a much greater number of runs for the simulation will get a better result as randomness will be smaller.
We made some assumptions to validate our methods as described in section 3.2. We could probably get better view of the reality, if we would know the chance of getting a delay in more detail and how long a delay will be. With this the delays are more like in real life and thus the results will give a better idea for practical uses.
A change of trains takes a few minutes when the platforms of the change are not next to each other. Therefore, this could be implemented in the model which might give a slight better view of the reality, but it is hard to approximate the number of minutes to take for the change. A too large number will let passengers wait for no reason and a too small a number will let some passengers miss their train.
Also the trains that wait could cause problems within a station. But there are other models that keep this in account. Thus, using another optimization model, like the one in Dollevoet et al. (2012c) will solve this problem. Of course the model will be a lot harder to implement. Actually any delay minimization model could replace the one we used. Thus, by taking a model that does not need the assumptions we made, will result in a better approximation of reality.

Appendix

Run	1	2	3	4	5	6	7	8	9	10
Delay nr										
1	9	6	10	14	9	16	14	16	12	18
2	13	18	18	13	11	13	14	7	18	11
3	8	9	17	10	11	16	17	6	17	16
4	11	14	11	16	15	11	6	10	11	13
5	6	15	16	18	12	6	14	9	14	16
6	13	16	6	10	18	17	8	13	10	18
7	18	17	14	11	6	16	12	17	14	18
8	11	18	7	16	7	11	12	16	12	13
9	8	13	16	11	17	7	7	9	6	8
10	10	14	12	6	11	10	16	8	10	18
11	11	7	13	16	17	17	13	6	12	11
12	11	12	17	12	16	8	10	9	6	14
13	6	9	15	13	10	14	13	13	18	14
14	11	11	16	18	7	10	18	17	6	10
15	7	16	6	9	14	7	9	14	18	13
16	13	8	18	12	14	13	13	8	9	18
17	10	16	10	18	17	17	17	13	13	18
18	11	9	12	7	16	7	18	12	7	18
19	11	16	14	11	12	13	16	9	15	14
20	15	13	7	13	7	10	6	16	7	6
21	7	14	8	9	15	16	16	12	11	18
22	13	15	13	14	13	11	16	6	17	12
23	10	7	9	9	12	12	18	14	18	7
24	14	13		9	11	11	11	18	10	
25		11		17	13	12	10	6	14	
26				10	7	10	16	11	6	
27					14	15	9	9		
28					15	18	17	18		
29					14	14	16	7		
30					8	9		16		
31					13			18		
32					18			18		
Average delay	10.71	12.68	12.39	12.38	12.50	12.23	13.17	11.91	11.96	14.00
Total nr of delays	24	25	23	26	32	30	29	32	26	23

Table a.1: Table of all the generated delays in the simulation of 10 runs with the small dataset. It also contains the average delay and total number of delays.

References

T. Dollevoet, D. Huisman, M.Schmidt, and A. Schöbel, “Delay Management with Rerouting of Passengers”. *Transportation Science*, 46(1):74-89 (2012b)

T. Dollevoet and D. Huisman “Fast Heuristics for delay management with passenger rerouting”. *Technical Report EI2011-35, Econometric Institute Report Series* (2011)

T. Dollevoet, D. Huisman, M.Schmidt, and A. Schöbel, “Delay Management including Capacities of Stations”. *Technical Report EI2012-22, Econometric Institute Report Series* (2012c)

T.Dollevoet, F. Corman, A. D’Ariano, and D. Huisman, “An iterative optimization framework for delay management and train scheduling”. *Technical Report EI2012-10, Econometric Institute Report Series* (2012a)

A. Schöbel. Integer Programming Approaches for Solving the Delay Management Problem. *In F. Geraets, L. G. Kroon, A. Schöbel, D. Wagner, and C. Zaroliagis, editors, Algorithmic Methods for Railway Optimization, number 4359 in Lecture Notes in Computer Science, pages 145-170. Springer (2007)*