Railway Crew Rescheduling with Retiming

L.P. Veelenturf

Erasmus School of Economics
Erasmus University Rotterdam
August 2009

Master Thesis
Econometrics and Management Science
Specialisation: Operations Research and Quantitative Logistics

Supervisors:
Dr. D. Huisman
Dipl.-Wirt.-Inf. D. Potthoff

Acknowledgments

For my bachelor thesis I have made a simulation program to test railway disruption management tools of Netherlands Railways (NS). Thereafter, Leo Kroon has given me the opportunity to work further on this research as a student assistant. During this further research, I discovered that there were some possibilities to extend the current crew rescheduling tool of NS. This discovery has become the basis of my Master thesis. I would like to thank the Rotterdam School of Management and the innovation group of NSR Logistics that they have given me the opportunity to do the research. Moreover, they have offered me the opportunity to do further research in railway disruption management as a PhD-student.

I also want to thank the Erasmus School of Economics at which I have followed my Bachelor and Master courses in Econometrics and Management Science. The teachers of the Erasmus School of Economics have provide me a lot of information and research skills in the area of quantative logistics and operations research.

For the research of my Master thesis I am thankful to all employees of the innovation group of NSR Logistics which have helped me with Java, C++ and LaTeX. But, I will especially thank my supervisors Dennis Huisman and Daniel Potthoff.

After a first research in platform scheduling and a second research project for my Bachelor thesis, Dennis Huisman supervised me for the third time in a research project about railway scheduling. He (together with Leo Kroon) definitely improved my reporting skills. His critical view on a good report, enormously decreased the number of pages of this thesis.

In my Bachelor thesis I used a tool of Daniel Potthoff and in this Master thesis I was allowed to make an extension for his tool. This tool was written in C++, which I did not have used before. With a lot of patience, Daniel Potthoff guided me through his C++ code and helped me to solve some bugs in my additional code. Moreover, he checked the methods which I described in this thesis, in the smallest details.

Summary

Railway operators have to deal with several disruptions during the day. These disruptions can make the timetable, rolling stock schedule and crew schedule infeasible. In this thesis, we present a column generation approach to reschedule the crew if a disruption made the crew schedule infeasible. This crew rescheduling approach uses retiming. With retiming it is allowed to slightly adapt the timetable if that would result in a better crew schedule. We use the retiming especially to come up with crew schedules with less canceled trains.

First, we solve the crew rescheduling problem without using retiming. We do this by using an existing approach of Potthoff et al. (2008). This approach selects only a subset of the duties and tasks to solve the problem. If the crew schedule generated by this existing approach has still some uncovered tasks, we use retiming and a neighborhood selection of duties to improve the schedule.

Experiments on data instances of train drivers of Netherlands Railways (NS) have shown that our retiming approach results in better crew schedules than an approach without retiming. Especially, the approach with retiming comes up with crew schedules with less canceled trains. For the used instances of NS, the approach with retiming was able to generate a totally new crew schedule within three minutes. In the worst case it required only one minute computation time more than the approach without retiming. Both approaches (with and without retiming) have even some potential to become faster since some parts can be computed parallel on different cores or computers.

Contents

Chapter 1

Introduction

Almost every nation does have one or more railway operators which operate trains on a certain railway network. These operators want to provide a high as possible service to their customers. Commonly used service levels for railway operators are: The number of operated trains, the number of trains which arrive on time (*punctuality*) and the seat availability. If there are not enough seats on a train or if the train arrives too late or even worse if the train does not run at all, the customers are disappointed.

For the accurate running of trains, railway operators construct three schedules: a timetable, a rolling stock schedule and a crew schedule. To ensure a high service level, these schedules contain some margins to deal with irregularities. Usually, the three schedules are not made at the same time, but after each other. Many operators use a fixed order in which these schedules are made: first the timetable is constructed, thereafter the rolling stock schedule is defined, and in the last step the crew is assigned to the trains. The three schedules for a specific day are made in advance. However, during the day, these schedules must be adjusted if despite the margins, some disruptions make them infeasible. To decide which adjustments must be made in the schedules, a trade off must be made between the different service elements. Must the operator allow some additional delays to decrease the number of canceled trains, or must the operator cancel more trains to decrease the number of delays?

There are several types of disruptions which require adjustments of the original schedules. Some examples are: accidents, defect rolling stock or defect light signals which can block a certain track for some minutes, or even for hours. Imagine the situation in which in the Netherlands all tracks between Utrecht and Amsterdam are blocked. In such a situation, it is not possible to run trains on those tracks. The timetable must be adjusted, by canceling the trains which originally have to run between Utrecht and Amsterdam during the time of the blockage. After the timetable adjustments, parts of the rolling stock and crew schedule are not feasible any more. Rolling stock and crew which had to run on a train between Utrecht

and Amsterdam which is now canceled, must receive a new duty. For example a crew member which had to run on a train from Utrecht to Amsterdam, has to find another way to reach Amsterdam on time to run the next train of his duty which starts in Amsterdam. It also can happen that he cannot find another way to reach Amsterdam on time for his next trip. Then, for example, a totally new duty must be made for this crew member.

Next to blockages, running time disruptions can also require to adjust the schedules. We have a running time disruption if the actual running time of a train is larger than the planned running time of a train. A running time disruption can be caused by temporary speed limitations or red light signals for which the train has to wait. Moreover, dwell times which take longer than planned are also defined as running time disruptions. The dwell time is the time to let passengers get on and off the train. If it takes longer than planned to let people get on and off the train, it is clear that the train will depart later than planned.

If a train is delayed due to some running time disruption, the crew and rolling stock of that train arrive later than planned at the arrival station of the train. Due to the delay it could happen that the rolling stock and crew arrive too late for the next trip in their schedule. A train cannot depart from its location before all necessary rolling stock and crew have arrived at that location. In some cases this means that a train must depart at a later time than the planned departure time, because it has to wait for rolling stock and/or crew. In such cases, a railway operator has two options. The first option is to let the train depart at a later time with the originally planned rolling stock and crew. The other option is to search for other crew and/or rolling stock which can run the train on time or with less delay. In the first option only a timetable adjustment is made and in the second option next to a possible timetable adjustment, the rolling stock and crew schedule are changed. In general the second option leads to a higher punctuality.

Just as all other railway operators, Netherlands Railways (NS), which is the largest passenger railway operator in the Netherlands, wants a high as possible service level for their passengers. According to a report (Ramaekers et al. (2009)) of Statistics Netherlands (CBS), in 2006 of the 27 members of the European Union, the Netherlands had the most train kilometers per kilometer of rail track. A train kilometer is equal to a kilometer traveled by a passenger train or freight train. Moreover, after 2006 the number of trains in the Netherlands has grown rapidly by the introduction of the whole new timetable of NS in December 2006 (Kroon et al. (2009)). Due to the large and even growing amount of trains in the Netherlands, the margins in the schedules become less and less. On the tracks, trains run quite close after each other. As a consequence of this, running time disruptions of a certain train mostly also influence running times of other trains. To reach a high service level NS needs a fast approach to detect and handle disruptions in the schedules.

After disruptions are detected, it is necessary to directly reschedule the timetable, the rolling stock and the crew in such a way that the negative effects of the disruption are minimized. In this thesis we suggest a fast crew rescheduling approach, which we test on data instances of NS. This approach is focused on the rescheduling of crew and it does not take the rolling stock rescheduling into account. Therefore, we assume that during our crew rescheduling approach, the rolling stock schedule is fixed. Due to the fixed rolling stock schedule, we do not use the seat availability as a service level.

In the crew rescheduling approach, the timetable is not fixed. The approach is allowed to cancel trains and to change the departure and arrival times of some trains. Changing the departure and arrival times of trains is also called *retiming*. We take two service levels into account: the number of operated trains and the punctuality. The number of operated trains gets the highest priority in this thesis. Consequently, retiming, which decreases the punctuality, may be used to reduce the number of canceled trains.

Overall this thesis makes a start with integrated rescheduling of the crew and the timetable, which is a new research area in railway rescheduling. Until now, in most approaches the timetable and the crew are rescheduled independently. However, in the approach of this thesis, major timetable changes needed to handle blockages must still be given as an input. Summarizing, the objectives of this thesis can be stated as:

Develop a fast approach to construct a new crew schedule in which as many trains as possible can be operated. During the approach, the rolling stock schedule is fixed and it is allowed to slightly change the timetable if that reduces the number of canceled trains. Between all possible schedules with the same amount of canceled trains, choose the schedule with the fewest changes in the timetable.

In the remainder of this thesis, a problem description is given in Chapter 2 and a literature review is given in Chapter 3. Thereafter, we introduce two mathematical formulations in Chapter 4. In Chapter 5 we construct an approach to solve the problem. This approach assumes that we can cancel a train without any problem. However, canceling a train will definitely influence the rolling stock schedule. To take the problems which occur by canceling a train into account, we have to make a more advanced approach which is described in Chapter 6. This approach is a start of integrating rolling stock rescheduling into the crew rescheduling approach. We test the approach of Chapter 5 on data instances of NS. The results are presented in Chapter 7. For the approach of Chapter 6 we do not come up with results because more research is required. In Chapter 8 we draw some conclusions and give some recommendations for further research.

Chapter 2

Problem description

In this thesis we define a fast crew rescheduling approach to deal with disruptions in the railway network. In this thesis we only focus on the rescheduling of train drivers. However, the proposed solution approach can also be used for the rescheduling of conductors. The rescheduling of conductors is quite similar to the rescheduling of train drivers, but it requires some other rules.

In this chapter, we first introduce in Section 2.1 some railway terminology for a better understanding of the problem. In Section 2.2 the problem definition with the assumptions of the thesis are given. Potthoff et al. (2008) have already made a fast railway crew rescheduling approach. However, this approach does not allow retiming to create a better crew schedule. In Section 2.3 we use the tool of Potthoff et al. (2008) to show that allowing retiming will lead to better solutions. With this example we show that it is worth to investigate in an approach which allows retiming.

2.1 Railway terminology

We first introduce some railway terminology which is necessary to describe the problem in a clear way. Passenger railway operators make use of *lines*, which are determined by a start and end location and some intermediate stops. On each line some trains are operated. In the Netherlands, NS operates most passenger lines. All lines of NS are operated in both directions. An example of such a line is the 800-line between Maastricht (Mt) and Alkmaar (Amr) with 12 intermediate stops. In the rush hours the 800-line is extended to a line between Maastricht and Schagen (Sgn) with 14 intermediate stops. Every half an hour, in both directions a train runs on the 800-line.

Rolling stock and crew must be assigned to every train, before the trains can run. However, for the crew it is not necessary to run the complete train from the begin to the end location

of the line. During stops at certain stations, crew members can leave their current train to switch to another train which usually uses other rolling stock. The stations at which it is allowed for crew members to switch to other rolling stock compositions are called *relief points*. Between two relief points crew cannot switch to other rolling stock and therefore we can divide trains into *trips* between relief points. Moreover, we also have some special trips to reallocate or shunt rolling stock. There are no passengers on the trains of these special trips.

Performing a trip between two relief points is also called a *task*. Not all crew members may run every task. For example, to be allowed to drive a trip, a driver must have knowledge about the route and about the rolling stock of the trip.

In reality, it take some time to switch at a relief point from one train to another train. At a transfer from one train to another train, crew members first have to leave a certain train, then they have to walk from one platform to another platform and finally they have to enter a new train. The total duration of the transfer is called the *transfer time*. At a transfer crew does not always have to walk to another platform. For example, instead of walking to another platform the crew member could have to walk from one side to the other side of the same rolling stock composition to turn a train. In the crew schedule the minimum allowed transfer times are mostly larger than the actually necessary transfer times. Railway companies have two reasons why they do not use the actual transfer times. First of all, it is very difficult to determine exactly all transfer times between all possible platform combinations. Secondly, the transfer times can be used as a kind of buffer in the crew schedule. At NS, in the planning phase, the minimum allowed transfer time is in most cases 20 minutes.

A *duty* consists of a sequence of certain tasks which must be performed by one crew member. Duties start and end at the same crew base to ensure that a crew member which starts at a certain crew base will be back at the same crew base after his duty. The set of crew bases is a subset of the relief points because tasks start and end only at relief points. Sometimes a duty contains a so called *deadhead task*, which is used for repositioning. If a crew member has a deadhead task in his duty, he does not have to drive the train of the deadhead task, but he will be a passenger on that train, while the train is driven by another crew member.

Another possibility is that the duty contains a *repositioning task*. The repositioning task is comparable with the deadhead task, with the difference that it uses another way of transportation. The repositioning task requires the crew member to be a passenger on another mean of transport than a train performed by the operator for which he works. The crew member has to take for example a bus, taxi or a train of another railway operator.

Next to driving, deadhead and repositioning tasks, the duties can also contain certain

education, *training* and *reserve tasks*. Reserve tasks are scheduled such that some crew is available to run a task if some other crew member does not show up. The education, training and reserve tasks are taken into account in our cases, but they get the lowest priority.

2.2 Problem definition

The main objective of crew rescheduling is to find a driver for every train; all tasks must be covered by the crew. To cover all driving tasks, the railway operator has some crew members which are busy with driving trains, but it also has some crew members which are stand-by. In the crew schedule the *stand-by crew* does not have to perform any tasks. However, after a disruption they can be necessary to take over some tasks of other crew. If a task cannot be covered by any crew member, the task must be canceled. Moreover, the train which corresponds to the canceled task must be canceled.

In this thesis the most important service level is the number of operated trains. If a train is canceled, not only the passengers are disappointed but also the rolling stock schedule must be adjusted. Moreover, we assume that the passengers have less negative feelings against small delays of trains than to canceled trains. A canceled train means that the passengers have to take the next train, which is mostly at least a quarter of an hour later; the passengers have a delay of at least 15 minutes. Moreover, the first train which departs after the canceled train becomes very busy since the passengers of two trains must be transported by it. By this classification in the service levels, we allow limited retiming possibilities for trains to decrease the number of canceled tasks. Since passengers do not expect that trains depart earlier than planned, it is only allowed to change the departure times to a later point in time. As a second objective, we have to minimize the number and duration of the additional delays caused by the retiming of trains.

By changing the departure and arrival times of trains, we assume that this can be done without causing any additional delays for trains which run on the same infrastructure. In reality this is not always true. For example, due to retiming of a fast train, it could happen that the fast train departs after a slow train on the same track instead of before. In such a case the fast train has a longer running time since it cannot pass the slow train. The assumption above is reasonable if the delays are relatively small. Moreover, if the assumption is not valid, i.e. when we detect in the solution some additional delays caused by the order of the trains on the track, we could again use the rescheduling tool with the newly computed departure and arrival times of the conflicting trains as input.

Since this thesis focuses on crew rescheduling, the crew rescheduling approach only makes changes to the timetable which are necessary to construct an optimal crew schedule. We only

cancel a train, if the corresponding task cannot be covered by a crew member. Timetable changes to deal with the primary disruption, such as the cancellation of trains due to the blockage, the use of extra trains for the passengers and the selection of new departure and arrival times, are not made by the crew rescheduling approach. Such changes must be made beforehand and must be given as an input to the approach.

The approach must be able to handle all kinds of situations. However, in this thesis we mainly focus on one type of situation. This situation is a blockage of (part of) a route. In such a situation, some or all tracks of a route are blocked and some trains must be canceled. As just mentioned, the trains which must be canceled due to the blockage, must be determined beforehand as input for the approach. The rescheduling after a blockage mostly requires many adjustments in the crew schedule.

In another situation, real-time schedules could be given as input to the approach. The approach must then be able to detect and handle transfers of crew members which cause delays. A transfer of a crew member between task t_1 and t_2 causes a delay if the crew member will arrive too late to run t_2 on time. Then, it is better to assign t_2 to another crew member which can perform it on the original departure time. If that is not possible, it is useful to assign t_2 to the crew member who can start the task with as less delay as possible. Notice that the crew member who originally had to run t_2 could be that crew member with the shortest delay. To handle these transfers, in contradiction with the handling of blockages, often only a few changes in the crew schedule are needed. Therefore we focus in this thesis mainly on the blockages.

In most cases, we already have a simple feasible solution to handle a transfer which causes delays. That solution is the delayed original plan: The original plan is used but the departure and arrival times of tasks which cannot be run on time, are changed. In such a case, we only use retiming and do not change the crew schedule. Moreover, all tasks are certainly covered. Sometimes, the delayed original plan is not feasible because, due to the retiming, the duties can have conflicts with some rules of the railway operator. For example, due to the retiming of some tasks, the duty can become too long or the meal break in the duty can become too short, which makes the duty infeasible. However, if the delayed original plan is feasible, the remaining goal is to find a schedule which needs the fewest timetable changes.

During the crew rescheduling approach, for every duty which has not been finished yet, a replacement duty must be selected. The replacement duty consists of a fixed and a variable part. The fixed part consist of the tasks of the original duty which already started. These are tasks which started before the rescheduling took place. All these tasks are already finished or the crew member is at the time of rescheduling busy with one of these tasks. It may be

clear that this part of the schedule cannot be changed anymore. However, we can change the variable part which is the completion of the duty. A feasible completion of a duty is a sequence of tasks which completes the duty in such a way that the total replacement duty satisfies all rules of the railway operator. In this thesis we use the same rules as Potthoff et al. (2008) use for their crew rescheduling approach. These rules are summarized below:

- A replacement duty needs to start and end at the same crew base as the original duty.
- A replacement duty may end up to 60 minutes later than the planned end time of the original duty.
- If, in a replacement duty, two subsequent tasks must be performed on different rolling stock units, a certain minimum transfer time has to be taken into account.
- A replacement duty which is longer than 5 1/2 hours must contain a meal break of at least 30 minutes at a relief point with a canteen. Before and after the meal break it is not possible to work more than 5 1/2 hours.
- A replacement duty may only contain tasks on parts of the railway network for which the crew is licensed.
- A replacement duty may only contain tasks which use rolling stock for which the crew is licensed.

The minimum transfer time used for the crew rescheduling is less than the minimum transfer time used in the planned crew schedule, since, due to the urgency of the rescheduling, less buffers are taken into account. For crew rescheduling the minimum transfer time becomes closer to the actually necessary transfer time.

In the original planning phase, NS takes more rules into account for the crew schedule. However, because of the urgency of the rescheduling, some of these rules are relaxed. An example of such an rule is the rule that the "sweet" and "sour" tasks must be divided equally over the crew. This rule defines that the attractive tasks and the unattractive tasks must be divided more or less equally over the duties. Such a rule is not of high importance during crew rescheduling.

In Figure 2.1 an example of crew rescheduling is given. For simplicity we have used a case in which a transfer causes a delay. For an example of a blockage we refer to Potthoff et al. (2008). After the delay of the transfer is detected, crew rescheduling can be used. First of all we have a crew member which has an original duty a). At 8:26 the railway operator discovered that due to some reason the train of task t_2 will arrive with a delay of 23 minutes at Amf. With this new information the operator discovers that in duty a) the tasks t_2 and t_3 overlap

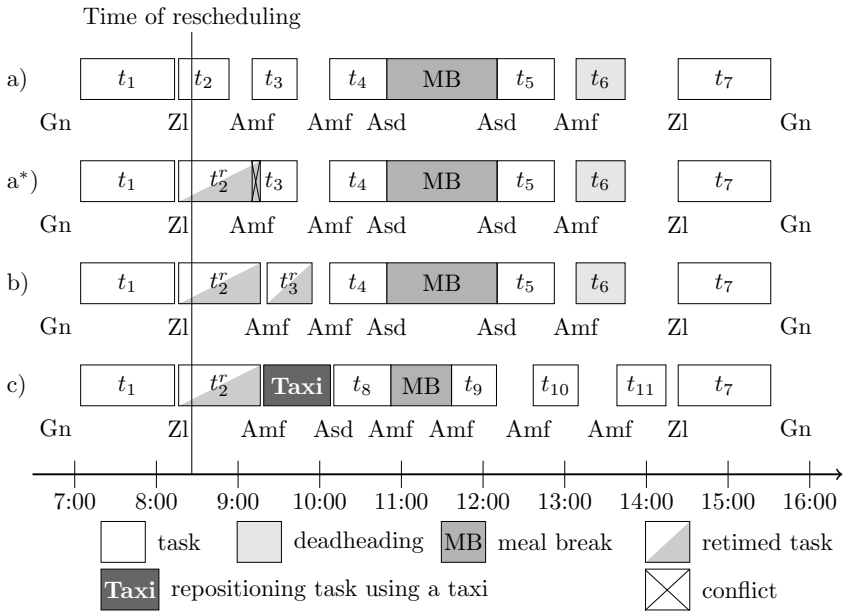

Figure 2.1: Example of crew rescheduling results

each other as shown in duty a^*). This means that the crew member would not arrive on time to run task t_3. To handle this caused delay, several options are available. As a first option we could use retiming and delay the trains of task t_3 with 11 minutes. The resulting replacement duty is given by duty b). However it may be better if another crew member takes over task t_3, to run that task on time. Then the replacement duty of a) becomes more different. An example of such a replacement duty is given by c). The crew rescheduling approach must determine which replacement duty, b) or c), must be chosen.

The approach which we develop in this thesis assumes that the computation time of the approach is negligible. Moreover, it is assumed that every crew member does know his replacement duty immediately. Both assumptions are in contradiction with reality, but it is difficult to deal with them. It is beforehand not known what the computation time of the solution approach will be and how long it will take to make all crew members aware of their new schedules. An option could be to make estimations of these durations and to take them into account. In this current research, we neglect both times. However we give a penalty on the number of changed duties. With this penalty, we try to minimize the number of phone calls which must be made.

In this thesis, the rolling stock rescheduling is not taken into account. Therefore, we have to make some assumptions about the rolling stock schedule. First of all, in case of crew rescheduling after a blockage, we assume that all old and new trips of the trains are covered with

rolling stock. This requires that changes to the original rolling stock schedule must be given as input. Moreover, we assume that the rolling stock schedule is fixed during the approach. This means that if the rolling stock schedule contains some transfers of rolling stock which cause delays, the corresponding trains must be retimed automatically. For example, if we have a rolling stock unit which is used for both task t_1 and task t_2 and the departure time of t_2 is 3 minutes before the arrival time of t_1, the train corresponding to task t_2 must be delayed with at least 3 minutes. Notice that this retiming of trains can also remove or introduce delays caused by transfers of the crew. By retiming a certain train, we assume again that it does not influence the departure and arrival times of other trains on the same track.

2.3 Example of using retiming

In this section, an example is given in which we show that allowing retiming during the crew rescheduling approach results in fewer canceled trains. With this example we show that it is worth to investigate in a crew rescheduling approach which allows retiming. The retiming options to reduce the number of canceled trains are found by examination of the data and by trial and error. In the remainder of this thesis we define an approach which automatically selects the trains which could be worth to retime.

For the example we use the crew rescheduling approach of Potthoff et al. (2008). This approach does not allow retiming. However, we can do some experiments by changing the timetable beforehand. This means that the retimings must be determined in advance. If we make different timetables, we can compare the resulting crew schedules of the approach and we can determine which timetable was best.

We use a instance of NS which is used in the article of Potthoff et al. (2008). The selected instance is *Ht B*, which consist of a two sided blockage between 15:30 and 18:30 around 's Hertogenbosch (Ht). Moreover, we use subset R_1 for the stand-by crew. With the original timetable the crew rescheduling approach of Potthoff et al. (2008) results in one uncovered task. The canceled task is the task of train 854 between 's Hertogenbosch and Utrecht (Ut). However, we have found two alternative timetables in which the crew rescheduling approach succeeds in covering all tasks.

In Figure 2.2 the original timetable of train 854 and train 871 is given with a black line. In this figure not all intermediate stops of the train are given. We only show the intermediate relief locations: 's Hertogenbosch, Utrecht, Amsterdam (Asd) and Alkmaar (Amr). Since Schagen (Sgn) does not belong to the relief locations, a crew member which arrives at Schagen with a certain rolling stock composition must depart at Schagen on the same rolling stock. However, the crew member has to walk at Schagen from one side to the other side of the

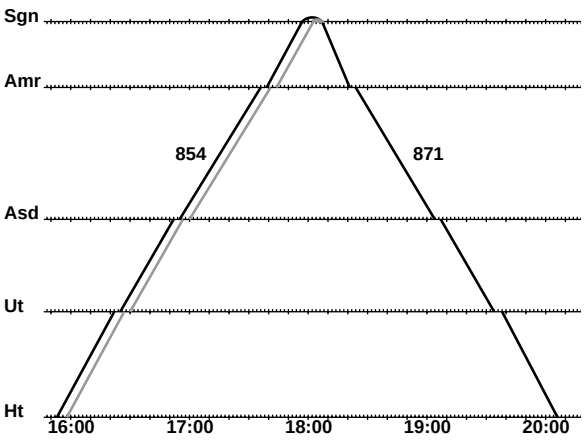

Figure 2.2: Delaying the 800-series

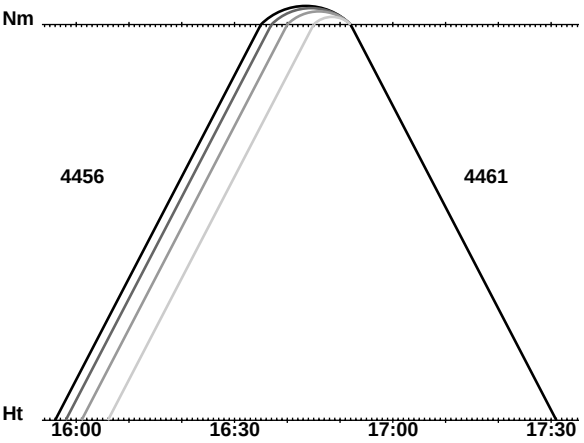

Figure 2.3: Delaying the 4400-series

rolling stock composition to turn the train. In this example, it takes 5 minutes to walk at Schagen from one side to the other side of the rolling stock composition. Crew tasks must always start and end at a relief location, therefore train 854 from Alkmaar to Schagen is combined with train 871 from Schagen to Alkmaar into one task from Alkmaar to Alkmaar.

In the first timetable which prevent the cancellation of train 854 between 's Hertogenbosch and Utrecht, we delay train 854 with 5 minutes at every location from 's Hertogenbosch to Schagen. This new timetable is shown with the gray line in Figure 2.2. In this new timetable we assume that there is no margin in the timetable or in the dwell times between 's Hertogenbosch and Schagen to absorb the delay of 5 minutes. In reality some margins are available to reduce these delays during the route of the train. It is possible to take the margins into account by delaying the later tasks less than the earlier tasks. In this case, without making use of these margins, 5 minutes have to be added to the departure and arrival times of the crew tasks *Ht-Ut*, *Ut-Asd* and *Asd-Amr*.

The task *Amr-Amr* is a special task since the rolling stock turns at Schagen at train 871. For this turning, the crew member has 10 minutes available of which only 5 minutes are really necessary. Because Schagen is no relief location, a crew member which runs from Alkmaar to Schagen must also run the train from Schagen to Alkmaar. If train 854 arrives 5 minutes late at Schagen, the crew member has still 5 minutes left to walk from one side to the other side of the rolling stock composition. These 5 minutes are enough for the crew member to run train 871 on the original planned departure time. If train 854 is delayed for more than 5 minutes, train 871 also gets a delayed departure.

Since train 871 can depart on time, only the departure time of the task *Amr-Amr* has to be delayed by 5 minutes and the arrival time remains the same. If we delay the crew tasks as described in the input of the crew rescheduling approach, all tasks are covered by the crew.

The second option to prevent the cancellation (of a part) of train 854 is to delay train 4456 with some minutes. In Figure 2.3 several options to delay train 4456 are presented in gray and the planned timetable of the trains 4456 and 4461 is given with the black lines. The shown delays are 2, 5 and 10 minutes. After trial and error, delaying train 4456 with 2 minutes is enough to ensure that train 854 between 's Hertogenbosch and Utrecht will not be canceled.

The rolling stock of train 4456 will turn in Nijmegen (Nm) on train 4461. In the original plan train 4461 departs 18 minutes after the arrival of train 4456. Since Nijmegen is a relief location, we can delay the departure and arrival of train 4456 with a maximum of 18 minutes without delaying the departure of train 4461. If the driver of train 4456 has to drive train 4461, he has to walk for 5 minutes at Nijmegen. Therefore, if train 4456 is delayed for more than 13 minutes, the driver of train 4456 cannot drive train 4461 on time. Since the current crew rescheduling tool does not use retiming, it is in case train 4456 is delayed for more than 13 minutes, not possible that one driver has to run both train 4456 and train 4461.

Chapter 3

Literature review

The airline industry is comparable with the railway industry: It has passengers which travel between two airports (stations) and which want to depart and arrive on time. The flights (trains) cannot depart without having aircraft (rolling stock) and crew. The advantage for the airline industry is that the airplanes can pass each other in the air which make them less dependent of each other than trains which cannot pass each other on the track. However, flights can still be dependent of each other since the airports have limited capacity (just as the stations) and there is a limited number of available crew and aircraft. Flights of airlines consist mostly of only one or two trips, in which a trip is the movement between two locations: airports for flights and stations for trains. In comparison, trains often consist of many trips.

In general, all trips of the same train must use the same rolling stock, to ensure that passengers can reach their destination without transfers. It is only allowed by exception to use other rolling stock for different trips of a train. In terms of crew scheduling this means that the same rolling stock must be used for all tasks of the same train. If a train is delayed during its first trip, all other trips of the same train will be delayed due to the fact that the rolling stock is delayed. Usually, the railway operator will not use other rolling stock to perform the other trips of the train on time. So, if the operator decides to retime the first trip and corresponding crew task of a train, the next trips and corresponding crew tasks of the same train must also be retimed. In the airline industry it is in case of delays more allowed to use other aircraft than planned since there are less passengers which want to perform two trips directly after each other with the same aircraft.

Approaches of the airline industry could be used, but the problem instances of the airline industry are mostly much smaller than the instances of the railway companies. For example, NS has more than 10,000 driving tasks a day which must be assigned to almost 1000 drivers. Also the time to react is for railway companies less than for airlines. NS is judged on the number of trains which have a delay of at most 3 minutes and delays for airlines are mostly

measured in quarters or hours. In this section we give some overview of some useful papers in the railway and airline industry.

Huisman (2007) constructed a column generation approach for railway crew rescheduling. This method is able to deal with planned modifications of the timetable. For example, this method can be used to generate a new crew schedule if we know in advance that some tracks are out of order due to maintenance. However, we require an approach which deals with non-planned, real-time disruptions.

As already mentioned Potthoff et al. (2008) introduce a fast railway crew rescheduling approach for generating a new crew schedule after the occurrence of a real-time disruption in the railway network. Within this approach all departure and arrival times are fixed and retiming is not allowed. Potthoff et al. (2008) use a column generation approach in which replacement duties are used as the columns. The columns are generated by representing the tasks in a directed weighted acyclic graph. The nodes represent departures and arrivals and the tasks are represented by the arcs. Moreover, there also exists an arc between the arrival node of task t_1 and the departure node of task t_2 if it is possible for a crew member to perform task t_2 directly after t_1. In the end, a path from source to sink represents a possible completion of the duty. Such a completion does not have to be feasible. The best feasible completions are generated with a resource constrained shortest path approach based on the approach of Irnich and Desaulniers (2005). The approach is tested on different data instances of NS. Because of the fact that NS has a lot of original duties (more than 900) the approach selects in an intelligent way a subset of duties. With this subset of duties and with the tasks in these duties, the crew rescheduling approach generates a (near) optimal crew schedule.

The approach of Potthoff et al. (2008) is tested on instances in which less trains than planned are running. In these instances, all trains were running on time, which is not very realistic. However, it is possible to use the approach of Potthoff et al. (2008) for instances in which the timetable uses the real-time departure and arrival times. If we use the approach, after we have detected a transfer which causes a delay, it searches for a crew member who can run the task, which will be delayed, on time. The current crew member to which this task is assigned is not allowed to run the task, while the approach does not allow transfers which cause delays. If the tool does not find any crew member to run a task on time, the task must be canceled. Such a solution is sub-optimal, because mostly in such cases it is better to use the delayed original plan.

Since the method of Potthoff et al. (2008) does not allow retiming, we can use the costs of the solution of the approach of Potthoff et al. (2008) as an upper bound for the costs of the solution created by the approach of this thesis.

Walker et al. (2005) is the pioneer in simultaneously rescheduling the timetable and crew

schedule. However, this method is only tested on small instances. If we want to reschedule the whole timetable and crew schedule of NS, the problem instance becomes too large. In the research of this thesis, we assume that we can delay some trains without creating any conflict in the timetable. The method of Walker et al. (2005) guarantees a solution without conflicts in the timetable.

Eggenberg et al. (2008) worked on the rescheduling of aircraft. Their approach is general such that it also can be used for passenger or crew rescheduling. Just as in Potthoff et al. (2008), a column generation approach is used, but now retiming is included. The columns represent recovery schedules for the aircraft. Each aircraft has to pick at most one recovery schedule and because flights cannot be run by two different aircraft, every flight may be covered by at most one recovery schedule. For the generation of the columns they use a directed graph. The arcs represent the trips, which are called legs in the airline industry and the nodes represent the departures and arrivals. For the retiming part, they use duplicates of legs, departures and arrivals. If for example a leg can be delayed for 5 minutes, an additional departure and arrival node is made in which the time is changed to 5 minutes later. Between these two new nodes, there is an arc representing the delayed leg. However, if by using duplicates of legs cycles appear in the graph, it is necessary to be aware of the fact that one flight may not be chosen twice in one recovery scheme. For this case, they use a method of Beasley and Christofides (1989).

For railway crew rescheduling we cannot use this approach since in our case multiple crew members may use the same task. Since this approach allows only one aircraft to use a certain leg, the report does not give any solution to ensure that all duties which cover a certain task use the same duplicate leg of the task. Moreover, in railway crew rescheduling, tasks are not independent of each other by assuming that the rolling stock schedule is fixed. In the example, in Figure 2.2 in Section 2.3, it is not possible that one crew has to run the delayed task between 's Hertogenbosch and Utrecht and that some other crew has to run the original non-delayed task between Utrecht and Amsterdam. All tasks of train 854 and 871 use the same rolling stock, which means that if the task between 's Hertogenbosch and Utrecht is delayed, the rolling stock will arrive delayed in Utrecht. It is in that case not possible to run the train from Utrecht to Amsterdam at the originally planned time. The proposed aircraft rescheduling approach of Eggenberg et al. (2008) does not take into account such dependencies.

Klabjan et al. (2002) developed an approach for crew scheduling with time-windows. Notice that this paper covers crew scheduling and not rescheduling. With their approach they want to partially integrate the timetabling step into the crew scheduling step. They allow the departure time to be forwarded or delayed with up to ten minutes. Here the same

problem as in Eggenberg et al. (2008) arises, a flight may only appear in one duty. This means that the paper does not give any solution to ensure that, if multiple duties cover the same flight, every duty uses the same start time for a flight.

Abdelghany et al. (2008) also constructed a method in which at most one duty of a resource unit may cover a certain flight. In their paper, they do not use duplicates of arcs for delay possibilities, but they introduce a variable for the actual departure time of a flight.

Yan and Yang (1996), Yan and Young (1996), Yan and Tu (1997) and Yan and Lin (1997) use a network flow model to solve the aircraft rescheduling problem with retiming. In their approach they use duplicates of arcs to represent delayed flights. They add a restriction to ensure that only one of the duplicated arcs is used and in Yan and Yang (1996) this restriction is relaxed in a Lagrangian way. However, like the other papers they use a set partitioning problem: only one aircraft can be assigned to every flight. In contradiction, we will use a set covering problem since multiple crew members may be assigned to one train.

Mercier and Soumis (2007) introduce an integrated aircraft routing, crew scheduling and flight retiming model. In this integrated model, for every aircraft type a subproblem is made containing all aircraft of that specific type and all flights which have to use an aircraft of that specific type. For all combinations of two subsequent flights the connection is called short if the time to rest for the crew between these two flights is too short. It is clear that for flights which are in a connection which is short, it is possibly worth to change the departure and arrival times. In this case it is also possible to depart earlier than the planned time, which is definitely not allowed in railway rescheduling. For each flight they use copies for all possible departure times, like Eggenberg et al. (2008). A leg covering constraint ensures that only one of the copies is used per leg. Since both aircraft and crew rescheduling is involved, they suggest also an approach which uses Benders decomposition (Benders (1962)). The approaches allow that multiple crew members can take the same flight, which means that the extra crew members travel as a passenger. These deadhead tasks in which crew travels as passengers are also used in the crew schedule of railway companies. Potthoff et al. (2008) also allow deadhead tasks. We do not use the approach of Mercier and Soumis (2007) while we only look at the crew rescheduling with a fixed rolling stock schedule. In addition, the reported computation times are too long for real-time railway rescheduling.

Chapter 4

Mathematical formulation

In this chapter, we define two mathematical formulations of the problem described in Chapter 2. For this mathematical formulation, we use the same notation as Potthoff et al. (2008). However, some additional notation is introduced since the models require more sets, parameters, variables and constraints. In Section 4.1, we present two general model formulations: A and B. The general models are based on the model of Potthoff et al. (2008), but they also use copies of tasks to represent the retiming possibilities, like Eggenberg et al. (2008) and Mercier and Soumis (2007) propose. Such a model with copies of tasks, limits the retiming possibilities since the departure time cannot be chosen continuously and the retiming possibilities of a task must be determined beforehand. Model A requires some additional variables in comparison with the model of Potthoff et al. (2008). Model B is formulated in such a way that additional variables are not needed. In Section 4.2, some options are introduced to change the models to some preferences. After we have made a size comparison of the models in Section 4.3, we select one of the models to use for our solution approach in Section 4.4.

4.1 General model formulations

In this section we introduce the general model formulations. Therefore, we first introduce some sets, parameters and decision variables. Thereafter, in Sections 4.1.1 and 4.1.2 the two general models are described.

- S: Set of stations (in our case limited to relief points).
- $D \subseteq S$: Set of crew bases.
- N: Set of tasks which have not started at the time of rescheduling. For every $i \in N$ we have:

  - ds_i: Departure station.
  - dt_i: Planned departure time.
  - as_i: Arrival station.
  - at_i: Planned arrival time.
  - R_i: Rolling stock composition.

- E: Set of all copies of the tasks. For every copy $e \in E$ we have:
  - dtd_e: Departure time delay.
  - atd_e: Arrival time delay.
  - i_e: Task i which the copy represents.

- $E_i \subseteq E$: Set of all copies of task i. This set contains at least the copy of task i in which the originally planned departure and arrival times are used, furthermore it could contain copies of task i with changed departure and arrival times.

- $N_e \subseteq N, |N_e| = 1$: Task i_e which copy e represents.

- $E_e^{link} \subseteq E, |E_e^{link}| \leq 1$: Set which contains the copy which is by rolling stock dependent of copy e. If the rolling stock composition has to perform task t_1 and t_2 directly after each other, E_e^{link} contains the earliest copy of task t_2 which may be used after we have used copy e for task t_1. In general railway terms, E_e^{link} is the copy on which the rolling stock composition of copy e is turned. After we have made the sets according to this rule, we have to clean up some sets because a certain copy d may be selected in only one set E_e^{link}. Therefore, if copy d is selected in multiple sets E_e^{link}, we first determine D_{link} which is the set of all copies e for which $d \in E_e^{link}$. Thereafter, we only keep d in E_e^{link} of the latest copy e in D_{link}. Notice that all copies in D_{link} are copies of the same task.

- $E_e^{bef} \subseteq E$: Set of all copies which represent the same task as copy e and which have a smaller or equal departure time than copy e. So, copy e is included in this subset.

- $\Delta = \Delta_A \cup \Delta_R$: Set of unfinished original duties, where Δ_A are active and Δ_R are stand-by duties. Moreover, for every $\delta \in \Delta$ we have:
  - cs_δ: The station where the original duty is at the time of rescheduling or the arrival station of the task which is performed by the driver at the time of rescheduling.
  - b_δ: The crew base where the original duty starts and ends.

- K^δ: Set of all feasible completions for original duty $\delta \in \Delta$. For every feasible completion $k \in K^\delta$ we have:
  - c_k^δ: Cost of feasible completion k for original duty δ. The cost of a feasible completion is zero if the duty is not modified. Otherwise, the cost is the sum of costs like cost for changing a duty, cost for new repositioning tasks and cost for overtime.
  - a_{ik}^δ: Binary parameter which is 1 if task i is covered by feasible completion k otherwise a_{ik}^δ is 0. A task can only be covered by a feasible completion k if the driver of duty δ is allowed to drive the task.
  - b_{ek}^δ: Binary parameter indicating if copy e is used by feasible completion k (1) or not (0). For the parameter b_{ek}^δ it is not important if the driver is allowed to drive the copy. This means that $b_{ek}^\delta = 1$ could indicate that the driver is a passenger on that copy.
- x_k^δ: In both model formulations, binary variables x_k^δ indicate if duty δ uses feasible completion k (1) or not (0).
- f_i: Cost for canceling task i.
- y_i: The binary variables y_i indicate if task i is canceled (1) or not (0).
- g_e: Costs for using copy e. Notice that for copies in which the times are not shifted these costs are 0.
- v_e: Model formulation A requires additional binary variables v_e indicating if copy e is used (1) or not (0)

4.1.1 Model A

In this section, the first general model is presented. Note that model A uses the binary variables v_e. Therefore, we can take the costs for using a certain copy into account in the objective function. In this model, constraints (4.2), (4.3), (4.7) and (4.9) are the same as in the model of Potthoff et al. (2008). Constraints (4.2) determine that every task is either covered by some feasible completions or canceled. Constrains (4.3) ensure that every original duty uses exactly one feasible completion. Constraints (4.7) and (4.9) are straightforward: They ensure that the variables are equal to either zero or one. Constraints (4.4) set a binary variable v_e equal to one if copy e is used in any feasible completion which is selected by an original duty. Only one copy per task may be used, which is modeled in constraints (4.5). Moreover, these constraints guarantee that no driver can be a passenger on a train which is

$$\min \sum_{\delta \in \Delta} \sum_{k \in K^\delta} c_k^\delta x_k^\delta + \sum_{i \in N} f_i y_i + \sum_{e \in E} g_e v_e \tag{4.1}$$

$$\text{s.t.} \quad \sum_{\delta \in \Delta} \sum_{k \in K^\delta} a_{ik}^\delta x_k^\delta + y_i \geq 1 \quad \forall i \in N \tag{4.2}$$

$$\sum_{k \in K^\delta} x_k^\delta = 1 \quad \forall \delta \in \Delta \tag{4.3}$$

$$v_e - \sum_{k \in K^\delta} b_{ek}^\delta x_k^\delta \geq 0 \quad \forall \delta \in \Delta, \forall e \in E \tag{4.4}$$

$$\sum_{e \in E_i} v_e + y_i = 1 \quad \forall i \in N \tag{4.5}$$

$$y_i + \sum_{f \in E_e^{bef}} v_f - v_d \geq 0 \quad \forall i \in N, \forall e \in E_i, \forall d \in E_e^{link} \tag{4.6}$$

$$x_k^\delta \in \{0,1\} \quad \forall \delta \in \Delta, \forall k \in K^\delta \tag{4.7}$$

$$v_e \in \{0,1\} \quad \forall e \in E \tag{4.8}$$

$$y_i \in \{0,1\} \quad \forall i \in N \tag{4.9}$$

canceled. This is an important difference with the model of Potthoff et al. (2008) in which it is allowed for crew members to use a deadhead task of a trip which has no driver. Using constraints (4.4) and (4.5) without constraints (4.2) is not enough because then, if no copy of a certain task i is used in the feasible completions, still one binary variable v_e of the set $\{v_e | e \in E_i\}$ can be chosen equal to one. Because the cost of retiming is less than the cost of canceling, without using constraints (4.2), the model sets, for every task i, y_i equal to zero and one v_e of the set $\{v_e | e \in E_i\}$ is set equal to one. Nevertheless, it is not guaranteed that this copy e is used in any feasible completion. Constraints (4.6) ensure that a delayed train could be delayed more, but never be delayed less. These constraints take the route of the rolling stock into account: If a train is delayed, the rolling stock is also delayed, which means that it is not possible to depart at the original time at the next station. At the original time, the rolling stock which must run the train, has not arrived yet.

In Figure 4.1 an example of this problem is given. We have a set of tasks $i_1, ..., i_4$ of train 871, which all use the same rolling stock. Three layers of copies are available. A layer is a sequence of copies which are dependent of each other due to time and rolling stock constraints. For every copy e in a layer, the next copy of the layer is the copy which is in E_e^{link}. Every copy can only be part of one layer. If some copy of a layer is chosen to represent a task, we have to select for further tasks of the rolling stock a copy out of the same layer or out of a layer with more delays. All copies within the same layer must use the same rolling stock and may not

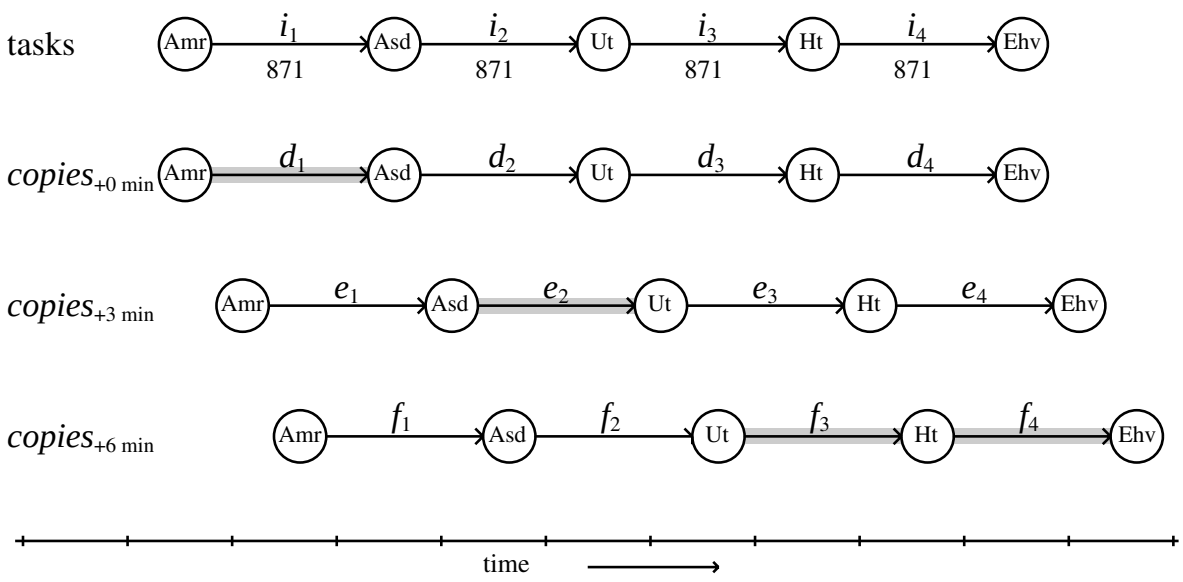

Figure 4.1: Using copies of the same layer

have an overlap in time. Overlap in time is not allowed, because otherwise the fixed rolling stock schedule causes additional delays, which could have been known beforehand. Mostly, layers consist of copies which have the same amount of delays.

In the figure we have three layers in which for j=1,2,3 and $a \in \{d, e, f\}$, $E_{a_j}^{link} = \{a_{j+1}\}$. The first layer, consisting of copies $d_1, ..., d_4$, represents copies of the tasks without delays. The second layer of copies $e_1, ..., e_4$ and third layer of copies $f_1, ..., f_4$ represent copies of the tasks with respectively 3 and 6 minutes delay. After copy d_1 is chosen to represent task i_1 it is clear that for representing task i_2 one of the copies d_2, e_2 and f_2 may be used. If a rolling stock duty contains two subsequent tasks t_1 and t_2, it is by constraints (4.6) allowed to use a copy e to represent task t_2 if for task t_1 a copy d is used which is of the same layer as e or of a layer with less delay than the layer of e. Moreover, every copy of t_2 may be used to represent the task if t_1 is canceled. Notice that in such a case a new rolling stock schedule must be made.

The copies d_2, e_2 and f_2 may be used to represent task i_2 if respectively $v_{d_1} + y_{i_1} = 1$, $v_{d_1} + v_{e_1} + y_{i_1} = 1$ and $v_{d_1} + v_{e_1} + v_{f_1} + y_{i_1} = 1$. Since in the current situation $v_{d_1} = 1$, all these conditions are met. Therefore, we can set one of the variables v_{d_2}, v_{e_2} or v_{f_2} equal to 1. If we select copy e_2 to represent task i_2, d_3 cannot be used to represent task i_3 because then, the arrival time of task i_2 at Utrecht (Ut) will be after the departure time of task i_3 at Utrecht. This is also modeled with the constraints (4.6) because they do not allow that v_{d_3} will be set equal to 1 since $v_{d_2} = 0$. However, e_3 and f_3 can still be chosen to represent task

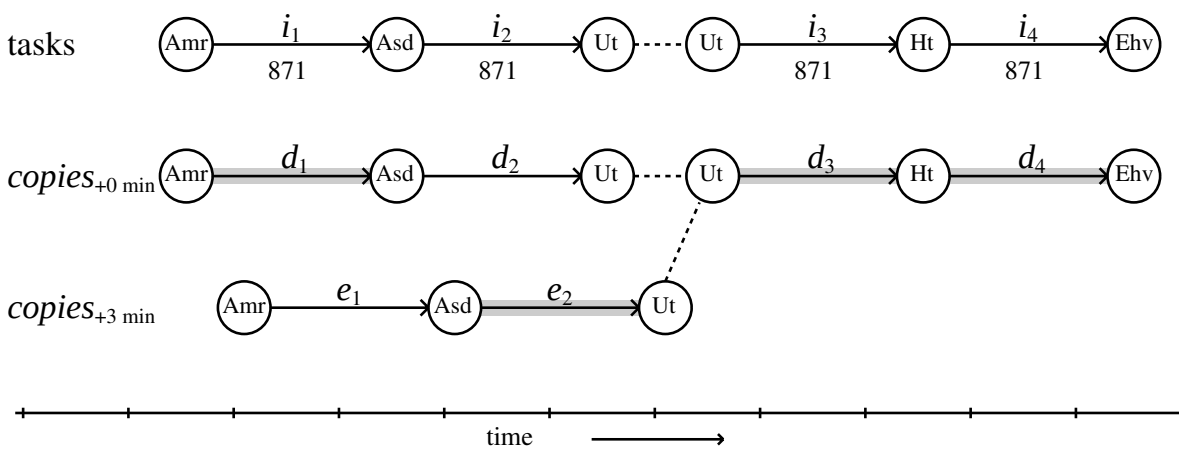

Figure 4.2: Margins in the timetable

i_3 while respectively $v_{d_2} + v_{e_2} + y_{i_2} = 1$ and $v_{d_2} + v_{e_2} + v_{f_2} + y_{i_2} = 1$. After copy f_3 is selected to represent task i_3, copy f_4 is the only possible copy which task i_4 may use.

In reality, delays of a train can also become less, if the train for example has some margins in the timetable or dwell times. In such cases, we can determine with the set E_e^{link} which copy is the first possible copy at which the train could run further. If we have some margin after a current retimed copy e, it is possible that the first allowed copy after copy e is a copy with the original planned departure and arrival time. In the end the delay is removed due to some margin in the timetable or dwell time. In Figure 4.2 an example is given in which copies use the margins in the timetable.

In this example, which is based on the example of Figure 4.1, we use the fact that $E_{e_2}^{link} = \{d_3\}$. Moreover, $E_{d_2}^{link} = \emptyset$ since a copy could only appear in one layer. In this case, the layers are $\{d_1, d_2\}$ and $\{e_1, e_2, d_3, d_4\}$. If the train is delayed for three minutes, the train could use the margin in the dwell time at Utrecht. If the planned dwell time is 5 minutes where only 2 minutes are necessary, a train which arrives 3 minutes late, can still depart on time. This means that, if copy e_2 is selected to represent task i_2, the train will be on time available to depart from Utrecht. In the model in which d_3 is a member of the set $E_{e_2}^{link}$ it is by constraints (4.6) allowed to use d_3 to represent task i_3.

4.1.2 Model B

It is also possible to formulate a model in which we do not have to use the variables v_e. In this model, the costs of using a delayed copy are measured in the parameters c_k^δ. A disadvantage of taking retiming cost into account in c_k^δ is that, if two crew members use a certain copy e, the cost for using this copy is counted twice. In this model, constraints (4.4) and (4.5)

$$\min \sum_{\delta\in\Delta}\sum_{k\in K^\delta} c_k^\delta x_k^\delta + \sum_{i\in N} f_i y_i \tag{4.10}$$

$$\text{s.t.} \quad \sum_{\delta\in\Delta}\sum_{k\in K^\delta} a_{ik}^\delta x_k^\delta + y_i \geq 1 \quad \forall i \in N \tag{4.11}$$

$$\sum_{k\in K^\delta} x_k^\delta = 1 \quad \forall \delta \in \Delta \tag{4.12}$$

$$\sum_{k\in K^\delta} b_{ek}^\delta x_k^\delta + \sum_{k\in K^\zeta} b_{dk}^\zeta x_k^\zeta + y_i \leq 1 \quad \forall i \in N, \forall e \in E_i, \forall d \in E_i \setminus \{e\}, \tag{4.13}$$
$$\forall \delta \in \Delta, \forall \zeta \in \Delta$$

$$y_i + \sum_{\zeta\in\Delta}\sum_{k\in K^\zeta}\sum_{f\in E_e^{bef}} b_{fk}^\zeta x_k^\zeta - \sum_{k\in K^\delta} b_{dk}^\delta x_k^\delta \geq 0 \quad \forall i \in N, \forall e \in E_i, \forall d \in E_e^{link}, \forall \delta \in \Delta \tag{4.14}$$

$$x_k^\delta \in \{0,1\} \quad \forall \delta \in \Delta, \forall k \in K^\delta \tag{4.15}$$

$$y_i \in \{0,1\} \quad \forall i \in N \tag{4.16}$$

of model A are replaced by constraints (4.13). These constrains ensure that only one copy of a task is used and they ensure that drivers do not run as a passenger on a train which is canceled. Moreover, Constraints (4.6) of model A are replaced by constraints (4.14), which take the rolling stock route into account.

4.2 Modification options for the models

In this section we present some possible modifications of the models of Section 4.1. First we shall discuss the modification possibilities of model A in Section 4.2.1 and thereafter we present the possible modifications of model B in section 4.2.2.

4.2.1 Modifications for model A

For model formulation A, there are some possibilities to rewrite some constraints and thereby to reduce the number of constriants. First of all, the number of constraints (4.4) can be reduced if we change them into constraints (4.17), in which M is the maximum number of duties which may cover the same task.

$$Mv_e - \sum_{\delta\in\Delta}\sum_{k\in K^\delta} b_{ek}^\delta x_k^\delta \geq 0 \quad \forall e \in E \tag{4.17}$$

Since, by allowing deadhead tasks, more than one duty can cover a certain copy, v_e must be multiplied with a certain big M in Constraints (4.17). Using a big M results in general in weaker LP-relaxations.

Without loss of generality, constraints (4.6) can also be written as constraints (4.18) since every set E_e^{link} contains at most one copy. By using constraints (4.18) instead of (4.6), we have more constraints if some sets E_e^{link} are empty. However, if E_e^{link} is empty, the constraints are always satisfied.

$$y_i + \sum_{f \in E_e^{bef}} v_f - \sum_{d \in E_e^{link}} v_d \quad \geq \quad 0 \quad \forall i \in N, \forall e \in E_i \tag{4.18}$$

The constraints (4.6) and (4.18) are only required for tasks which have multiple copies. This means that the number of restrictions can be reduced by using a new subset of tasks N^c which indicates which tasks use multiple copies.

- $N^c \subseteq N$: set of all tasks i which have more than one copy. $\{i \in N || E_i| \geq 2\}$

In constraints (4.6) and (4.18), $\forall i \in N$ can be replaced by $\forall i \in N^c$.

4.2.2 Modifications for model B

Some of the ideas of the modifications of model formulation A can also be used for model formulation B. For example, since $|E_e^{link}| \leq 1$ and since every copy is a copy of exactly one task, Constraints (4.14) can be replaced by Constraints (4.19).

$$y_{i_e} + \sum_{\zeta \in \Delta} \sum_{k \in K^\zeta} \sum_{f \in E_e^{bef}} b_{fk}^\zeta x_k^\zeta - \sum_{k \in K^\delta} \sum_{d \in E_e^{link}} b_{dk}^\delta x_k^\delta \quad \geq \quad 0 \quad \forall e \in E, \forall \delta \in \Delta \tag{4.19}$$

To reduce the total number of constraints, we can replace Constraints (4.13) and Constraints (4.14) by respectively Constraints (4.20) and Constraints (4.21). In these new constraints we use a big M which is the maximum number of drivers which could be assigned to the same task.

$$M \sum_{k \in K^\delta} b_{ek}^\delta x_k^\delta + \sum_{\zeta \in \Delta} \sum_{k \in K^\delta} \sum_{d \in E_i \setminus \{e\}} b_{dk}^\zeta x_k^\zeta + My \leq M \quad \forall i \in N, \forall e \in E_i, \delta \in \Delta \tag{4.20}$$

$$M(y_{i_e} + \sum_{\zeta \in \Delta} \sum_{k \in K^\zeta} \sum_{f \in E_e^{bef}} b_{fk}^\zeta x_k^\zeta) - \sum_{\delta \in \Delta} \sum_{k \in K^\delta} \sum_{d \in E_e^{link}} b_{dk}^\delta x_k^\delta \geq 0 \quad \forall e \in E \tag{4.21}$$

For model formulation B the same holds as for model formulation A, that Constraints (4.14), (4.19) and (4.21) are only required for tasks which have retiming possibilities. In these constraints we can replace $\forall i \in N$ by $\forall i \in N^c$ and $\forall e \in E$ by $\forall i \in N^c, \forall e \in E_i$.

4.3 Size comparison with the model of Potthoff et al. (2008)

In this section we compare the size of both models with the size of the model of Potthoff et al. (2008). These size comparisons give some insight into the complexity of the models.

4.3.1 Size comparison for model A

If we do not allow retiming, we want to solve the same problem as Potthoff et al. (2008). However, our model is larger since we have an additional restriction. This restriction is that we do not allow that a crew member uses a deadhead task of a trip that has no driver.

Without retiming, $|E|$ is equal to $|N|$ and the total number of constraints in model A has in comparison with the model of Potthoff et al. (2008) increased with $|N||\Delta|$ (4.4) $+|N|$ (4.5) + (at most) $|N|$ (4.6) $+|N|$ (4.8). Since tasks have only one copy, we can neglect Constraints (4.6). By removing these constraints, model A has in total at most $(|\Delta| + 2)|N|$ additional constraints. We cannot neglect constraints (4.4) and (4.5) since these constraints together do not allow that someone is a passenger on a trip which is not covered by a driver.

As already mentioned in the possible modifications, the number of constraints can be reduced by using constraints (4.17) instead of (4.4). This requires to use a big M in the formulation, but reduces the total number of additional constraints to at most $3|N|$.

If we use retiming, for every task which get multiple copies we first have to add constraint (4.6) for the copy without delay. Moreover per retimed copy of a task, the number of additional constraints is $|\Delta|$ (4.4) + (at most) 1 (4.6) +1 (4.8). In total this means that the first retimed copy of a task requires at most $|\Delta|+3$ additional constraints in the model and every additional retimed copy of a task requires at most $|\Delta| + 2$ extra constraints. By using constraints (4.17) instead of (4.4), the number of additional constraints per extra copy is at most 3. But again this requires to use a big M in the formulation.

Next to the additional constraints this model does also have additional binary variables. For every copy, retimed or not, a variable v_e is needed. The number of binary variables increases from $|N|$ in the model of Potthoff et al. (2008) to $|N|+|E|$ in our model formulation.

4.3.2 Size comparison for model B

Model B is made in such a way, that it does not require additional variables in comparison with the model of Potthoff et al. (2008). However by leaving the number of variables the same, the number of constraints has grown enormously.

In case retiming is not allowed, the model has already $|N|(|\Delta|)^2$ additional constraints of type (4.13) and $|N||\Delta|$ additional constraints of type (4.14). Just as for model formulation A, constraints (4.14) are not necessary if retiming is not allowed. Per additional copy, at least $(|\Delta|)^2$ additional constraints of type (4.13) are necessary and at most $|\Delta|$ constraints (4.14). In total this model does need much more constraints than model formulation A or the model of Potthoff et al. (2008).

4.4 Model selection

We have discussed that model formulation A has more variables than model formulation B, but that model formulation B has a lot more constraints than model formulation A. We have to make a choice between a model with more variables or a model with more constraints. The number of additional constraints of model B is much larger than the number of additional variables of model A. Moreover without using additional variables, it is not possible in model B to give an explicit cost for using a certain copy. If a certain copy is covered by two crew members the cost for the use of this copy is counted twice.

Without using retimed copies, the models have the same LP-relaxation. We give a short explanation why these LP-relaxations are the same. Without retiming, every task has only one copy. This copy can be used without any cost: g_e =0. For both models the objective function then becomes the same. Moreover, both models use the same constraints as Potthoff et al. (2008). This means that we only have to compare Constraints (4.4), (4.5) and (4.6) of model A with Constraints (4.13) and (4.14) of model B.

First of all we show that Constraints (4.4) and (4.5) of model A together are the same as Constraints (4.13) of model B. We can rewrite Constraints (4.4) and (4.5) as Constraints (4.22) which becomes equal to Constraints (4.23) if we subtract $v_e + y_i$ from both sides. These Constraints (4.23) are the same as Constraints (4.13) of model B.

$$v_e - \sum_{k \in K^\delta} b^\delta_{ek} x^\delta_k \geq v_e + y_i - 1 \quad \forall i \in N, e \in E_i, \forall \delta \in \Delta \tag{4.22}$$

$$-y_i - \sum_{k \in K^\delta} b^\delta_{ek} x^\delta_k \geq -1 \quad \forall i \in N, e \in E_i, \forall \delta \in \Delta \tag{4.23}$$

Secondly, we can prove that Constraints (4.6) of model A and Constraints (4.14) of model B do not restrict any of the variables. In Constraints (4.6) we have that $y_i + v_e - v_d \geq 0$ for all $i \in N$ with $e \in E_i$ and $d \in E_i^{link}$. However, since $y_i + v_e$ is equal to 1 by Constraints (4.5) and since v_d is binary $y_i + v_e - v_d$ can never become less than 0. So, these Constraints (4.6) have no additional value to the model.

Since every task i has one copy e, $b^\delta_{ek} \geq a^\delta_{ik}$ for all $\delta \in \Delta$ and for all $k \in K^\delta$. This also means that $\sum_{\delta \in \Delta} \sum_{k \in K^\delta} b^\delta_{ek} x^\delta_k \geq \sum_{\delta \in \Delta} \sum_{k \in K^\delta} a^\delta_{ik} x^\delta_k$ and so $\sum_{\delta \in \Delta} \sum_{k \in K^\delta} b^\delta_{ek} x^\delta_k + y_i \geq \sum_{\delta \in \Delta} \sum_{k \in K^\delta} a^\delta_{ik} x^\delta_k + y_i \geq 1$. With this knowledge, we can rewrite Constraints (4.14) of model B as Constraints (4.24) and since $\sum_{k \in K^\delta} b^\delta_{ek} x^\delta_k$ is always lower or equal to 1 by Constraints (4.13), Constraints (4.14) are always met and do not have any additional value.

$$1 - \sum_{k \in K^\delta} b^\delta_{ek} x^\delta_k \geq 0 \quad \forall i \in N, e \in E_i, \forall \delta \in \Delta \tag{4.24}$$

Chapter 5

Solution approach

In this chapter, we construct an approach to find a (near) optimal solution for the problem described in Chapter 2. In Section 5.1 the general solution approach is given. This general approach, which is based on Potthoff et al. (2008), can be used for both mathematical formulations of Chapter 4. In the remainder of the chapter, the steps of the approach are explained in more detail for model formulation A.

The problem becomes very large if we take all duties and tasks into account. Therefore, we define small core problems in which subsets of duties and tasks are used. These subsets must be chosen in an intelligent way since both the computation time and the quality of the solution are dependent of them. In Section 5.2 is discussed how we select a core problem. After a core problem is determined, the core problem must be solved which is explained in Section 5.3.

5.1 General solution approach

NS has about 1000 driver duties a day, to which more than 10,000 driving tasks are assigned. If we take all these duties and especially all possible feasible completions of these duties into account, the problem gets very large. For example, a duty which has not started yet could have many millions possible feasible completions.

To make the problem solvable, we use just as Potthoff et al. (2008) column generation to generate the feasible completions, which means that we do not have to determine all possible feasible completions beforehand. Moreover, by using only a subset of tasks, the number of possible feasible completions is reduced substantially. During the approach, promising feasible completions are added to the model. Such a column generation approach could reduce the computation time enormously. However, the problem is still too large if all duties are taken into account.

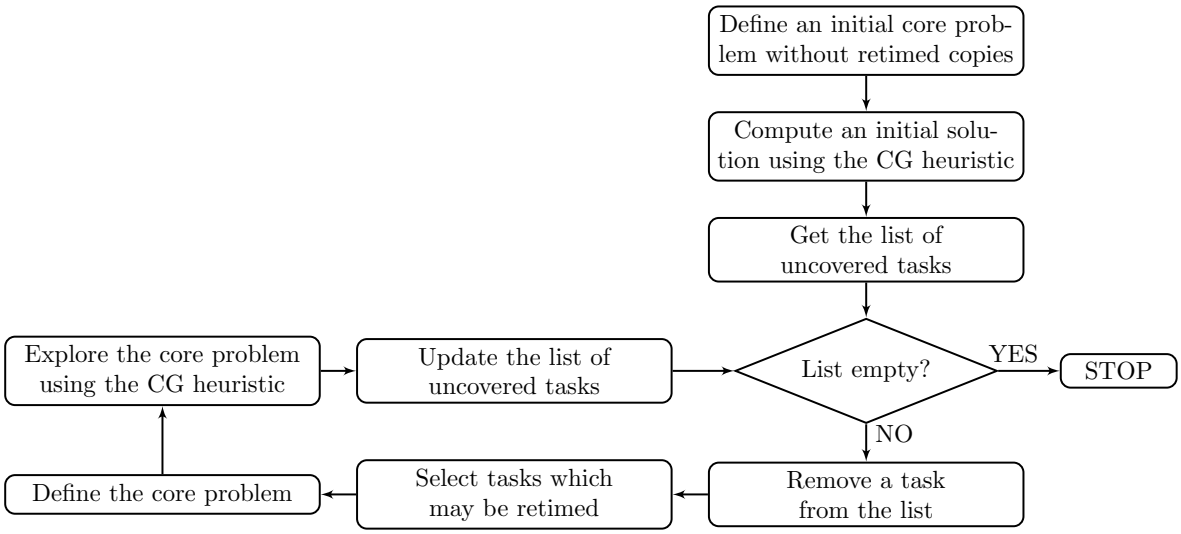

Figure 5.1: Overview of the algorithm

Especially, since we want to use the approach for real-time cases, the computation time must be very short. This enforces us to use efficiently chosen subsets of duties and tasks. We want to use the smallest subset which is still large enough to handle the disruption against reasonable costs. Since we use subsets, it is not certain that we find the optimal solution. However, for real-time cases the aim is to find a reasonable solution within a few minutes instead of getting the optimal solution to handle the disruption.

Sometimes, the dispatchers could be dissatisfied with the solution of the core problem, for example if some tasks are still uncovered. According to Potthoff et al. (2008), it is in such a case not recommended to resolve the problem with a larger subset of duties, as Lettovský et al. (2000) and Nissen and Haase (2006) propose. A larger subset of duties will increase the computation time a lot. Potthoff et al. (2008) propose another solution approach. In their algorithm, if the solution of a core problem still contains some uncovered tasks, a new core problem is defined. Duties which are not selected by the new core problem use the feasible completion which is selected for them by the previous core problem.

Based on the algorithm of Potthoff et al. (2008), our general solution approach becomes as shown in Figure 5.1. First, we start with an initial core problem without retiming possibilities. Thereafter, for every uncovered task in the solution we use retiming to cover that task. Iteratively, for every uncovered task a new core problem is made with retiming possibilities. The duties selected by the new core problem, get a new feasible completion in the solution. Since retiming is allowed, the duties have more possible feasible completions. Probably, by using these additional feasible completions, we can cover the task, which was uncovered until now.

5.2 Defining a core problem

To define a core problem, a subset of the duties, the tasks and the retimed copies must be selected. As described in the previous section, we have two situations in which we need to select a core problem. First, at the start of the approach we need an initial core problem and thereafter we have to generate a core problem for every uncovered task in the solution.

For the initial core problem we need to focus on selecting an efficient subset of duties, based on the tasks which are rescheduled. This selection procedure is described in Section 5.2.1. For defining a core problem for an uncovered task, we also have to decide which tasks are allowed to have retiming possibilities. Moreover, for each of these tasks the allowed retiming possibilities must be chosen. We discuss how we select the retiming possibilities and the subsets of duties and task in Section 5.2.2. In Section 5.2.3, we use the new subsets to give the mathematical formulation of the core problem.

5.2.1 Selecting subsets for the initial core problem

For a given disruption, the initial core problem uses a subset $\bar{\Delta}$ of the original duties. This subset is generated with exactly the same method as Potthoff et al. (2008) use. The subset $\bar{\Delta}$ contains all stand-by duties and all duties which contain at least one task which is in N_1, N_2 or N_3:

- N_1: Set of all modified tasks
- N_2: Set of all tasks j which are planned on the same route as a task $i \in N_1$ and whose departure time dt_j is between t_0 and $t_1 + d_2$. Here, d_2 is a predefined duration, t_0 is the earliest departure time of all tasks in N_1 ($t_0 = \min\{dt_i \,|i \in N_1\}$) and t_1 is the latest arrival time of all tasks in N_1 ($t_1 = \max\{at_i \,|i \in N_1\}$). In set notation $N_2 = \{j \in N \,|\exists i \in N \text{ with } (ds_i, as_i) = (ds_j, as_j)\} \cap \{j \in N \,|t_0 \leq dt_j \leq t_1\}$
- N_3: Set of all tasks which are part of the same train as tasks in N_1 and N_2.

Let us define $\Delta_N = \{\delta \in \Delta_A | \delta \text{ covers at least one task in } N_1 \cup N_2 \cup N_3\}$, then $\bar{\Delta} = \Delta_R \cup \Delta_N$. After selecting the subset $\bar{\Delta}$ of duties, we have to determine the subset of tasks which will be used. This subset of tasks must contain all tasks which are used in the subset of duties and all newly introduced tasks which are necessary to handle the disruption. These new tasks are currently not covered by any duty. If $N_4 = \left\{i \in N | i \text{ is in at least one duty } \delta \in \bar{\Delta}\right\}$ and $N_5 = \{i \in N | i \text{ is not covered by any duty } \delta \in \Delta\}$, then $\bar{N} = N_4 \cup N_5$. For every task $i \in \bar{N}$ the subset of copies E_i does only contain the copy with the original departure and arrival time. Since we do not have any retimed copy, the constraints about the linkages can

be neglected. Therefore, for all $e \in \bar{E}$ the set $E_e^{\bar{link}}$ is empty. Moreover, since every task does only have one copy, for every $e \in \bar{E}$, $E_e^{bef} = \{e\}$. $\bar{K^{\delta}}$ is the subset of all possible feasible completions for duty δ which can be made by using the tasks in $\bar{N}$.

5.2.2 Selecting subsets for the core problem of an uncovered task

In the initial core problem, no retiming possibilities are used. However, if the initial solution contains still uncovered tasks, iteratively new core problems in which retiming is allowed are made to cover an uncovered task. In this section, we give two possibilities to construct the core problem. Both options determine the retiming possibilities with the same method. If a certain task i is uncovered in the previous solution, we define the tasks which would have retiming possibilities by: $\bar{N^c} = N_6 \cup N_7$.

- N_6: Set of all tasks j which have the same start location as task i and whose departure time is at most d minutes before or after the departure time of task j. $N_6 = \{j \in N | ds_j = ds_i, dt_j \in [dt_i - d_6, dt_i + d_6]\}$.
- N_7: Set of all tasks which are linked to a task of N_6 or N_7 by rolling stock. This set must be generated iteratively since sometimes, if a new task is added to N_7, another task has to be added to N_7.

The two options only differ in the selection of the subset of duties.

Option 1:

In the first option we could use the neighborhood method of Potthoff et al. (2008). Then, the subset of duties contains all duties selected by the neighborhood selection of Potthoff et al. (2008) and all duties which contain at least one task of $\bar{N^c}$. All duties which contain a task of $\bar{N^c}$ are needed to ensure that every duty uses the same copy of a task, which is taken into account in the model. If some duties are not taken into account in the model, the duties $\delta \in \bar{\Delta}$ do not have to use the same copies as the duties $\delta \notin \bar{\Delta}$.

Option 2:

In a second option, the set $\bar{\Delta}$ contains all duties of the initial core problem and all duties which contain at least one task $i \in \bar{N^c}$.

After selecting the subset of duties by either Option 1 or 2, we have to determine the subset of tasks $\bar{N}$. This subset of tasks is the same as for an initial core problem: It contains all tasks which are used in at least one duty $\delta \in \Delta$ and all uncovered tasks ($\bar{N} = N_4 \cup N_5$). For every task $i \in \bar{N^c}$, the subset of copies $\bar{E_i}$ consists of the copy with the original times and some retimed copies. For all other tasks $i \in \{\bar{N} \setminus \bar{N^c}\}$, $\bar{E_i}$ only contains the copy with the original departure and arrival time. The linkages $\bar{E}_e^{link}$ are made according to the rolling stock

schedule and between copies with the same amount of delay. The sets $\bar{E}_e^{bef}$ are constructed as by the definition. Again, $\bar{K}^\delta$ is the subset of feasible completions for duty δ which can be made by using the tasks in $\bar{N}$. Next to these subsets, still one set ($\ddot{N}$) must be defined. This set $\ddot{N}$ is the set of tasks in $\bar{N}$ which are already covered by duties $\delta \notin \bar{\Delta}$. Then, $\bar{\bar{N}} = \{\bar{N} \setminus \ddot{N}\}$ is the set of all tasks which must be covered by the subset of duties Δ.

5.2.3 The core problem formulation

In the previous sections, we have defined the subsets which are needed to construct a core problem. With this subsets the core problem formulation becomes:

$$\min \sum_{\delta \in \bar{\Delta}} \sum_{k \in \bar{K}^\delta} c_k^\delta x_k^\delta + \sum_{i \in \bar{\bar{N}}} f_i y_i + \sum_{e \in \bar{E}} g_e v_e \tag{5.1}$$

$$\text{s.t.} \quad \sum_{\delta \in \bar{\Delta}} \sum_{k \in \bar{K}^\delta} a_{ik}^\delta x_k^\delta + y_i \geq 1 \quad \forall i \in \bar{\bar{N}} \tag{5.2}$$

$$\sum_{k \in \bar{K}^\delta} x_k^\delta = 1 \quad \forall \delta \in \bar{\Delta} \tag{5.3}$$

$$v_e - \sum_{k \in \bar{K}^\delta} b_{ek}^\delta x_k^\delta \geq 0 \quad \forall \delta \in \bar{\Delta}, \forall e \in \bar{E} \tag{5.4}$$

$$\sum_{e \in \bar{E}_i} v_e + y_i = 1 \quad \forall i \in \bar{\bar{N}} \tag{5.5}$$

$$y_i + \sum_{f \in \bar{E}_e^{bef}} v_f - \sum_{d \in E_e^{link}} v_d \geq 0 \quad \forall i \in \bar{\bar{N}}, \forall e \in \bar{E}_i \tag{5.6}$$

$$x_k^\delta \in \{0,1\} \quad \forall \delta \in \bar{\Delta}, \forall k \in \bar{K}^\delta \tag{5.7}$$

$$v_e \in \{0,1\} \quad \forall e \in \bar{E} \tag{5.8}$$

$$y_i \in \{0,1\} \quad \forall i \in \bar{\bar{N}} \tag{5.9}$$

In this formulation we have used Constraints (4.18) instead of Constraints (4.6). Notice that Constraints (4.18) restrict exactly the same as Constraints (4.6), but they are written in another way.

We replace N by $\bar{\bar{N}}$ since all tasks in $\ddot{N}$ are already covered and therefore these tasks do not have to be covered by a duty $\delta \in \bar{\Delta}$. Moreover, tasks in $\ddot{N}$ do not have retimed copies, which makes the copy constraints not necessary for them. $\bar{E}$ only contains copies of tasks $i \in \bar{\bar{N}}$.

Notice that all tasks in $\bar{N}$ are taken into account to generate feasible completions for the duties. This ensures that a duty can always use its original completion if that is still feasible.

5.3 Exploring a core problem

After we have defined the core problem, we have to find a (near) optimal solution for this core problem. This section explains how this solution is generated. As an overall approach we use column generation. By using column generation we do not have to enumerate all possible feasible completions in advance for every duty. During each column generation iteration, new feasible completions are added to the model. These feasible completions are generated with the approach as described in Section 5.3.4. After each iteration, we solve the core problem with all currently known feasible completions. The approach stops if the found feasible solution is reasonable. To check whether the solution is reasonable, we need a lower bound on the optimal solution. To generate such a lower bound, we use Lagrangian relaxation, which is described in Section 5.3.1. In Section 5.3.3, a subgradient method is described to find Lagrangian multipliers which result in a good lower bound.

We use the best Lagrangian multipliers of the Lagrangian relaxation to generate a feasible solution for the core problem. The Lagrangian multipliers which give the best lower bound do not have to be the multipliers which lead to the best feasible solution. Therefore, we use every column generation iteration also X other sets of multipliers to generate feasible solutions. The generation of these feasible solutions is described in Section 5.3.2. Each feasible solution is an upper bound on the optimal solution of the core problem.

The column generation approach stops when a maximum number of iterations is reached or when the difference between the Lagrangian lower bound and the feasible upper bound has become small enough. Several ways to accelerate the column generation approach are described in Section 5.3.5.

5.3.1 Lagrangian subproblem

In this section we explain how we can apply Lagrangian relaxation to Constraints (5.2), Constraints (5.4) and Constraints (5.6) of the core problem formulation of Section 5.2.3. For Constraints (5.2) we use Lagrangian multipliers λ_i, $\forall i \in \bar{\bar{N}}$, for Constraints (5.4) we use the multipliers μ_e^δ, $\forall e \in \bar{E}, \forall \delta \in \bar{\Delta}$ and for Constraints (5.6) we use the multipliers η_e, $\forall e \in \bar{E}$. Notice that Constraints (5.6) are not necessary for tasks which do not have retimed copies. To take this into account, we set η_e equal to 0 for all copies e which are a copy of a task with no retimed copies. For all these copies e, it is not allowed to change η_e during the solution approach. By applying the Lagrangian relaxation to the core problem formulation,

the Lagrangian subproblem equals:

$$\begin{aligned}\Theta(\lambda,\eta,\mu) &= \min \sum_{\delta\in\Delta}\sum_{k\in K^\delta} c_k^\delta x_k^\delta + \sum_{i\in\bar{\bar{N}}} f_i y_i + \sum_{e\in\bar{E}} g_e v_e + \sum_{\delta\in\bar{\Delta}}\sum_{e\in\bar{E}} \mu_e^\delta(-v_e + \sum_{k\in K^\delta} b_{ek}^\delta x_k^\delta) \\ &+ \sum_{i\in\bar{\bar{N}}} \lambda_i(1 - \sum_{\delta\in\bar{\Delta}}\sum_{k\in\bar{K}^\delta} a_{ik}^\delta x_k^\delta - y_i) \\ &+ \sum_{i\in\bar{\bar{N}}}\sum_{e\in\bar{E}_i} \eta_e(\sum_{d\in\bar{E}_e^{link}} v_d - \sum_{f\in\bar{E}_e^{bef}} v_f - y_i) \\ &\quad \text{s.t. } (5.3), (5.5), (5.7), (5.8) \text{ and } (5.9)\end{aligned} \tag{5.10}$$

By using some different notation, the objective function (5.10) of this subproblem can be rewritten as (5.11). Notice that $\sum_{e\in\bar{E}}$ is equal to $\sum_{i\in\bar{\bar{N}}}\sum_{e\in\bar{E}_i}$, since every copy e is a copy of exactly one task i. Moreover, for simplification, two new sets E_e^p, E_e^q and a new variable γ_e are used for all $e \in \bar{E}$.

- $E_e^p = \left\{ d \in \bar{E} | e \in E_d^{\bar{link}} \right\}$
- $E_e^q = \left\{ d \in \bar{E} | e \in E_d^{\bar{bef}} \right\}$
- $\gamma_e = \sum_{d\in E_e^p} \eta_d - \sum_{d\in E_e^q} \eta_d$

$$\begin{aligned}\Theta(\lambda,\eta,\mu) &= \min \sum_{i\in\bar{\bar{N}}} \lambda_i + \sum_{\delta\in\Delta}\sum_{k\in K^\delta} (c_k^\delta + \sum_{e\in\bar{E}} \mu_e^\delta b_{ek}^\delta - \sum_{i\in\bar{\bar{N}}} \lambda_i a_{ik}^\delta) x_k^\delta + \sum_{i\in\bar{\bar{N}}} (f_i - \lambda_i - \sum_{e\in\bar{E}_i} \eta_e) y_i \\ &+ \sum_{i\in\bar{\bar{N}}}\sum_{e\in\bar{E}_i} (g_e + \gamma_e - \sum_{\delta\in\bar{\Delta}} \mu_e^\delta) v_e \\ &\quad \text{s.t. } (5.3), (5.5), (5.7), (5.8) \text{ and } (5.9)\end{aligned} \tag{5.11}$$

For given vectors λ, η and μ, the value $\Theta(\lambda,\eta,\mu)$ can be calculated with a simple approach. First, we determine the values for all binary variables x_k^δ. To ensure that Constraints (5.3) are not violated, for every duty $\delta \in \bar{\Delta}$ we set x_k^δ equal to 1 for exactly one $k \in \arg\min \left\{ \bar{c}_k^\delta(\lambda,\eta,\mu) \middle| k \in \bar{K}^\delta \right\}$. In this method $\bar{c}_k^\delta(\lambda,\eta,\mu) = (c_k^\delta + \sum_{e\in\bar{E}} \mu_e^\delta b_{ek}^\delta - \sum_{i\in\bar{\bar{N}}} \lambda_i a_{ik}^\delta)$. The values of the variables y_i and v_e can be chosen independently from the values of the variables x_k^δ. The algorithm in Figure 5.2 determines for every task $i \in \bar{\bar{N}}$ the values for the variables y_i and v_e ($\forall e \in \bar{E}_i$) in such a way that Constraints (5.5) are not violated.

5.3.2 Combining column generation with Lagrangian relaxation

The Lagrangian dual problem of the Lagrangian subproblem of Section 5.3.1, is to find Θ^*.

$$\Theta^* = \max \Theta(\lambda,\eta,\mu), \quad \lambda \geq 0, \eta \geq 0 \text{ and } \mu \geq 0 \tag{5.12}$$

For all $e \in \bar{E}_i$ determine $\bar{g}_e = (g_e + \gamma_e - \sum_{\delta \in \bar{\Delta}} \mu_e^\delta)$;
Select $e^* \in \arg\min \{\bar{g}_e \,|e \in \bar{E}_i\}$;
if $\bar{g}_{e^*} \leq f_i - \lambda_i - \sum_{e \in \bar{E}_i} \eta_e$ **then**
    Set $y_i = 0$, $v_{e^*} = 1$ and for all $e \in \bar{E}_i \setminus \{e^*\}$, set $v_e = 0$
else
    Set $y_i = 1$ and for all $e \in \bar{E}_i$, set $v_e = 0$
end

Figure 5.2: Algorithm to determine y_i and v_e

As already described, we apply column generation since the number of possible feasible completions of a certain duty can be enormous. Therefore, we do not solve the Lagrangian dual problem for the overall core problem, but we solve it after every column generation iteration for a restricted master problem, with the then known subset of feasible completions. As a result, the restricted master problem uses a subset of the x_k^δ variables.

Every column generation iteration, some additional x_k^δ variables are added to the restricted master problem. After the n^{th} iteration, of all feasible completions which can finish a certain duty δ, a subset K_n^δ is used by the restricted master problem. Since in every iteration columns could only be added to the restricted master problem and not be removed, $\bar{K}_{n-1}^\delta \subseteq \bar{K}_n^\delta$. K_0^δ always includes the feasible completion which contains no tasks. This completion is called the artificial completion: After the completion of his current task, the crew member will directly take a taxi to its own crew base. The artificial completion is taken into account to ensure that every crew member has always at least one feasible completion to choose. However, we give high costs to the use of an artificial completion. If the end location of the current task of a duty is equal to the crew base of that duty, the cost of using an artificial completion is significantly lower, since a taxi is not necessary. Next to the artificial completion, K_0^δ contains, if it is feasible, the original completion of the duty which has no cost: $c_k^\delta = 0$.

During every column generation iteration, we use subgradient optimization to approximate Θ_n^*, which represents the optimal value of the Lagrangian dual problem of the n^{th} restricted master problem.

$$\Theta_n^* = \max \Theta_n(\lambda, \eta, \mu), \qquad \lambda \geq 0, \eta \geq 0 \text{ and } \mu \geq 0 \tag{5.13}$$

The difference between $\Theta_n(\lambda, \eta, \mu)$ and $\Theta(\lambda, \eta, \mu)$ is that we do not know all possible feasible completions in $\Theta_n(\lambda, \eta, \mu)$. In $\Theta_n(\lambda, \eta, \mu)$, the sets $\bar{K}^\delta$ are replaced by $\bar{K}_n^\delta$. In Section 5.3.3, the subgradient optimization method is explained in more detail. The vectors λ^n, η^n and μ^n are the Lagrangian multipliers corresponding to the best approximation of Θ_n^*. After the

subgradient optimization, we can check with these multipliers if we need additional feasible completions in the next column generation iteration. Moreover, we can use the multipliers to calculate a lower bound on Θ^*.

For the $(n+1)^{th}$ column generation iteration, at most P feasible completions per duty are added to the restricted master problem. Given the best found multipliers λ^n, η^n and μ^n of the n^{th} iteration, a feasible completion k is added for duty δ if it has lower reduced costs $\bar{c}_k^\delta(\lambda^n, \eta^n, \mu^n)$ than all currently known feasible completions of δ in the restricted master problem. We add per duty at most P feasible completions per iteration and ideally, these P feasible completions are the best ones. It requires a lot of computations to select the best P feasible completions of a duty. However, by formulating that problem as a resource constrained shortest path problem we can select the best feasible completion $k^* \in K^\delta$. The best feasible completion k^* is the feasible completion with the lowest reduced costs given the currently best known multipliers: $k^* \in \arg\min \left\{ \bar{c}_k^\delta(\lambda^n, \eta^n, \mu^n) \,\middle|\, k \in \bar{K^\delta} \right\}$. With this best feasible completion we can calculate a lower bound LB_n on Θ^*. The best known lower bound of Θ^* is equal to $\Theta_n^* + \sum_{\delta \in \bar{\Delta}} (\bar{c}_{k^n}^\delta(\lambda^n, \eta^n, \mu^n) - \bar{c}_{k^*}^\delta(\lambda^n, \eta^n, \mu^n))$, but since we only approximate Θ_n^*, we cannot calculate this lower bound. However, because $\Theta_n(\lambda^n, \eta^n, \mu^n) \leq \Theta_n^*$, $LB_n = \Theta_n(\lambda^n, \eta^n, \mu^n) + \sum_{\delta \in \bar{\Delta}} (\bar{c}_{k^n}^\delta(\lambda^n, \eta^n, \mu^n) - \bar{c}_{k^*}^\delta(\lambda^n, \eta^n, \mu^n))$ is also a lower bound on Θ^*.

During the approach to find the best feasible completion k^*, which is described in Section 5.3.4, some other feasible completions are generated. Of these other feasible completions we select at most $P-1$ completions with the lowest reduced cost and we put them together with k^* in a set Kp_{n+1}^δ. Let $k^n \in \arg\min \left\{ \bar{c}_k^\delta(\lambda^n, \eta^n, \mu^n) \,\middle|\, k \in \bar{K_n^\delta} \right\}$ be the optimal feasible completion of the current n^{th} restricted master problem. Then, the new subset of feasible completions for the $(n+1)^{th}$ iteration $K_{n+1}^\delta = K_n^\delta \cup \left\{ k \in Kp_{n+1}^\delta \,\middle|\, \bar{c}_k^\delta(\lambda^n, \eta^n, \mu^n) < \bar{c}_{k^n}^\delta(\lambda^n, \eta^n, \mu^n) \right\}$.

Next to an accurate lower bound, we are especially interested in a tight upper bound on the cost, which is the objective value of a feasible solution of the core problem. Since in the Lagrangian subproblem some of the restrictions of the core problem are relaxed, the solution of the Lagrangian subproblem does not have to be feasible for the core problem. To generate a feasible solution for the core problem given some Lagrangian multipliers λ, η and μ, we use the greedy algorithm in Figure (5.3).

The greedy algorithm is based on the greedy algorithm of Potthoff et al. (2008). In this algorithm, we select for every duty the best feasible completion. If it is the first time that a certain task appears in a selected feasible completion, the copy which is used for that task, will be the copy that is allowed to be used in all duties. After a certain copy for a task is selected, all feasible completions which use another copy of the same task will be ignored. Moreover, we ignore feasible completions which use copies, which, due to the rolling stock schedule, cannot be used together with the current copy. Since every set $\bar{K_n^\delta}$ contains the

1 Order the original duties $\delta \in \bar{\Delta}$ based on the reduced cost;
2 Set $y_i = 1$ for all $i \in \bar{\bar{N}}$ and set $v_e = 0$ for all $e \in \bar{E}$;
3 Set $\hat{\lambda} = \lambda$, $\hat{\eta} = \eta$ and $\hat{\mu} = \mu$;
4 **forall** $\delta \in \bar{\Delta}$ **do**
5 Choose $k^*(\delta) \in \arg\min\{\bar{c}_k^\delta(\hat{\lambda}, \hat{\eta}, \hat{\mu}) | k \in \bar{K}_n^\delta\}$ and set the corresponding $x^\delta_{k^*(\delta)} = 1$;
6 Set $\hat{\lambda}_i = 0$ and $y_i = 0$ for all $i \in \bar{\bar{N}}$ with $a^\delta_{ik^*(\delta)} = 1$;
7 **forall** $e \in \bar{E}$ with $b^\delta_{ek^*(\delta)} = 1$ **do**
8 Define E^*: the set of copies which, by using copy e, are not allowed to be used;
9 Define K^*: the set of completions which use at least one copy $d \in E^*$;
10 Ignore $\forall \delta \in \bar{\Delta}$ the completions $k \in K^*$ out of $\bar{K}_n^\delta$;
11 Set $v_e = 1$ and $\hat{\eta}_e = 0$;
12 **end**
13 **end**
14 Set $\hat{\lambda}_i = f_i$ if $y_i = 1$ for all $i \in \bar{\bar{N}}$;
15 Construct the set of idle stand-by duties $\bar{\Delta}_I = \{\delta \in \bar{\Delta}_R | a^\delta_{ik^*(\delta)} = 0 \text{ for all } i \in \bar{\bar{N}}\}$;
16 **forall** $\delta \in \bar{\Delta}_I$ **do**
17 Set $x^\delta_{k^*(\delta)} = 0$;
18 Repeat lines 5 until 12;
19 **end**
20 Check if $\sum_{e \in \bar{E}_i} \sum_{\delta \in \bar{\Delta}} \sum_{k \in K^\delta} b^\delta_{ek} x^\delta_k = 0$ for all $i \in \left\{ i \in \bar{\bar{N}} | y_i = 1 \right\}$. If this condition holds, a feasible solution is found.

Figure 5.3: Greedy procedure to construct feasible solutions

artificial completion, it is ensured that every duty always has at least one remaining feasible completion. Since the artificial completion does not use any copy, it will never be ignored from $\bar{K}_n^\delta$.

If after the feasible completions of the duties have been selected, still some tasks are uncovered, we check if the idle stand-by duties can cover those tasks. A stand-by duty is idle if the selected feasible completion does not cover any tasks.

After copies and tasks have been covered, we adapt the Lagrangian multipliers to make it preferable for the algorithm to select copies of other tasks. In this way we enforce the algorithm to select feasible completions which cover different tasks than already selected feasible completions do. In Line 11 of the algorithm, we set $\bar{\eta}_e$ equal to 0, since that makes it more preferred to select a copy linked to copy e and less preferred to take a copy (with less

or equal delay) of the same task as copy e. We do not set $\hat{\mu}_e^\delta$ equal to 0, $\forall \delta \in \Delta$ since that will make it more preferred to use this copy e again instead of copies of other tasks.

The greedy algorithm in Figure (5.3) does not guarantee that we find a feasible solution, however in most cases it will. Only in the extraordinary case that a crew member is assigned to be a passenger on a train which is not covered by a driver, the solution is not feasible. This condition is checked in Line 20. The problem that a driver must be a passenger on a train which has no driver only occurs on the task level. Within a task, due to the algorithm, it is not possible that a driver uses a deadhead copy representing another copy than the selected copy. For example, imagine that from task i with 2 copies (d and e), copy e is selected by the algorithm. The algorithm is made in such a way that it ignores all completions which contain a (deadhead) copy representing copy d. This means that we ensure that no other copies than the selected copy e will be used as a deadhead copy. However on the task level, if a task i is not covered at all, the completions which use a deadhead task, representing task i are not ignored by the algorithm.

After every column generation iteration, we use the greedy algorithm to generate at most X feasible solutions for the overall problem. We use the greedy algorithm to generate solutions for the last X sets of multiplier vectors λ, η and μ of the subgradient optimization algorithm. Of the X generated solutions, we only look at the feasible solutions and we save the feasible solution with the lowest costs. The cost of this solution is an upper bound (UB^*) on the cost of the optimal solution of the core problem. The column generation stops if the gap between the upper bound UB^* and the lower bound LB_n is small enough or if a maximum number of iterations is reached.

5.3.3 Subgradient optimization

In the n^{th} column generation iteration, the optimal $\Theta_n^*(\lambda, \eta, \mu)$ is approximated with subgradient optimization. First, we present the general subgradient optimization as described in Beasley (1993). Thereafter, we introduce some modifications which will speed up the algorithm. Within the subgradient optimization, the Lagrangian multipliers λ, η and μ are updated several times. For a given multiplier $\chi(1)$, the updated value $\chi(2) = \chi(1) + T \bullet S$ in which T is the step size and S is the search direction. In a general subgradient optimization method, T is equal to a chosen norm and S is equal to the subgradient: The left hand side of the relaxed constraint corresponding to χ.

Notice that the subgradient optimization does not necessarily reach Θ_n^*. The notation λ^n,η^n and μ^n is used for the best multipliers which are found during the whole subgradient optimization. These best multipliers do not necessarily have to be the multipliers generated in the last subgradient optimization iteration. Although the value of $\Theta_n(\lambda^n, \eta^n, \mu^n)$ will not

necessarily reach Θ_n^*, the value $\Theta_n(\lambda^n, \eta^n, \mu^n)$ can still be used for computing a lower bound. Better approximations of Θ_n^*, generally lead to better lower bounds.

Let Z_{UB} be the objective value of the current best feasible solution of the overall problem. Then, the subgradient optimization algorithm becomes as presented in Figure 5.4. The algorithm uses five stopping criteria:

- Stop if: $(Z_{UB} - Z_{max}) < maxGap$, for a chosen maximum allowed gap $maxGap$.
- Stop if: $g \geq maxG$, in which $maxG$ is a selected maximum number of iterations.
- Stop if: $T < minT$, in which $minT$ is the minimum step size.
- Stop if: $norm$ is equal to 0. In such a case, all constraints of the core problem are met. This means that if the norm is equal to 0, the Lagrangian solution is a feasible solution for the core problem.
- Stop if: all $\lambda(g), \eta(g)$ and $\mu(g)$ are equal to 0 and all subgradients $F(g), G(g)$ and $H(g)$ are lower than or equal to 0. Under these conditions, the Lagrangian solution is a feasible solution for the core problem.

Beasley (1993) suggests, based on his experiences, some modifications which could be applied to the subgradient optimization. The first modification is to halve π if Z_{max} has not improved the last n iterations. In that case, we could add an additional stopping criterion that stops the optimization if π becomes too small. Moreover, he suggests to multiply Z_{UB} by 1.5 in the formula of the gap. This prevents that the step size T becomes very small if the lower bound $\Theta(\lambda(g-1), \eta(g-1), \mu(g-1))$ becomes close to the upper bound Z_{UB}.

Another modification of Beasley (1993) is to use an alternative norm. If a Lagrangian multiplier $\lambda(g)$, $\eta(g)$ or $\mu(g)$ is equal to 0 and the corresponding subgradient $F(g)$, $G(g)$ or $H(g)$ is less than or equal to 0, the Lagrangian multiplier will not change. However, if the subgradient is less than 0, the square of the subgradient is greater than 0 and will therefore increase the norm. Beasley (1993) suggests that if we have Lagrangian multipliers $\chi(g)$ and subgradients $S(g)$, we must use the new norm $\sum_{i \in I}(S_i^*(g))^2$ instead of the norm $\sum_{i \in I}(S_i(g))^2$, where:

$$S_i^*(g) = \begin{cases} \max(0, S_i(g)) & \text{if } \chi(g) = 0 \\ S_i(g) & \text{otherwise} \end{cases}$$

Next to the modifications of Beasley (1993), we also use a modification of Camerini et al. (1975). Camerini et al. (1975) suggest to use for the search direction S not only the current

$g = 0$ (counter for the iteration number);
Choose $\pi \in [0, 2]$;
Let $\lambda(0)$, $\eta(0)$ and $\mu(0)$ all be 0 vectors;
Determine the optimal solution of $\bar{x}_k^\delta, \bar{y}_i, \bar{v}_e$ for $\Theta(\lambda(0), \eta(0), \mu(0))$;
$Z_{\max}(0) = \Theta(\lambda(0), \eta(0), \mu(0))$;
while stop criteria are not met **do**
    $g = g + 1$;
    $F_i(g) = 1 - \sum_{\delta \in \bar{\Delta}} \sum_{k \in \bar{K}^\delta} a_{ik}^\delta \bar{x}_k^\delta - \bar{y}_i, \forall i \in \bar{\bar{N}}$;
    $G_e(g) = \sum_{d \in \bar{E}_e^{link}} \bar{v}_d - \sum_{f \in \bar{E}_e^{bef}} \bar{v}_f - y_{i_e}, \forall e \in \bar{E}$;
    $H_e^\delta(g) = \sum_{k \in \bar{K}^\delta} b_{ek}^\delta \bar{x}_k^\delta - \bar{v}_e, \forall e \in \bar{E}$ and $\forall \delta \in \bar{\Delta}$;
    Let $norm = \sum_{i \in \bar{\bar{N}}} (F_i(g))^2 + \sum_{e \in \bar{E}} (G_e(g))^2 + \sum_{e \in \bar{E}} \sum_{\delta \in \bar{\Delta}} (H_e^\delta(g))^2$;
    Let $gap = \pi(Z_{UB} - \Theta(\lambda(g-1), \eta(g-1), \mu(g-1)))$;
    Then $T = \frac{gap}{norm}$;
    Set $\lambda_i(g) = \max\{0, \lambda_i(g-1) + T \bullet F_i(g-1)\}, \forall i \in \bar{\bar{N}}$;
    Set $\eta_e(g) = \max\{0, \eta_e(g-1) + T \bullet G_e(g-1)\}, \forall e \in \bar{E}$;
    Set $\mu_e^\delta(g) = \max\left\{0, \mu_e^\delta(g-1) + T \bullet H_e^\delta(g-1)\right\}, \forall e \in \bar{E}$ and $\forall \delta \in \bar{\Delta}$;
    Determine the optimal solution of $\bar{x}_k^\delta, \bar{y}_i, \bar{v}_e$ for $\Theta(\lambda(g), \eta(g), \mu(g))$;
    $Z_{\max}(g) = \max\{Z_{\max}(g-1), \Theta(\lambda(g), \eta(g), \mu(g))\}$;
end

Figure 5.4: Subgradient optimization for the Lagrangian dual problem

subgradients $F(g), G(g)$ or $H(g)$, but also part of the last search direction. For a given multiplier $\chi(g)$ with corresponding subgradient $I(g)$, the search direction $S(g)$ can be calculated with the following formula: $S(g) = I(g) + \xi_g S(g-1))$ with:

$$\xi_g = \begin{cases} -\frac{S(g-1)I(g)}{(S(g-1))^2} & \text{if } S(g-1)I(g) \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

5.3.4 Solving the pricing problems

For every duty in the core problem, we construct a directed acyclic graph which contains all possible feasible completions. In these graphs, a node represents an arrival or a departure of a copy and an arc represents a copy or a transfer between two copies. To construct a graph for a duty, we start with using the departure and arrival nodes of all copies $e \in \bar{E}$. Thereafter, we can make arcs from an arrival node to a departure node or from a departure node to an arrival node.

First, we generate for every copy $e \in \bar{E}$ an arc from its departure node to its arrival node. This arc is a copy arc if the driver of the duty is allowed to drive this copy and a deadhead arc if he is not allowed to drive the copy. Secondly, all possible transfer arcs are generated. Between an arrival node of copy e_1 and a departure node of copy e_2, a transfer arc is made if it is, by location and time, allowed to perform e_2 directly after e_1.

Moreover, a source and a sink are added to the graph. The source represents the arrival of task i_x, in which i_x is the last task in the original duty which started before the time of rescheduling. Transfer arcs are made from the source to all departures which are at the same location and to which enough transfer time is available for the driver. The sink represents the latest allowed arrival of the crew member at his crew base. From every arrival node at the same location as the crew base, an arc is constructed to the sink if the arrival time is at least the sign-off time before the latest allowed end time of the duty. If the source and the sink are at the same location, an additional transfer arc is made from the source to the sink. Since arcs are only made from an earlier point in time to a later point in time, the graph does not contain any cycles.

Every path from source to sink represents a possible completion of the duty. This means that we can remove all paths which do not start at the source or end in the sink. However, not all paths from source to sink represent automatically a feasible completion. We have to check if the completion that the path represents is feasible according to the meal break constraint. The meal break constraint requires that duties with a duration longer than 5 1/2 hours must contain a meal break of at least 30 minutes. Moreover, to be feasible, the duration of the duty before and after the meal break may not exceed 5 1/2 hours.

We have to find the feasible completion with the lowest reduced cost. The subproblem to find the optimal feasible completion is also called the pricing problem. Every arc contains cost for using the arc, values of the Lagrangian multipliers belonging to the arc, a duration and an indication if a meal break fits into it. Notice that also paths with more than one meal break possibility are feasible completions. The duration of an arc is equal to the difference between the times of its start and end node.

To check if a path is feasible according to the meal break constraint we also need information on the part of the duty which is already finished. At the time of rescheduling, the crew member has already done some tasks and maybe he already had his meal break. To take this into account, the duration of an arc which starts at the source is not equal to the difference between the time of its start node and the time of its end node. The duration of such an arc is equal to the difference between the start time of the duty and the time which the end node of the arc represents. If the crew already had a meal break possibility, we indicate this on the first arc.

If a path from source to sink is shorter than 5 1/2 hours, the path is always a feasible completion. For all other paths, to determine if the path represents a feasible completion, we have to check if the meal break constraint is met.

Concluding, the aim is to find the shortest path which meets the meal break constraint. Notice that a shortest path means a path with the lowest costs and not a path with the shortest duration. To find the shortest path which meets the meal break constraint, we use a resource constraint shortest path algorithm based on the algorithm of Irnich and Desaulniers (2005). This algorithm, is presented in Figure 5.5.

```
1  Set U = {(source)} and R = ∅;
2  while U ≠ ∅ do
3      Choose a path Q ∈ U and remove Q from U;
4      for all arcs (v(Q), w) ∈ A do
5          if the path (Q, w) is feasible then
6              if w is equal to the sink then
7                  Add the path (Q, w) to R;
8              else
9                  Add the path (Q, w) to U;
10             end
11         end
12     end
13     Remove dominated paths in U;
14 end
15 The optimal path Q* = arg min {C(Q) |Q ∈ R}
```

Figure 5.5: Resource constraint shortest path algorithm

In the algorithm, A is the set of arcs and $v(Q)$ is the last node of the partial path Q. This algorithm contains a dominance step in Line 13. In this step a path Q dominates another path Y if $v(Q) = v(Y)$, $TMB(Q) \leq TMB(Y)$ and $C(Q) \leq C(Y)$. Here, $TMB(Q)$ indicates the end time of the last possible break in path Q and $C(Q)$ represents the reduced cost of path Q. The last possible break must always start before 5 1/2 hours after the start time of the duty. Line 3 checks if the meal break constraint is met by checking if $TMB(Q) \leq 5$ 1/2 hours.

For an efficient dominance step, it is necessary that we do not arbitrarily choose a path $Q \in U$ in Line 3. Therefore, we introduce $TN(w)$ which indicates the arrival or de-

parture time that the node $v(Q)$ represents. Then, we must select in Line 3 path $Y \in \arg\min\{TN(v(Q)) \,|Q \in U\}$.

As already mentioned in Section 5.3.2, we do not only add the best feasible completion Q^* to the model. We also add $P-1$ other paths in R to the model, if they exist and if they have low enough costs.

It is clear that every feasible completion may contain only one copy per task. However, this resource constraint shortest path procedure does not exclude paths which use two different copies of the same task. In the current solution approach, if the delay of a copy is larger than the duration of the task, it could happen that one path contains two different copies of the same task. We do not add direct arcs between copies which represents the same task, but it is possible that there is a path between the two copies of the same task via a copy of another task. We can take this into account in the feasibility check in Line 5. However, this will increase the computation time. Another option to deal with it could be to restrict the allowed retiming possibilities. First, we determine t_min which is the minimum duration of a task in $\bar{N}$. For task i we then must not allow delays which are longer than the duration of task i plus t_min. A last option could be, to remove some arcs from the graph. For every task we can determine how many copies of it are in the graph. If multiple copies of one task i are in the graph, we remove the copies of task i whose departure time is more than t_min later than the arrival time of the earliest copy of task i. Again all paths which do not start at the source and/or end at the sink must be removed from the graph. A disadvantage of this option is that it decreases the number of possible feasible completions per duty. If we do not want such a decrease, we have to select the first option.

In Figure 5.6, an example of a graph for a duty is given. In this graph we have three tasks $\{2, 4, 5\}$ and 4 copies. In this figure $\text{arc}(t_i^d, t_i^a)$ represents a copy of task i, s_1 represents the source and t_1 represents the sink. For task 2, next to the original copy $(\text{arc}(t_2^d, t_2^a))$ there is also a copy $(\text{arc}(t_{2^*}^d, t_{2^*}^a))$ available with a delay of 10 minutes.
All arcs have certain costs:

- C_c: Cost for changing a duty
- C_o: Cost for covering a task which was in the original duty
- C_n: Cost for covering a task which was not in the original duty
- C_o^t: Cost for a transfer which was in any original duty
- C_n^t: Cost for a transfer which was not in any original duty

In the figure, if the duty selects the original copy of task 2 $(\text{arc}(t_2^d, t_2^a))$, the waiting time between the last arrival of the duty and the departure of the $\text{arc}(t_2^d, t_2^a)$ is not enough to have

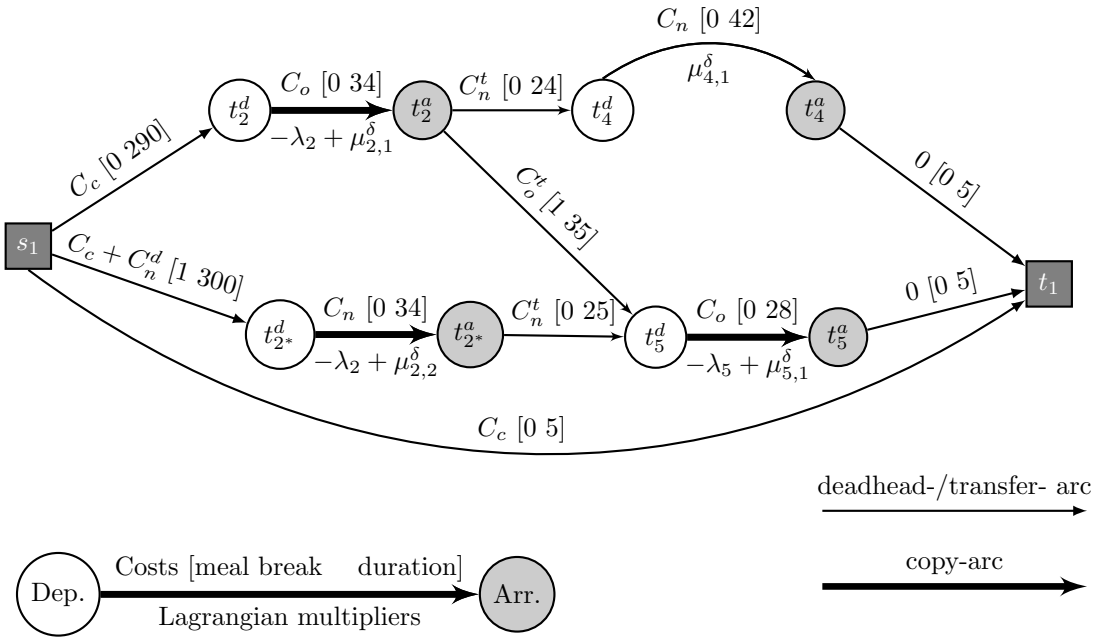

Figure 5.6: Example of a column generation for duty δ of model A

a meal break. However, if it takes the delayed copy (arc($t^d_{2^*}, t^a_{2^*}$)), the transfer time is enough to have a meal break. Between the original copy of task 2 and the original copy of task 5, there is also a meal break possibility. The driver is not allowed to drive task 4, but he can use it to transfer as a passenger to another location. However, the driver will never use this deadhead copy, because the only path which uses this copy exceeds the 5 1/2 hours and does not have a meal break possibility.

5.3.5 Acceleration strategies

Potthoff et al. (2008) use two approaches to speed up the procedure. The first approach is to use partial pricing in which the approach stops solving the pricing problems if it already found better feasible completions (columns) for more than a certain percentage of the duties. This means that it is possible that for only a part of the duties a new feasible completion is added to the restricted master problem. For this strategy, a remark must be made that LB_n can only be computed if all pricing problems are solved.

The second approach, suggested by the paper, is to fix columns. With column fixing we select a fixed feasible completion k for a duty δ and we set the variable x^{δ}_k equal to 1 and x^{δ}_l equal to 0 for all other feasible completions $l \in \bar{K}^{\delta}$. During all further steps of the solution approach no other feasible completion can be assigned anymore to duty δ. Moreover, if we fix a column we also fix the copies which are used in the column. From that moment on, copies

which are in conflict with the fixed copies, will be ignored in the remainder of the approach. So, every time some columns are fixed, the number of possible copies will be reduced. This also means that the graphs can change during the approach. The column fixing phase does not guarantee to speed up the procedure but it leads in most cases to better feasible solutions.

We switch to a column fixing phase if the relative gap between the lower bound (LB_n) and the approximation of Θ_n^* is smaller than a certain threshold value. Every step of the fixing phase, we start with fixing at most Y columns. Thereafter we continue the column generation process for the non-fixed duties. If the improvement of the newly added columns of the non fixed duties is less than a certain threshold, we start with the next step of the fixing phase and again fix at most Y columns. An overview of this algorithm is given in Figure 5.7.

1 **while** not all duties are fixed **do**
2 Fix at most Y duties;
3 Determine $\bar{\Delta}^* = \{\delta \in \bar{\Delta} \,|\delta \text{ is not fixed}\}$;
4 Apply column generation until the first iteration n for which:
$\sum_{\delta \in \bar{\Delta}^*}(c^{\bar{\delta}}_{k^n}(\lambda^n, \eta^n, \mu^n) - c^{\bar{\delta}}_{k^*}(\lambda^n, \eta^n, \mu^n)) < \epsilon$;
5 **end**

Figure 5.7: Fixing phase

In Line 2 of the algorithm, we have to fix at most Y columns. We fix the columns which were selected often in the solutions of the Lagrangian subproblems. To determine which columns must be fixed, we calculate for every original duty δ and for every feasible completion k the ratio $R_k^\delta = \frac{s_k^\delta}{g}$. In this ratio, g is the number of subgradient iterations and s_k^δ is the number of times x_k^δ was set equal to 1 during the subgradient optimization iterations. The feasible completions are ordered by decreasing values of R_k^δ. We select the highest R_k^δ and fix the corresponding feasible completion k to the corresponding duty δ. Then, we ignore all feasible completions k^n which are not allowed in combination with completion k and we ignore their corresponding $R_{k^n}^\delta$. Thereafter, we apply the same procedure to the feasible completion k_1 and duty δ_1 belonging to the next largest value of R_k^δ as long as $R_k^\delta \geq \beta$ and the number of fixed columns has not passed Y. This procedure of Potthoff et al. (2008) is closely related to the α-fixing procedure proposed by Holmberg and Yuan (2000).

The disadvantage of column fixing is that it is possible that an infeasible solution is fixed. The only constraint that can be violated is the constraint that a crew member may not be a passenger on a train which has no driver. If we fix a column in which a driver has a deadhead task, it could happen that in a further stage no driver can be found to drive the trip of that

task. Then, after the column is fixed, no better feasible solution can be found anymore. This problem with the fixing phase could happen since we look only at how many times a column is selected in a Lagrangian relaxation solution. This Lagrangian relaxation solution does not have to be feasible. Notice, that we are interested in the best feasible solution found during the process and that solution does not have to contain all fixed columns.

In the initial core problem we do not use retimed copies, which allows us to use the method of Potthoff et al. (2008) to solve the initial core problem. Since the model suggested in Chapter 4 is larger than the model of Potthoff et al. (2008), the computation time can be reduced if we use the method of Potthoff et al. (2008) to solve the initial core problem. However, the model of Potthoff et al. (2008) allows that a crew member is a passenger on a train which has no driver. Since this is not allowed in our case, there are two options to deal with this. The first option is that we allow such infeasible solutions in the initial core problem since they will be made feasible in further core problems. If a task is canceled in the initial core problem, a new core problem is determined to cover this task. During the approach to solve this new core problem it is not allowed anymore to be a passenger on a train which has no driver. The disadvantage of this method is that the solution of the initial core problem cannot be used as an upper bound.

In the second option, we neglect infeasible solutions generated by the greedy heuristic of Potthoff et al. (2008), just as in our suggested approach in this chapter. Since in contrast to our approach, the approach of Potthoff et al. (2008) does not correct for these infeasible solutions, the subgradient optimization could point to a direction which only contains infeasible solutions. In such a case the approach could end up with a very bad feasible solution.

After the initial core problem is solved, iteratively for every uncovered task a core problem must be solved. In Section 5.2.2 we have given two options to determine such a core problem. In Option 2, the core problem contains all duties which were used in the initial core problem. Moreover, some additional duties are added, if that is necessary for the retiming options. Mostly, the sets of additional duties and additional retimed copies are very small. Therefore, we suggest to make one overall problem which is equal to the union of all core problems for the uncovered tasks. This overall problem is in most cases not much larger than one individual core problem. Instead of iteratively solving the core problems, we have to solve an overall problem once. This can significantly reduce the computation time and can lead to better solutions.

Chapter 6

Cancellation of trains

Imagine, a train must be canceled because no crew member is found to drive that train. In that case the rolling stock schedule is not feasible anymore. For example, if a train is canceled between Breda (Bd) and Dordrecht (Ddr), some rolling stock composition RA will end at Breda and a new rolling stock composition RB is needed at Dordrecht to complete the duty of RA. Multiple questions arise in this case. Is there anyway some rolling stock available at Dordrecht? Is there some place at Breda were RA can be placed such that it does not block the tracks? How do we manage that RA and RB are parked at their planned shunting yard at the end of the day?

If the crew rescheduling approach is not able to cover all crew tasks, it could be useful to take some of these questions into account. In this chapter we give an introduction to how the model can be adapted in such a way that the rolling stock schedule is taken into account if a train must be canceled. However, we will not implement this since more research is required. A pitfall in the suggested model of this chapter is that the core problems become very large.

The assumptions which are used in this chapter are: At every location with a shunting yard there is always rolling stock available to cover a task and at every location there is enough place to store the rolling stock. For example, consider two succeeding tasks i and j which use the same rolling stock. If task i ends at a location with a shunting yard, it is reasonable to assume that there is always some rolling stock available at the shunting yard to cover task j if task i is canceled. However, if task i is canceled and the task does not end at a location with a shunting yard, it is not reasonable that there is rolling stock available at that location to cover task j. Therefore, task j must be canceled. We call the case that task j must be canceled due to the fact that another task i is canceled, *cancel dependency*.

The assumption that at every location there is enough space to store the rolling stock, does not match reality. Especially on locations without a shunting yard this assumption is very strong. However, it is difficult to leave this assumption out of the model. For example,

if we cannot find a crew member for a train between Breda (a location without a shunting yard) and Dordrecht and the rolling stock is already in Breda, the rolling stock definitely must end at Breda. In such a case we assume that some rolling stock rescheduling can be done by the dispatchers. For example, the rolling stock composition can be combined with the rolling stock composition of another train.

Moreover, in this chapter we also assume that it is not a problem that the rolling stock ends at another shunting yard than planned. For the re-balancing of rolling stock at the end of a day we refer to Budai et al. (2008) and for a real-time rolling stock rescheduling model we refer to Nielsen (2008).

6.1 Adaptations to the model formulations

If we want to take the cancel dependencies into account in model formulation A of Chapter 4, we have to change and add some constraints. First, we must change (4.6) into (6.1). The set E_e^{link} determines which cancel dependencies must be taken into account. However, the sets E_e^{link} do not take the shunting yards into account. This means that if there is a shunting yard between two tasks and the first task is canceled, still both tasks must be canceled. If only canceling a task which does not end at a location with a shunting yard must lead to the cancellation of the next task of the rolling stock, we have to use the parameter q_i which is 1 if task i ends at a location with a shunting yard and 0 otherwise.

$$y_i q_i + \sum_{f \in E_e^{bef}} v_f - v_d \quad \geq \quad 0 \quad \forall e \in E_i, \forall d \in E_e^{link} \tag{6.1}$$

These constraints (6.1) do not ensure that all cancel dependencies are taken into account. If we have for example a copy d which is not used in any set E_e^{link}, the model allows us to use this copy without cancel dependency. But maybe in reality, this copy d may not be used due to a cancel dependency. To deal with these exceptional cases, we have to add some new constraints. For these constraints we need to define a new set:

- $N_i^{link} \subset N$: the task which the rolling stock composition of task i must perform directly after it has performed task i.

The necessary additional constraints become:

$$y_i - y_j \quad \geq \quad 0 \quad \forall i \in N, \forall j \in N_i^{link} \tag{6.2}$$

Notice that we can reduce the number of Constraints (6.2) by using these constraints only for the exceptional cases for which they are necessary. An example could be to replace $\forall i \in N$ by $\forall i \in \{h \in N | q_h = 0\}$.

6.2 Defining the core problems

To select a core problem in the case that cancel dependencies must be taken into account, the sets $\bar{N}$ and $\bar{\Delta}$ must be defined iteratively. For cancel dependency, $\bar{N}$ must be made in such a way that all linked tasks to tasks in $\bar{N}$ are also in $\bar{N}$. Moreover, $\bar{\Delta}$ must still be equal to all duties which contain a task in $\bar{N}$ and all tasks of the duties in $\bar{\Delta}$ must be in $\bar{N}$. These restrictions can make the sets $\bar{N}$ and $\bar{\Delta}$ very large. Moreover, $\ddot{N}$, the set which contains all tasks which are covered by at least one duty $\delta \notin \bar{\Delta}$, must be empty. Or in other words, $\bar{\bar{N}}$ must be equal to $\bar{N}$.

To define the core problem we start with the same core problem as in case we have no cancel dependencies. Thereafter, we adjust $\bar{N}$ in such a way that it meets the restrictions of this set. Then, a new $\bar{\Delta}$ must be made and possibly $\bar{N}$ must be adjusted again. This iterative process stops the first time that $\bar{N}$ does not have to be adjusted.

6.3 Solution approach

We apply the same overall column generation approach as before. However, some small steps of this approach are changed. We have for example other Lagrangian relaxations and another greedy heuristic.

The Lagrangian relaxation is changed significantly, since we have additional constraints which we relax. For model formulation A, by using the idea of Constraints (6.1) and by relaxing constraints (6.2) with multipliers π_i, the Lagrangian relaxation is equal to (6.3) in which $q_i^* = (1 - q_i)$. For all tasks i for which q_i is 1, π_i is equal to 0.

$$
\begin{aligned}
\Theta(\lambda, \eta, \mu, \pi) \;=\; & \min \sum_{i \in \bar{N}} \lambda_i + \sum_{\delta \in \bar{\Delta}} \sum_{k \in \bar{K}^\delta} (c_k^\delta + \sum_{e \in \bar{E}} \mu_e^\delta b_{ek}^\delta - \sum_{i \in \bar{N}} \lambda_i a_{ik}^\delta) x_k^\delta \\
+ \; & \sum_{i \in \bar{N}} (f_i - \lambda_i - q_i \sum_{e \in \bar{E}_i} \eta_e - q_i^* \pi_i + \sum_{\{j \in \bar{N} | i \in N_j^{link}\}} q_j^* \pi_j) y_i \\
+ \; & \sum_{i \in \bar{N}} \sum_{e \in \bar{E}_i} (g_e + \gamma_e - \sum_{\delta \in \bar{\Delta}} \mu_e^\delta) v_e \\
& \text{s.t. } (5.3), (5.5), (5.7), (5.8) \text{ and } (5.9)
\end{aligned}
\tag{6.3}
$$

With this new Lagrangian subproblem, the approach to determine the x_k^δ variables remains the same as in the approach without cancel dependencies. In the algorithm to determine the variables v_e and y_i, only Line 6 in the algorithm of Figure 5.2 had to be changed. The new algorithm is given in Figure 6.1. It seems that the values of π do not influence the selection of the feasible completions. However, they do not do this directly, but the values of π do

influence the values of λ. So, indirectly, the values of π do influence the selection of the values for the variables x_k^δ.

For all $e \in \bar{E}_i$ determine $\bar{g_e} = (g_e + \gamma_e - \sum_{\delta \in \bar{\Delta}} \mu_e^\delta)$;
Select $e* \in \arg\min \{\bar{g_e} : e \in \bar{E}_i\}$;
if $\bar{g_{e*}} \leq f_i - \lambda_i - q_i \sum_{e \in \bar{E}_i} \eta_e + q_i^* \pi_i - \sum_{\{j \in \bar{N} | i \in N_j^{link}\}} q_j^* \pi_j$ **then**
    Set $y_i = 0$, $v_{e*} = 1$ and for all $e \in \bar{E}_i \setminus \{e*\}$ set $v_e = 0$
else
    Set $y_i = 1$ and for all $e \in \bar{E}_i$ set $v_e = 0$
end

Figure 6.1: Algorithm to determine y_i and v_e

The idea of the subgradient optimization remains the same. Now we have some additional constraints which are relaxed and some additional multipliers. To deal with this we have to introduce new subgradients for the tasks i where $q_i = 0$. These subgradients are: $I_i(g) = \sum_{j \in N_j^{link}} y_j - y_i$. The sum of squares of these subgradients must be added to the *norm*. Moreover, every iteration the multipliers $\pi_i(g)$ are updated with the following formula: $\pi_i(g) = \max\ \{0, \pi_i(g-1) + T \bullet I_i(g-1)\}$.

We determine two ways to take cancel dependencies into account in the greedy algorithm to generate feasible solutions given some multipliers. The first option is to use the same algorithm as the approach without cancel dependencies. Then, after a solution has been generated, we can check if the solution is feasible according to the cancel dependencies. If the solution is not feasible, we will ignore it.

Another option is to use an iterative procedure as shown in Figure 6.2. We must quit the procedure if the algorithm in Figure 5.3 does not return a feasible solution. Then the procedure in Figure 6.2 does not return any solution.

1 Execute the algorithm of Figure 5.3;
2 Define the set of canceled tasks $N^{canc} = \{i \in \bar{N} | y_i = 1\}$;
3 **while** $|N^{canc}| > 0$ **do**
4 Select a task $i \in N^{canc}$; i will from now on be ignored in the set N^{canc} ;
5 Ignore task i and all rolling stock dependent tasks from $\bar{N}$;
6 Ignore all copies corresponding to the removed tasks out of the copy subsets;
7 Ignore all feasible completions which contain a removed copy out of the feasible completion subsets;
8 Execute the algorithm of Figure 5.3 (if the solution is not feasible **STOP**);
9 Define the set of canceled tasks $N^{canc} = \{j \in \bar{N} | y_j = 1\}$.
10 **end**

Figure 6.2: Greedy procedure to construct feasible solutions with cancel dependency

Chapter 7

Experiments with data of Netherlands Railways

In this chapter we run some experiments with data instances of NS. First, we introduce in Section 7.1 the selected parameters and instances. We use instances of NS, which are comparable with the instances used in Potthoff et al. (2008). These instances are not exactly the same as in Potthoff et al. (2008), since some other decisions are made to adapt the timetable and rolling stock schedule to handle the disruptions. In Table 7.1, a summary of the 10 used instances is presented. In Section 7.2 we test the instances on different groups of stand-by crew.

7.1 Parameter settings

The solution approach as discussed in Chapter 5 requires a lot of settings. In this section we first discuss the settings of the approach of Chapter 5 in Section 7.1.1 and the settings of the neighborhood selection are given in Section 7.1.2. Thereafter, we give the costs and rules of the feasible completions in Section 7.1.3 and we give some additional information of the instances in Section 7.1.4.

7.1.1 Settings of the retiming approach

First of all, we have some settings which are required to determine the core problems. In the definition of N_2 we set $d_2 = 60$ minutes and to determine N_6 we set $d_6 = 5$ minutes. For every copy in $\bar{N}^c$ we generate three delay copies of which the delays are 1, 3 and 5 minutes.

Every column generation iteration, we start with adding at most $P = 20$ columns per duty. To speed up the procedure, we use partial pricing in which we stop adding columns if

Location	ID	Time	Type
Abcoude	*Ac C*	11:00-14:00	two sided blockage
	Ac D	16:30-19:30	two sided blockage
Beilen	*Bl C*	07:00-10:00	two sided blockage
	Bl D	16:00-19:00	two sided blockage
's-Hertogenbosch	*Ht C*	08:00-11:00	two sided blockage
	Ht D	15:30-18:30	two sided blockage
Lelystad	*Lls C*	04:00-07:00	two sided blockage
	Lls D	13:00-16:00	two sided blockage
Zoetermeer	*Ztm C*	08:00-11:00	reduced number of trains
	Ztm D	11:30-14:30	reduced number of trains

Table 7.1: Summary of the different instances

we have already found better columns for 30 % of the duties.

After the new columns have been added, we start a subgradient optimization in which $\pi = 1.1$. This subgradient optimization has six stopping criteria of which four require a parameter. The subgradient optimization stops if the gap between the upper and lower bound is smaller than $maxGap = 0.001$. Moreover, the method stops if the step size becomes below a threshold value $minT = 0.01$ or if π becomes smaller than 0.001. The method has a maximum number of iterations $maxG = 2500$.

After the subgradient optimization we use, if they exist, the last $X = 100$ sets of Lagrangian multipliers to generate feasible solutions. The column generation approach stops if the cost of the best feasible solution is less than 1% higher than the lower bound LB_n. Moreover, the column generation process stops after it has reached its maximum number of iterations which equals 5000.

The fixing phase starts if the relative gap between the lower bound (LB_n) and the approximation of Θ_n^* is lower than 0.01%. If the fixing phase has started, we fix in every step at least one column. We fix more columns if those columns are selected in more than 75% (β) of the Lagrangian solutions. However, in every fixing phase step, we are allowed to fix columns for at most 5% of the number of duties in the core problem. Every step of the fixing phase we apply at most 10 column generation iterations.

7.1.2 Neighborhood selection

To determine the neighborhood of an uncovered task, we first need to select 8 tasks which start at the same location as the uncovered task. Of these 8 tasks, 4 start before the uncovered

task and 4 start after the uncovered task. We then make a basic selection which contains all duties which are, by the current solution, assigned to one of these tasks. Thereafter, for every duty in this basic selection, we select at most 4 similar duties. The basic selection together with the similar duties, are put in $\bar{\Delta}$. Notice that $\bar{\Delta}$ also contains the duties which are selected by the retiming possibilities.

7.1.3 Rules and costs of feasible completions

In Section 5.3.4 we have already introduced some types of costs which are taken into account in the feasible completions. First, the cost of changing a duty is equal to 400. If the driver is directly sent home, this cost is not 400 but 3000. The cost of covering a task is 0 if the duty was already covering that task and 50 otherwise. Moreover, the cost of a transfer is 0 if the transfer was already in any original duty and 1 otherwise. If the duty uses a new repositioning task, the fixed cost of using this task is 1,000.

If a driving task is not covered by any duty, the penalty cost is equal to 3,000 if the begin and end location of the task are the same and 20,000 otherwise. An uncovered driving task which has the same begin and end location does not require difficult rolling stock rescheduling. In the rolling stock schedule we can remove the task, without making the schedule infeasible. Therefore, the penalty of a uncovered driving task which has the same begin and end location is much lower than the penalty of an uncovered driving task which has a different begin and end location. If a planned education, training or reserve task is not covered, the penalty cost is only 250. If a certain task is retimed, we add a penalty of 200 per minute.

Next to the costs, there are also some rules which must be taken into account. We have already introduced that a duty must contain a break if it is longer than 5 1/2 hours. The time before and after the break may not exceed 5 1/2 hours. In addition to this, a duty may not end more than an hour later than its original end time. Hereby, we have to take a sign-on time of 10 minutes and a sign-off time of 5 minutes into account. The sign-on time of a stand-by crew member is equal to 0 minutes.

If the crew member has to transfer between two tasks, we take a transfer time to walk into account. It is clear, that if the crew member drives further on the same rolling stock composition, there is no necessary transfer time. However, if he has to transfer to another rolling stock composition, the minimal transfer time is equal to 10 minutes.

7.1.4 Instances

In this chapter we only do some experiments on instances of train drivers. We use the original crew schedule of September 24, 2007 as input for our experiments. This schedule contains

about 9300 driving tasks and 925 train driver duties. For every instance in Table 7.1 we use the same duties, but the tasks can be different. In an instance some of the tasks of the 24th of September 2007 are canceled and some new tasks are added to handle the disruption. Moreover, some instances introduce repositioning possibilities. In that case crew members are allowed to use a new repositioning task between a given set of locations and within a given time interval. The duration of this repositioning task is given as input and if a crew member uses such a new repositioning task, we have a fixed cost of 1000. In reality the use of such a repositioning task between locations A and B means that the crew member uses a taxi to travel from A to B.

If the instances contain two duties which cover the same task, it is not always clear which of the crew members is the driver and which is the passenger. The role of the crew member is not given as an input and therefore we decide that all crew members which are assigned to the task and which are allowed to drive the task, may be used as the driver of that task.

We can run the instances with different sets of stand-by crew. For these subsets we use the four subsets (R_0, R_1, R_2 and R_3) which are used in Potthoff et al. (2008). R_0 is the subset which contains all stand-by duties and R_3 is the empty set of stand-by duties. The other two subsets, R_1 and R_2, are randomly generated subsets.

The tasks also contain information about the next task of the rolling stock. We use this information to determine the sets E_e^{link} for the rolling stock linkages. Between linked copies there must be a certain dwell time *dwell*. For *dwell* we use the originally planned dwell time between the tasks represented by the copies. However, we use a maximum dwell time of 2 minutes. Therefore, if the originally planned dwell time was larger than 2 minutes, we can use the margins to reduce the delays. For the rolling stock dependencies we take the whole route of the rolling stock and we do not break these dependencies up at locations with shunting yards.

7.2 Results

In this section we first determine the best approach to solve the initial core problem. Thereafter, we test our approach to solve core problems for uncovered tasks. For the tests we use a quad core 2.99 GHz CPU processor with 3.25 GB RAM memory. However, the tests are only performed on a single core.

7.2.1 Results for the initial core problem

First, the best option to solve the initial core problem must be determined. We have three options which we will test. The first option is the unchanged method of Potthoff et al. (2008)

in which we allow that the solution could be infeasible for our problem. In the second method, we use the method of Potthoff et al. (2008) but during the approach, we neglect the infeasible solutions generated by the greedy heuristic. The last option is to use our approach of Chapter 5 for the initial core problem. Notice that the last two options always end up with a feasible solution.

If we solve the initial core problem of the 10 instances with R_1 and R_3, the unchanged method of Potthoff et al. (2008) results in 19 feasible solutions and 1 infeasible solution. If we compare the 19 feasible solutions of the unchanged method of Potthoff et al. (2008) with the solutions of the other methods we notice that in most cases the solution of the unchanged method of Potthoff et al. (2008) was the best. Moreover, the total cost of all 19 solutions of the unchanged method of Potthoff et al. (2008) were lower than the total cost of the other solutions. However, the method of Chapter 5 had in most cases the best lower bound on the cost. The gap between the sum of the cost of the 19 solutions and the sum of the lower bounds is equal to 0.5% for both methods based on Potthoff et al. (2008) and equal to 0.8% for the method based on Chapter 5. For a heuristic approach, like the three methods, such small gaps are a good result. The solutions found by the methods are quite close to optimality.

For instance $Ac\ C$ with the set R_3 of stand-by crew, the unchanged method of Potthoff et al. (2008) did not result in a feasible solution since there was one crew member which was a passenger on a train which did not have a driver. The changed method of Potthoff et al. (2008) which ignores infeasible solutions resulted in a very bad solution. The solution which the method found for this instance was the trivial solution in which at least all drivers with an infeasible duty due to the disruption take a taxi to their crew base. This leads to a large amount (233) of uncovered tasks. The method searches in a direction in which someone is a passenger on a train which cannot be driven by any crew member. All solutions generated by the greedy heuristic are infeasible and the method does not correct for this. The method of Chapter 5 comes up with a feasible solution for instance $Ac\ C$ which even has lower cost than the solution of the unchanged method of Potthoff et al. (2008).

The total computation time to solve the 20 initial core problems was 974 seconds if we use the unchanged method of Potthoff et al. (2008) and 966 seconds if we use the method of Potthoff et al. (2008) which neglects infeasible solutions. The method of Chapter 5 required a lot more computation time. It needed 3449 seconds to solve the 20 initial core problems with this method. The unchanged method of Potthoff et al. (2008) was for 5 initial core problems the fastest method and the changed method of Potthoff et al. (2008) was for 4 initial core problems the fastest method. For the other 11 initial core problems the changed and unchanged method of Potthoff et al. (2008) were equally fast. The method of Chapter 5 was for all cases the slowest method.

All three methods have a disadvantage. The disadvantage of the unchanged method of Potthoff et al. (2008) is that it not certainly results in a feasible solution. On the other hand the changed method of Potthoff et al. (2008) always result in a feasible solution, but sometimes it comes up with a very bad trivial solution. The method of Chapter 5 always results in a good feasible solution, but has larger computation times.

Evaluating the disadvantages, we select the unchanged method of Potthoff et al. (2008) as the best method to solve the initial core problem. The method of Chapter 5 is too slow to use for real time instances and therefore it was the first method which was rejected. We had to make a selection between the changed and unchanged method of Potthoff et al. (2008), which were both equally fast. We made our decision based on the results of the instance *Ac C*. For this instance the result of the changed method of Potthoff et al. (2008) would be a bad start schedule for the following iterations but the infeasible result of the unchanged method can be used as a reasonable start for further iterations. Notice that in the further iterations the infeasible solution of the initial core problem will be made feasible. After the last core problem is solved, we know that the solution must be feasible, since after the first iteration it is not allowed anymore to be a passenger on a train which has no driver.

7.2.2 Results for the total problem

In this section we test the total approach of Chapter 5. The interesting cases are the instances in which it was not possible to cover all tasks by solving the initial core problem. These instances are *Ac C* (R_3), *Ac D* (R_2) and *Ht D* (R_0, R_1, R_2, R_3). It is surprising that case *Ac D* has an uncovered task if we use R_2 and no uncovered tasks if we use R_3. Since the set R_2 is a subset of the stand-by duties and R_3 is the empty set of stand-by duties, the optimal solution of the instance by using R_2 must be better than the optimal solution by using R_3. The reason why we were not able to cover all tasks with R_2 is that the method has stopped with a gap of 10% between the lower and upper bound. The lower bound of R_2 was lower than the lower bound of R_3, but R_3 was able to find a better upper bound before the method stopped.

We have determined two ways to select a core problem to deal with an uncovered task. The general solution approach is to solve a core problem for every uncovered task iteratively after each other. This seems to be a good idea for the first option, which uses for a core problem a small neighborhood selection and some retiming options. In the remainder of this chapter, we call this *Option 1*. For the second option (*Option 2* in the remainder of this chapter) we use one overall problem, which contains at least the same duties as the initial core problem and which contains retiming options around all uncovered tasks. We compare our results with the method of Potthoff et al. (2008) which does not allow retiming, but we

ID	It	Fixed	$\lvert\bar{\Delta}\rvert$	$\lvert\bar{\bar{N}}\rvert$	UB	Gap (%)	Sol	TT (s)	A-B	A-A	PU	D	Min
Potthoff	1	0	130	629	64277	5.2	64277	92	1	1	1	0	0
et al. (2008)	2	33645	58	257	29691	0.0	63336	101	1	0	1	0	0
Option 1	2	31086	66	355	15502	14.0	46588	118	0	1	0	1	1
	3	37059	50	346	7031	0.0	44090	138	0	0	0	1	1
Option 2	2	0	144	781	46234	10.3	46234	676	0	1	0	1	1

Table 7.2: Results of instance *Ac C* by using R_3

must take into account that the solutions of Potthoff et al. (2008) could be infeasible.

In the tables with the results we use some abbreviations: "It" is the iteration number, of the algorithm of Figure 5.1. The cost in the column "Fixed" are the cost of the duties which are not selected in the current core problem and the cost in the column "UB" are the cost of the best solution of the current core problem. The column "UB" only contains cost of duties which are selected in the current core problem. The "Gap" represents the relative difference between the best solution and the lower bound of the core problem. The column "Sol" represents the cost of the total solution: the cost of the core problem ("UB") plus the cost of the duties which were not selected by the core problem ("Fixed"). The total computation time in seconds up to the current iteration is given in the column "TT". The columns "A-B" and "A-A" represent the number of uncovered tasks for respectively tasks which have a different start and end location and tasks which have the same start and end location. The column "PU" indicates how many times a driver is a passenger on a train which has no driver. A 0 in the column "PU" indicates that the solution is feasible. The last two columns give information about the used retimed copies. The column "D" contains the number of retimed tasks and the column "Min" represents the total number of retimed minutes.

Ac C

The first instance that we discuss is *Ac C*. If we use R_0, R_1 or R_2, it is possible to cover all tasks by solving the initial core problem. But if we use R_3, it is not possible to cover all tasks with the initial core problem. The results of using R_3 are presented in Figure 7.2. After the initial core problem is solved, we have still 2 uncovered tasks. The first of these uncovered tasks, is a task to drive train 839 between Amsterdam and Utrecht starting at 11:07 A.M..

If a task starts *at* the time of rescheduling, it is only allowed to assign this task to a stand-by crew member or to the crew member to which the task was originally assigned. A task may be assigned to other original duties than planned if its start time is *after* the time of rescheduling. The time of rescheduling was in this instance 11:07 and therefore only stand-by

crew or the driver who was originally assigned to this task, may drive the task.

In this case, the task is rerouted due to the disruption and takes half an hour longer. The crew member which was originally assigned to this task does not have the knowledge of the new route and is therefore not allowed to drive this train. We also have, with R_3, no stand-by crew available. As a conclusion, no driver is allowed to drive the train of this task. However, if the task is delayed by at least one minute (as done by Options 1 and 2) the start time of the task is after the time of rescheduling. Then, it is possible to assign other crew members than the original driver, to this task.

In the solutions of the second core problem of Options 1 and 2, we notice that the task of train 839 between Amsterdam and Utrecht is delayed by 1 minute. By delaying this task with 1 minute, we were able to cover this task. The method of Potthoff et al. (2008) was, as expected, not able to cover train 839 between Amsterdam and Utrecht. However, it was in the second iteration able to cover the other uncovered task. Option 1 was also able to cover that task without using retiming, but that required a third iteration. Option 2 was not able to cover the second uncovered task.

Option 1 has the best solution in terms of costs but, in comparison with the other methods, it needed an extra iteration to reach this solution. The second best solution is the solution of Option 2. Option 2 and the method of Potthoff et al. (2008) both have one uncovered task. The difference is that the uncovered task of Option 2 is a task which has the same begin and end location and the uncovered task of the method of Potthoff et al. (2008) is a task which has a different begin and end location. Therefore, the solution of Option 2 is much better.

In the table we notice some relatively large gaps. The difference between the lower bound and upper bound of the initial core problem is 5.2%. In absolute terms this gap is larger than the cost of 1 uncovered task of type A-A. This means that it could be that the optimal solution for the initial core problem has only one uncovered task. The relative gap between the upper and lower bound of the second core problem is more than 10 % for Option 1 and Option 2. For Option 1 the absolute gap is still reasonable, but for Option 2 the absolute gap is again larger than the cost of 1 uncovered task of type A-A. So, the optimal solution of Option 2 could be a solution without any uncovered task.

In the second iteration, we notice that Option 1 uses next to the 58 duties of the neighborhood selection of the method of Potthoff et al. (2008), 8 more duties to deal with the retiming possibilities. Moreover, as required by the definition, Option 1 takes a larger subset of tasks into account than the neighborhood selection of Potthoff et al. (2008). In the second iteration Option 2 uses the same subsets as the first iteration. However, 14 additional duties and 152 additional tasks are necessary for the retiming possibilities.

Due to the large subsets of Option 2, the computation time of that method is very large.

The computation time of Option 1 is larger than the computation time of the method of Potthoff et al. (2008), but that is expected since we have larger subsets and retiming possibilities. However, the additional computation time of Option 1 is only 37 seconds.

The solution of Option 1 contains in comparison with the solution of Potthoff et al. (2008), 2 more modified duties and 2 more duties which end later than planned. The solution of Option 2 contains 1 modified duty less than the solution of Potthoff et al. (2008) but contains 3 more duties which end later than planned. The crew schedules of Option 1 and the method of Potthoff et al. (2008) both contain 1 new repositioning task and the crew schedule of Option 2 contains 2 new repositioning tasks. These differences in the solutions of the methods are not very large. A remark must be made that the solution of the method of Potthoff et al. (2008) is infeasible since it has one crew member which is a passenger on a train without a driver.

Ht D

The instance *Ht D* is comparable with the instance *Ht B* as discussed in Section 2.3. For all subsets of stand-by duties, it was not possible with the initial core problem to cover the task of train 854 between 's Hertogenbosch and Utrecht. In the further iterations, by using R_0, R_1 or R_2, both methods of Chapter 5 are able to cover this task by delaying train 4456 by 3 minutes. Option 2 is also able to cover this task by using R_3. Without using retiming we were never able to cover this task.

In Table 7.3a we present the results of the instance *Ht D* by using R_1. In case we use R_0 or R_1 we have two uncovered tasks after solving the initial core problem. As described before, one of these tasks is the task to drive train 854 between 's Hertogenbosch and Utrecht. For R_0 and R_1, Options 1 and 2 were able to cover this task by delaying train 4456 by 3 minutes. Without retiming, the method of Potthoff et al. (2008) was not able to cover this task. None of the methods was able to cover the other uncovered task. In total the solution of Option 2 has the lowest cost. This solution differed at most 0.2 % with the optimal solution. The solutions of the core problems of the other methods are optimal.

The solutions of Options 1 and 2 have lower costs than the solution of the method of Potthoff et al. (2008) since they contain less canceled tasks, however to achieve this, Options 1 and 2 use next to the retimed task, 1 more stand-by crew member and modify 2 more duties than the method of Potthoff et al. (2008). Again, Option 2 has an enourmous computation time, which could be explained by the fact that the method uses subsets of duties and tasks which are almost twice as large as the subsets used in the core problems of Option 1 and the method of Potthoff et al. (2008).

If we use R_2 we have next to the uncovered task of train 854 between 's Hertogenbosch

ID	It	Fixed	$\lvert\bar{\Delta}\rvert$	$\lvert\bar{\bar{N}}\rvert$	UB	Gap (%)	Sol	TT (s)	A-B	A-A	PU	D	Min
Potthoff	1	0	126	660	61645	0.1	61645	90	1	1	0	0	0
et al.	2	32502	72	389	29143	0.0	61645	98	1	1	0	0	0
(2008)	3	30453	78	411	31192	0.0	61645	117	1	1	0	0	0
Option 1	2	29995	75	448	31650	0.0	61645	113	1	1	0	0	0
	3	29949	85	516	14300	0.0	44249	168	0	1	0	1	3
Option 2	2	0	132	780	42799	0.2	42799	376	0	1	0	1	3

(a) Results using R_1

ID	It	Fixed	$\lvert\bar{\Delta}\rvert$	$\lvert\bar{\bar{N}}\rvert$	UB	Gap (%)	Sol	TT (s)	A-B	A-A	PU	D	Min
Potthoff	1	0	104	660	65250	0.4	65250	114	1	2	0	0	0
et al.	2	32055	51	374	30649	0.0	62704	124	1	1	0	0	0
(2008)	3	31512	56	425	31192	0.0	62704	140	1	1	0	0	0
Option 1	2	31758	52	435	32151	0.0	62704	135	1	1	0	0	0
	3	30204	63	530	15104	0.0	45308	172	0	1	0	1	3
Option 2	2	0	111	789	45656	0.3	45656	467	0	1	0	3	12

(b) Results using R_2

ID	It	Fixed	$\lvert\bar{\Delta}\rvert$	$\lvert\bar{\bar{N}}\rvert$	UB	Gap (%)	Sol	TT (s)	A-B	A-A	PU	D	Min
Potthoff	1	0	90	660	65548	0.1	65548	90	1	2	0	0	0
et al.	2	33863	39	403	29539	0.0	63402	100	1	1	0	0	0
(2008)	3	32866	43	387	30536	0.0	63402	110	1	1	0	0	0
Option 1	2	31758	42	472	31644	3.1	63402	119	1	1	0	0	0
	3	32413	50	491	30989	0.0	63402	150	1	1	0	0	0
Option 2	2	0	97	789	46707	0.1	46707	370	0	2	0	1	3

(c) Results using R_3

Table 7.3: Results of instance *Ht D*

and Utrecht, 2 additional uncovered tasks. The results of R_2 are given in Table 7.3b. Both methods of Chapter 5 end up with a solution with one uncovered task. In both solutions train 4456 is delayed by 3 minutes to cover the task of train 854 between 's Hertogenbosch and Utrecht. In the second iteration the method of Potthoff et al. (2008) and Option 1 are able to cover one of the uncovered tasks of type A-A (tasks which have the same begin and end location). Since Option 2 does not use neighborhood selection, it was not able to find directly a driver which could cover one of the two uncovered tasks of type A-A. However, by using retiming, Option 2 was able to cover one of the two uncovered tasks of type A-A. Option 2 delays a task from Breda to Breda with 5 minutes. Since this task is by rolling stock linked with one minute margin to a task from Breda to Roosendaal, the task from Breda to Roosendaal will be delayed with 4 minutes. Option 1 reached in the second iteration a solution of the same cost as the solution which the method of Potthoff et al. (2008) obtained after the second iteration. However, the method of Potthoff et al. (2008) was not able to come up with a better solution in the next iteration.

Every method resulted in two uncovered tasks if we use R_3. The results of this instance are given in Table 7.3c. Only Option 2 was able to cover the task of train 854 between 's Hertogenbosch and Utrecht by delaying train 4456 by 3 minutes. In the end, Option 2 has a much better solution since it has only uncovered tasks which have the same begin and end location. Although Option 1 takes in the core problem more duties and tasks into account than the method of Potthoff et al. (2008) does, it was not able to come up with a better solution than the method of Potthoff et al. (2008).

The set R_2 contains less stand-by crew than R_1, therefore the subsets of duties in the core problems by using R_2 are in general smaller than the subset of duties in the core problems by using R_1. In the same reasoning, the subset of duties in the core problems by using R_3 must be smaller than the subsets of duties in the core problems by using R_1 or R_2. In Table 7.3 we notice that this is true for the instance *Ht D*. However, if a method uses a smaller subset of duties, it is not guaranteed to get a lower computation time. Indeed the number of pricing problems to be solved are less, but it also means that it could be more difficult to solve the overall problem. For instance *Ht D*, we have the smallest computation times by using R_3, but thereafter we notice that we can solve the instance faster with R_1 than with R_2.

Obviously, in terms of cost, using R_1 results in the lowest cost for all methods and thereafter R_2 results in the second best solutions. If we have more stand-by crew available, it is reasonable that we can generate better crew schedules.

For every subset of reserve crew, Option 2 results in low cost, but the computation time of this method is a lot larger than the computation time of the other methods. Option 1 has two times a much better solution than the method of Potthoff et al. (2008) and once an equal

solution. To reach these better solutions, the computation time was at maximum 51 seconds larger than the method of Potthoff et al. (2008).

Ztm C

With the current crew rescheduling rules, we were able to cover all tasks of instance *Ztm C*, by solving the initial core problem. To see how the method works under some different circumstances we have changed the rules a bit. With these new rules a duty may not end 60 minutes later than planned but only 15 minutes. In Table 7.4, the results of instance *Ztm C* with the new rules are given. R_0 is the used subset of stand-by crew. After solving the initial core problem, the solution has one uncovered task. Within the next core problem, Option 1 was able to cover this uncovered task by delaying one task by 5 minutes. Option 2 and the method of Potthoff et al. (2008) were not able to cover this task in the next iteration. The set R_0 contains 84 stand-by crew members which makes the subset of duties very large. However, the computation times are much lower in comparison with the instances *Ac C* and *Ht D*. Option 1 needed only 18 more seconds than the method of Potthoff et al. (2008). Again Option 2 has the largest computation time.

ID	It	Fixed	$\lvert\bar{\Delta}\rvert$	$\lvert\bar{\bar{N}}\rvert$	UB	Gap (%)	Sol	TT (s)	A-B	A-A	PU	D	Min
Potthoff	1	0	153	502	33394	0.5	33394	37	1	0	0	0	0
et al. (2008)	2	9943	127	303	23451	0.2	33394	41	1	0	0	0	0
Option 1	2	9943	127	397	5505	0.0	15448	59	0	0	0	1	5
Option 2	2	0	158	624	33341	0.6	33341	161	1	0	0	0	0

Table 7.4: Results of instance *Ztm C* by using R_0

Summary of the results

We have noticed in the 6 cases that using retiming mostly results in less uncovered tasks. Option 1 comes in 5 of the 6 cases with a solution with less uncovered tasks than the method without retiming and Option 2 has 3 times a solution with less uncovered tasks than the method without retiming. For the other 3 cases, Option 2 has two times a much better solution since it has more uncovered tasks of type A-A in exchange of less canceled tasks of the type A-B (tasks which have a different begin and end location).

However, using retiming increases the computation times. The maximum difference between the computation times of Option 1 and the method of Potthoff et al. (2008) is 51 seconds. Especially, Option 2 has a large computation time. For instance *Ac C*, it needed 575 seconds more than the method of Potthoff et al. (2008) to solve the case. The computation times of Option 1 and the method of Potthoff et al. (2008) were in all 6 cases less than 3

minutes, which is reasonable if the methods must be used in practice. However, the computation times of Option 2, which can be more than 10 minutes, are too large. This means that Option 2 cannot be used in practice.

Chapter 8

Conclusions and recommendations

In this chapter we first briefly summarize the research and draw some conclusions. Thereafter, we have some recommendations for further research.

During a day, railway operators have to deal with several disruptions in the railway network and with irregularities in the running times of the trains. These disruptions can make the timetable, rolling stock schedule and crew schedule infeasible. In this thesis we have constructed a column generation approach to generate a new crew schedule if the crew schedule was infeasible due to a disruption. This column generation approach is also allowed to slightly change the timetable, which is called retiming. The retiming part is especially used to ensure that less trains are canceled.

In the approach described in this thesis, only a few trains may be retimed. The approach is not able to select the optimal departure time of a train continuously. Therefore, some allowed departure times of the train must be selected in advance.

The approach starts with searching for a new crew schedule without using retiming. For this initial iteration we decided to use the crew rescheduling approach of Potthoff et al. (2008) although this method could come up with infeasible crew schedules. A crew schedule is infeasible if we have at least one crew member which is a passenger on a train without a driver. However, in further iterations of the approach the schedule will be made feasible with certainty.

To generate the initial crew schedule, we only use a subset of the duties and the tasks of the railway operator. If we take all duties and tasks into account the problem becomes too large.

After we have generated an initial new crew schedule, we check if this new crew schedule has still some uncovered tasks. If so, we search iteratively, by allowing retiming, for a better schedule. For every uncovered task we create a subproblem in which we try to cover the uncovered task. We have made two approaches to select a subset of duties and tasks which

will be used for constructing the new schedule. The first approach, Option 1, uses the neighborhood selection of Potthoff et al. (2008) and the second approach, Option 2, uses the same duties as we have used to generate the initial new schedule. In general, Option 2 uses larger subsets of duties and tasks than Option 1.

We have tested the approaches on data instances of train drivers of NS. The initial iteration of the approach of Potthoff et al. (2008) rarely results in a crew schedule with uncovered tasks. We had only 5 cases in which the crew schedule contained uncovered tasks after the initial iteration. To construct an additional case we have changed our selected crew rescheduling rules a bit. We have tested our two approaches on the 6 cases and compared the results with the method of Potthoff et al. (2008) which does not use retiming.

In 5 of the 6 cases, Option 1 results in less uncovered tasks than the method of Potthoff et al. (2008). To come to such a result, Option 1 had to delay only one train with 1, 3 or 5 minutes. It may be clear that allowing retiming increases the computation time. In the worst case, Option 1 was 51 seconds slower than the method of Potthoff et al. (2008).

On the other hand, Option 2 has much larger computation times than the method of Potthoff et al. (2008). In the worst case, Option 2 needed 575 more seconds than the method of Potthoff et al. (2008) to solve the case. Option 2 has in 5 of the 6 cases a much better crew schedule than the method of Potthoff et al. (2008), but in only 3 cases Option 2 had less canceled tasks than the method of Potthoff et al. (2008).

We can conclude that using retiming leads to less canceled trains. Especially, our approach of Option 1 can be useful in practice since it has not that large computation times. In the crew schedules of Option 1, only one task is delayed. However, in practice this does not mean that the train corresponding to this task really has to be delayed. To construct the new crew schedule, we had for example strict rules for the transfer time between two tasks. The transfer time between two tasks with different rolling stock must, according to the rules, be at least 10 minutes. It could be that a task is only delayed to ensure that the transfer of the driver is at least 10 minutes. Imagine, for example, that we have an end time of a task t_1 and a start time of a task t_2 which differ 9 minutes. The approach has suggested that one crew member must perform both task t_1 and t_2 and therefore task t_2 is delayed by 1 minute. However, if in reality, the crew member walks a bit faster to the second train, he could perform task t_2 on the original planned time. So, a scheduled delay of a train does not have to mean that the train will be delayed in practice. The scheduled delay can be seen as an upper bound on the delay which will be reached by performing the crew schedule in reality.

We suggest that Option 1 can be used in practice as a decision supporting tool for real-time cases. Moreover, with further research the approach can become even better. First of all the computation time could be reduced by using multiple computers or multiple cores of

a computer. For example, the pricing problems can be solved independently of each other. We also can generate the feasible solutions out of the Lagrangian multipliers independently of each other.

Not only the computation time is a way in which the method can be improved, but the method can also be made more in line with reality. In the current approach, we assume for example that we can delay a train without causing delays for other trains on the same track. This certainly does not match with reality. We suggest that further research must be done, in which we take next to rolling stock dependencies also dependencies between trains on the same track into account.

Moreover, to take the rolling stock dependencies more into account, we have to investigate a method with cancel dependency. In Chapter 6 a start of such an approach is made.

In the current approach we assume that the computation time is negligible. However, the computation times are currently between 1 and 3 minutes. To take this into account we could set the time of rescheduling to a later point in time and assume that till that time every crew member performs his original duty. However, in reality it is not that easy, since some people can directly from the start of the disruption not perform the original plan anymore. Moreover, we must take time into account to communicate the new crew schedule to the drivers.

Another assumption we use is that the duration of the disruption is fixed and known. In reality it is very difficult to estimate the duration of a disruption. Further research must be done on how we must deal with the uncertainty in the duration of the disruption. How can we construct a method which results in a robust crew schedule?

We have developed an approach which can also be used for cases in which the crew schedule is infeasible due to a transfer which causes a delay. However, in such a case it could be useful to construct the subset of duties and tasks in another, more clever way. More research must be done, in which the approach will be tested and harmonized for instances with transfers which cause delays.

In the current approach, we have focused on the rescheduling of train drivers. For the conductors there are some slightly other rules and preferences. In another research it could be worth full to adapt the approach to these rules and preferences and test the approach on instances of conductors.

Next to the fact that the current approach minimizes the number of canceled trains, the approach could also be used for reducing the number of new repositioning tasks used by the crew. However, this means that it must be allowed to delay trains to reduce the number of new repositioning tasks. Moreover, we then need another method to select the delay possibilities.

Bibliography

K. F. Abdelghany, A. F. Abdelghany, and G. Ekollu. An integrated decision support tool for airline schedule recovery during irregular operations. *European Journal of Operational Research*, 185(2):825–848, 2008.

J. E. Beasley. Lagrangian relaxation. In C. R. Reeves, editor, *Modern heuristic techniques for combinatorial problems*, pages 243–303. John Wiley & Sons, Inc., New York, 1993.

J. E. Beasley and N. Christofides. An algorithm for the resource constrained shortest path problem. *Networks*, 19(4):379–394, 1989.

J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.

G. Budai, G. Maróti, R. Dekker, D. Huisman, and L. G. Kroon. Rescheduling in railways: the rolling stock balancing problem. Technical Report ARRIVAL-TR-0120, Algorithms for Robust and online Railway optimization: Improving the Validity and reliAbility of Large scale systems (ARRIVAL), 2008.

P. M. Camerini, L. Fratta, and F. Maffioli. On improving relaxation methods by modified gradient techniques. In M. Balinski and P. Wolfe, editors, *Nondifferentiable Optimization*, number 3 in Mathematical Programming Studies, pages 26–34. Springer Berlin Heidelberg, 1975.

N. Eggenberg, M. Salani, and M. Bierlaire. Constraint Specific Recovery Networks for Solving Airline Recovery Problems. Technical Report TRANSP-OR 081001, Transport and Mobility Laboratory, Ecole Polytechnique Fédérale de Lausanne, 2008.

K. Holmberg and D. Yuan. A Lagrangian Heuristic Based Branch-and-Bound Approach for the Capacitated Network Design Problem. *Operations Research*, 48:461–481, 2000.

D. Huisman. A column generation approach for the rail crew re-scheduling problem. *European Journal of Operational Research*, 180(1):163–173, 2007.

S. Irnich and G. Desaulniers. Shortest Path Problems with Resource Constraints. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, pages 33–65. Springer, New York, 2005.

D. Klabjan, E. L. Johnson, G. L. Nemhauser, E. Gelman, and S. Ramaswamy. Airline crew scheduling with time windows and plane-count constraints. *Transportation Science*, 36(3): 337–348, 2002.

L. G. Kroon, D. Huisman, E. Abbink, P.-J. Fioole, M. Fischetti, G. Maróti, L. Schrijver, A. Steenbeek, and R. Ybema. The New Dutch Timetable: The OR Revolution. *Interfaces*, 39(1):6–17, 2009.

L. Lettovský, E. L. Johnson, and G. L. Nemhauser. Airline Crew Recovery. *Transportation Science*, 34(4):337–348, 2000.

A. Mercier and F. Soumis. An integrated aircraft routing, crew scheduling and flight retiming model. *Computers & Operations Research*, 34(8):2251–2265, 2007.

L. K. Nielsen. A Decision Support Framework for Rolling Stock Rescheduling. Technical Report ARRIVAL-TR-0158, Algorithms for Robust and online Railway optimization: Improving the Validity and reliAbility of Large scale systems (ARRIVAL), 2008.

R. Nissen and K. Haase. Duty-period-based network model for crew rescheduling in European airlines. *Journal of Scheduling*, 9:255–278, 2006.

D. Potthoff, D. Huisman, and G. Desaulniers. Column generation with dynamic duty selection for railway crew rescheduling. Technical Report EI 2008-28, Econometric Institute, 2008.

P. Ramaekers, T. de Wit, and M. Pouwels. Hoe druk is het nu werkelijk op het nederlandse spoor? (Dutch). Technical report, Statistics Netherlands (CBS), 2009. URL `www.cbs.nl/en-GB/menu/themas/verkeer-vervoer/publicaties/artikelen/archief/2009/2009/2027-wm.htm(English)`.

C. G. Walker, J. N. Snowdon, and D. M. Ryan. Simultaneous disruption recovery of a train timetable and crew roster in real time. *Computers & Operations Research*, 32:2077–2094, 2005.

S. Yan and C. G. Lin. Airline scheduling for the temporary closure of airports. *Transportation Science*, 31(1):72–82, 1997.

S. Yan and Y.-P. Tu. Multifleet routing and multistop flight scheduling for schedule perturbation. *European Journal of Operational Research*, 103(1):155–169, 1997.

S. Yan and D. H. Yang. A decision support framework for handling schedule perturbation. *Transportation Research Part B: Methodological*, 30(6):405–419, 1996.

S. Yan and H.-F. Young. A decision support framework for multi-fleet routing and multi-stop flight scheduling. *Transportation Research Part A: Policy and Practice*, 30(5):379–398, 1996.