Bachelor Thesis Econometrics

Delay management with capacity considerations.

Should a train wait for transferring passengers or not, and which train goes first?

Kai Huiskamp

348882

Content

Chapter 1: Introduction

In the train sector there are several points of issue. Maybe the most important one of these issues is delay management. This is because one of the main goals of a railway company is to minimize the (extra) traveling time of their customers. Of course, the train travels as fast as it can. But sometimes there occurs a delay. The task is now to update the planned timetable π to a *disposition timetable X* in such a way that the inconvenience for the passengers is as small as possible. When making this disposition timetable we have to make two decisions. The first decision is to decide which connections should be maintained in case of delay. For example: if you are in a delayed train A, and you need to transfer to train B. Should this train B wait and give more passengers delay, or should this train depart on the scheduled time with the possibility that passengers will miss their transfer. The second decision is to decide which train goes first. This occurs when there is limited capacity on the railway network, and two trains or more are driving on the same track. If an Intercity then comes behind a Sprinter there would be no chance to overtake this Sprinter, and thus there would be extra travelling time for the passengers.

In the article of Schachtebeck & Schöbel (2010) an integer programming formulation is given to solve this problem. In this article, this formulation is tested on data from a German railway network. In my research, I am also going to implement this formulation, and use it on data of the NS: the Dutch railway company. First I will solve the uncapacitated delay management problem (UDM) and after that I will solve the capacitated delay management problem (CDM). When solving both problems I must determine the delay for each event. Because there is no real data of the delay I will use two methods. In the first method I will solve both problems with 'known' delay: some events of trains are delayed by hand. These events are selected by their chance on delay in reality.

The second method to determine the delay is described in the paper of Schachtebeck & Schöbel (2010). I will generate 100 times a random delay vector, and try to see how this impacts the results. First I will compare the results of Schachtebeck & Schöbel, (2010) and the results of my own research. But because the focus of this research is to solve the problem, I am also going to discuss some operational results. The content of my thesis is as follows: in chapter 2 I will explain the model and especially which sets of parameters I need. In chapter 3 I show and explain the formulations of the UDM and CDM problem. Chapter 4 describes the methods used to get the results. Also in this chapter is described how the delays are derived. After chapter 4, I will show the results in chapter 5 and tell the conclusion in chapter 6. Finally, I discuss some further research that could be done.

Chapter 2: The Model

As described before I am going to implement two models. To implement these two I make use of the concept of an *event-activity network* (Schobel, 2007). An event-activity network is a directed graph *N = (E, A)*, where E consist of departure events E_{dep} and arrival events E_{arr} . These events are connected via activities (arcs) *A*. For the UDM model we need two types of activities: train activities and transfer activities. A train activity can either be a driving train, $A_{drive} \subset E_{dep} \times E_{arr}$ or a train which is waiting on a station, $A_{wait} \subset E_{arr} \times E_{dep}$. A transfer activity is always a connection

between an arrival of train A, and a departure of train B: $A_{transfer} \subset E_{arr} \times E_{dep}$. For the CDM problem I need an extra set of activities: headway activities. A headway activity describes the capacity of a part of the network, which will be explained in section 2.4. A headway activity is a connection between two different departures, from the same station: $E_{dep\,i} \times E_{dep\,j}$. An example of an event-activity network is described in Figure 1. Events are shown as a circle, the solid arcs represent train activities, the dashed arcs represent transfer activities and the dotted arcs represent headway activities.

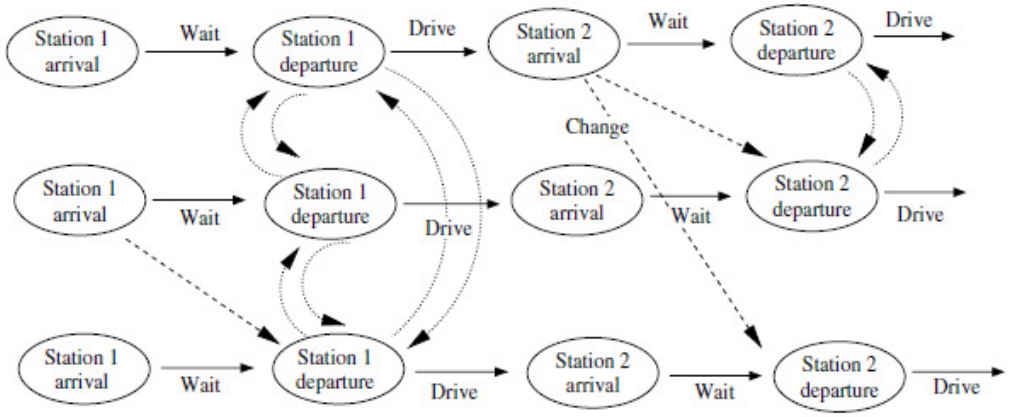

Figure 1, Example of an event-activity network (Source: Schobel, 2007)

So, the data for my research is provided by the NS, the Dutch railway company. This data covers the part of the railway network between the cities Eindhoven, Den Bosch, Tilburg, Nijmegen, Arnhem and Utrecht. The specific stations and lines are shown in Figure 2. Each line can either be a Sprinter, which stops at every station, or an Intercity, which only stops at the 'bigger' stations.

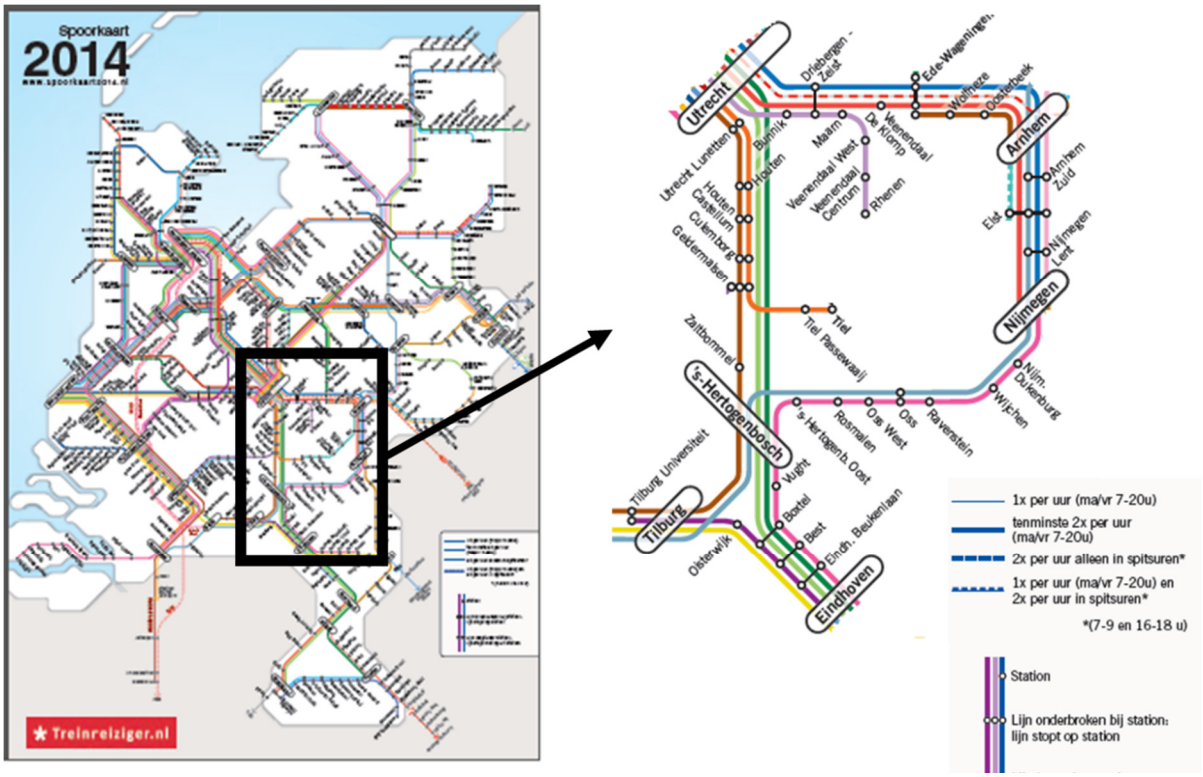

Figure 2, Railway network that is investigated in this thesis. (Source: Treinreiziger.nl)

In the data are 3 different sets, which are together describing an event-activity network without capacity. In the first set, every event is described, in the second set the train activities are described and in the third set the transfer activities are described. These sets are explained in more detail in sections 2.1 until 2.3. I need to determine the headway activities myself, this set is needed for the CDM problem. I will explain this in section 2.4, with making use of the other 3 sets and the known capacities.

2.1 Events

An event can either be an arrival at a station, or a departure from a station. Every event is scheduled at a certain time, this gives us a timetable π. If a delay occurs, we want to update this timetable into a *disposition timetable X*. In the data set the time is measured in the number of 6 seconds after midnight. So, one minute is stated as 10 time units. Also stated in the data is how much passengers want to get off the train. Only when an event is an arrival, passengers will get off the train. Next, we have got the service number. This number shows us which line is involved in this event. The first 2 or 3 digits determine the line; the last two digits determine the direction and the time. There is also a column for the station of this event. Last, there is a vector with the delay for every event. For example, an event can be delayed because the train driver was late.

2.2 Train activities

The second set involves activities of a train. As mentioned before, a train activity can either be a driving activity or a waiting activity. These two activities can be treated as the same because they have got the same parameters, variables and restrictions. First, there is an index for each activity. This index is needed for implementing the problem in AIMMS. The train activities are always a pair of 1 arrival event and 1 departure event. For each train activity a minimal time is needed: $L_a , a \in A_{train}$. Last, there is a vector with the delay for every activity. This can occur because of an accident with the train, or because of a failure of the train. The waiting and driving activities are combined in the train activities: $A_{train} = A_{wait} \cup A_{drive}$

2.3 Transfer activities

In the third set the needed transfer activities are described. A passenger can transfer from arrival event A to departure event B, such that he or she can take another line into another direction. For such a transfer a minimal time is needed, which is also mentioned as L_a, with $a \in A_{transfer}$. Again, every activity has got its own index. The total number of passengers who need this transfer is also stated. When this transfer is not maintained, these people will have to wait until the next train leaves. With help of this set I have determined the extra waiting time of a passenger, when he misses his transfer: $T_a, with\ a \in A_{transfer}$. Every line has got a structured timetable. So they leave every hour, every half hour, etc. Some lines are sharing the same track, and therefore the number of trains that a passenger can take in one hour increases. As example, if there are two train lines on the same track, both with a half-hour service, the service for the passengers becomes a quarter service. By knowing the line which is involved in a transfer activity I know also which lines can be taken by the passengers, and how long they have to wait before the first train of these lines leaves. This extra waiting time is shown in Table 1, per line.

Line:	800	1900	3000	3100	3500	3600	4400	5200
Extra time:	15 min	30 min	15 min	15 min	15 min	30min/ 15min	30 min	30 min
Line:	**6000**	**7400**	**7500**	**7600**	**9600**	**13600**	**16000**	**174000**
Extra time:	30 min	30 min	30 min	30 min	30 min	30 min	30 min	30 min

Table 1, Minimal extra waiting time for passengers when there transfer is not maintained, per line. (For line 3600 it differs because of the different departure stations.

Lines 800&3500 and 3000&3100 have got a quarter service, because there are 2 InterCity's driving at the same track. Also, some other InterCity's have got less extra waiting time, because passengers can also take a Sprinter train. When a passenger needs a Sprinter, I can't say that he can also take an InterCity, because I don't know his final destination. We can see now that some lines have got less extra waiting time, so it's less important to maintain the transfers to these lines. But if a lot of passengers needs this transfer, it becomes important again. Very important are the passengers who want to transfer to a last train of the day for that track. When they miss there transfer, they will have to wait all night long. Therefore, the penalty time for these transfers should be very high, such that the optimization process will always maintain these transfers.

2.4 Headway activities

By using the previous 3 sets I can solve the UDM problem. But in reality there is also a capacity on the railway network. For example: between Nijmegen en Arnhem there is only one track for each direction, so when a Sprinter leaves Nijmegen first, the Intercity can never overtake this train until Arnhem. This example can give the passengers of the InterCity extra traveling time. That's why it is important to make a decision which train is allowed to go first. To model these capacities (and decisions) I use a 4th set of parameters: Headway activities. As described before, a headway activity is a connection between two departures from different lines, from the same station: $A_{headway} \subset E_{dep\ i} \times E_{dep\ j}, with\ dep\ i\ and\ dep\ j\ from\ the\ same\ station.$ When there is, for example, only one track for each direction at a certain line, only one train can leave per time unit from a station which is involved with this track. So, a decision variable g_{ij} has to decide which of these two departures i and j may leave first. For each kind of headway activity I use a pair of activities: $(i,j)\ and\ (j,i)$. This is necessary because before solving the model I don't know yet which event will take place first. To conclude: two departure events are only combined by a headway activity if:

- The departure events occur at the same station.
- At this station, there is room to overtake the other train (there is more than 1 platform).
- The two trains are traveling in the same direction.
- The scheduled time of the two departures is maximally 2 hours of each other.
- There are capacity constraints, namely:
    - 1 track for both Sprinters and InterCity's: there is an activity between both kinds of trains.
    - 1 track for Sprinters and 1 track for InterCity's: only trains with the same type have got a headway activity.

In our network there are 39 stations, but for the headway activities I do not need all stations. Only the stations were there is place to overtake a train, or where trains can go into a different direction, are important. These stations, and there special properties, are shown in Figure 3.

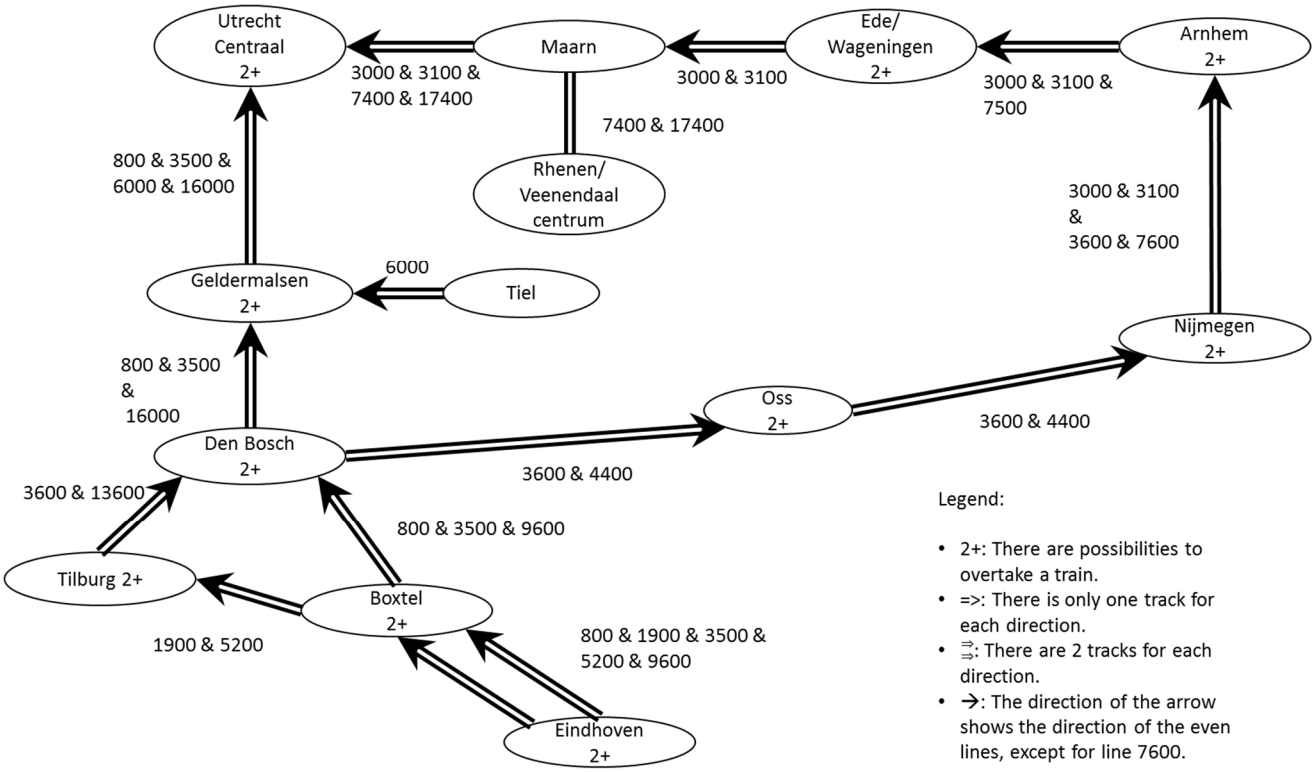

Figure 3, Railway network with the important stations and tracks.

For each station with two platforms or more I have to decide which train leaves first. The stations Maarn, Tiel, Boxtel, Geldermalsen and Rhenen are (also) important because they provide different directions for the trains. For example, a train in Maarn can either go to Arnhem or to Rhenen. I have left out two things: the stations between Boxtel and Eindhoven, and the stations between Geldermalsen and Utrecht Central. This is because I think their (limited) capacity is not necessary to take into account. To model the headway activities we take a look at Figure 4 and 5. In these figures are two different kinds of headway activities shown.

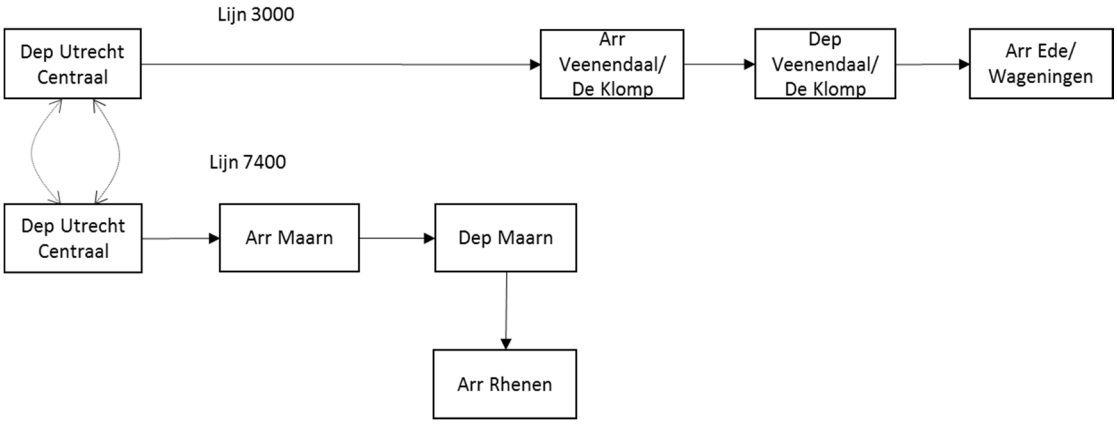

Figure 4, Example of a Semi Headway activity.

In both of the examples there is capacity: only one train can leave at the time, and this train cannot be overtaken by another train. But there is one main difference: in Figure 5 the headway activity at Nijmegen is followed by an event at Arnhem, while the two lines in Figure 4 have only got a combined event at Utrecht Central. So, in Figure 5 we can say that the train which leaves first must

also arrive first at Arnhem, but in the example of Figure 4 this is not possible. Because of this difference we need two kinds of headway activities to model the capacity: $full\ headway\ activities\ H_f$ and $semi\ headway\ activities\ H_h$. These two activities are further described in sections 2.4.1 and 2.4.2.

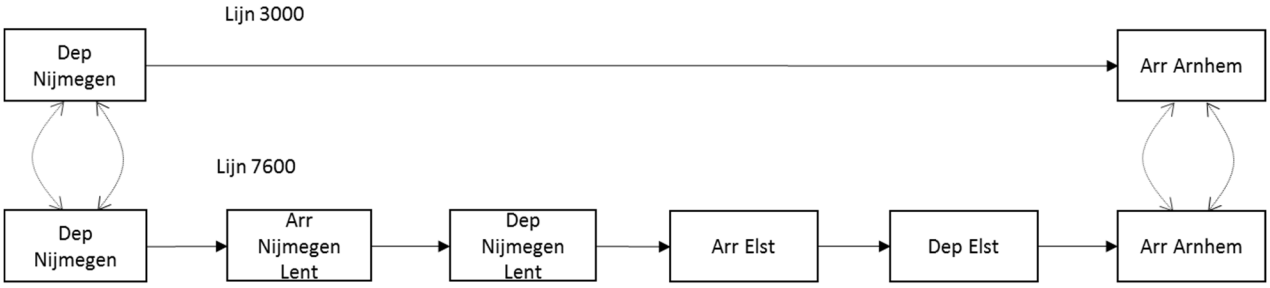

Figure 5, Example of a Full Headway Activity.

2.4.1 Full Headway Activities

For the full headway activities we look at Figure 5. As I already said, the two lines have got two common stations: a departure at Nijmegen and an arrival at Arnhem. Because of the railway capacity there is no place to overtake another train. In Dollevoet (2013) a method is described to model these activities. Let's say the departures at Nijmegen are events $i\ and\ j$. The arrivals of these trains at Arnhem are stated as $i'\ and\ j'$. If departure i takes place before departure j, then arrival i' should also be scheduled before arrival j'. The minimal time between two departures and arrivals $L_{a,}\ a \in H_f$, is always two minutes. For each pair of headway activities, we define both a pair of constraints for the departures of the train and a pair of constraints for the arrivals. For each important departure station, the lines with a full headway activity are shown in Table 2. Also shown is the common next arrival station of these lines. With help of Table 2 I determined which events have got a full headway activity. I used the program Matlab, the Matlab code can be found in the appendix.

Depart Station:	Eindhoven		Tilburg		Den Bosch		Oss	
	Lines:	**Next station:**	**Lines:**	**Next station:**	**Lines:**	**Next station:**	**Lines:**	**Next station:**
	800&3500	Den Bosch	3600& 13600	Den Bosch	800&3500	Utrecht	3600 & 4400	Nijmegen
	5200&6900	Boxtel			801&3501	Eindhoven	3601 & 4401	Den Bosch
					3600&4400	Oss		
					3601&13601	Tilburg		
Depart Station:	Nijmegen		Arnhem		Ede/Wageningen		Utrecht	
	Lines:	**Next station:**	**Lines:**	**Next station:**	**Lines:**	**Next station:**	**Lines:**	**Next station:**
	3000&3100&3600 &7601	Arnhem	3000&3100 & 7500	Ede/ Wageningen	3000&3100	Arnhem	801& 3501	Den Bosch
	3601&4401	Oss	3001&3101 &3601& 7600	Nijmegen	3001&3101 &7501	Utrecht	3001 & 3101	Geldermalsen
							7400 & 17400	Rhenen

Table 2, Full Headway activities between lines, per station. (Geldermalsen is not included because they have got no Full Headway Activities)

2.4.2 Semi Headway Activities

Beside the full headway activities I also create the semi headway activities. The semi headway activities are introduced in Schachtebeck & Schöbel (2010). As is already said, these semi headway activities have only got a common departure station. Because of this, I cannot say that the train which departs first must also arrive first at the following common station. For example, in Figure 4 there are two trains that want to depart from station Utrecht Central. The Sprinters goes to Maarn and then to Rhenen, the Intercity goes straight to Geldermalsen. Because of the capacity, the InterCity cannot overtake the Sprinter until Maarn, but after Maarn the track is free to go. So, I have to model this without making use of arrival events at Maarn (the InterCity has got no arrival event here). To solve this I make use of a minimal time L_{ij} between departure i and departure j.

I distinguish two different kinds of semi headway activities: 2 events with the same train types and 2 events with different train types. For the first of these activities, the minimal time between the departures of the two trains is always 2 minutes. For the second kind of activities the minimal time is 2 minutes if the InterCity leaves first, but when a Sprinter departs before an InterCity, this minimal time needs to be large enough, such that the extra time for the Intercity is modelled as it should be. I found these minimal times at the site of the Dutch Railway Company (www.ns.nl). The depart stations and lines with a semi headway activity are shown in Table 3. The minimal time in this table is the time when a Sprinter leaves before an InterCity. Again, with help of this table, I determined which events have got a semi headway activity by using Matlab. Also this code can be found in the Appendix.

Depart Station:	Eindhoven		Tilburg		Den Bosch		Geldermalsen	
	Lines:	**Minimal time**	**Lines:**	**Minimal time:**	**Lines:**	**Minimal time:**	**Lines:**	**Minimal time:**
	800&1900	2 minutes	1901&5201	8 minutes	800/3500 &16000	6 minutes	6000 &16000	2 minutes
					801/3501 &9601	12 minutes		

Depart Station:	Utrecht Central	
	Lines:	**Minimal time:**
	801/3501 & 6001/ 16001	10 minutes
	3001/3101 & 7401/ 17401	8 minutes
	6001 & 16001	2 minutes

Table 3, Semi Headway Activities between lines, per station.

Chapter 3: Model formulations

Now I know which sets of parameters I need I can formulate the mathematical problems. But for these formulations I need to describe the most important parameters: the delay parameters. Since without delay there is no need for a solution. There are two types of delays: an event can be delayed or an activity can take longer than expected. An event delay is mentioned by d_i , $with\ i\ \in E$, and an activity delay is mentioned by d_a , $with\ a\ \in A_{train}$. Both kind of delays can cause different kinds of extra travelling/waiting time, as is described in Schachtebeck & Schöbel (2010).

Now, let's start with the objective function. The objective function is the same for both formulations, and is as follows:

$$\text{min f(x,z)} = \sum_{i \in E} w_i(x_i - \pi_i) + \sum_{a \in A_{Transfer}} (z_a w_a T_a)$$

With:

- w_i : $number\ of\ passengers\ getting\ off\ at\ arrival\ event\ i.$

- w_a : $number\ of\ passengers\ that\ are\ using\ transfer\ activity\ a.$

This function is divided into two parts: one part for the extra traveling time and one part for the extra waiting time. For the extra travelling time I take the extra time of all passengers that are arriving too late. For the extra waiting time I measure the total extra waiting time of all passengers that are missing there transfer. The sum of these two parts must be minimized to get an optimal solution. This is done by the optimization program AIMMS. As is already said there are 2 formulations: UDM and CDM. These formulations are shown and described in section 3.1 and 3.2.

3.1 Uncapacitated Delay Management

For the formulation of the UDM problem I use the first 3 sets. The formulation is as follows:

(UDM) $$\text{min f(x,z)} = \sum_{i \in events} w_i(x_i - \pi_i) + \sum_{a \in activities} z_a w_a T_a$$

Such that

$$x_i \geq \pi_i + d_i \ \forall\ i \in E \tag{1}$$

$$x_j - x_i \geq L_a + d_a \ \ \forall\ a = (i,j) \in A_{train} \tag{2}$$

$$M z_a + x_j - x_i \geq L_a \ \forall\ a = (i,j) \in A_{transfer} \tag{3}$$

$$x_i \in \mathbb{N} \ \forall\ i \in E \tag{4}$$

$$z_a \in \{0,1\} \ \forall\ a \in A_{transfer} \tag{5}$$

The goal of the objective function is to minimize the total extra traveling time. Restriction 1 says that a disposition event x_i is always after the scheduled time plus the delay of this event. Restriction 2 says that the time between two events of an activity must be large enough, even if a delay occurs

during this activity. Last, restriction 3 takes care of the transfer times: a transfer of a passenger takes a minimal time L_a, thus the departure of his next train must be later then his arrival plus the minimal time needed. In this restriction the first decision variables is used (beside the disposition timetable x):

$$z_a = \begin{cases} 0 & if\ transfer\ activity\ a\ is\ maintained, \\ 1 & otherwise, \end{cases} \quad \forall\ a\ \in\ A_{transfer}$$

This decision variable is necessary because when a transfer is not maintained, it doesn't matter that the departing train leaves earlier then the arriving train of an activity $a, a\ \in\ A_{transfer}$. When z_a is equal to 1, the big M makes a transfer activity where the departing train event occurs first also feasible, such that the problem can still be solved. This big M is calculated as follows (Schachtebeck & Schöbel, 2010): $M = \ _{i\in Events}^{Max} d_i + \sum_{a\in A_{train}}(d_a) + \sum_{(i,j)\in A_{Head}^{Back}}(\pi_i - \pi_j + L_{ij})$, with A_{Head}^{Back} the set of Headway activities for which holds: event j is planned before event i. The last part of the summation is not yet necessary because there are no headway activities in this problem.

3.2 Capacitated Delay Management

Now I will discuss the CDM formulation. Within this formulation there are also restrictions that model the capacity by using the headway activities. The CDM model formulation is as follows:

(CDM) $$\min f(x,z) = \sum_{i\in events} w_i(x_i - \pi_i) + \sum_{a\in activities} z_a w_a T_a \quad (1)$$

Such that

$$x_i\ \geq\ \pi_i + d_i, \forall\ i\ \in E \quad (2)$$

$$x_j - x_i\ \geq\ L_a + d_a,\ \forall\ a = (i,j)\in\ A_{train} \quad (3)$$

$$Mz_a + x_j - x_i\ \geq\ L_a,\ \forall\ a = (i,j)\in\ A_{transfer} \quad (4)$$

$$Mg_{ij} + x_j - x_i \geq\ L_{ij},\ \forall\ (i,j)\ \in A_{headway} \quad (5)$$

$$Mg_{ij} + x_{j'} - x_{i'} \geq\ L_{ij},\ \forall\ (i,j)\ \in\ A_{headway\ full} \quad (6)$$

$$g_{ij} + g_{ji} = 1, \quad \forall\ (i,j)\ \in\ A_{headway} \quad (7)$$

$$x_i\ \in\ \mathbb{N}, \quad \forall\ i\ \in E \quad (8)$$

$$z_a \in \{0,1\}, \quad \forall\ a\ \in\ A_{transfer} \quad (9)$$

$$g_{ij}\ \in \{0,1\}, \quad \forall\ (i,j)\ \in\ A_{headway} \quad (10)$$

To model the capacity of the network, restrictions 5, 6 and 7 are added to the formulation. In (5) it is taken into account that the time between x_i and x_j is large enough and also that these events take place in the right order. This order is chosen by AIMMS with help of the decision variable g_{ij}:

$$g_{ij} = \begin{cases} 0 & if\ event\ i\ takes\ place\ befor\ event\ j, \\ 1 & otherwise, \end{cases} \quad \forall\ (i,j) \in A_{headway}$$

To repeat: there is a headway activity (i, j) and there is a headway activity (j, i). So, for each pair of headway activities there are two restrictions, this is because I don't know which of the two events will be chosen first. Therefore, only of these two restrictions ensures that the events take place in the right order. For the full headway activities I also use a second restriction (6). This restriction says that the train who leaves first at common station *A* must also arrive first at the following common station *B*. Again, these restrictions comes in pairs. Last, restriction 7 says that if event A takes place before event B, then B can't take place before A, which is quite logical. Finally, when using this formulation to solve the CDM problem, an exact solution is giving when the 'never-meet property' for headway activities holds (Schachtebeck and Schobel, 2010).

Chapter 4: Methods

Now I have implemented the formulations in AIMMS I need to decide which methods I want to use to get the results. The first step is to distinguish two types of methods: one type when delay is known and one type when the delay is simulated (not known). In the first method are some events chosen, and I assigned a given delay to these events. The events are not randomly chosen: there is looked which lines have got the most delay in reality. The next method is to simulate the delay. Events are randomly picked, and are assigned with delay which is uniformly distributed. What I especially want to investigate by using these two methods is explained in sections 4.1 and 4.2.

4.1 Methods when delay is know

There are two different approaches to get results when the delay is known. The first approach is to determine if the problem can be solved. If this is possible, you can investigate the solving time, the number of iterations, type of problem etc. This is the first thing I want to determine. The second method when the delay is known is more an operational method. I want to determine after how much minutes of delay a transfer is no longer maintained. I do this by looking at only one transfer and give the arriving train of this transfer a certain delay. In this research I chose 5 different transfers, which all take place at the station of Den Bosch. After setting the delay AIMMS will solve the problem and will determine of these transfer will be maintained or not. When repeating this method for different amounts of delay there will be a turnover point at some moment, where the transfers are no longer maintained. Of course, this method can be used for all transfers, but because of time limitations this is not possible.

4.2 Methods when delay is not known

After using the method with known delay I will us a method with unknown delay. Also for this method I use two different approaches: one approach to test if the problem can be solved and one approach to get some operational results. During this method I will give AIMMS 100 different situations to solve. Every situation (iteration), about 10 percent of the events will be randomly selected and assigned with some delay. This delay is simulated following a uniform distribution between 1 and 15 minutes of delay. To check if the different situations can be solved I let AIMMS save some parameters: the solving time, the gap between the lower and upper bound and the value of the objective function. The gap between the LB and UB is only important when the problem isn't finished in time. The maximal time for solving a problem is set at 15 minutes, because otherwise the time required is too long.

The second approach is somewhat comparable with the operational approach from section 4.1. Again I choose 5 transfers to investigate. But this time not the arriving event of these transfers will be delayed, but the delayed events are randomly chosen. The question is whether or not these transfers will be maintained, and why they will be maintained. Therefore AIMMS must not only save the decision variable z_a, but also the arrival times of the arriving trains. Then I can look after how much minutes of delay the transfer is no longer maintained and whether this result matches with the result of the previous method.

Chapter 5: Results

Now I have explained all sets, model formulations and methods I will analyse the results. I do this both for the UDM problem as for the CDM problem, such that I can not only analyse the results separately, but also compare them with each other for each problem. But first, to check if both models are implemented correctly, I will solve the problems without any delay in section 5.1. In section 5.2 the results of the problems with known delay are shown and in section 5.3 the results of the CDM problem with simulated delay are shown.

5.1 Results with no delay

Why should I first solve the problems without delay? I do this to check if the model formulations work. If there is no delay, the optimal solution should give an objective value of 0: there is no extra waiting time for all passengers, because every train can depart and arrive at the scheduled time. In Figure 6 the results for both problems are shown. We can see now that both optimal solutions indeed have got an objective value of 0, and therefore I conclude that the models are working.

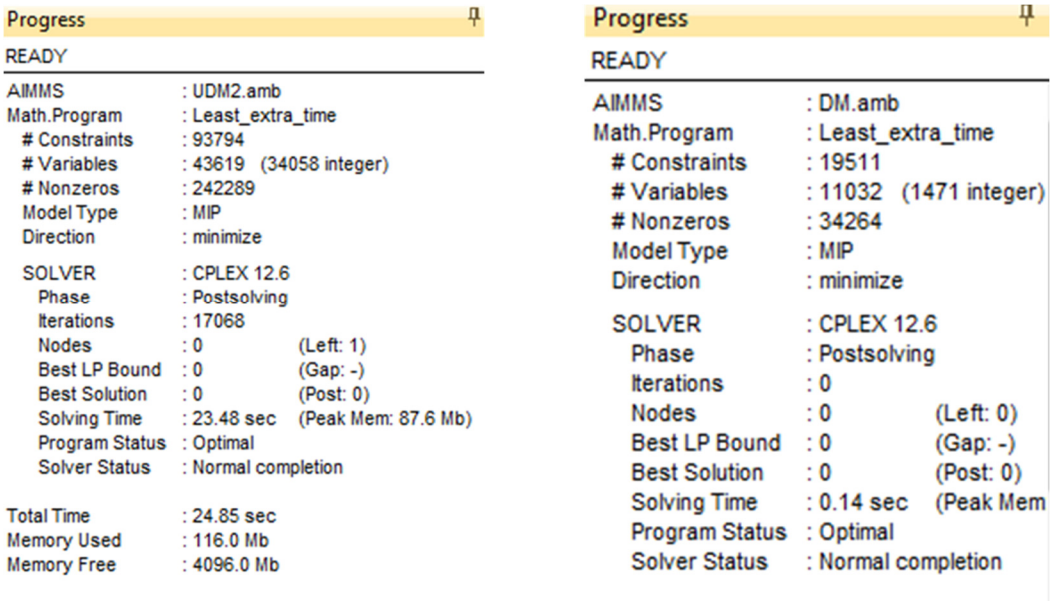

Figure 6, Results UDM & CDM problem without delay.

5.2 Results UDM and CDM with known delay

Now we have seen that the models are working I solved the problems with fixed delay. First, about 50 arrival events are delayed with a delay between 1 and 10 minutes. With this delay I want to know if the problems can be solved, and how long it takes AIMMS to solve the problems. The results of the solving process for both problems is shown in Figure 7. It is easy to see that the problems can be solved and that it takes much more time to solve the CDM problem than the UDM problem. Also, due to the extra headway constraints that comes with the CMD problem, this problem has got a higher objective value.

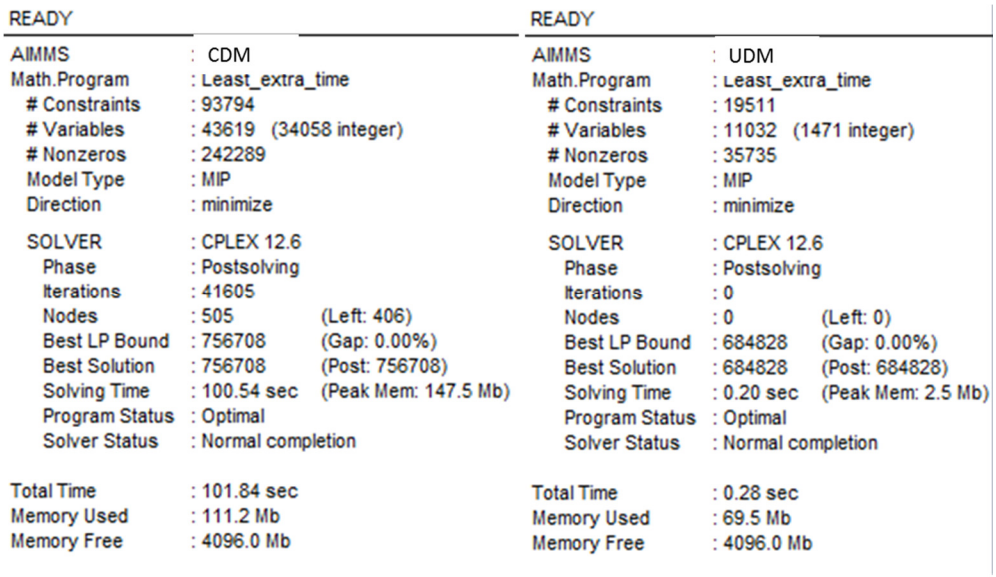

Figure 7, Results of the UDM & CDM problem with fixed delay.

It is nice that I can solve the problems, but I also want some operational results. The main operational question is: after how much minutes of delay will a transfer be no longer maintained? As I described before, I will determine this turnover point by solving the problem multiple times, every time with a different delay for one and the same arrival event. I repeated this method for 5 different transfer activities which all take place at the station of Den Bosch:

INDEX ACTIVITY	# PASSENGERS	FROM LINE	TO LINE	MINIMAL TIME	SCHEDULED TIME	PENALTY TIME
264	63	827	3627	40	130	300
1071	56	9660	4464	20	20	300
1120	115	3664	864	40	130	150
1290	47	3571	13673	40	60	300
1401	36	4481	881	40	60	150

Table 4, Examined transfer activities and their characteristics.

Of course, the minimal time needed for a transfer is very important, but a delay is only annoying when the extra time is longer than the scheduled time minus the minimal time that is needed for a transfer. When this occurs, passengers are missing their transfer, or the train has to wait. For example, if train 827 is delayed with 8 minutes, there is no extra travelling time for the passengers that are transferring to train 3627 because the transfer can still be maintained. So, only when the delay of the arriving train is bigger than the scheduled time minus the minimal time needed for the

transfer there is a decision needed. The results of whether maintaining the transfers of Table 4 or not are shown in Figure 8.

	Results of CDM						Results of UDM				
Delay in minutes:	Transfer 264	Transfer 1070	Transfer 1120	Transfer 1290	Transfer 1401		Transfer 264	Transfer 1070	Transfer 1120	Transfer 1290	Transfer 1401
1											
2											
3											
4											
5											
6											
7											
8											
9					1						
10					1						
11					1						
12		1			1			1			1
13		1			1			1			1
14		1			1			1			1
15		1	1		1		1	1	1		1
16		1	1		1		1	1	1		1
17		1	1		1		1	1	1		1
18		1	1		1		1	1	1		1
19	1	1	1		1		1	1	1		1
20	1	1	1	1	1		1	1	1		1

Figure 8, Operational results CDM & UDM.

In this figure there stands a 1 if a transfer is no longer maintained. We can see now that transfer 1290 is always maintained when we solve the UDM problem. When we solve the CDM problem the transfer will not be maintained if there is more than 20 minutes of delay. Looking at the characteristics of this transfer I see nothing special. In fact, the schedule for this transfer is very tight and not that much passengers are needing this transfer. If there is more than 2 minutes of delay, the next train will have to wait. It is not yet clear for me why this transfer is so important for the model. Also when I look at the other transfers it look likes there is no direct relation between the variables from Table 4 and the fact if a transfer will be maintained. Perhaps this relation will be evident in section 5.3. Also, there seems to be no remarkable difference between the results of the UDM problem and the results of the CDM problem. Because it looks likes that there is no difference between UDM and CDM I will continue with solving only the CDM problem for simulated delay.

5.3 Results CDM with simulated delay

In the previous section I let AIMMS solve the problems for known delay where there is only one event delayed. Now, I simulate the delay by given 1000 random events a delay between 1 and 15 minutes. I let AIMMS repeat this 100 times, and after every iteration I saved the following variables: the delay and the decision variables of the transfers from Table 4, the objective value, the solving time of the problems and the gap between the lower bound and the optimal solution. The amount of delay and the transfer decision variable are shown in a scatter plot in Figure 9.

The delay is stated on the y-axis and the binary decision variable is stated on the x-axis. For each delay and for each transfer I took the averages of the decision variables. For example, if there are 4 observations from the same transfer and with the same amount of delay, but only in 3 cases the transfer is not maintained, the value in the scatter plot is ¾.

	Average of transfer maintaining				
Delay in minutes	264	1070	1120	1290	1401
<1	0	0	0	0	0
1-2	0	0	0	0	0
2-3	0	0	0		0
3-4	0	0	0		0
4-5	0	0	0	0	0
5-6	0	0		0	0
6-7	0	0	0		0
7-8		0			0
8-9		0	0	0	0
9-10	0	0		0	0,375
10-11		0	0	0	0,6
11-12	0	0,333333	0		0,615385
12-13	0	0,75			0,666667
13-14		0,3	0	0	0,5
14-15	1	1	1		1
15-16				0	

Figure 9, Results CMD with simulated delay.

We can see now that transfer 1290 is always maintained, which is the same result as in section 5.2. And again, the transfers 1070, 1120 and 1401 are dropped as first. But it is still not clear why these transfers are more often not maintained than the two other transfers. And it becomes even less clear because a transfer is sometimes maintained, and sometimes not maintained, while the arriving train has got the same delay. To investigate this further I take a look at the objective value, the solving time of the problems and the gap between the lower bound and the optimal solution. The results of these variables are shown in Figure 10. The gap between the lower bound and the optimal solution founded is always around zero because every problem is solved within the 15 minutes.

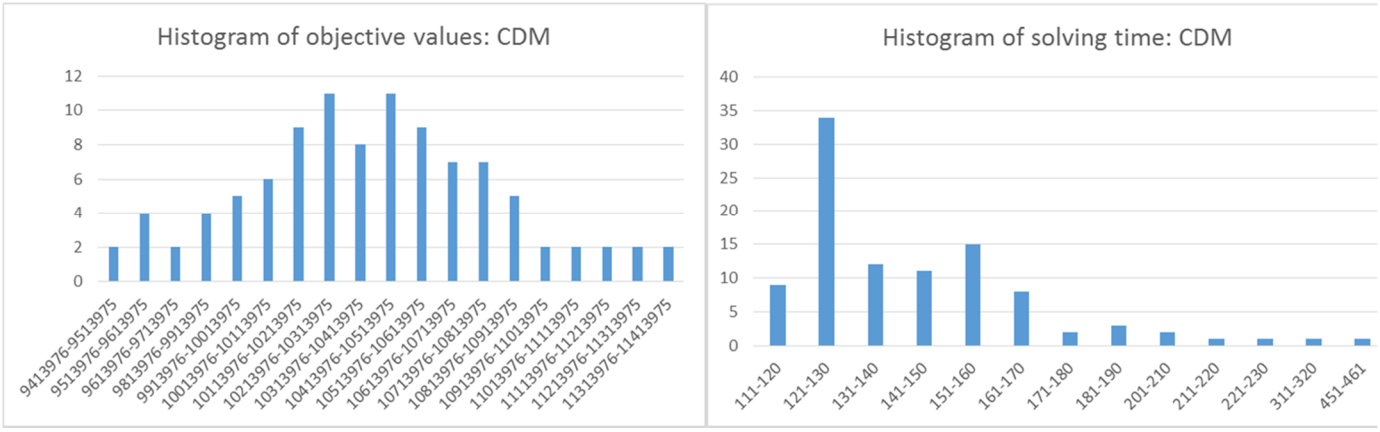

Figure 10, Histogram of the objective values and of the solving times.

The histogram of the objective values looks like a normal distribution. I expected this because of the randomly chosen events with uniformly chosen delay. When you repeat such a distribution often enough it converges to the normal distribution. But because there are differences between the objective values, this may cause some of the different decision variables. The histogram of the solving times follows a Weibull distribution with a fat tail. Again, because of this fat tail, it is possible that the solving time has got influence on the transfer decision variable. Furthermore we can see that every problem is solved within 500 seconds, and the most of them even within 180 seconds (3 minutes). Therefore, I think it is a good model to use in practice.

In the previous parts I have suggest some hypotheses about variables which could have got influence on the decision variable. I want to test these hypotheses with help of regression analyses. The dependent variable is in this case the binary transfer decision variable and the explanatory variables are the variables that we have seen before: the amount of delay, the penalty time, the number of passengers, the objective value etc. Because the dependent variable is binary we need to estimate a Logit regression. Such a regression makes use of probabilities: what is the chance that a transfer will be maintained given a set of explanatory variables? Because I think that maintaining a transfer or not depends not only on the transfer itself, but also on the upcoming events, I introduce some extra explanatory variables, which are shown in Table 5:

INDEX ACTIVITY	# PASSENGERS WAITING	TIME TIGHTNESS*
264	405	8
1071	95	3
1120	650	30
1290	83	14
1401	77	21

Table 5, Two extra characteristics of the five transfers. (*Time Tightness is in the number of six seconds)

The description of these two variables is as follows:

- Time tightness for next train activity: the minimal time needed for the next driving activity can be smaller than the scheduled time. In this case it is possible that a train leaves too late but arrives at the right time.
- Number of passengers for the next train activity: How more passengers are waiting for the transferring passengers, how more extra time if they have got delay as well.

Now that I have chosen all variables I can solve the Regression. For the Logit regression I make use of the program Eviews. The results of the regression analysis are as follows:

Dependent Variable: TRANSFER_MAINTAINED
Method: ML - Binary Logit (Quadratic hill climbing)
Date: 06/30/14 Time: 13:30
Sample: 1 500
Included observations: 500
Convergence achieved after 7 iterations
Covariance matrix computed using second derivatives

Variable	Coefficient	Std. Error	z-Statistic	Prob.
C	6.828986	4.237551	1.611541	0.1071
PENALTY_TIME	-0.053679	0.014313	-3.750331	0.0002
TIME_TIGHTNESS_TRAIN	-0.357072	0.111756	-3.195116	0.0014
delay, delay> tightness train	0.076742	0.013915	5.515123	0.0000

McFadden R-squared	0.598719	Mean dependent var	0.080000
S.D. dependent var	0.271565	S.E. of regression	0.197315
Akaike info criterion	0.239730	Sum squared resid	19.31079
Schwarz criterion	0.273447	Log likelihood	-55.93242
Hannan-Quinn criter.	0.252960	Deviance	111.8648
Restr. deviance	278.7694	Restr. log likelihood	-139.3847
LR statistic	166.9045	Avg. log likelihood	-0.111865
Prob(LR statistic)	0.000000		

Figure 11, Results of the regression analysis for the CMD problem.

Only the variables with a significant p-value are added to the regression. The McFadden R-squared shows us that these 3 explanatory variables estimate the decision variable properly. Also, the results are quite normal: when there is more delay, the chance of maintaining the transfer will decrease. And if the penalty term is high, or the schedule is not that tight, the chance of maintaining the transfer will increase. I already explained that the delay has only got influence when the delay is higher than the scheduled time minus the minimal time. Therefore, I estimate the effect of delay on the decision variable only when this equation holds. But the three relations described in this section could be expected. And a real answer on the main question is not been given. I think that, because of the deeper relationship between all events and activities, this question is not easy to answer. For example: when an event is delayed it can have influence on ten other events and activities. There is much more time needed if you would know what exactly happens if multiple events are delayed.

Chapter 6: Conclusion

After all these formulations, methods and results it's time to come with a conclusion. Let's start with the uncapacitated delay management problem. This problem and the CDM problem were introduced in Schachtebeck and Schobel (2010). We have seen that the UDM problem is quite easy to implement. With help of the arrival and departure events, the train activities and the transfer activities, I solved this problem using the optimization program AIMMS. In section 5.1 is shown that the model works correct when there is no delay and in section 5.2 I have solved the model with known delay. The solving time of this problem was really short, so it can be used in practice. For the UDM problem I have answered one operational question, which is shown if Figure 8. We can see in this figure that it is possible to determine when a transfer will no longer be maintained. But why this transfer will no longer be maintained cannot be answered by this method.

After solving the UDM problem I solved the capacitated delay management problem. For this problem I needed an extra set of activities: the headway activities. These activities modelled the network in such a way that there were no points in time with more trains on the tracks than there are tracks. Creating these activities was really hard, because it was of great importance to know the railway network. As I have shown in Figure 3, it was necessary to determine the capacity of each part of the network. And after I was finished with investigating the network I needed to create the right headway activities, such that, for example, trains were not able to overtake each other. Finally I was able to create the so called Full- and Semi Headway activities with the program MATLAB.

With help of these new sets of activities I implemented the CMD formulation in AIMMS. After implementing this model it was up to AIMMS to solve the problem. AIMMS solved the problem remarkable quickly: when using the fixed delay it took AIMMS no longer than 110 seconds. The first conclusion to this problem is that it is pretty easy to solve, and therefore also this method can be used in practice. But if we want to use this method in practice we also want operational results and not just solving times. That's why I used a method to determine when a transfer is no longer maintained. The results of this method was almost the same of that from the UDM problem: there was a relationship between the delay and whether or not a transfer was maintained, but I was not able to tell why transfer i was maintained with 13 minutes of delay and why transfer j was not maintained.

To try if it was possible to give an answer on this question I used a second operational method. The aim of this method was to let AIMMS solve the problem multiple times, always with a different delay vector. Again, 5 transfers were investigated with help of their characteristics: the extra time when missing the transfer, the number of passengers that needed the transfer, the tightness of the timetable etc. The results of this method were similar to the results of the previous method: it is possible to say when a transfer is no longer maintained, but this is different to say for each transfer. While using this method there arises a new question: when looking at the same delay, why is one and the same transfer sometimes maintained and sometimes not? It looks like that there are more variables needed to answer the question.

Therefore, I introduces two new variables which were able to model the effect of a delay better. One of them was the number of passengers that are waiting in the departing train, the other one was the tightness of the driving activity schedule of this train. With help of these new variables and the previous variables I tried to answer the question by making use of a Logit regression analysis. The results of this regression are (again) shown below:

Dependent Variable: TRANSFER_MAINTAINED
Method: ML - Binary Logit (Quadratic hill climbing)
Date: 06/30/14 Time: 13:30
Sample: 1 500
Included observations: 500
Convergence achieved after 7 iterations
Covariance matrix computed using second derivatives

Variable	Coefficient	Std. Error	z-Statistic	Prob.
C	6.828986	4.237551	1.611541	0.1071
PENALTY_TIME	-0.053679	0.014313	-3.750331	0.0002
TIME_TIGHTNESS_TRAIN	-0.357072	0.111756	-3.195116	0.0014
delay, delay> tightness train	0.076742	0.013915	5.515123	0.0000
McFadden R-squared	0.598719	Mean dependent var		0.080000

Figure 12, Results of the regression analysis.

The three variables from Figure 12 have got the most (and also significant) influence on the chance of maintaining a transfer or not. But again: it is still not clear why sometimes a transfer is maintained and sometimes not.

To conclude: both the UDM problem and CDM problem can be solved by making use of the formulations in Schachtebeck and Schobel (2010). The solving time of the UDM problem is negligible and the solving time of the CDM problem is on average no longer than 2 minutes. Both models can thus be used in practice! I tried to use some methods myself to investigate when a transfer is no longer maintained. It is possible to say when a transfer is no longer maintained, but it is not possible to say why a transfer is no longer maintained. Probably there is more research needed into all the effects of one delay on the other events and activities. Then I can really say why it is important to drop a transfer or not.

Chapter 7: Further Research

I have already mentioned it, but there are a lot of possibilities for further research. The first possibility is to better model the capacity of the network. By creating extra events for the InterCity it is possible to model the capacity, with use of full headway activities only. For example, if an InterCity has also got an arrival and a departure event at the station of Maarn, there can be a headway activity with the Sprinter in the direction of Rhenen. The train activity between this arrival and departure of this InterCity has got a minimal time of zero minutes, because the InterCity doesn't need to stop.

The second possibility to improve the model is already noticed in the results: when the network is better examined you can exactly predict what is going to happen when one train is delayed. When you know these consequences, you know how important it is to maintain a transfer or not and, this is even more important, why such a transfer is maintained or not. The regression analysis I made in section 5.3 can then be extended with extra explanatory variables, and so it can be improved. But understanding such a network is not easy and takes a lot of time, therefore I did not applied it in this research.

Chapter 8: References

Michael Schachtebeck, Anita Schöbel, (2010) To Wait or Not to Wait—And Who Goes First? Delay Management with Priority Decisions. Transportation Science 44(3):307-321.

Twan Dollevoet, (2013) Delay Management and Dispatching in Railways

Anita Schobel, (2007) Integer programming approaches for solving the delay management problems

www.ns.nl
www.treinreiziger.nl

Chapter 9: Appendix

8.1 Matlab code for creating the headway activities:

```
data = xlsread('Data.xlsx',1);

stations = [0 3 1 12 4 8 9 25 2];

lijnen{1} = [ 800 1900 3500 5200 9600]; % Eindhoven
lijnen{2} = [ 1901 3600 5201 13600]; % Tilburg
lijnen{3} = [ 800 801 3500 3501 3600 3601 4400 9601 13601 16000]; % Den
Bosch
lijnen{4} = [ 3600 3601 4400 4401]; % Oss
lijnen{5} = [ 3000 3100 3600 3601 4401 7601]; % Nijmegen
lijnen{6} = [ 3000 3001 3100 3101 3601 7500 7600]; % Arnhem
lijnen{7} = [ 3000 3001 3100 3101 7501]; % Ede/Wageningen
lijnen{8} = [ 6000 6001 16000 16001]; % Geldermalsen
lijnen{9} = [ 801 3001 3101 3501 6001 7401 16001 17401]; % Utrecht Centraal


combinaties{1} = [1 2 1 0 0; 2 1 2 0 0; 1 2 1 0 0; 0 0 0 1 1; 0 0 0 1 1];
combinaties{2} = [1 0 3 0; 0 1 0 1; 3 0 1 0; 0 1 0 1];
combinaties{3} = [1 0 1 0 0 0 0 0 0 3; 0 1 0 1 0 0 0 3 0 0; 1 0 1 0 0 0 0 0
0 3; 0 1 0 1 0 0 0 3 0 0; 0 0 0 0 1 0 1 0 0 0;
    0 0 0 0 0 1 0 0 1 0; 0 0 0 0 1 0 1 0 0 0; 0 3 0 3 0 0 0 1 0 0; 0 0 0 0
0 1 0 0 1 0; 3 0 3 0 0 0 0 0 0 1];
combinaties{4} = [1 0 1 0; 0 1 0 1; 1 0 1 0; 0 1 0 1];
combinaties{5} = [1 1 1 0 0 1; 1 1 1 0 0 1; 1 1 1 0 0 1; 0 0 0 1 1 0; 0 0 0
1 1 0; 1 1 1 0 0 1];
combinaties{6} = [1 0 1 0 0 1 0; 0 1 0 1 1 0 1; 1 0 1 0 0 1 0; 0 1 0 1 1 0
1; 0 1 0 1 1 0 1; 1 0 1 0 0 1 0; 0 1 0 1 1 0 1];
combinaties{7} = [1 0 1 0 0; 0 1 0 1 1; 1 0 1 0 0; 0 1 0 1 1; 0 1 0 1 1];
combinaties{8} = [1 0 2 0; 0 1 0 0; 2 0 1 0; 0 0 0 1];
combinaties{9} = [1 0 0 1 3 0 3 0; 0 1 1 0 0 3 0 3; 0 1 1 0 0 3 0 3; 1 0 0
1 3 0 3 0; 3 0 0 3 1 0 2 0; 0 3 3 0 0 1 0 1; 3 0 0 3 2 0 1 0; 0 3 3 0 0 1 0
1];


aankomst_vol{1} = [1 -1 1 -1 -1; -1 3 -1 -1 -1; 1 -1 1 -1 -1; -1 -1 -1 20
20; -1 -1 -1 20 20];
aankomst_vol{2} = [0 -1 -1 -1; -1 1 -1 1; -1 -1 0 -1; -1 1 -1 1];
aankomst_vol{3} = [2 -1 2 -1 -1 -1 -1 -1 -1 3; -1 0 -1 0 -1 -1 -1 3 -1 -1;
2 -1 2 -1 -1 -1 -1 -1 -1 3; -1 0 -1 0 -1 -1 -1 3 -1 -1; -1 -1 -1 -1 12 -1
12 -1 -1 -1;
    -1 -1 -1 -1 -1 3 -1 -1 3 -1; -1 -1 -1 -1 12 -1 12 -1 -1 -1; -1 3 -1 3 -
1 -1 -1 20 -1 -1; -1 -1 -1 -1 -1 3 -1 -1 3 -1; 3 -1 3 -1 -1 -1 -1 -1 -1
25];
aankomst_vol{4} = [4 -1 4 -1; -1 1 -1 1; 4 -1 4 -1; -1 1 -1 1];
aankomst_vol{5} = [8 8 8 -1 -1 8; 8 8 8 -1 -1 8; 8 8 8 -1 -1 8; -1 -1 -1 12
12 -1; -1 -1 -1 12 12 -1; 8 8 8 -1 -1 8];
aankomst_vol{6} = [9 -1 9 -1 -1 9 -1; -1 4 -1 4 4 -1 4; 9 -1 9 -1 -1 9 -1;
-1 4 -1 4 4 -1 4; -1 4 -1 4 4 -1 4; 9 -1 9 -1 -1 9 -1; -1 4 -1 4 4 -1 4];
aankomst_vol{7} = [2 -1 2 -1 -1; -1 8 -1 8 8; 2 -1 2 -1 -1; -1 8 -1 8 8; -1
8 -1 8 8];
aankomst_vol{8} = [2 -1 2 -1; -1 23 -1 -1; 2 -1 2 -1; -1 -1 -1 1];
aankomst_vol{9} = [1 -1 -1 1 3 -1 3 -1; -1 9 9 -1 -1 3 -1 3; -1 9 9 -1 -1 3
-1 3; 1 -1 -1 1 3 -1 3 -1; 3 -1 -1 3 25 -1 2 -1; -1 3 3 -1 -1 31 -1 31;
    3 -1 -1 3 2 -1 25 -1; -1 3 3 -1 -1 31 -1 31];
```

```
teller_vol = zeros(size(stations,2),1);
teller_half_normaal = zeros(size(stations,2),1);
teller_half_speciaal = zeros(size(stations,2),1);

headway_vol{1} = zeros(981,5);
headway_vol{2} = zeros(693,5);
headway_vol{3} = zeros(1968,5);
headway_vol{4} = zeros(936,5);
headway_vol{5} = zeros(2324,5);
headway_vol{6} = zeros(2809,5);
headway_vol{7} = zeros(1330,5);
headway_vol{8} = zeros(404,5);
headway_vol{9} = zeros(1256,5);

headway_half_normaal{1} = zeros(470,4);
headway_half_normaal{8} = zeros(242,4);
headway_half_normaal{9} = zeros(230,4);

headway_half_speciaal{2} = zeros(259,4);
headway_half_speciaal{3} = zeros(1001,4);
headway_half_speciaal{9} = zeros(1641,4);

for station = 1:9
    station_huidig = stations(station);
    events_huidig = data(data(:,5) == station_huidig,:);
    events_huidig = events_huidig(events_huidig(:,9) == 1,:);
    lijnen_huidig = lijnen{station};
    combinaties_huidig = combinaties{station};

    for j = 1:size(events_huidig,1)

        lijn_huidig = floor(events_huidig(j,2)/100)*100 +
mod(events_huidig(j,2),2);
        index_lijn_huidig = find(lijnen_huidig == lijn_huidig);
        type_trein_huidig = events_huidig(j,4);

        for k = j+1: size(events_huidig,1)
            lijn_vergelijk = floor(events_huidig(k,2)/100)*100 +
mod(events_huidig(k,2),2);
            index_lijn_vergelijk = find(lijnen_huidig == lijn_vergelijk);
            type_trein_vergelijk = events_huidig(k,4);

            if abs(events_huidig(j,6)-events_huidig(k,6)) <1200

                combinatie_soort =
combinaties_huidig(index_lijn_huidig,index_lijn_vergelijk);


                if combinatie_soort == 1 % Volle combinatie
                    teller_vol(station) = teller_vol(station) +1;
                    index_i = data(data(:,2) == events_huidig(j,2) &
data(:,5) == aankomst_vol{station}(index_lijn_huidig,index_lijn_vergelijk)
& data(:,9) ==0,1);
                    index_j = data(data(:,2) == events_huidig(k,2) &
data(:,5) == aankomst_vol{station}(index_lijn_huidig,index_lijn_vergelijk)
& data(:,9) ==0,1);
```

```
                    headway_vol{station}(teller_vol(station),:) =
[events_huidig(j,1) events_huidig(k,1) index_i index_j 2];

                elseif combinatie_soort == 2 % Halve combinatie, gelijke
wachttijd
                    teller_half_normaal(station) =
teller_half_normaal(station) +1;

headway_half_normaal{station}(teller_half_normaal(station),:) =
[events_huidig(j,1) events_huidig(k,1) 2 2];

                elseif combinatie_soort == 3
                    teller_half_speciaal(station) =
teller_half_speciaal(station) +1;
                    if type_trein_huidig == 1

headway_half_speciaal{station}(teller_half_speciaal(station),:) =
[events_huidig(j,1) events_huidig(k,1) 2 12];

                    elseif type_trein_huidig == 2

headway_half_speciaal{station}(teller_half_speciaal(station),:) =
[events_huidig(j,1) events_huidig(k,1) 12 2];
                    end
                end
            end
        end
    end
end
```

8.2 Matlab code for creating the random events

```
events = xlsread('Data.xlsx',1,'a2:a9561');
random_events = zeros(1000,100);

for i = 1:100

    hulp_events = events;

    for j = 1: 1000

        random_event = randi([1 size(hulp_events,1)],1);

        random_events(j,i) = hulp_events(random_event);

        hulp_events(random_event) = [];

    end
end
```