Erasmus University Rotterdam

Master Thesis

Crew Rescheduling in the Tram Network of Rotterdam

Timo van Dockum *376457*

Supervisor
dr. Twan Dollevoet

Second accessor
dr. Remy Spliet

Supervisors RET
ir. Richard Both
drs. Cees Boogaard
dr. Judith Mulder

MSc Econometrics and Management Science
Operations Research and Quantitative Logistics
Erasmus School of Economics

13th December 2018

Abstract

In this thesis, we consider the crew rescheduling problem for the tram network of Rotterdam. In recent years, interlining has been introduced into the crew schedule. Interlining means that drivers operate multiple vehicles and lines during their duty. Consequently, crew rescheduling has become more complicated. Therefore, we investigate whether an algorithm could assist in this task.

In the last two decades, crew rescheduling methods have been developed and applied in the rail industry. The contribution of this thesis is that we apply such a method to a regional tram network.

The problem is formulated as a set covering problem. In this problem, a column corresponds to a driver duty which covers pieces of work. The problem is solved with column generation and therefore split in a restricted master problem and a subproblem. The subproblem is modelled as a resource constrained shortest path problem.

The results show that the algorithm is able to repair a disrupted crew schedule within minutes. An experiment with a realistic case shows that the crew rescheduling method allows for new vehicle rescheduling actions. Those actions disturb the crew schedule but are preferable from the passenger perspective. The vehicle rescheduling problem for a tram network is left for further research.

Keywords: rail crew rescheduling, set covering problem, column generation, resource constrained shortest path problem

Contents

Chapter 1

Introduction

The RET is the public transport operator in Rotterdam. They are responsible for both infrastructure and operations of bus, tram, metro and ferry. We will focus on operations of the tram. In this first chapter, we give a general introduction of the tram network and insight into the scheduling and rescheduling practices at the RET.

In recent years, the RET has introduced interlining into its crew schedules for the tram. Interlining means that drivers operate on multiple vehicles and lines during their duties. Interlining has complicated recovering operations in case of a disruption. Therefore, we develop an algorithm which could assist the central dispatchers in this task.

Rezanova (2009) has developed a method for recovering train driver duties. The contribution of this thesis is that we test this method, which has been developed for a rail network, on real-world data of a tram network.

1.1 Tram network

The tram network consists of nine daily lines. All lines, except for one, pass the central station. All lines usually operate between two endpoints and stop at all stops. The longest trip, between west and east, takes about one hour. The network is divided into a northern and southern part by the river. The Erasmus bridge is the only connection between these parts. Both the northern and southern part have a depot, where the trams are parked during the night. A map of the line network is shown in Figure 1.1.

Two types of trams operate on the network, of which one is the newer edition of the other. From an operational perspective, the two types can be considered equal. The RET possesses around a hundred trams, of which about ninety percent is in daily use. The remaining ten percent is being maintained or available as backup vehicle.

The trams have only one drivers cabin and can therefore only move in forward direction. Only in exceptional cases, a tram may drive backwards at a very low speed. By design of the rail network, trams cannot drive wrong-way and therefore a head-on collision is physically impossible. The usual way to get a tram in opposite direction is by turning the tram at one of the loops. In the city centre, the network is

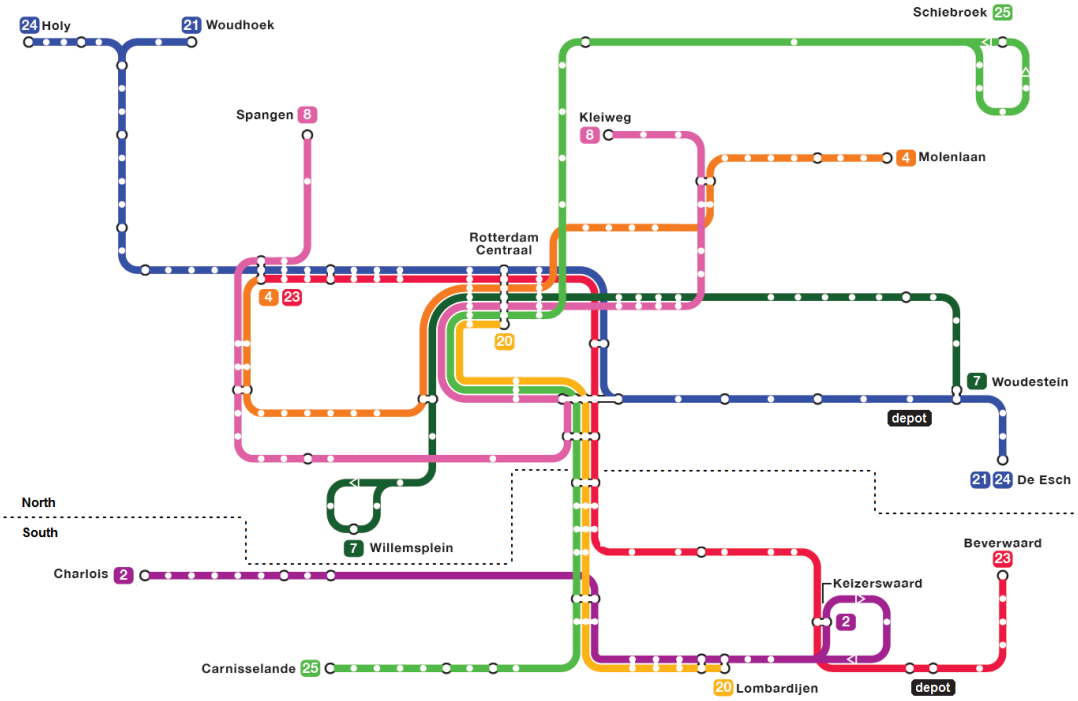

FIGURE 1.1: Tram network map (2018).

dense and relatively many loops exist. At the endpoints, there are small loops which are built for turning.

In contrast to railway networks, the tram network is not divided into safety sections. Drivers do not have to wait until other trams have left the track. They could approach other trams as closely as they want.

1.2 Scheduling approach

The scheduling is divided into four stages: (i) route network planning, (ii) vehicle scheduling, (iii) crew scheduling and (iv) crew rostering. The output of each stage serves as input for the next stage. The stages are not completely separated though: a schedule in a previous stage may be adapted if it leads to more attractive results in a later stage. We will describe each stage in this section.

The RET uses the HASTUS system for scheduling vehicles and crew. Some of the explanations in this section originate from Blais et al. (1990) about the HASTUS system.

1.2.1 Route network planning

The first step is to decide on the routes and the frequency at which the trams should operate. This is based on requirements of authorities, passenger demand and the budget available. The interval between two departures is usually 7.5 or 10 minutes in peak hours and 15 or 20 minutes in off-peak hours. The tram network map from

Figure 1.1 is a result of the route network planning. Line 20 is only operated during peak hours.

1.2.2 Vehicle scheduling

In the second stage, trips are assigned to vehicles by building vehicle blocks. A trip is the movement of a vehicle from endpoint to endpoint. The structure of a vehicle block is shown in Figure 1.2. A block starts and ends in the depot. From the depot, the vehicle travels off service to the closest stop of the line to be served. From here, the vehicle repeatedly travels between the endpoints. The vehicle block finishes with a deadhead trip from the closest stop to the depot.

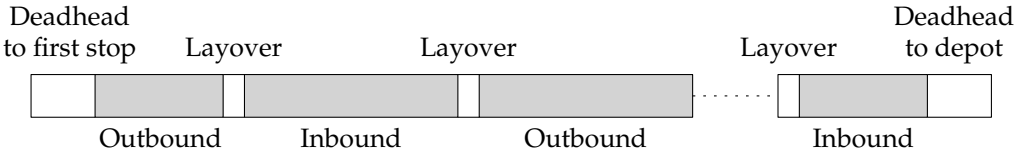

FIGURE 1.2: Structure of a vehicle block with trips in gray.

At the endpoints, a layover time is scheduled in between trips. A fixed number of minutes is scheduled for turning the tram and catching up with delays of previous trips. Furthermore, a tram may wait up to a few minutes until the departure of the next trip. This waiting time is a result of a clock-face schedule. For example, suppose that a return trip takes 73 minutes and the interval between two trams is set at 10 minutes. Suppose that 5 minutes are scheduled for catching up with delays. In this case, the tram waits another 2 minutes since 73 + 5 + 2 = 80 is divisible by 10.

To meet the desired frequency from the network planning, multiple vehicle blocks are created to serve a line. Each block has to be covered by a vehicle. During a block the vehicle is almost continuously operating. Most vehicle blocks span a whole day, but shorter blocks exists for extra capacity during peak hours.

1.2.3 Crew scheduling

The third stage is crew scheduling. The vehicle blocks cannot be handled by one driver, and therefore they are split into pieces of work, as shown in Figure 1.3. The idea is to assign these pieces of work to drivers, such that the vehicle has a driver at every moment in time.

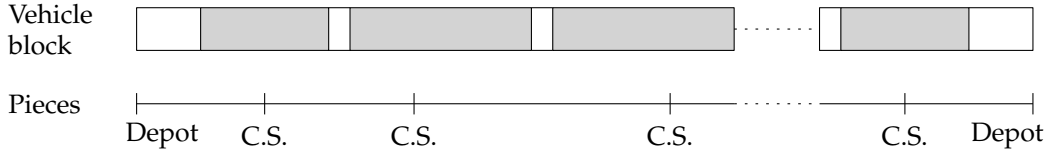

FIGURE 1.3: A vehicle block is split into pieces of work.

The vehicle block is split at relief points: a location where trams can be taken over and where drivers can have a break or are allowed to start or end their duty. The central station and the two depots are the most used relief points.

An algorithm combines pieces of work into driver duties. The driver duties should meet the labour restrictions. For example, there is a maximum duty length and each duty should contain two meal breaks. The most important objective of the algorithm is to minimise the number of duties and working hours such that all pieces are covered.

In the past, a duty consisted of pieces on the same line only. Since a few years, pieces on different lines may be combined, which we call interlining. The introduction of interlining has given the algorithm more flexibility in creating driver duties with less slack. Slack is the time in excess of the required break time.

Table 1.1 shows an example of a driver duty. The driver carries out subsequent pieces on the same vehicle, which are not specified in the table. The driver switches vehicle three times. At 8:39 and 11:31, the driver has a meal break.

TABLE 1.1: Example of a driver duty.

Vehicle	Line	Start		End	
1	4	Depot Kralingen	6:34	Rotterdam Centraal	8:39
83	24	Rotterdam Centraal	9:03	Rotterdam Centraal	10:29
1	4	Rotterdam Centraal	10:39	Rotterdam Centraal	11:31
46	8	Rotterdam Centraal	11:54	Rotterdam Centraal	13:39

1.2.4 Crew rostering

In the last step, rosters are created. A roster is a set of duties which can be assigned to a person. Labour rules also apply to rosters. Examples of such rules are that there should be sufficient rest time between two duties and each week should contain a free period. Rosters are assigned to persons a few weeks ahead.

Once the rosters are assigned, planners manually adapt the rosters to include vacations, days off, exchanges between drivers or the abscence of a driver. These changes occur until the day of operation.

1.3 Disruption management

In the case of a disruption, such as a delay or obstruction, operations run differently from scheduled and eventually rescheduling is needed. At the RET, the central dispatchers are responsible for the rescheduling. They are mostly focused on rescheduling the vehicles. They are assisted by an advanced system which shows them the vehicles in real-time and which allows them to activate disruption scenarios. Each segment of the network has a disruption scenario that tells how to reroute tram lines in case of an obstruction.

1.3.1 Vehicle rescheduling

The traffic control software has options for adjusting the vehicle routes. These include skipping stops, cancelling complete trips or setting detours. The new routes are automatically sent to the tram board computers and the advanced traveller information system.

For most disruptions, previously defined scenarios are available that can be activated in the system such that all route adjustments can be executed at once for all vehicles.

A limitation of the system is that it only changes the routing, not the driving times. If a detour is scheduled that takes longer than the original tour, no additional time is scheduled and the vehicle ends up with a delay by the end of the detour. If possible, the delay is caught up by the layover time or by skipping stops.

The traffic controllers currently aim at getting each vehicle back at the time and place it was originally planned. The advantage of this is that the crew schedule remains largely feasible. The disadvantage is that it may not be preferable from a passenger perspective.

1.3.2 Crew rescheduling

In most situations, the crew schedule is largely not affected by the vehicle rescheduling actions. Often, a line is shortened or a detour can be scheduled that takes less or the same amount of time. In these cases, the vehicles just wait longer at the endpoints such that they return at the relief point around the originally scheduled point in time.

However, some drivers are directly affected by the disruption. Take for example a faulty vehicle. The driver has to stay with its vehicle and cannot reach the relief point in time for his next piece of work on a different vehicle. Also, at the relief point another driver is waiting for the vehicle which has broken down. In this case, some crew has to be rescheduled.

The rescheduling of crew is done manually by the dispatchers. They have to decide which drivers could potentially take over pieces of work of other drivers. Usually, drivers are assigned multiple subsequent pieces on the same vehicle. This gives the dispatchers, if they are lucky, time to find rescheduling options. At the central station, one reserve driver is available which is often used first in case of a disruption. All changes in crew duties are communicated by phone and therefore the number of changes that can be carried out is limited.

The traffic control software is focused on the vehicles and does not assist the dispatchers in rescheduling the crew. Therefore, the dispatchers have to track manually which conflicts in the crew schedule arise. This can be a hard job for even small disruptions.

In case a large disruption occurs, in which no detours are possible and parts of the network get isolated, the dispatchers may decide to freeze the crew. This means that drivers stay on their current vehicle and do not switch to other vehicles anymore. They hold their breaks at endpoints. The advantage is that a situation in which a tram has no driver is unlikely. The disadvantage is that labour restrictions are very likely to be violated. Fortunately, these circumstances are rare.

1.4 Contribution of the thesis

The RET currently has limited options for crew rescheduling. At the same time, disruptions have a more disruptive effect on the crew schedule because of the introduction of interlining. Therefore, in this thesis, we develop an algorithm for crew rescheduling.

The method we use is similar to the method which has been implemented at train operator S-tog (Copenhagen) and which is developed by Rezanova (2009). A similar approach is also taken by Potthoff et al. (2010) at Netherlands Railways (NS). By experimenting with this method on real-world data of the RET, we will answer the question whether these methods can be of value for tram networks.

The crew rescheduling problem is modelled by a set covering formulation. This formulation is widely used for scheduling problems, not only for train driver crew rescheduling. Therefore, our method may also be of value for other rescheduling problems.

Chapter 2

Problem description

In Figure 2.1, we show the three phases of rescheduling after a disruption in a tram network. In this thesis, we consider the crew rescheduling phase. A set of (rescheduled) pieces of work from the vehicle rescheduling is given as input. The problem consists of reassigning the pieces of work such that each driver has a feasible duty again and as many pieces as possible are covered.

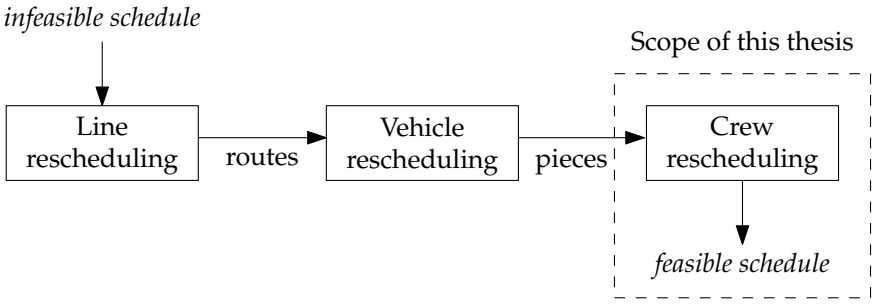

FIGURE 2.1: Rescheduling phases

2.1 Recovery duty

A duty is a set of pieces of work a driver is assigned to. We distinguish the original duty and the recovery duty. The original duty is a driver's duty before crew rescheduling, which might have become infeasible because of a disruption. The recovery duty is a driver's duty after crew rescheduling.

2.2 Distinction between disruptions

We distinguish small, medium and large disruptions. The distinction is not a strict line though.

For small disruptions, only limited vehicle rescheduling actions are necessary and the crew schedule is not affected by the disruption. For example, a short delay of a single vehicle or the shortening of a line. These disruptions can be handled with the current traffic control software.

The focus in this thesis is on medium disruptions. In these cases a number of vehicles is affected and the crew schedule has become infeasible. The disruption does not expand since other vehicles can be rerouted.

Large disruptions include cases in which parts of the network get isolated. For example, a track is blocked at a critical point in the network such that detours are not possible, or take a very long time. Large conflicts in both the vehicle schedule and crew schedule arise. In other words, the situation is out-of-control.

2.3 Interlining

In the past, rescheduling of crew was easier since crew was not scheduled to change lines or not even to change vehicles. If a disruption occurs, the other vehicles on the same line are likely to be delayed as well. Thus, if the piece of work of a duty is delayed, the subsequent piece of work is also likely to be delayed. In that case, no conflict in the duty arises.

In recent years, interlining has been introduced into the schedule. Interlining means that a driver operates on multiple lines during its duty. Interlining leads to a more efficient schedule but also makes rescheduling more difficult. If a disruption occurs, the driver may be stuck in a certain part of the network and may not be able to start his next piece of work on a different line. Therefore, the piece of work has to be transferred to some other driver.

At the moment, only a few lines interline. The RET is considering introducing more interlining into its schedule. However, before doing this, they need a robust crew rescheduling approach.

2.4 Labour restrictions

The driver duties have to meet the labour restrictions. In case of a disruption, those are partly relaxed. We use the following labour restrictions for rescheduling:

- The start time of a duty may not change.
- The duty length may be extended by at most 60 minutes.
- A rest time of at least 15 minutes counts as a break.
- The working time before and after a break may not exceed 4:15 hours.
- A duty that lasts more than 6 hours must contain two breaks.
- A duty should start and end at the same relief point as originally planned.

We chose the numbers ourselves, they differ slightly from the real labour restrictions for scheduling. The numbers are coded as parameters in the algorithm and can therefore be easily changed.

We would like to place a comment on the break restriction. The original duties contain apart from break time also some time to walk to the break room and to give information to passengers. We do not force to schedule this extra time. In that sense, the break restriction is relaxed.

Duties have to start and end at the same relief point as originally planned. However, we allow for transfers. A transfer is the movement of a driver as a passenger to some other location. A transfer can also be included at the end of a duty to return to the original end relief point. The transfer time counts for the duty length. We chose to count transfer time also as a break.

2.5 Comparison to train networks

At some points, the tram network of Rotterdam is less complex than the train networks which have been researched in literature. First, most pieces of work start and end at the same location: the central station. As a result, it may be easier to reassign pieces to different duties. Second, the network is not divided into safety sections and there is no fixed order in which trams should approach a platform. Third, the number of relief points are few and they are relatively close. Usually, other modes of transport are available to travel between relief points. The costs for travelling between relief points are relatively low.

The type of disruptions are different in a tram network. Trams are frequently exposed to small disruptions, because the road is often shared with all other traffic.

The disruption management for a tram network is similar to that for a train network. An important difference is the line rescheduling. For tram networks, often detours are possible. Those detours do not take too much extra time and give more options and flexibility for recovering operations.

The trams in Rotterdam cannot easily be turned and also, at the central station there is no space to park trams. Therefore, the cancellation of a piece of work often causes an infeasible vehicle schedule.

2.6 Problem statement

Because of disruptions and the rescheduling of lines, the crew schedule might have to be rescheduled. The adapted schedule should prevent situations in which a tram has no driver. Furthermore, as many trips as possible from the original schedule should be operated to fulfil travellers demand. The RET is fined for delays and cancellations.

For operational issues, it is important that an adapted schedule can be computed fast and drivers can be notified directly of their new pieces of work. Also, to prevent confusion and new delays, the number of duties that are changed from the original schedule should be minimised.

The relaxed labour restrictions from Section 2.4 should be met. Preferably, the recovery duties end around the same time and location as originally scheduled.

The main objective is to find a crew schedule that is feasible and attractive from both a passenger and a crew perspective.

Chapter 3

Literature review

3.1 Disruption management

Research on disruption management has first been focused on the airline industry. Clausen et al. (2010) give an overview of the concepts, models and methods which have been developed for this industry. They emphasise that the solution approaches for scheduling and rescheduling are very similar to each other.

In the past fifteen years, attention has been given to disruption management for the rail industry. Jespersen-Groth et al. (2009) describe the disruption management process in passenger railway transportation. They describe how the process is organised, which actors are involved and where challenges arise. They also discuss some details of the process at the Netherlands Railways (NS) and DSB S-tog, which are railway operators in The Netherlands and the area of Copenhagen, respectively. The NS exploits an intensively used rail network.

Some years later, multiple mathematical models to deal with disruptions have been developed for the rail industry. We refer the reader to Cacchiani et al. (2014) for an extensive overview of recovery models and algorithms for real-time railway rescheduling. Quite some methods have been tested on datasets of the NS.

As far as we know, research about disruption management for tram networks is scarce. Recently, Roelofsen et al. (2018) have assessed strategies for rescheduling tram lines from a passenger perspective. They present a framework to decide on splitting or diverting a line. The outcomes show that the tram network of The Hague, operated by the HTM, could benefit from the framework. The line rescheduling strategies that are found lead to less delay for passengers.

3.2 Crew rescheduling

3.2.1 Set covering models

Both the crew scheduling and rescheduling problem are often modelled as a set covering problem and the solution approaches applied are similar as well. The difference, however, is that in the scheduling problem a crew schedule is constructed

from scratch, while in the rescheduling problem the original crew schedule has to be maintained as much as possible.

Walker et al. (2005) describe a first attempt to solve the crew rescheduling problem and do this simultaneously with the timetable rescheduling. They tested their method on a relatively small case of the Wellington Metro line. The number of possible recovery duties is limited and no complex methods are needed to generate those.

Huisman (2007) describes a solution method for crew rescheduling a few days ahead, for example in case of unplanned maintance work. A set covering model is solved with an algorithm based on column generation and Lagrangian relaxation. Rezanova and Ryan (2010) and Potthoff et al. (2010) take a very similar approach but Rezanova and Ryan do not use Lagrangian relaxation. They all solve a shortest path problem in a graph to generate recovery duties. The methods of Rezanova and Ryan and Potthoff et al. have successfully been implemented by the suburban rail network of Copenhagen (S-tog) and Netherlands Railways (NS), respectively. S-tog and NS share the same software vendor SISCOG, which has implemented both methods.

Veelenturf et al. (2012) have extended the method of Potthoff et al. (2010) by allowing the retiming of trains. Trains may be delayed for at most a few minutes if otherwise no driver can be found. Sato and Fukumura (2011) also use the set covering model for rescheduling crew members for the Japanese Freight Railway Company. The computation time is short and therefore the method can be used for real-time rescheduling.

3.2.2 Other methods

Abbink et al. (2009) use a system in which the drivers themselves are responsible to swap parts of their duties with other drivers. The system is tested on real-world cases from Netherlands Railways (NS) and performs quite well for relatively small disruptions. However, for larger disruptions, the computation times are too high.

Verhaegh et al. (2017) developed a heuristic to reschedule crew during small disruptions. The idea is to remove pieces from duties that have become infeasible and to reinsert them in different duties to minimize the number of pieces that have to be cancelled. The heuristic performs very well for small disruptions and therefore Verhaegh et al. suggest to combine the heuristic with the method of Potthoff et al. (2010).

3.3 Methods used in this thesis

The method of Rezanova (2009) is used as base for the solution approach in this thesis. Thus, a set covering problem which is solved with column generation. New columns are generated with a shortest path algorithm in a graph. In size, that is the

number of lines, the network of S-tog in Copenhagen is similar to the tram network of Rotterdam.

To limit the complexity of our solution approach, we do not use the method with Langrangian relaxation of Potthoff et al. (2010). Also, we do not implement a retiming option as proposed by Veelenturf et al. (2012).

In this thesis, we speak of the Crew Rescheduling Problem (CRSP). In the literature, similar problems are known as the Train Driver Recovery Problem (TDRP) or the Operational Crew Rescheduling Problem (OCRSP).

Chapter 4

Mathematical models

In this chapter we decompose the crew rescheduling problem into well-known mathematical problems. In the next chapter, we present a method to solve these problems. The central idea of our solution approach is to generate many recovery duties and to select one of them for each driver, such that all pieces of work are covered. Our approach is based on the approach of Rezanova (2009).

First, we explain how to construct a duty graph for each driver in Section 4.1. The duty graphs will be used to generate candidate recovery duties in the next chapter. Second, we explain how to compute the penalty cost of a duty. Some recovery duties may be more attractive than others, and this is expressed in the penalty costs. Finally, we formulate a set covering problem in Section 4.3. In the next chapter, this mathematical program is used to select recovery duties such that all pieces of work are covered and the penalty costs are minimized.

4.1 Duty graphs

For each driver, we build a non-cyclic directed graph. An example of a duty graph is shown in Figure 4.1. A path from source to sink in this graph represents a recovery duty. The white nodes represent pieces of work. In case an arc is drawn between two nodes, the pieces can be carried out subsequently. In reality, the graph from the figure may contain way more nodes and arcs. Notice the path from the source directly to the sink, which represents an empty recovery duty. Also notice the path A-C-D-E-F, which does not contain a break and might therefore be infeasible.

4.1.1 Building the source and sink node

We use a source and sink node to represent the start and end of the recovery period for the respective driver. The recovery period is the period for which pieces from the original duty may be replaced by other pieces.

Figure 4.2 shows the original duties of three different drivers. The rectangles represent pieces of work. The disks show the points in time for the source and sink node. The source and sink are given a time and a relief point. By t_R we denote the earliest start of the recovery period for all drivers. For the source node, we have three options:

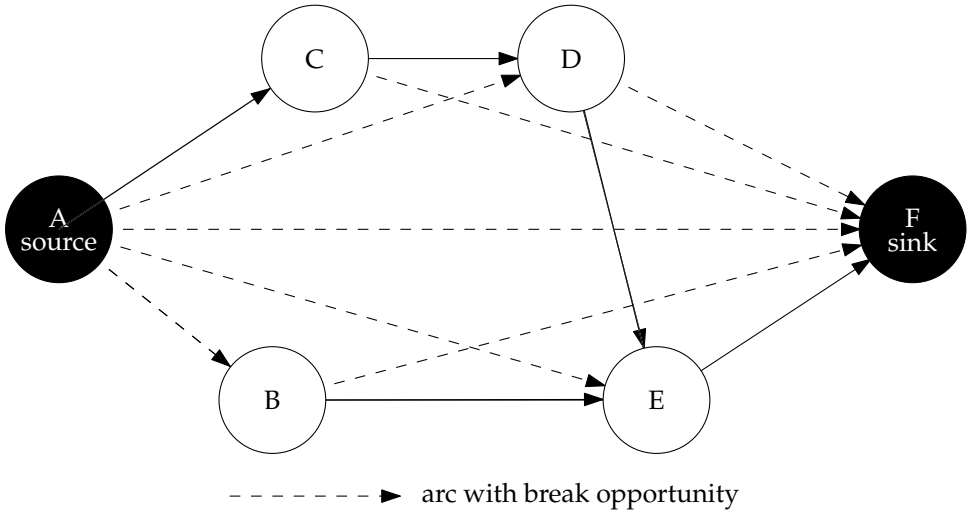

FIGURE 4.1: Example of a recovery duty graph.

I. The original duty has not started yet at t_R. The time and relief point of the source node are set equal to the start time and relief point of the first piece in the original duty.

II. The original duty is assigned a (rescheduled) piece of work at t_R. The time and relief point of the source node are set equal to the end time and relief point of this piece of work.

III. The original duty is in between pieces at t_R. The time of the source is set at t_R, unless the driver is transferring at t_R. In that case, we set the source time equal to the time the driver has finished transferring. The relief point of the source is set equal to the start relief point of the first piece that was originally planned after t_R.

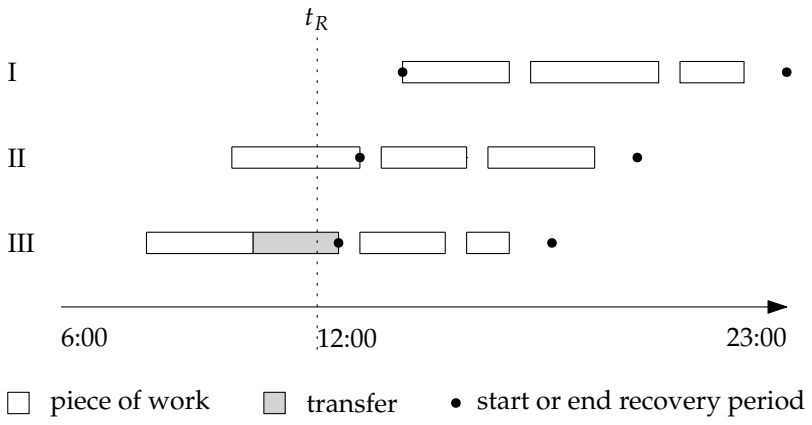

FIGURE 4.2: The original duties of three different drivers. The period in between the disks is the recovery period for that driver.

For the sink node, there is only one option. The relief point is set equal to the relief point where the duty originally ended. The time is set equal to t^* minutes later than the end time of the original duty, where t^* is the maximum overtime.

Note that the last piece of work of a recovery duty does not have to end at the same relief point as the sink, as long as it is possible to transfer to the original relief point within the maximum overtime.

4.1.2 Building the internal nodes

The internal nodes of a duty graph as shown in Figure 4.1 represent pieces of work. Initially, all pieces of work that start and end within the recovery period are considered to be included in the graph. After drawing all arcs between nodes, we exclude the nodes to which there is no path from the source and the nodes from which there is no path to the sink.

4.1.3 Drawing arcs

Lastly, we explain how to draw the directed arcs in a duty graph as shown in Figure 4.1. Arcs between nodes are drawn if the pieces of work can be carried out subsequently. For this purpose, we use the following checks:

1. The $from$ piece should end before the to piece starts.
2. If the pieces are on different vehicles, there should be sufficient time between the pieces to change vehicle.
3. In case of a transfer, there should be sufficient time between the pieces to travel to the other relief point.

We used a minimum change time of 10 minutes and a transfer time of 45 minutes for any pair of locations.

By deciding carefully which nodes to include and which arcs to draw, we have implicitly taken care of most constraints in the graph. However, not all, and therefore not all paths represent a feasible recovery duty. The remaining constraints, such as the break time and maximum driving time, are taken care of when generating recovery duties in the graph.

4.2 Penalty costs

From a passenger perspective, the most important objective is to cover as many pieces of work as possible such that as few trips as possible have to be cancelled. Therefore, we start by setting a big-M cost at not covering a piece of work, where M is an arbitrary large value. In the current implementation, M is set at $10,000$.

There may be multiple solutions to cover all pieces, and some are more attractive than others from a crew or robustness perspective. Therefore, we assign penalty costs to recovery duties such that the master problem can search for an attractive solution that covers all pieces. The costs are relative, meaning that they do not have a unit.

Table 4.1 gives an overview of the penalty costs for recovery duties. For each piece that was not in the original duty, a penalty cost is included. Furthermore, we prefer assigning subsequent pieces on the same vehicle to limit the impact of further disruptions on the crew schedule. Therefore, we set a penalty for a vehicle change. Lastly, an overtime penalty per minute is included since we want to limit overtime.

TABLE 4.1: Penalty costs for recovery duties

Type	Costs
New piece of work	20 per piece
Vehicle change	10 per change
Overtime	3 per minute

Currently, the maximum continuous driving time and the break requirements are modelled as hard constraints by not extending paths in the duty graph that violate these constraints. This will be explained in Section 5.3.3. It might be interesting though to allow violating these constraints at a high penalty cost if it leads to less uncovered pieces of work.

4.3 Set covering problem

The problem of selecting a recovery duty for each driver is formulated as a set covering problem. For each driver, a duty has to be selected from the set of candidate recovery duties such that all pieces are covered. The objective is to minimize the total penalty costs.

Actually, the set partitioning problem feels more natural to model the crew rescheduling problem. However, the set covering problem is easier to solve and it is trivial to construct a solution to the set partitioning formulation from the solution to the set covering formulation (Barnhart et al., 1998). Therefore, the set covering formulation is often used for crew scheduling problems.

We start by introducing the notation. The set N contains all pieces of work that need to be covered. By Δ, we denote the set of drivers and by R^δ the set of candidate recovery duties that are generated for driver $\delta \in \Delta$.

The parameter c_r^δ denotes the penalty cost of candidate recovery duty r of driver δ. The binary parameter a_{ir}^δ indicates whether piece i is covered by candidate recovery duty r from driver δ. The parameter M is an arbitrary large value which is incurred as a cost for each piece of work that is left uncovered.

The decision variable x_r^δ is 1 if candidate recovery duty r of driver δ is selected and 0 if not. Furthermore, the decision variable z_i is 1 if a piece of work is not covered by any selected recovery duty and 0 otherwise.

The problem can now be formulated as a set covering problem, comparable to the formulation from Rezanova (2009).

Sets

N	set of pieces of work
Δ	set of drivers
R^δ	set of candidate recovery duties for driver δ

Parameters

a_{ir}^δ	is 1 if piece i is covered by recovery duty $r \in R^\delta$, 0 otherwise
c_r^δ	cost of recovery duty r of driver δ
M	cost of not covering a piece of work

Variables

x_r^δ	is 1 if recovery duty $r \in R^\delta$ is selected, 0 otherwise
z_i	is 1 if piece i is not covered, 0 otherwise

Objective

$$\min \quad \sum_{\delta \in \Delta} \sum_{r \in R^\delta} c_r^\delta x_r^\delta + \sum_{i \in N} M z_i \tag{4.1}$$

Constraints

$$\text{s.t.} \quad \sum_{r \in R^\delta} x_r^\delta = 1 \qquad \forall \delta \in \Delta, \tag{4.2}$$

$$\sum_{\delta \in \Delta} \sum_{r \in R_\delta} a_{ir}^\delta x_r^\delta + z_i \geq 1 \qquad \forall i \in N, \tag{4.3}$$

$$x_r^\delta \in \{0,1\} \qquad \forall r \in R^\delta, \quad \forall \delta \in \Delta, \tag{4.4}$$

$$z_i \in \{0,1\} \qquad \forall i \in N \tag{4.5}$$

The driver constraints (4.2) state that for each driver δ exactly one recovery duty $r \in R^\delta$ has to be selected. There always is at least one recovery duty in R^δ.

The piece constraints (4.3) ensure that for each piece, at least one duty is selected that covers this piece. In case no duty can be found to cover the piece, the artificial variable z_i may be set to 1 at a very high cost (M). This way, we can discover which pieces cause infeasibilities. A piece may be covered by more than one duty, in which case one of these drivers travels as passenger.

The decision variables are required to be binary in (4.4) and (4.5). A recovery duty is either selected or not, and a piece of work is either covered or not.

The objective (4.1) is to minimise the cost of all selected recovery duties and the costs of the artificial variables. The big-M ensures that a solution with less uncovered pieces always has a lower objective value.

Chapter 5

Solution approach

5.1 Outline

The difficulty of solving our rescheduling problem lies in the fact that there is an enormous number of possible recovery duties. The time needed to generate all possible paths in the duty graphs rises exponentially in the number of nodes. Besides, we also face the set covering problem, whose solving time rises exponentially in the number of duties. Therefore, we take a column generation approach, which implies that we only generate a limited number of duties and iterate between solving the set covering problem and generating new candidate duties.

From now on, the set covering problem is referred to as the restricted master problem (RMP). The problem is called restricted, since we do not include all possible duties. In order to use column generation, we use the linear programming relaxation of the RMP. The RMP is initialised by including a limited set of duties. Thereafter, we iterate between solving the RMP and generating new candidate duties until a stopping criterion is met.

The key to finding a satisfactory solution is an efficient algorithm to find useful candidate recovery duties using the duty graphs. We refer to this problem as the subproblem. The values of the dual variables of the RMP give information about which pieces are uncovered or are costly to cover with the current set of duties. Therefore, these values are useful for the subproblem. The dual values are assigned to nodes in the duty graphs. Now, the problem of finding duties that could reduce the objective of the RMP is a resource constrained shortest path problem.

In Figure 5.1, we give an overview of our algorithm to solve the crew rescheduling problem. The approach we take is similar to the approach of Rezanova (2009). However, we do not implement branching. Computational results from Rezanova show that only 4 percent of the test cases benefited from branching, while it comes at a higher computation time. Also, leaving out branching limits the complexity of our solution method. Instead, we solve the root node with column generation. Then, in order to find a feasible solution, we solve an integer version of the RMP with all columns that were generated in the root node.

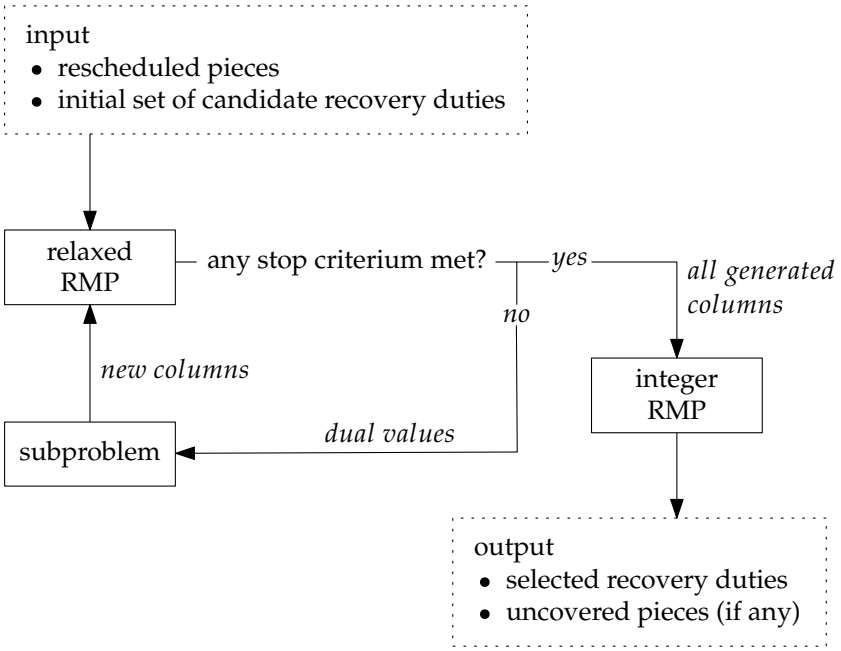

FIGURE 5.1: Outline of the solution approach to the crew rescheduling problem.

5.2 Restricted master problem

The actual restricted master problem (RMP) we use differs slightly from the set covering formulation in Section 4.3. The problem is relaxed by requiring the decision variables to be non-negative only instead of integer. This is necessary for a column generation approach in which dual values are used in a subproblem to generate new, interesting columns.

The RMP is initialised by including one recovery duty for each driver. For the original duties that are not affected by the vehicle rescheduling, the initial recovery duty is just a copy of the original duty. For the duties that are affected, a recovery duty is constructed by removing pieces of work from the original duty until it has no conflicts anymore.

The relaxed, linear RMP is solved by the CPLEX solver from IBM. The algorithm keeps iterating between solving the RMP and generating new columns until no new columns with negative reduced costs are found or the stopping criterion is met.

Since the RMP is relaxed, the algorithm may end up with a solution which is not integer and therefore not directly usable. For example, instead of selecting one recovery duty for a driver, the solver may have selected two recovery duties half. In that case two drivers are competing for the same piece of work. Therefore, an integer solution has to be searched for. The last set of columns is given as input to an integer version of the RMP and again the CPLEX solver is used. If the optimal solution is not found within 30 seconds, the solver is terminated. The objective value of the obtained solution may be equal or worse than the objective value of the relaxed RMP.

5.3 Subproblem: duty generation

The idea of the duty generation algorithm is to build paths from the source to the sink in the duty graphs. We start at the source and from there we consider extension in all directions. To limit the computation time, we only search in directions that look promising. The reduced cost of a path tells us whether a path has potential for reducing the objective of the RMP.

Each driver has its own duty graph and therefore the subproblem is solved for each driver individually. The paths with lowest reduced costs are added to the relaxed RMP. After each iteration of solving the RMP, the dual values have changed and the subproblem is solved again. In this section, we first present an overview of the algorithm after which we discuss components of the algorithm in more detail. Finally, we give a numerical example of the algorithm.

5.3.1 Shortest path algorithm

Similar to Rezanova (2009), we use a label setting procedure. A label represents a path in the duty graph, which starts at the source node and ends at a node. If a path represented by a label is extended, a new label is generated with a reference to the previous label. Therefore, a label does not need to store the full path. A label keeps track of the reduced costs, the number of breaks and the continuous driving time of the path. The algorithm is initialised by creating a label representing a path that starts and ends at the source node.

The algorithm visits the nodes one by one in topological order. The duty graphs are weighted, directed non-cyclic graphs. A property of these graphs is that for every combination of nodes v and w, there may exist a path from v to w or from w to v, but never both. This property is used when the nodes are sorted topologically. If there exists a path from v to w, then v comes before w in the topological order. In our case, we can sort the nodes topologically by sorting them on the end time of the corresponding piece of work.

The source node comes first in the topological order and is therefore visited first. When visiting a node v, we consider all labels that were generated before and that represent a path that ends at node v. We apply a dominance check to this set of labels. Every label that is dominated by any other label in the set, is discarded. Next, all remaining labels are extended by generating new labels for all outgoing arcs. If a newly generated label does not represent a feasible path, it is discarded immediately.

Algorithm 1 gives the pseudo code of the shortest path algorithm. Let N denote the topologically sorted set of all nodes in the duty graph and A^v the set of outgoing arcs of node v. Furthermore, by L^v we denote the set of generated labels which represent a feasible path that ends at node v and by c_R^i the reduced cost of label i.

Our approach differs slightly from Rezanova (2009). They use a first-in-first-out strategy for extending labels and apply a dominance check directly when a label is

Algorithm 1: Shortest path algorithm

```
Initialise: generate a label for the source node.
foreach node v ∈ N do
    Remove all dominated labels from L^v.
    foreach label i ∈ L^v do
        foreach arc (v, w) ∈ A^v do
            Generate a new label j for node w with a reference to label i.
            If label j is feasible, add it to L^w.
        end
        if v is the sink node and c^i_R ≤ 0 then
            Generate a recovery duty by backtracking label i.
        end
    end
end
```

generated. We visit the nodes one by one and extend all labels which are associated with the node. Potthoff et al. (2010) use this node visiting approach as well. The result of the two approaches is the same and also the computation time should be very similar. We felt that this approach is more natural and easier to understand.

5.3.2 Reduced cost computation

The reduced cost of a path is the sum of penalty costs and dual values. The reduced cost tells whether the path has potential to reduce the objective value of the RMP.

The penalty costs as introduced in Section 4.2 are assigned to arcs in the duty graph. Therefore, the penalty cost of a path is just the sum of the penalty costs of the arcs. If we consider the source-sink path $p = o^k \to v_1 \to v_2 \to \ldots \to v_M \to s^k$, then the penalty cost c_p of path p is:

$$c_p = c(o^k, v_1) + c(v_1, v_2) + \ldots + c(v_M, s^k) \tag{5.1}$$

where o_k is the source node, v_i is a node on the path, s_k is the sink node and $c(v_i, v_j)$ is the penalty cost of the arc between v_i and v_j. Note that $c(v_M, s^k)$ includes a possible penalty for overtime.

The values of the dual variables from the RMP are set as costs to the nodes in the graph. Let λ^k be the dual value of the constraint of driver k, then the cost of the source node is set at $-\lambda^k$. Furthermore, let π_i be the dual value of the constraint of piece i, then node v_i is given cost $-\pi_i$. The sink node is not given a cost. Now, the formula for the reduced cost $\bar{c}_p$ of path p becomes:

$$\bar{c}_p = c_p - \lambda^k - \pi_1 - \pi_2 - \ldots - \pi_M \tag{5.2}$$

The cost of a path that does not end at the sink is computed similar, except that $c(v_M, s^k)$ is left out in Equation 5.1.

5.3.3 Feasibility check

Every time a new label is generated, a feasibility check is performed which tells if the subpath could lead to a feasible recovery duty. By design of the duty graph, pieces of work in a subpath can always be carried out subsequently considering the start and end time and location. During the label feasibility check, we check the following:

- The path does not violate the continuous working time constraint. A driver is not allowed to drive more than 4:15 hours without a break.
- The path contains enough breaks. If the path ends at the sink node and the duty length is at least 6 hours, the path should contain at least two breaks.

5.3.4 Dominance check

During the dominance check, we check for each label in the list L^v if it is dominated by any other label in the list. Label i dominates label j if:

1. it has lower reduced costs, and
2. it has a higher or equal number of breaks, and
3. it has a lower continuous driving time.

Each label that is dominated is removed from L^v. If all of the above conditions should be met, we speak of a strict dominance check. If not all conditions are used, we speak of a soft dominance check. In a soft dominance check, labels are discarded faster and less paths are considered. By using the strict dominance check, one knows for sure that the optimal solution is obtained if the algorithm is not terminated early.

The dominance check differs slightly for the sink node. At that moment, we are not interested in conditions 2 and 3 anymore, since the path cannot be further extended. Therefore, we only consider the reduced costs in that case.

5.3.5 Numerical example

We will show how the duty generation algorithm works by means of a numerical example. Figure 5.2 shows a duty graph with penalty costs on the arcs and dual costs on the nodes. The sequence A-B-C-D-E-F denotes a topological order.

Table 5.1 shows the progress of the shortest path algorithm. The table shows for each node (i) which labels were generated, (ii) which of those are dominating and (iii) the extension of the dominating labels. We use a soft dominance check by only comparing the reduced costs and the number of breaks (conditions 1 and 2 as described in Section 5.3.4).

When treating node D, the generated label ACDE is infeasible since it violates the maximum continuous driving time constraint. When treating node E, label ADE dominates each of the other labels and therefore those are discarded. Label ADE has

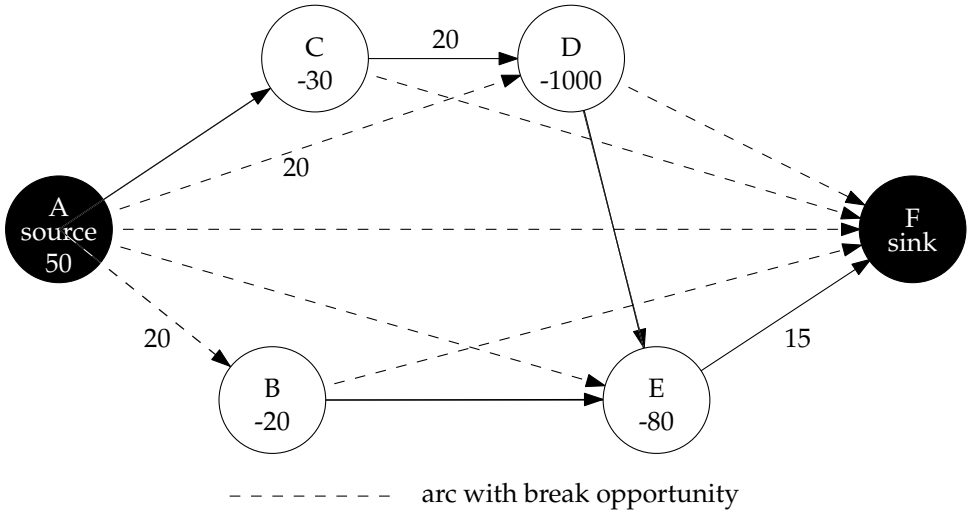

FIGURE 5.2: Example of a duty graph with costs on nodes and arcs.

TABLE 5.1: Progress of the shortest path algorithm.

Node	Labels	Dominating	New labels
A (source)	A	-	AB, AC, AD, AE, AF
B	AB	-	ABE, ABF
C	AC	-	ACD, ACF
D	AD, ACD	-	ADE, ADF, ACDE (infeasible), ACDF
E	AE, ABE, ADE	ADE (-1010)	ADEF
F (sink)	AF, ABF, ACF, ADF, ACDF, ADEF	ADEF (-995)	-

lower reduced costs ($50 + 20 - 1000 - 80 = -1010$) while having the same number of breaks. Similarly, for node F, the path ADEF has the lowest reduced costs and has enough breaks and therefore dominates each of the other paths.

The algorithm ends up with ADEF as shortest path. A recovery duty is generated by backtracking label ADEF, as is shown in Figure 5.3. The recovery duty is added to the relaxed RMP.

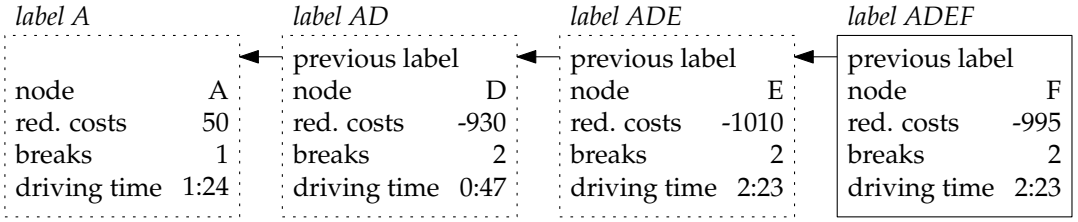

FIGURE 5.3: The labels that are generated for path ADEF.

5.4 Limiting the search

Column generation has limited the computation time since less paths in the duty graph have to be inspected while in theory still the optimal solution can be found. However, still too many paths exist and therefore we consider further limiting the searches through the duty graphs. The cost of limiting is that we are not guaranteed to find the optimal solution for the relaxed RMP. Therefore, the key is to ensure that still attractive recovery duties are found to end up with a solution that is close to optimal.

5.4.1 Driver neighbourhood

The first way to limit the search is by limiting the number of original duties that is considered to change. We call the set of duties that may be changed the neighbourhood. We do not develop a refined method to decide on which duties to include in the neighbourhood, but we always include the duties that contain disrupted pieces of work.

Rezanova (2009) have implemented a neighbourhood expansion. If no solution can be found with all pieces of work covered, more drivers are added to the neighbourhood. Potthoff et al. (2010) use an approach in which they search for drivers with similar pieces of work as those that are disrupted.

5.4.2 Soft dominance check

The second way to limit the search, is dropping one or more conditions of the dominance check from Section 5.3.4. As a result, it will happen more often that a label is dominating some other label. More paths are discarded early such that less paths are extended. However, note that also useful paths may be discarded.

Early experiments showed that dropping the third condition, a lower continuous driving time, substantially decreased the computation time. At the same time, the objective value increased only marginally. Therefore, we decided to drop this condition for all of our experiments in Chapter 6.

5.4.3 Maximum time span between two pieces of work

Duties that contain few pieces of work and long breaks are not likely to be useful for finding a solution. Therefore, we limit the time span between two pieces of work at 120 minutes. Thus, in the duty graphs, there exist no arcs between pieces of work with a gap of more than 120 minutes.

5.4.4 Stopping criterion

The algorithm could continue for a very long time finding new columns with negative reduced cost. However, after some time the new columns only marginally

improve the objective of the RMP. Therefore, we stop the algorithm if the objective function has not improved by x over the last n iterations.

Chapter 6

Computational results

In this chapter, we test our crew rescheduling algorithm on a real crew schedule of the RET. In Section 6.1, we give a general description of this schedule. Next, in Section 6.2, we discuss which parameters for the algorithm were found to be effective for finding good solutions. In Section 6.3, we will test the algorithm on many randomly generated disruptions and finally we will test the algorithm on a realistic disruption scenario in Section 6.4.

The algorithm is implemented in Java and run on a common business laptop. For the master problem, we have used a code example by Bouman (2018) that shows how to use CPLEX and Java for solving a problem with column generation.

6.1 General description of the original crew schedule

All of our tests are performed on a crew schedule for a weekday timetable, valid from December 2018. The schedule contains 204 driver duties. In total, those duties include 1,605 pieces of work.

In the crew schedule, there is no interlining for lines 2, 20, 23 and 25. Drivers on these lines switch vehicles during their break, but they do not switch between lines. The challenge lies in lines 4, 7, 8, 21 and 24. The crew for these lines interline, and on top of that, these lines are most sensitive to disruptions. The vehicles have to share the road with automobile traffic. Hence, disruption management for these lines is currently a difficult task for the traffic controllers. Testing our crew rescheduling algorithm is therefore most interesting for these disruptions.

Most pieces of work in the crew schedule start and end at the central station. Often, the first or last piece of work in a duty ends at a different location, usually the depot or an endpoint close to the depot.

In Figure 6.1, we show the number of drivers which are in service and which of them are holding a break over the day. Clearly, we see that the number of drivers in service raises until 7:30, which is explained by the vehicles coming into service. Also, we see that more drivers are working during the rush hour. This is because of line 20 which only operates during rush hour. After the evening rush hour, line frequencies decrease. Half of the vehicles are returned to the depot and less drivers are needed.

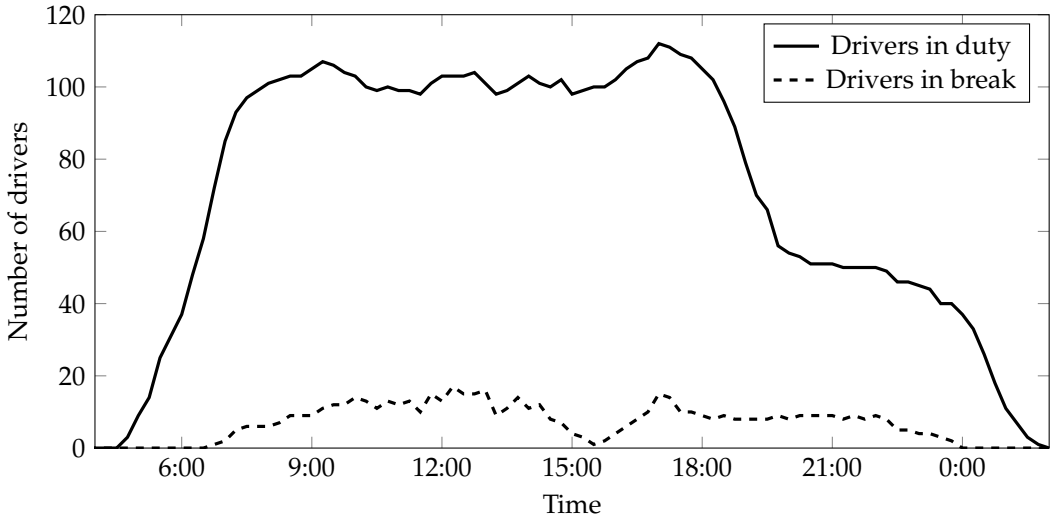

FIGURE 6.1: Drivers in duty and holding a break over the day.

From the figure, we may formulate hypotheses about whether our algorithm is able to find good solutions. Until around 6:30, no drivers have a break and the schedule is therefore tight. The same holds for 15:30, when many drivers end their morning shift, while other drivers start their evening shift. Around 12:30, many drivers are having a break and therefore conflicts in the crew schedule may be easier to resolve.

6.2 Parameter settings

6.2.1 Driver neighbourhood

Initially, we experimented with including all drivers. This does not mean that the algorithm is likely to end up with many changed duties, since changing duties involves penalty costs. It just means that the algorithm is given many possibilities. This approach led to reasonable computation times for disruptions in the late afternoon or evening. However, high computation times were found for disruptions during the morning or early afternoon since more duties and pieces of work are then available for rescheduling.

Therefore, we decided to implement a neighbourhood as proposed in Section 5.4.1. The neighbourhood approach shows way better computation times while the algorithm is still given enough possibilities to find solutions with all pieces covered.

We will discuss how we have selected a neighbourhood for the random disruptions in Section 6.3. We included:

- All disrupted duties: duties that contain a rescheduled piece of work.
- Non-disrupted duties that start between $t_R - a$ and $t_R + b$.

We chose a and b such that we obtained a neighbourhood of the size we wanted. Obviously, we did not include duties that ended before t_R.

6.2.2 Stopping criterion

We introduced the stopping criterion in Section 5.4.4. The algorithm is terminated if the objective value has improved less than x over the last n iterations.

The objective value is the sum of the penalty costs. Remember that we set the penalty cost on an uncovered piece of work very high. Therefore, very large improvements are made if new columns are added to the RMP such that an extra piece of work can be covered. Small improvements are made if more attractive duties are found, for example if less drivers have to work overtime. We saw that the objective value does not necessarily decrease in every iteration.

An important performance measure is the gap between the objective value of the relaxed or linear RMP and the integer RMP. A large gap implies that the integer solution has more pieces of work uncovered. We saw that if we terminated the algorithm just after an iteration which had largely improved the objective value of the relaxed RMP, the integer solution is not likely to have an objective value that is close. If we continued the algorithm for a few more iterations, more columns and thus more options are given to the RMP and the integer problem is way more likely to have a similar solution.

To support our statements, we have solved the integer RMP to optimality in every iteration for one instance. The result is shown in Figure 6.2. The objective value of the linear RMP drops smoothly. The objective value of the integer RMP drops in steps and with a lag. Take for example the moment that the linear RMP has a solution with three pieces left uncovered $(10; 3{\cdot}10^4)$. It takes three more iterations before the integer RMP has a solution with the same objective value.

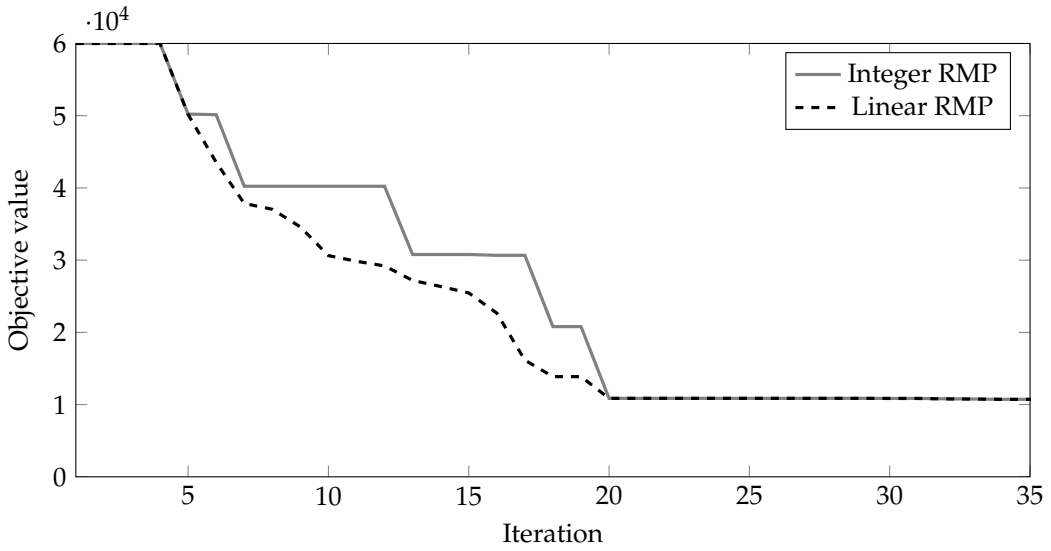

FIGURE 6.2: Course of the gap between the integer and linear RMP.

We experimented with a few values for x and n. The objective value should be as low as possible and the gap between the linear and integer solution should be small. We chose for $x = 50$ and $n = 40$, which is on the safe side. We almost always found good solutions within a reasonable time.

6.3 Random disruptions

6.3.1 Creating disruptions in the vehicle blocks

We create test instances by randomly disrupting some vehicle blocks in the vehicle schedule. Those instances only partly represent a realistic scenario. However, they are useful to examine the performance of our crew rescheduling algorithm. The algorithm is expected to perform similarly for real-life scenarios.

The vehicle blocks are disrupted as shown in Figure 6.3. The vertical lines denote the split of the vehicle block into pieces of work. In the schedule of the RET, vehicle blocks start and end in the depot and the blocks are usually split at the central station. The disruption starts at t_D. From t_R, crew may be rescheduled and from t_Z the original vehicle block is restored. The figure shows that two pieces of work are merged by cancelling the relief around 7:45.

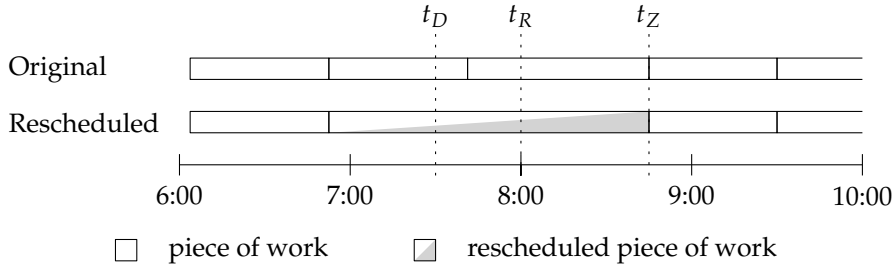

FIGURE 6.3: A vehicle block before and after creating the disruption.

At the traffic control centre, we saw that the dispatchers are focused on rerouting the disrupted vehicles such that they get back to their original schedule. The vehicle block rescheduling from Figure 6.3 is therefore quite realistic. For example, consider the following disruptions, which occur regularly:

- A vehicle and accompanying driver are stuck because of an obstruction. When the vehicle can continue its way, the vehicle turns before reaching the end point to catch up with the delay.
- A vehicle breaks down and has to be brought to the depot. Here, the driver picks up a spare vehicle which takes over the vehicle block. By merging the pieces of work, the driver has enough time to do this.

The set of test instances is created by varying (i) the disruption time t_D, (ii) the number of vehicle blocks that are disrupted and (iii) the neighbourhood of duties that are considered. For the rescheduling time we use $t_R = t_D + 30$.

In total, we created sixteen instances. The disruption time is either 8:00, 12:00, 15:00 or 18:00. The number of vehicle blocks that are disrupted and rescheduled is either 10 or 30 and the size of the neighbourhood is either small or large.

The disruptions in the vehicle blocks lead to conflicts in the crew schedule. Pieces of work in a duty may overlap or the maximum driving time is exceeded. Some pieces become uncovered: they cannot be operated anymore in the original schedule.

6.3.2 Results

In Table 6.1 we present a summary of the instances and the results of the crew rescheduling algorithm for those instances. Generally speaking, the algorithm is able to reduce the number of uncovered pieces to a minimum within a reasonable time.

TABLE 6.1: Results for the random test instances.

Instance						Results							
Disruption time (t_D)	Disrupted vehicles	Uncovered pieces	Duties in neighb. a	Duties in neighb. b		Objective IRMP	Objective LRMP	Uncovered pieces	Changed duties	Overtime (h:mm)	Overtime (duties)	Comp. time (m:ss)	
8:00	10	12	26	60	60	1,895	1,774	0	20	2:15	5	0:13	(1)
			61	120	120	1,546	1,478	0	20	0:32	2	4:15	(2)
	30	21	53	0	0	2,647	2,444	0	35	0:56	2	1:47	(3)
			83	120	120	32,533	2,354	3	34	1:10	2	9:21	(4)
12:00	10	6	17	0	0	10,707	10,707	1	10	1:19	2	0:01	(5)
			59	120	120	10,707	10,634	1	11	0:39	2	3:53	(6)
	30	19	45	0	0	12,077	12,075	1	32	3:01	9	0:20	(7)
			76	120	120	12,056	12,052	1	33	2:44	8	4:26	(8)
15:00	10	5	28	15	15	20,666	20,666	2	9	0:46	2	0:30	(9)
			58	180	0	1,715	1,715	0	19	1:59	4	0:55	(10)
	30	14	40	0	0	2,928	2,851	0	29	3:12	8	1:12	(11)
			58	60	60	32,193	2,718	3	25	1:45	5	6:17	(12)
18:00	10	8	22	120	120	11,263	11,014	1	15	1:38	3	0:05	(13)
			41	180	180	11,041	10,958	1	18	0:27	1	0:38	(14)
	30	19	35	0	0	2,012	1,846	0	26	2:04	4	0:12	(15)
			47	180	180	12,069	1,690	1	27	1:53	5	1:25	(16)

The objective value of both the IRMP and the LRMP are presented in the table. This objective value is the sum of the penalty costs, introduced in Section 4.2. By IRMP, we denote the integer restricted master problem. This is the problem that is solved once at the end of the algorithm. By LRMP, we denote the linear restricted master problem, which is solved in every iteration. The objective of the LRMP is a lowerbound for the IRMP.

We will continue this section by discussing the properties of the solutions: uncovered pieces, computation time, overtime and number of changed duties.

Uncovered pieces

For ten instances, uncovered pieces are left after rescheduling. There are three reasons for this:

1. Covering the piece of work with the current constraints is likely to be impossible (instances 5 to 8, 13 and 14).
2. The neighbourhood is too small or not chosen usefully (instance 9).
3. The integer RMP is not solved to optimality (instance 4, 12 and 16).

Considering the neighbourhood selection, we saw that our simple rule with $a = b$ did not perform well for instances 9 and 10 since almost all duties end or start around this time. Therefore, we chose $b = 0$ for instance 10.

We would like to elaborate on the last reason. If we compare instances 3 and 4, we see that the objective of the linear RMP is lower for instance 4. This is the case for all instances, a larger neighbourhood leads to a lower objective for the LRMP. However, the objective of the integer RMP is much higher, implying that pieces are uncovered. By the end of the algorithm, the IRMP is solved and it is actually a matter of chance whether the branch-and-cut algorithm of CPLEX is able to find a good solution within the given time limit. Especially for the large neighbourhoods for which way more columns are added to the RMP, this is an issue. We will give ideas to overcome this issue in Section 7.2.2.

Computation time

The results show that good solutions can already be found with small neighbourhoods. The computation time raises in the size of the neighbourhood, for two reasons. First, for each driver in the neighbourhood the subproblem is solved in every iteration. Second, it takes more iterations before the stop criterion is reached. The RMP becomes bigger and solving the integer RMP becomes computationally more expensive.

Large disruptions indirectly raise the computation time. For the instances with 30 disrupted vehicles, we saw that the neighbourhoods already became quite large by including the disrupted duties only. As just explained, a larger neighbourhood causes a higher computation time. An option to reduce the computation time would be to split the problem into two or more problems.

Overtime

Every solution contains some overtime. As expected, we see that increasing the size of the neighbourhood leads to solutions with less overtime (instance 2, 6, 8 and 14).

We would like to note that not all overtime can be avoided. The merge of two pieces of work could immediately lead to overtime, if it is the last piece in a duty. The piece starts before t_R and therefore it cannot be assigned to some other driver.

We expected that the solutions to the instances around 15:00 would contain more overtime, because many duties start or end around this time. However, the results do not support this hypothesis.

Changed duties

The algorithm changes quite some duties to find a solution. This is because currently no penalty is set at changing a duty. Instead, a penalty is set at assigning a different piece of work to a driver. The algorithm is well able to change duties minimally. Most of the pieces of work from the original duty are still contained in the recovery duty. Also, the recovery duties contain few vehicle changes, mostly two or three, which is preferable for a robust schedule. This is partly because of the vehicle change penalty, but also just because it is more efficient. Changing vehicles involves slack time because of the minimum change time. We saw that dropping the vehicle change penalty only marginally increased the number of vehicle changes.

6.4 The Oudedijk case

6.4.1 Description

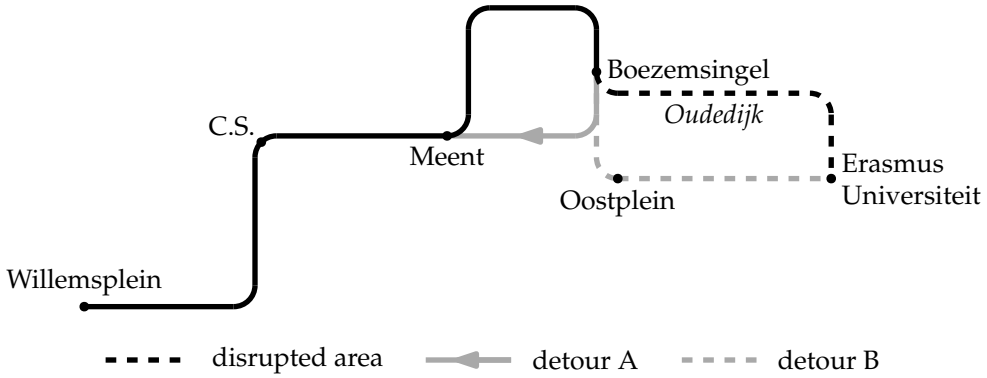

FIGURE 6.4: The disrupted line 7 and its detour options.

In this section, we test the crew rescheduling algorithm on a more realistic scenario. Consider line 7 as shown in Figure 6.4. Suppose that a disruption occurs on the Oudedijk. As a result, the part from Boezemsingel to Erasmus Universiteit cannot be operated anymore. Three trams are stranded in this part and those trams and their accompanying drivers have to wait until the disruption is over. Assume that the disruption lasts an hour. We set the recovery start time 20 minutes from the disruption start time, thus $t_R = t_D + 20$. The figure shows two possible detouring options. Note that turning at Boezemsingel is not possible.

The disruption scenario currently in use by the RET says that the line is rescheduled according to detour B. The detour takes about the same time as the original route. Therefore, only drivers in the disrupted area may get conflicts in their duty.

However, since three trams are stranded, this detour implies that the frequency interval of 10 minutes cannot be met anymore. In particular, there will be one gap of 40 minutes between two successive vehicles. Between t_D and $t_D + 40$, no vehicles are departing from Boezemsingel in direction Willemsplein.

From the passenger perspective, detour A is preferred. Since the line is shortened, the frequency interval of 10 minutes can be retained for the non-disrupted area. The loop Meent-Boezemsingel-Meent is only operated in the specified direction. Travellers for destination Erasmus Universiteit can take tram 21 or 24, which operate between Oostplein and Erasmus Universiteit. The walk to stop Oostplein takes four minutes.

We create four test instances at different times during the day by rescheduling the vehicles according to detour A. Vehicles end their trips at Boezemsingel and from here they take over trips in opposite direction from the vehicles that were supposed to be there. The vehicle rescheduling leads to infeasible duties for the drivers in the disruption area. Their duties contain up to six pieces of work which become uncovered as a result of the disruption. Those have to be taken over by drivers outside the disruption area.

For the neighbourhood, we include all drivers which are in duty at t_R. Thus, all duties that originally started before t_R and ended after t_R. We further limit the neighbourhood by only considering drivers from the interlining lines 4, 7, 8, 21 and 24. In Table 6.2 a summary of the instances are presented together with the results. We do not show the number of drivers in the neighbourhood. Instead, we show the number of pieces, which is a more accurate indicator for the size of the problem.

6.4.2 Results

The results of the crew rescheduling algorithm are shown in Table 6.2. Solutions which cover all pieces of work are found for all instances within three minutes.

TABLE 6.2: Results for the Oudedijk case.

Instance			Results						
Disruption time (t_D)	Uncovered pieces	Pieces in neighb.	Objective IRMP	Objective LRMP	Uncovered pieces	Changed duties	Overtime (h:mm)	Overtime (duties)	Comp. time (m:ss)
8:00	6	302	843	799	0	11	0:11	1	2:22
12:00	3	227	473	442	0	7	1:21	3	0:48
15:00	3	204	927	903	0	13	0:49	3	0:31
18:00	2	184	951	644	0	12	1:17	2	0:50

For the instance at 15:00, we expected more overtime since many duties are ending or starting around this time. However, there are enough duties that do not start

or end around this time and therefore the algorithm is able to find a solution with limited overtime.

The instance at 18:00 differs from the others since from this time half of the vehicles are returned to the depot. In the evening, the line is operated at a 20 minute interval. The algorithm is still able to find a solution, but with a lot of changes and also some transfers.

For the latter three instances, the computation time is rather short. This is explained by the number of pieces in the neighbourhood. The algorithm has less possibilities and less columns are generated and added to the RMP. We asked ourselves whether a complete enumeration of recovery duties would be possible. We tried running an enumeration but the algorithm took too long and we terminated early. Therefore, we conclude that the column generation technique has been successful in finding a good solution within a short time.

Chapter 7

Conclusion and discussion

7.1 Conclusion

In this thesis, we considered the crew rescheduling problem in the tram network of Rotterdam. We used a method which has previously been developed and implemented in the rail industry. The problem is modelled as a set covering problem. The LP relaxation of this problem is solved with a column generation approach. The subproblem of generating columns is modelled as a resource constrained shortest path problem.

We tested our solution method on a real-life crew schedule of the RET, the public transport operator in Rotterdam. The results show that the algorithm is able to repair small to quite large disruptions in the crew schedule while respecting the labour restrictions. The Oudedijk case shows that the algorithm also performs well on a realistic disruption scenario. The crew rescheduling algorithm would allow the dispatchers to choose detours which are preferable from a passenger perspective.

The computation time of our algorithm is low. Good solutions can be found within minutes. This is important from an operations perspective, since a quick rescheduling process can prevent further delays and cancellations. We think that our method can be further improved by choosing the neighbourhood more carefully. Small, carefully selected neighbourhoods decrease the computation time, while still good solutions can be found.

A crew rescheduling algorithm like the one presented in this thesis gives new opportunities for disruption management. Vehicles can be rescheduled from the passenger perspective, like proposed by Roelofsen et al. (2018). The crew rescheduling algorithm will reassign the uncovered pieces such that the new vehicle schedule can be operated.

7.2 Discussion

In this section, we discuss some aspects of the crew rescheduling which may be of interest for future research or implementation.

7.2.1 Neighbourhood

We used a simple rule to decide on which drivers to include in the neighbourhood. A more sophisticated method can be beneficial for the computation time and the quality of the solution. More information from the crew schedule, like Figure 6.1, can be used to select the neighbourhood of drivers more carefully. Potthoff et al. (2010) used a more thoughtful method for selecting the neighbourhood. Rezanova (2009) implemented a neighbourhood expansion.

7.2.2 Finding an integer solution

We saw that obtaining a good solution for the integer RMP within a short time may be difficult if the problem contains many columns. In our approach we added all columns that were generated. To overcome the problem of a high computation time for some instances, we suggest removing columns from the RMP that were not selected in the last k solutions to the linear RMP. The columns can be discarded or kept in a list to be re-added to the RMP in case they would again have negative reduced costs.

However, finding an integer solution from the linear solution might stay an issue. Another, more sophisticated method is branch-and-price as developed by Barnhart et al. (1998) and applied to the train crew rescheduling problem by Rezanova (2009).

7.2.3 Uncovered pieces

In this thesis, we have not discussed what to do with uncovered pieces. If pieces are left uncovered after the crew rescheduling, the vehicle schedule becomes infeasible. Multiple pieces need to be cancelled or new pieces have to be created since trams can not be parked near the central station. We suggest to tackle this problem when developing a vehicle rescheduling method.

7.2.4 Extensions

The algorithm may be extended or changed to find more attractive solutions. Here, we give some suggestions.

- Instead of a hard constraint on the break time, one could set a high penalty on skipping breaks. Sometimes, skipping a break is the only way to cover a piece of work. By doing this, the vehicle schedule remains feasible and no other pieces have to be cancelled.
- We have not thoroughly implemented driver transfers in our algorithm. One could consider splitting pieces of work at other relief points than the central station, or changing the start and end locations of a duty. More options could lead to more attractive solutions.

- We set a fixed, high penalty on not covering a piece of work. This cost could be made dependent on the length of the piece, the number of passengers affected or the number of trips that has been cancelled before on the same line.
- One could set a higher penalty on long overtime, such that overtime is spread more equally. For example, two drivers working 21 minutes overtime might be preferable over one driver working 40 minutes overtime.
- An interesting question would be at which times of the day reserve drivers are needed. In this thesis, we solved the crew rescheduling problem without reserve drivers.

7.2.5 Further softening of the dominance check

Originally, we proposed a dominance check with three conditions (Section 5.3.4). We decided quickly to drop the third one. The results from Table 6.1 are obtained with a dominance check which compares the reduced costs and the number of breaks. It would be interesting to see what happens if the break comparison is dropped as well.

Dropping the second condition could substantially reduce the computation time, for two reasons. First, since more labels are dominated, less paths are extended. Second, and more interesting, the algorithm can become faster because less information needs to be kept track of. Currently, many pairs of labels need to be compared during the dominance check. If only the reduced cost condition is used, a label for node v needs only to be compared to the lowest reduced cost found so far for node v. If the new label has a higher reduced cost, it can be discarded immediately.

Note that dropping the break check does not imply that the recovery duties do not contain breaks anymore. The feasibility check (Section 5.3.3) ensures that each generated duty has enough breaks.

7.2.6 Vehicle rescheduling

In Chapter 2, we introduced three rescheduling phases: line, vehicle and crew rescheduling. We tested a crew rescheduling method for a tram network. Roelofsen et al. (2018) have developed a framework for rescheduling tram lines from the passenger perspective. Now, rescheduling vehicles is the missing link.

We have seen that the crew rescheduling problem for trams is very similar to that of trains. The vehicle rescheduling problem is also likely to be similar and in the past two decades vehicle rescheduling methods have been developed for trains (Cacchiani et al., 2014). However, we expect that there is one aspect which makes the problem significantly different: the trams in Rotterdam operate in one direction only and can only turn at a limited number of locations.

Bibliography

Abbink, E. J., Mobach, D. G., Fioole, P. J., Kroon, L. G., van der Heijden, E. H., and Wijngaards, N. J. (2009). Actor-agent application for train driver rescheduling. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 513–520. International Foundation for Autonomous Agents and Multiagent Systems.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W., and Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329.

Blais, J.-Y., Lamont, J., and Rousseau, J.-M. (1990). The hastus vehicle and manpower scheduling system at the société de transport de la communauté urbaine de montréal. *Interfaces*, 20(1):26–42.

Bouman, P. C. (2018). Java and CPLEX example. `https://github.com/pcbouman-eur/JavaCplexExample`. [Online; accessed 26-november-2018].

Cacchiani, V., Huisman, D., Kidd, M., Kroon, L., Toth, P., Veelenturf, L., and Wagenaar, J. (2014). An overview of recovery models and algorithms for real-time railway rescheduling. *Transportation Research Part B: Methodological*, 63:15–37.

Clausen, J., Larsen, A., Larsen, J., and Rezanova, N. J. (2010). Disruption management in the airline industry—concepts, models and methods. *Computers & Operations Research*, 37(5):809–821.

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

Jespersen-Groth, J., Potthoff, D., Clausen, J., Huisman, D., Kroon, L., Maróti, G., and Nielsen, M. N. (2009). Disruption management in passenger railway transportation. In *Robust and online large-scale optimization*, pages 399–421. Springer.

Potthoff, D., Huisman, D., and Desaulniers, G. (2010). Column generation with dynamic duty selection for railway crew rescheduling. *Transportation Science*, 44(4):493–505.

Rezanova, N. J. (2009). *The Train Driver Recovery Problem - Solution Method and Decision Support System Framework*. PhD thesis, Technical University of Denmark (DTU).

Rezanova, N. J. and Ryan, D. M. (2010). The train driver recovery problem—a set partitioning based model and solution method. *Computers & Operations Research*, 37(5):845–856.

Roelofsen, D., Cats, O., Oort, N., and Hoogendoorn, S. (2018). Assessing disruption management strategies in rail-bound urban public transport systems from a passenger perspective. In *CASPT 2018, 23-25 July 2018, Brisbane*.

Sato, K. and Fukumura, N. (2011). Real-time freight train driver rescheduling during disruption. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 94(6):1222–1229.

Veelenturf, L. P., Potthoff, D., Huisman, D., and Kroon, L. G. (2012). Railway crew rescheduling with retiming. *Transportation research part C: emerging technologies*, 20(1):95–110.

Verhaegh, T., Huisman, D., Fioole, P.-J., and Vera, J. C. (2017). A heuristic for real-time crew rescheduling during small disruptions. *Public Transport*, 9(1-2):325–342.

Walker, C. G., Snowdon, J. N., and Ryan, D. M. (2005). Simultaneous disruption recovery of a train timetable and crew roster in real time. *Computers & Operations Research*, 32(8):2077–2094.