ERASMUS UNIVERSITY ROTTERDAM
ERASMUS SCHOOL OF ECONOMICS

Integrated Vehicle and Crew Rescheduling - Disruption management of the RET tram network

Master of Science (MSc) Econometrics and Management Science

Master Thesis

Operations Research and Quantitative Logistics

NAME: Lisanne van Huizen
STUDENT NUMBER: 427878

SUPERVISOR EUR:	SUPERVISORS RET:
Dennis Huisman	Cees Boogaard
	Judith Mulder
SECOND ASSESSOR EUR:	
Twan Dollevoet	

August 6, 2020

Abstract

In this thesis, we focus on the disruption management of the RET (Rotterdamse Elektrische Tram) tram network. On a daily basis, disruptions occur which affect the original timetable such that vehicles are not able to continue their route without delay, or even cancellations of parts of their route. As the vehicles are operated by different drivers during the day, such delays or cancellations may propagate towards vehicles and drivers not even affected by the disruption. In the existing literature, this problem is also known as the vehicle and crew rescheduling problem, including delay possibilities (VCRSP).

The goal of this thesis is to find a solution approach to solve the VCRSP in a very short amount of time, such that it can support the traffic controllers of the RET by forming recovery duties for the drivers and recovery routes for the vehicles, which can be used to cover for the disruption. In most literature, the problem is solved sequentially by first establishing a solution for the vehicle rescheduling problem, and then using this solution as fixed timetable to then solve the crew rescheduling problem. This is also done by different groups of students of the Erasmus University, for this particular problem. Therefore, the main goal of this thesis is to find an integrated solution approach which solves the VCRSP.

To quickly solve the integrated VCRSP, we propose a column generation approach for which the columns correspond to feasible recovery routes and duties for the vehicles and drivers, respectively. These routes and duties are generated by solving a (recourse constrained) shortest path problem in combination with a labeling procedure. Also, to improve the computation time, we suggest making a selection of vehicles and drivers for which changes in the original duty are allowed. This is done by introducing a neighbourhood or by setting an end time of the recovery period. Furthermore, we propose a pre-solving method for faster convergence to a feasible solution, such that the total computation time of the solution method will reduce. Finally, a strategy is used to reduce the number of times the integer master problem is solved, which will also improve the computation time. The latter is based on the gap between the objective value of the linear restricted master problem and the overall lower bound of the problem.

We test our solution method by disrupting an original timetable provided by the RET of a normal week day, on two locations in the network: 1) an isolated line (and without interlining) and 2) a location where multiple lines cross (and with crew interlining).

The results indicate that the introduction of the neighbourhood and the recovery period both lead to huge reductions in the size of the instances, which consequently improves the computation time of the solution method. The reduction is more noticeable for instance 1, as less vehicles and drivers are affected by the disruption for this instance compared to instance 2. Furthermore, with the use of the pre-solving method, and all other methods to reduce the total computation time, optimal solutions can be found within seconds for small instances like instance 1, and feasible solutions can be obtained for larger instances like instance 2, within the maximum computation time of 20 minutes. However, if the instance is much larger than instance 2, a feasible solution may not be obtained. Therefore, for further research, we recommend to improve the solution method to its full potential.

In conclusion, improvement can still be made in (the computation time of) the solution method by exploring the recommendations. However, despite the limitations, the solution method can be used to solve small disruptions, and it is able to support the traffic controllers of the RET for large disruptions, affecting many vehicles and drivers.

Contents

Chapter 1

Introduction

RET (*Rotterdamse Elektrische Tram*) is a public transport company that seeks perfectly organized and conducted public transport with the highest quality for the passengers. Public transportation of RET is carried out by tram, bus, metro and ferry in and around the city of Rotterdam, located in the Netherlands. In this thesis, the focus lies on the operation of the RET tram network. RET is responsible for satisfying passengers demand while operational cost is minimized. In order to maintain this for the tram network, vehicles and crew members operate according to a daily timetable in which is specified which and when trips need to be performed. In a perfect situation, every vehicle and crew member shows up at their stops on time. Unfortunately, on the day of operation, disruptions can take place which result in delays or cancellations of trips. The goal of this thesis is to quickly adjust the timetable by rescheduling the vehicles and drivers of the current day, to account for these disruptions, such that delays, cancellations and deviation from the original timetable is minimized.

Disruptions can have many causes such as open bridges, accidents and rail blockage. If a disruption occurs, the traffic control is responsible for rescheduling the vehicles and crew members such that all trams can continue their routes with a driver and changes to the timetable are limited. Some tram vehicles may be unable to continue their planned route and a detour or trip cancellation is needed. Furthermore, the crew duties should still satisfy the labour rules such as the maximum consecutive driving time and minimum number of required breaks during a long duty. At the moment, the adjustments made are based on the experience and creativity of the traffic controller as there is no tool to support them in determining which adjustments to make. However, even small disruptions might affect many vehicle and crew duties, because many duties pass by the same locations and disrupted duties propagate their disruption through the consecutive duties. As a result, a small disruption in the morning might still have a disruptive effect in the timetable in the afternoon.

To overcome this problem, a decision support tool is desired to support the traffic controllers. The tool must be able to quickly adjust the disrupted timetable into a feasible timetable, satisfying the vehicle and labour rules. This problem is referred to as the combined, or integrated, vehicle and crew rescheduling problem (VCRSP). However, because of the complexity of the problem and the necessity to find a feasible solution in a short amount of time, the problem is usually split into two separate problems: the vehicle rescheduling problem (VRSP) and the crew rescheduling problem (CRSP). Sequentially solving the two separate problems instead of the integrated problem reduces the dependency between the two problems, as the solution to one of the problems is assumed to be fixed while solving the other. This means that it is more likely to find a better solution if we use the integrated method instead of the two separate methods. However, in case of disruptions, a quick (feasible) solution is preferred over an optimal solution due to the time trade-off in finding an optimal solution. In other words, an integrated method may need a longer time to find even a feasible solution, while the two separate methods may find solutions of lower quality, but in very short time.

Because prior work of students of the Erasmus University has already led to some promising solution methodologies for the individual vehicle rescheduling and the individual crew rescheduling problem, in this thesis we will focus on finding a good integrated solution methodology. The trade-off, in solution quality and computation time, between the sequential and the integrated solution method can then be compared such that, eventually, the RET may decide which of the two methods can be used in practice.

To solve the integrated vehicle and crew rescheduling problem with delay possibilities, we use a column generation approach in which the columns correspond to recovery routes and duties. This is also done by, among others, van Dockum (2018), van Meer et al. (2019) and Potthoff et al. (2010) for the crew rescheduling problem. We extend their work by including the methods proposed by van Lieshout et al. (2018) for the vehicle rescheduling problem with delay possibilities. The recovery routes and duties are found by solving a (resource constrained) shortest path problem in combination with a labeling procedure. It turned out that the computation time was still too long and therefore, we also propose to use a neighbourhood of vehicles and drivers and a recovery period in which tasks are allowed to be changed. Furthermore, for faster convergence to a feasible solution, we implement a pre-solving method and only solve the integer restricted master problem (IRMP) during certain moments in the solution process, instead of every time a better overall lower bound is found. To compute the overall lower bound of the problem, we use the research performed by Huisman et al. (2005).

We test our solution method by disrupting an original timetable provided by the RET of a normal week day, on two locations in the network: 1) an isolated line (and without interlining) and 2) a location where multiple lines cross (and with crew interlining). We choose these instances such that we are able to compare the performance of the solution method for disruptions taking place on different locations.

The results show that huge improvements can be made in the reduction of the size of the instance by introducing a neighbourhood and a recovery period. This consequently leads to a huge improvement in the computation time. The reduction is more noticeable for small instances. For larger disruptions, all other methods to reduce the computation time are needed to find a feasible solution within a short amount of time, but for very large instances, this may even not be achievable within the maximum set computation time. Furthermore, the results show that changing the penalty cost can change the computation time of the solution method as well as the outcome of the solution method. We do not make any recommendations for which penalty cost to use, as we believe that this should be further discussed with the traffic controllers of the RET, based on their preferences for different recovery timetables. However, despite the limitations (of the computation time of) the solution method, we believe that the solution method can be used to solve small disruptions, and it is able to support the traffic controllers of the RET for large disruptions, affecting many vehicles and drivers.

The remainder of this thesis is organized as follows. In Chapter 2, we provide more insight into the problem by discussing the RET tram network, and the planning process currently applied. Moreover, we describe disruption management and formally state the problem for this thesis. Thereafter, in Chapter 3, we continue by providing an extensive literature review on disruption management. In Chapter 4, we propose a mathematical formulation of the integrated vehicle and crew rescheduling problem for which we discuss the solution method in Chapter 5. The data used to test our solution method is presented in Chapter 6. We also introduce our test instances in this chapter. In Chapter 7, we tune the algorithm for faster computation time while maintaining a feasible solution. The results using the final algorithm settings are presented in Chapter 8. In this chapter, we also perform a sensitivity analysis. Furthermore, we discuss the limitations and the practical use of our solution method in Chapter 9 and make some recommendations for further research in Chapter 10. Finally, in Chapter 11, we state our concluding remarks.

Chapter 2

Problem Description

In this chapter, we provide more information about the problem of this thesis. In short, the problem is to quickly find a feasible and cheap recovery timetable after a disruption has occurred in the RET tram network. The recovery timetable is feasible if it adheres all vehicle and labour rules, and it is cheap if deviation from the original timetable is minimized.

To fully understand the problem, we first provide more insight in the RET tram network, the current planning process and used terminology in Section 2.1. Thereafter in Section 2.2, we explain disruption management and the associated difficulties within this process. In this section we also focus on the underlying problems of disruption management, namely the vehicle rescheduling problem and the crew rescheduling problem. Finally, we continue in Section 2.3 with explicitly stating the goal of this thesis.

2.1 The RET Tram Network

The RET tram network is spread through the city Rotterdam, located in the Netherlands, and its surroundings. A map of this tram network can be found in Figure 2.1. The trams operate at most 11 lines on a day. Note that the map only shows the 9 main lines, as the other lines are only used for specific events or time periods. Table 2.1 states the start and end location for each of the tram lines, as well as the total number of vehicles that operate that line. Also, for each line the vehicle frequency and the time frequency are stated for both during off-peak hours (if the line is used during off-peak hours) and during rush hours. For our research, tram line 12 is not considered as this line is only used if an event takes place at *De Kuip*.

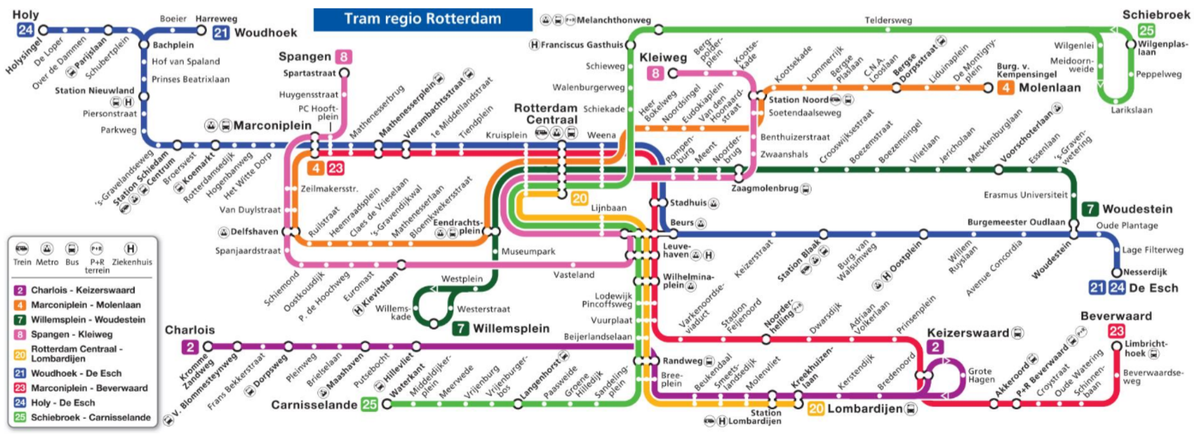


Figure 2.1: Abstract map of the RET tram network

As Table 2.1 shows, tram lines 2, 4, 7, 8, 23, 24 and 25 are operated both during off-peak hours and rush hours, with time frequencies ranging from every 7 to 20 minutes. Tram line 20 is only operated during rush hours on weekdays with a time frequency of every 6 minutes. Tram line 21 is operated every 20 minutes until 20:00 on weekdays and tram line 124 is only operated by 2 trams from 8:30 until 9:00 on weekdays for students of Erasmus University. There is no clear relation between the number of vehicles operating a line per day and the vehicle frequency. However, there is a correlation between the number of vehicles used per line per day and whether or not these vehicles also operate other lines during the same day. So, for example, vehicles operating tram line 2 only operate on this specific line while vehicles operating tram line 23 also operate tram line 20, and vice versa. Because vehicles can operate different lines during the day, a delay on one line can also cause a delay on another line. Moreover, as drivers also operate multiple vehicles of different lines during the day, the disruption can cause problems for vehicles and drivers not even passing the disruption.

At the end of each day, each vehicle returns to its corresponding depot. The RET has two depots where the vehicles are stationed during the night named *Remise Beverwaard* and *Remise Kralingen*. At *Remise Beverwaard*, planned maintenance on the vehicles is performed and thus sometimes vehicles normally stationed at *Remise Kralingen* should go there for repairs. Therefore, it is important that each vehicle returns to its planned end depot for that day.

Table 2.1: Tram lines of the RET during a normal weekday

Tram Line	Start Location	End Location	Vehicles (total per day)	Vehicle Frequency (total per hour)	Time Frequency (every ... min.)
2	Keizerwaard	Charlois	7	3 - 6	10 - 20
4	Molenlaan	Marconiplein	9	3 - 6	10 - 20
7	Woudestein	Willemsplein	8	3 - 6	10 - 20
8	Kleiweg	Spangen	11	3 - 6	10 - 20
20	Lombardijen	Centraal Station	18	9	6
21	de Esch	Woudhoek	19	4	15
23	Beverwaard	Marconiplein	28	4 - 8.5	7 - 15
24	de Esch	Holy	19	4 - 4	15 - 15
25	Carnisselande	Schiebroek	16	4 - 8	7 - 15
124	Centraal Station	de Esch	2	-	11
12*	Centraal Station - Stadion Feyenoord - P+R Beverwaard*				

This table states information about the 11 tram lines of the RET while operating on a normal day. For each line, the start and end location of the line, the total number of vehicles that operate that line, the vehicle frequency and the time frequency for both during off-peak hours (if the line is used) and during rush hours is stated.
* Tram line 12 is only used if an event takes place at *De Kuip*.

Currently, most of the planning process is done with the powerful modular solution tool HASTUS. Other used tools in the planning process are RIDS and Perdis. The planning process, shown as a flowchart in Figure 2.2, consist of six stages.

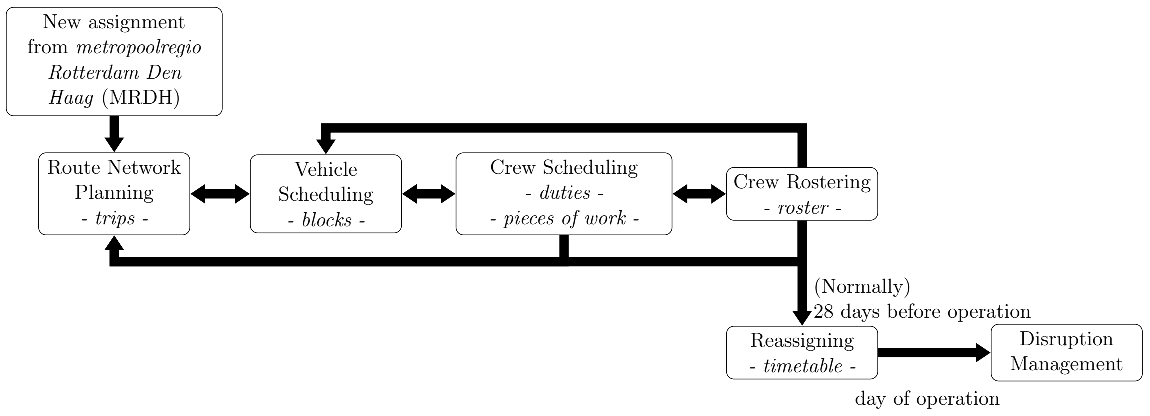

New assignment from *metropoolregio Rotterdam Den Haag* (MRDH)
Route Network Planning - *trips* -
Vehicle Scheduling - *blocks* -
Crew Scheduling - *duties* - - *pieces of work* -
Crew Rostering - *roster* -
(Normally) 28 days before operation
Reassigning - *timetable* -
Disruption Management
day of operation

Figure 2.2: Stages of planning process

The six stages are sequentially executed in the following order: route network planning, vehicle scheduling, crew scheduling, crew rostering, (crew or/and vehicle) reassigning and finally, disruption management. It is possible to adapt previous stages manually during the process, if this leads to improvements (e.g. lower operational cost) later on.

The process starts in the first stage by planning the route network after a new assignment has been given to the RET by *metropoolregio Rotterdam Den Haag* (MRDH). In this stage the tram lines are set up such that total operational cost is minimized while passengers demand is satisfied as well as the restriction on the number of available vehicles. Besides the set-up of the tram lines, also the frequency at which they should operate is determined. This results in many different *trips*, which are defined by a line number, a start and end location, a start and end time and its corresponding stops with departure times. The start time at each stop, which need to be visited during these trips, is presented by a *trip point*. Note that a trip is thus a collection of trip points, which can be split into smaller pieces.

In the second stage, vehicle blocks are made by merging trips from the first stage, and then assigning those vehicle blocks to vehicles using deadhead trips from and to either one of the two depots. By forming the vehicle blocks, some extra time between two trips is added according to the task the vehicle must perform. Such a task may be that the vehicle must turn around or that the vehicle must make an empty reallocation trip to the start location of another trip. It may also be that the vehicle just has to wait a couple of minutes before it is allowed to perform the next trip. An example of such a vehicle block is shown in Figure 2.3. As there are 9 lines during weekdays operating between 3 to even 9 times per hour, many vehicle blocks are formed which all have to be assigned to a vehicle. Note that, at this point, the vehicles blocks are not assigned to a specific vehicle, as it may be that a vehicle is out of service at the day of operation.

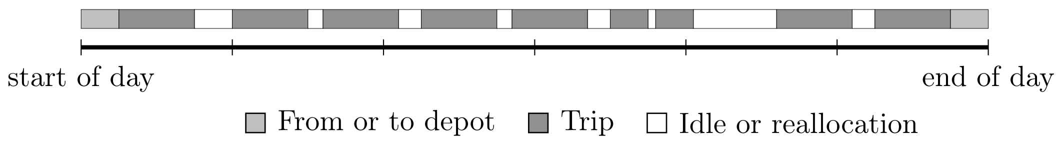


Figure 2.3: Example vehicle block

In the third stage of the planning process, the crew scheduling takes place. In this stage, crew *duties* are made based on the vehicle blocks from the previous stage. This is done by splitting the vehicle blocks into *work pieces* and merging them into feasible crew duties. If a duty has ended or a change of crew member occurs, we say that the crew member currently assigned to the vehicle is relieved. The work pieces specify the part between two reliefs of a crew member. The crew duties must (mutually) cover all vehicle blocks and satisfy all labour rules while again the total operational cost is minimized. The following labour rules must be considered:

1. A minimum number of breaks (b_{min}), with a minimal duration (l^b_{min}), in each duty with a duration of at least (h^{duty}_{min}) hours.
2. The working time between 2 breaks (or the start or the end of a duty/break) may not exceed (h^{work}_{max}) hours.
3. The maximum duty length (in hours), including break time, may not exceed (h^{duty}_{max}) hours.

Besides these labour rules, recovery duties must also start and end at the same locations as the planned duties (which may require a reallocation trip at the beginning or end of the duty). This is convenient for the crew member as he/she probably parks his/her car at this location. Reliefs can only take place at some specific locations, which are the two depots, *Rotterdam Centraal Station*, *Oostplein*, *Wilhelminaplein*, *P+R Beverwaard* and all end locations of the 11 lines (for reference, see Figure 2.1). Also breaks may only take place at those locations except for at the depots.

Up to recent years, one duty could only consist of work pieces of one line at the time. However, RET found that combining work pieces of different lines for one duty had great advantages and thus *crew interlining* was introduced. The advantages of crew interlining are that crew members could be scheduled more efficiently which results in less unnecessary break time (waiting until next piece of work can be done). Unfortunately, by introducing crew interlining, disruption management becomes more complicated because (a) disruption(s) on a single line or for one crew member can propagate to other lines and duties of other crew members. In the remainder of this thesis, we refer to *crew interlining* simply by *interlining.*

Besides interlining, the RET also uses *split duties.* A split duty is formed by combining two short duties. Because the time between the two duties is far more than the required meal break time, this time is not seen as meal break time and thus not part of the duty (and duty length). Therefore, the labour rules still hold as the split duty can be seen as two separate duties. Note however that crew members often not prefer to have a split duty, because for them it still feels as a very long working day. Therefore, the RET prefers a normal duty over a split duty and tries to minimize the number of split duties for the drivers.

In the fourth stage, crew members are assigned to the crew duties, which is called crew rostering. At the RET, a *roster* is not made for each crew member separately, but a *cycling roster* is introduced. This means that for a certain number of weeks a roster, satisfying the labour rules, is made and all crew members get assigned to different weeks at which his/her roster begins. So, for example, crew member A starts at the first week in the roster with working 3 days in a row after which he/she has 2 days off, then has yet another working day and he/she ends his/her week with another day off. Crew member B starts at the same time but has to start at week two of the roster. According to the roster, she/he has to work two days, is thereafter one day off, then has to work another 2 days and finishes with two days off. During the next week, crew member A begins at week two of the roster, which has been done by crew member B the previous week and crew member B starts at the third week of the roster. This way of rostering turns out to be very convenient as crew members know there working days far in advance.
At some days, a crew member is scheduled as *stand by*, this means that in case another crew member calls in sick, she/he can be used to cover his/her duty. In this stage of the planning process, the labour rules state the minimum required rest time between duties and split duties, and how often and for how long a free period must occur in the roster for a crew member. After a feasible and cost-minimizing roster has been made, every crew member gets assigned to a specific week at which his/her roster starts, and this information is given to all crew members.

The stages discussed so far are performed a few weeks ahead, while the next stage lasts until the day of operation. In this stage, the reassigning of vehicles, duties and crew members takes place. It may for example be that crew members are sick, or they want some days (or weeks) off. It can also happen that some vehicles are out of operation for a longer time or that parts of tram tracks are under maintenance. In all cases, the roster must be adjusted to account for these events. At the RET this is mostly done manually.

Finally, we arrive at the last stage of the planning process, which is the main topic of this research. This stage is called disruption management. Up to this point, at the start of the day of operation, a feasible cost-efficient roster for the day has been made, all vehicles are assigned to blocks of trips, all pieces of work now form duties and all duties are assigned to crew members. We refer to this one-day roster as the *timetable* of that day. Unfortunately, on most days it will however turn out this timetable cannot be executed as originally planned due to disruptions. To account for this, the traffic controllers of the RET manually try to adjust the original timetable to a recovery timetable once a disruption occurs. As our thesis focuses on improving this stage of the planning process, we explain this stage in further detail in the next section.

2.2 Disruption Management

In this section we discuss disruption management. By a disruption we mean an event (or series of events) that makes the current planning infeasible. In the tram network, those events can have multiple causes, such as open bridges, accidents and rail blockages. If a disruption occurs, the planning must be adjusted such that vehicles can continue their routes with a driver while changes to the timetable are minimized.

By making the recovery timetable feasible, some trips or trip points may get delayed, detours are needed for the vehicles and in some cases trips, or parts thereof, must even be cancelled. In all those cases, inconveniences arise for the crew members, as breaks may need to be rescheduled and/or they may need to work overtime, and for the passengers, as they may experience delay. It is therefore important that penalties for those actions ensure that the recovery timetable has minimal changes with respect to the original timetable while the crew duties should still satisfy the labour rules. Unfortunately, even small disruptions might affect many vehicle and crew duties, because many duties pass by the same locations and disrupted duties propagate their disruption through the following duties.

Because disruption management is complex, it is most often considered as two separate problems which are sequentially solved. We refer to these two separate problems as the vehicle rescheduling problem (VRSP) and the crew rescheduling problem (CRSP). In the next two subsections, the individual problems will be separately described. Note that combined, this problem is called the vehicle and crew rescheduling problem (VCRSP). In this thesis, we focus on quickly solving the VCRSP.

2.2.1 Vehicle Rescheduling

The VRSP is extensively researched during the past decades, and within most literature, the objective is to minimize the total operating and delay cost of the network once a disruption occurs. For our research this means that the recovery timetable must have minimal deviations compared to the original timetable. Besides the definitions given in the previous section, we introduce some new definitions and notation, based on Li et al. (2007b), to describe the VRSP.

We refer to a disrupted trip by a *cut trip.* The point in the cut trip where the disruptions occurs is referred to as the *breakdown point.* Furthermore, we refer to the trip operated by a vehicle during a disruption as the *current trip* of the vehicle. This trip can be an *in-service trip*, which is a trip with (possibly) passengers in the vehicle, or a *deadhead trip*, which means that no passengers are allowed in the vehicle. Lastly, we say that trip i is *compatible* with trip j if the start time of trip j is later than the end time of trip i plus the travel time from i to j.

The problem can be defined as follows. Given the timetable of trips for vehicles (with their corresponding start and ending times and locations), the travel time between locations, the list of relief locations and the occurring disruption (specified by start time, duration and affected locations or track segments), quickly find a feasible minimal-deviating recovery timetable such that each vehicle operates a feasible sequence of trips. In order to do so, trips may be delayed or cancelled. Furthermore, the following vehicle regulations must be satisfied:

1. Vehicles must start and end at their planned location.
2. Vehicles may arrive at the depot at most o^V_{max} minutes later than planned.
3. Minimum times for detours, turn-arounds and possible other vehicle related tasks need to be considered.
4. Vehicles cannot pass each other at stops or between every pair of stops. So, for two vehicles A and B with arrivals A_a and B_a and departures A_d and B_d must hold $A_a < B_a \implies A_d < B_d$. This must also hold for vehicles sharing a track between two stops.

5. Vehicles cannot drive over blocked track segments.
6. Vehicles can only use the track segments given in the data set (if not blocked) or the detours.
7. There are no additional vehicles available.
8. Reliefs (changing driver or the start/end of the route of the vehicle) can only take place at the relief locations.

2.2.2 Crew Rescheduling

For CRSP, crew members are assigned to recovery duties. These recovery duties must be as close as possible to the original duties and the following crew regulations must be satisfied:

1. Adjusted crew duties can at most be o^D_{max} minutes longer than planned and cannot start earlier than planned.
2. Break and duty rules need to be satisfied (same as rules 1 and 2 for crew scheduling stage).
3. Adjusted crew duties must start at the same location as planned.
4. Adjusted crew duties may end at a different location as planned, however in this case a travel piece has to be included, for a duration of T_{travel} minutes, at the end of the duty such that the duty still ends at the planned end location. The duty including this travel piece cannot be more than o^D_{max} minutes longer than planned.
5. There are no additional crew members available.
6. Reliefs (changing vehicle, or having a break, or the start/end of the duty) can only take place at the relief locations.

2.3 Problem Statement

The goal of this thesis is to improve the disruption management stage of the RET tram network. At RET, the traffic controllers are responsible for adjusting the vehicle blocks and crew duties in such a way that all trams can continue their routes with a driver and changes to the timetable are limited. The adjustments made are based on the experience and creativity of the traffic controller as there is no tool to support them in determining which adjustments to make. As has become clear from the previous sections, this is very complex and therefore there is enough reason to improve on this stage of the planning process.

In order to so, a decision support tool must be developed to support the traffic controllers. This tool should have a user interface which asks the traffic controller if a disruption occurs, and if this is the case, when, where and for how long the disruption will occur. Besides the user interface, the decision support tool must of course be able to come up with a recovery timetable. The requirements for this are that the recovery timetable should be obtained within reasonable time and with minimal deviation from the original timetable. Also, the recovery timetable must implement the vehicle and crew regulations which especially hold for the disruption management stage, as stated in the previous section. Thus, the main goal of the thesis is to find a solution approach to quickly solve the integrated vehicle and crew rescheduling problem (VCRSP).

A good quality solution within reasonable time means that if a disruption only lasts for 20 minutes, the new recovery timetable should be obtained within this time window. If a disruption lasts longer, it is still required that a new recovery timetable is found within at most 20 minutes, because otherwise, disrupted trams (and crew members) may not be able to continue their routes and get stuck on track segments. This is not only inconvenient for the vehicles and the crew members, but especially for the passengers using the tram. Note that besides quickly finding a solution, the recovery timetable must have minimal changes compared to the original timetable because all adjustments have to be reported to the crew members. The more changes the longer it takes to implement the new timetable as more crew members must be told about their new duties.

Chapter 3

Disruption Management: Literature Review

In the past decades, the research and the applications in the area of disruption management has increased as transportation problems are becoming more integrated and thereby complex. Disruption management is needed in a wide range of logistic processes such as public transport (with airplanes, trains, metros, trams, taxi's or buses), but also within inventory control, production management, machine scheduling etc. A framework, models and applications of disruption management are provided by Yu and Qi (2004). The objective of most of these models is to quickly recover to the original planned schedule, such that operations can run smoothly thereafter. As the focus of our thesis lies on public transportation, we only discuss the literature of disruption management conducted on this area. Within this area, disruption management is also referred to as the vehicle and/or crew rescheduling problem.

Compared to other forms of transportation, such as airplanes and trains, not much research has been conducted on disruption management of tram networks. However, many methodologies and solution approaches, especially those for train, metro and bus networks, can still be used for tram networks by making some modifications. For example, Kiefer et al. (2016) state that their proposed mixed integer linear programming model can be used for an arbitrary number of transport modes such as subway, tram and bus. Worth mentioning is however that they omit the crew rescheduling from their method, and from the results it is not clear how fast the solutions are or can be obtained. Because the network of airplanes differs much from the network of trams (much longer travel time between stops, less possibility to reallocate a driver, no blocked track segments etc.), the disruption management strategy developed for airlines can most often not simply be adapted to be used for trams, in contrast to those developed for trains and busses. An overview of models and solution approaches for disruption management of the train network is given by Cacchiani et al. (2014) and of the tram network by Lückerath et al. (2013).

Because many research in the area of disruption management of transportation modes only focuses on one aspect of the vehicle and crew rescheduling problem, we discuss the research conducted on the vehicle rescheduling problem (VRSP) in Section 3.1 and the research conducted on the crew rescheduling problem (CRSP) in Section 3.2. Thereafter, we discuss the research conducted on the integrated solution approaches for the vehicle and crew rescheduling problem (VCRSP) in Section 3.3. Finally, in Section 3.4, we provide a summary of the literature used in our thesis.

3.1 Vehicle Rescheduling

In the area of disruption management of transportation modes, the VRSP is the most researched aspect. The VRSP is closely related to the dynamic vehicle scheduling problem (see Huisman et al. (2004)). Most solution approaches for the latter start with an initial feasible solution, which is thereafter improved until a certain stopping criterion is met and can therefore also be used for solving the VRSP. The most often used stopping criteria are a limitation on the solving time or on the number of iterations of the solution approach, and a relaxation on the optimality gap of the solution approach (e.g. stop if optimality gap is less than $x\%$ instead of 0%).

An advantage of the dynamic solution approach is that most often, the computation time is not too long, which is also required for the solution approach of the VRSP. In the literature, various extension of the VRSP are considered: with or without the possibility to retime (delay) the vehicles, the possibility to cancel trips, the use of stochastic travel times, the inclusion of uncertainty about the disruption length, et cetera. Besides that, also different objectives for the problem exist, such as maximizing the customer service by highly penalizing every minute of delay, maximizing other service measures (e.g. vehicle frequency at stops), or minimizing the cancellation of trips. In other papers, the objective is to stick to the original planning as much as possible, such that each task that is not performed by the planned vehicle but another one, is heavily penalized. Fortunately, the majority of the solution approaches for those different objectives can be adjusted or extended to account for the problem that needs to be solved.

Most experiments to test the solution approaches of the VRSP are performed for the train network, as for example done by Meng and Zhou (2014). They develop an innovative integer programming model for the problem, by introducing network flow variables and solving the model using a procedure based on Lagrangian relaxation. Unfortunately, obtaining the results for larger instances does require a lot of time. They suggest incorporating parallel computing techniques into the proposed solution algorithms to speed up the solution process. The advantages of this are shown by Bettinelli et al. (2017), who are able to obtain excellent quality solutions within 2 seconds of computation time with their proposed solution approach. Furthermore, Dávid and Krész (2017) propose two mathematical models and a recursive and a local search algorithm to solve the VRSP. The results for bus instances are, besides using the algorithms, also obtained exactly such that the solution quality of the algorithms can be evaluated. They found that the algorithm was able to find good quality solutions while having a low running time. Also Li et al. (2007a), Li et al. (2007b) and Li et al. (2009) propose different solution methods, such as a Lagrangian heuristic, to solve the problem. The benefit of their methods is that they can be used for all transportation modes.

Finally, van Lieshout et al. (2018) discuss the vehicle rescheduling problem with retiming. Retiming means that the departure of the vehicle may be delayed, which leads to more scheduling flexibility. They suggest to expand the network by adding copies of trips, each with different starting times, presenting the delay possibilities. The results, obtained within 3 minutes by using Lagrangian relaxation and an iterative neighborhood exploration heuristic, show that the number of cancelled trips can be reduced significantly if delays are allowed.

3.2 Crew Rescheduling

The other aspect of disruption management is the CRSP. Not much research has been conducted solely on the CRSP as most disruptions also cause infeasibility within the vehicle schedule, which leads to an integrated or sequential vehicle and crew rescheduling problem. For the CRSP, it is assumed that the vehicle schedule is feasible, and that only the crew duties have to change. Among others, Potthoff et al. (2010), Potthoff (2010) and Veelenturf et al. (2012) propose different models and solution methods for this problem. All their research has been conducted for the train network but can be reformulated or remodeled to account for the tram network.

Potthoff et al. (2010) present an algorithm based on column generation techniques combined with Lagrangian heuristics. They start with a core problem of tractable size, but when tasks remain uncovered, a neighborhood exploration is performed to improve the solution. Good quality results can be obtained by the algorithm within minutes for real-life instances. Potthoff (2010) provides extensive research on the CRSP. The contribution of his thesis is twofold, he first extends the CRSP with the possibility of retiming. Furthermore, he also considers the uncertainty of the duration of a disruption in the disrupted situation. Mathematical formulations are given for both versions of the CRSP, as well as solution approaches to solve the problem. Veelenturf et al. (2012) also allow for retiming and they use the same solution approach as Potthoff (2010).

3.3 Integrated Methodologies & Solution Approaches

The VCRSP is very closely related to the integrated and dynamic vehicle and crew scheduling problem. For the latter, Huisman (2004) shows that integrated approaches can lead to significant improvements over sequential approaches. In our research, we intend to prove the same within the framework of disruption management, which is disregarded by most researchers. Namely, up to this point, most research has been conducted solely on the VRSP and/or the CRSP instead of the integrated VCRSP, especially for tram networks. Fortunately, in the last years, research in this area has increased.

Dávid and Balogh (2016) propose an algorithmic framework for the bus network, that guides the solution process regardless of the used method. This framework is very useful, also for other transportation modes, and can easily be implemented. The algorithmic framework is tested using recursive and local search heuristic methods provided by Dávid and Krész (2014). The results show short running times for both heuristics. Also Walker et al. (2005) are able to find good (even optimal) solutions within a short amount of time for their proposed methods. They provide a mathematical formulation and solve the problem using branch and bound with column and constraint generation. The average time to solve instances consisting of 36 trains and 564 work pieces is 50 seconds, which suggest that their method could be used in practice. More recently, Malucelli and Tresoldi (2019) conducted a case study for delay and disruption management in local public transportation via real-time vehicle and crew rescheduling. They propose a simulation based optimization system which incorporates retiming and vehicle and driver rescheduling.

In most literature, the VCRSP is formulated as an extended version of the set partitioning problem. The major issue with solving the integrated VCRSP is the increase in the solution time compared to solving only the VRSP or the CRSP. This is because more possibilities exist for creating alternative vehicle routes and crew duties, especially if we also include retiming. Therefore, the computation time of solving the VCRSP will be much higher than the computation time of solution methods for solving the VRSP and the CRSP. Elhallaoui et al. (2008) propose bi-dynamic constraint aggregation and subproblem reduction to deal with this problem. The main idea of their method is to cluster tasks from which one expects that they are performed by the same vehicle and driver, such that the number of variables and constraints decrease. During the solution method, one may split these clusters into smaller clusters if this leads or could lead to a better solution.

3.4 Summary used Literature

In this section we discuss the literature used for developing our mathematical formulation and solution approach. Inspiration is taken from all papers discussed in the previous sections, however, we mostly rely on the formulation given by van Lieshout et al. (2018) for the vehicle rescheduling problem with retiming.

This model is extended using previous research conducted by Timo van Dockum, another student of the Erasmus University, who focused on the crew rescheduling problem. His research, which can be found in van Dockum (2018), is very important for our research, as his methods are also developed especially for the RET. Besides his research, also 5 groups of students of the Erasmus University have proposed mathematical formulations and solution approaches for the RET disruption management problem. Most of them however did not focus on an integrated approach (vehicle and crew rescheduling) but a sequential approach. Nonetheless, our solution method has been inspired by theirs. The formulations, solution methods and results of the 5 groups can be found in Abouelrous et al. (2019), Kunst et al. (2019), Blokland et al. (2019), van de Pol et al. (2019) and van Meer et al. (2019), however, the most usable paper among these is that of van Meer et al. (2019).

For our research, we intend to extend the proposed mathematical formulation given by van Lieshout et al. (2018) to also include crew members. To solve this extended problem, we can use an exact approach, however due to the complexity, this will probably not provide a solution within a short amount of time. Therefore, as also done by van Dockum (2018) and van Meer et al. (2019), we could use column generation to solve the problem. During the solution approach, it may be convenient to use the research conducted by Elhallaoui et al. (2008). By this we mean that, initially, we do not allow all possibilities (e.g. not all possible retiming options or detours) such that solutions can be found faster than is the case if all possibilities are initially included. Once tasks remain uncovered for a certain number of iterations, we may include new possibilities. Also, other smart ways of decreasing the initial possibilities exist, such as an iterative neighborhood search heuristic which only allows tasks to be delayed if they remain uncovered for a certain number of iterations, or if tasks are in the neighborhood of such cancelled tasks. Moreover, we could also exclude vehicles and drivers for which we do not expect any changes compared to their original route or duty.

Chapter 4

Mathematical Problem Formulation

In this chapter, we provide a mathematical formulation of the integrated vehicle and crew rescheduling problem. First, in Section 4.1 we describe how we create the network used for the mathematical formulation. We explain how a task list is generated and how the vehicle network and the driver network is constructed. Moreover, we explain how to incorporate retiming of tasks into the formulation. Lastly, in Section 4.2, we provide the mathematical formulation for the VCRSP.

4.1 Creating the Network

Once a disruption has occurred, the original timetable becomes infeasible. Some tasks may need to be cancelled or delayed, and in some cases, a detour for the vehicle is needed. In this section, we first describe how we generate a new list of tasks consisting of trips or sets of trip points, which need to be rescheduled due to the disruption. Thereafter, we describe the underlying vehicle and driver networks. These networks specify which vehicles and drivers are able to execute tasks from the new list (and in which order), while taking into account the (labour) rules specified in Chapter 2. Finally, we explain how to incorporate delay possibilities into the resulting network.

4.1.1 Task List

As explained earlier, we distinguish between trips with and without passengers by *in-service* trips and *deadhead* trips, respectively. These trips consists of multiple trip points, which specify the departure time and location of each stop. A driver starts a trip at a relief location, performs the sequence of corresponding trip points and ends again at another relief location. During the trip, the driver may also pass other relief locations and locations at which the vehicle can take a detour.

In case a disruption occurs, some of the trips may not be feasible anymore. We refer to these trips as *cut trips*. More precise, a trip is a cut trip if the disruption occurs at one or more of the track segments during the time the vehicle and driver need to use these segments. In order to reschedule the trips, it is necessary to split the cut trips into feasible *subtrips*, such that a recovery route for the vehicle and a recovery duty for the driver can be established. We split the cut trips by relief locations and by locations at which a detour can be taken. All other trips are also split by relief locations, but only by detour locations if these provide new connections within the networks. The latter means that, for example, if the cut trips are split by detour locations A, B and C, we only split the other trips by detour locations which can be taken to locations A, B and C or from locations A, B and C.

If the cut trip is a deadhead trip, a detour can be taken such that the driver and vehicle still arrive on time at the first location of the next trip, or at another stop of the cut trip. If a detour can be taken, but the driver and the vehicle do not arrive on time at the location, the remaining part of the cut trip must be assigned to another driver and vehicle, or the trip must be delayed.

In extreme cases, a trip must be cancelled. Cancelled (parts of) deadhead trips do not matter as much as those of in-service trips because the former does not affect the passengers. Therefore, no penalties are incurred for such a cancellation.

If a disruption occurs at one (or multiple) track(s) of an in-service trip, we have to consider the passengers currently present in the vehicle. Again, it may be possible to take a detour such that trip points, which are not executed because of the detour, are cancelled. In this case, passengers waiting at the stops of the cancelled trip points are not served anymore. This is very inconvenient for the passengers, because due to this, some of them may not be able to reach their end destination. Another possibility is to wait until the disruption is over, which may lead to delay for the remaining part of the trip or a reassignment of the remaining trip to another driver and vehicle.

Important to note is that only the in-service (sub)trips need to be executed, while deadhead (sub)trips and detours are optional. Therefore, the penalties for delays and cancellations are only taken into account for the in-service (sub)trips. We denote all the trips in the recovery period (so the (sub)trips and all other sorts of trips) by $i \in N$ with $N = \{1, 2, \ldots, n\}$. From this point forward, we refer to this set as the *task list.* Each *task* in the task list is defined by a start and end time, start and end location, original vehicle and driver, line number, whether or not the task is an in-service trip, and set of trip points corresponding to the task. Note that the task list can contain all tasks until the end of the day such that more flexibility exists in creating the recovery timetable. However, if the computation time is too long because too many tasks are included, one may decide to only include tasks which start up to h hours after the disruption has ended such that after that time, the original timetable has to be followed.

4.1.2 Vehicle Network

In this section, we explain how the vehicle network is created. We present the set of vehicles by $V = \{1, \ldots, v\}$, with v the number of available vehicles. For each vehicle, we define a start node specifying the time at which the vehicle becomes available, and its corresponding location. We let s^v present the start node of vehicle $v \in V$, and we let $S^V = \cup_{v \in V} s^v$. Furthermore, we define an end node for each vehicle, specifying the latest allowed arrival time at the planned end location in the original timetable of the vehicle, and the corresponding end location. As stated in Chapter 2, the latest allowed arrival time is o^V_{max} minutes later than the planned arrival time of the vehicle at its end location. We let t^v present the end node of vehicle $v \in V$, and we let $T^V = \cup_{v \in V} t^v$.

The set of arcs can be obtained by connecting the start and end nodes to the nodes given in the task list N as follows:

1. For each $s^v \in S^V$, draw an arc from s^v to $i \in N$ if the start node of vehicle $v \in V$ is compatible with task i.
2. For each $i \in N$, draw an arc from i to $j \in N$ if task i is compatible with task j.
3. For each $i \in N$, draw an arc from i to $t^v \in T^V$ if task i is compatible with the end node of vehicle $v \in V$.
4. For each vehicle $v \in V$, connect its start node s^v with its end node t^v.

Node i is said to be *compatible* with node j if the start time of node j (st_j) is later than the end time of node i (et_i) plus the travel time from node i to node j (tt_{ij}), such that holds that

$$st_j = et_i + tt_{ij} + sl_{ij}. \tag{4.1}$$

In this formula, sl_{ij} denotes the slack time between node i and j. If the slack time is negative, the two nodes are not compatible. If the slack time is positive, we have to make sure that $sl_{ij} \leq w_i^{max}$, with w_i^{max} the maximum allowed waiting time at node i. If holds that $l^b_{min} \leq sl_{ij} \leq w_i^{max}$, and node i is a relief location, it is possible to schedule a break for the crew member at node i.

The maximum allowed waiting time at node i is determined based on the location of node i and the time until the next vehicle arrives at node i. If it is possible to use a side track at the location, such that other vehicles can pass by, without delay, no maximum allowed waiting time has to be taken into account. However, it may be convenient to still choose a w^{MAX}, such that $w_i^{max} \leq w^{MAX}$ for all $i \in N$, to ensure that vehicles and drivers do not remain idle for a long period and the vehicle network remains small.

We represent the set of arcs over which the vehicles can travel by

$$A = \{(i, j) : i \text{ compatible with } j,\ i \in S^V \cup N,\ j \in N \cup T^V\}.$$

The vehicle network without retiming is given by $G = (\mathcal{V}, A)$, where $\mathcal{V} = S^V \cup N \cup T^V$. Now that vehicle network is created, we can find all feasible paths within the network for each vehicle, corresponding to feasible recovery routes for that vehicle. We present the set of recovery routes for vehicle $v \in V$ by $\mathcal{R}^v$.

4.1.3 Driver Network

The network for the drivers can be created in a similar way as done for the vehicles. We denote the drivers by set $D = \{1, \dots, d\}$ with d the number of available drivers. In a similar manner as done for the vehicles, we denote s^d as the start node of driver $d \in D$, and the set of all driver start nodes by $S^D = \cup_{d \in D}\ s^d$. Besides a start time and location, we also add the number of hours the driver has worked and the number of breaks the driver has had up to the start time of the start node, to the information of the start node. Furthermore, we denote by t^d the end node of driver $d \in D$. Again, the latest allowed arrival time at this node is o^D_{max} minutes later than the planned arrival time of the driver at its end location. We let $T^D = \cup_{d \in D}\ t^d$ denote the set of all driver end nodes.

If one decides to use a recovery period of h hours after the disruption in which the reassignment of tasks may take place, and after which the original timetable should be executed again, it is necessary to keep track of more information about the duty after the end node of the driver at the end of the recovery period. Namely, the labour rules hold for the entire duty and not only for the recovery period. Therefore, we need to know the driving time, duty length and number of breaks after the recovery period for the specific driver. Then, by creating the network, and thus the recovery duties, we need to take this information into account in the same manner as we do this for the start node of the driver. Note that in this case, the end node may not be the end location of the duty and thus no overtime is allowed for the driver at the end node in such case.

Besides drivers with a normal duty, we also have drivers with a split duty. We let $D^s \subseteq D$ denote the set of drivers with a split duty, who are still performing their first part of the split duty (the drivers with a split duty performing the second part of their duty are considered as drivers without a split duty). For those drivers in D^s, the same regulations apply as for the drivers with a normal duty. This means that we do allow overtime of o^D_{max} minutes for the first piece of the duty of a driver with a split duty. The reason for this is that it may happen that the driver just cannot arrive at the end location of the first piece of its split duty on time, because of the disruption. However, by allowing for overtime after the first piece, the driver may not have sufficient rest time between the two pieces of the duty, which may result in a violation of the labour rules.

In order to account for this, we create a rest node for every driver with a split duty, denoted by r^d for $d \in D^s$. This node presents the end location and latest arrival time at the end location corresponding to the first part of the split duty of driver $d \in D^s$. Even though it is allowed to have overtime at node r^d and node t^d, the combined overtime may still not exceed the maximum allowed overtime (o^D_{max} minutes).

We denote the set of all rest nodes by R. Note that if we set a recovery period in which we may change the original timetable (and thereafter the original timetable must be executed), instead of allowing changes in the timetable until the end of the day, most of the rest nodes turn into start or end nodes. If this is the case, we consider the duty of the driver as a normal duty, and not as a split duty.

Given the list of tasks N and the start and end nodes of the drivers, we can for each driver obtain a set of feasible recovery duties. In order to do so, a network has to be created for each driver separately, in which feasible paths over the arcs correspond to feasible duties. This can be obtained by checking whether or not performing a task, after performing the tasks done so far, does 1) not result in violation of the maximum consecutive working time and 2) not result in violation of the minimum number of required breaks.

For each driver $d \in D \setminus D^s$, the following arcs are included:

1. An arc from s^d to $i \in N$ if the start node of driver d is compatible with task i, driver d is currently not operating a vehicle, and performing task i does not result in a violation of the labour rules.
2. An arc from s^d to $s^v \in S^V$ if driver d is able to operate vehicle $v \in V$.
3. An arc from $i \in N$ to $j \in N$ if task i is compatible with task j, and performing task j after task i does not result in a violation of the labour rules. If the end location of task i is a possible detour location and not a relief location, the driver is not allowed to change vehicle nor to have a break.
4. An arc from $i \in N$ to t^d if the end node of driver d is compatible with task i.
5. An arc from t^v to t^d, if driver d is able to drive vehicle v to its end location.
6. An arc from s^d to t^d.

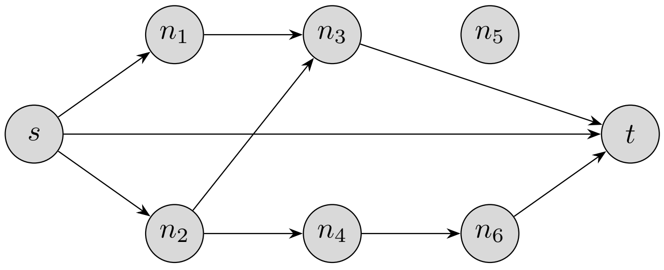


Figure 4.1: Network for single driver (without a split duty)

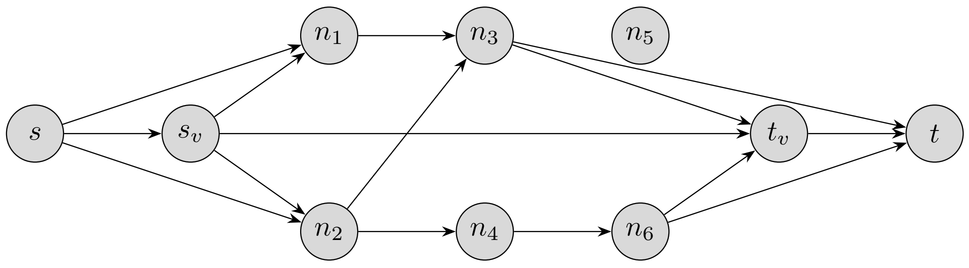


Figure 4.2: Network for single driver (without a split duty, incl. vehicle pick up/delivery)

Figures 4.1 and 4.2 show two examples of created networks, based on above rules, for a single driver without a split duty. Note that it is not necessarily the case that both a start and a end node of a **vehicle** is included in the network. This is because a driver may have a break at the beginning of the recovery period, or because the driver finishes his/her duty before the end of the recovery period, while the vehicle does not end at this location.

For drivers $d \in D^s$, the following arcs are included:

1. An arc from s^d to $i \in N$ if the start node of driver d is compatible with task i, driver d is currently not operating a vehicle, the end time of task i is not later than the latest arrival time at node $r^d \in R$, and performing task i does not result in a violation of the labour rules.
2. An arc from s^d to $s^v \in S^V$ if driver d currently operating vehicle $v \in V$.
3. An arc from r^d to $i \in N$ if the rest node of driver d is compatible with task i, and performing task i does not result in a violation of the labour rules.
4. An arc from $i \in N$ to $j \in N$ if task i is compatible with task j, tasks i and j both end before r^d or start after r^d, and performing task j after task i does not result in a violation of the labour rules. If the end location of task i is a possible detour location and not a relief location, the driver is not allowed to change vehicle nor to have a break.
5. An arc from t^v to t^d and/or r^d, if driver d is able to drive vehicle v to its end location.
6. An arc from $i \in N$ to r^d or to t^d if the rest node or the end node of driver d is compatible with task i.
7. An arc from s^v to r^d and from r^d to t^d.

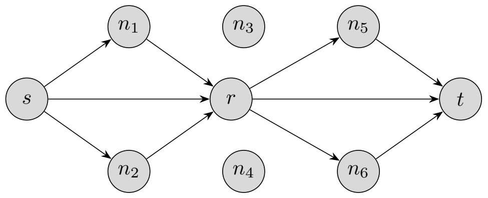


Figure 4.3: Network for single driver (with a split duty)

Figure 4.3 shows an example of a created network, based on the above rules, for a single driver with a split duty. Note that it may be the case that also start and end nodes of vehicles are included in this network, as also shown in Figure 4.2. As can be seen by the structure of the network, a split duty can easily be seen as two separate duties, with the rest node functioning as either a start or an end node.

Finally, for each driver we remove all nodes that cannot be reached, such that only feasible paths, from s^d (to r^d) to t^d, remain in the network. For example, tasks 2 and 4 will be removed from the list of reachable tasks for the driver of the network shown in Figure 4.3. Note that also in this case two nodes, i and j, are said to be compatible if the start time of node j is later than the end time of node i plus the travel time from node i to node j. For the travel time from a rest node r^d to other nodes, we include a minimum required rest time such that labour rules are not violated. We present the set of arcs over which driver $d \in D$ can travel by

$$A^d = \{(i,j) : i \text{ compatible with } j,\ i \in s^d \cup S^V \cup N \cup T^V,\ j \in S^V \cup N \cup T^V \cup t^d\}.$$

The driver network, for driver $d \in D$, without retiming is given by $G^d = (\mathcal{V}^d, A^d)$, where $\mathcal{V}^d = s^d \cup S^V \cup N \cup T^V \cup t^d$. Now that a driver network is established for each driver, we can find all feasible paths within the network, which corresponds to feasible recovery duties. We represent the set of recovery duties for driver $d \in D$ by $\mathcal{R}^d$.

It should be noted that it may not be possible to find a recovery network for each vehicle and driver, affected by the disruption. For example, if a driver is scheduled to have a break shortly after 12:00 due to reaching its maximum driving time, and a disruption of 30 minutes occurs at 11:50 which affects this driver, the driver almost immediately violates its maximum driving time, even if a break can be scheduled later. In practice, this situation could also occur and there is no other way than to violate this labour rule. Therefore, in our solution approach, the labour rule of maximum driving time does not apply for those drivers for which this situation holds.

4.1.4 Introducing Retiming

Up to this point, tasks are performed on time or cancelled. Here we explain how to incorporate retiming into our model. For this, we heavily rely on the notation, description and formulation given by van Lieshout et al. (2018). Note that we only allow for full minutes of delay (1 minute, 2 minutes, etc.), because the original timetable is given in minutes as well.

To expand all networks to include retiming possibilities, we first introduce a set W containing all nodes in the network. This list of nodes is then sorted on starting time such that the first node in W has the earliest starting time and the last node in W the latest starting time. By constructing the list W in this manner, each node only has to be considered once. For each node $i \in W$, we let q_i^{max} denote the maximum allowed delay for this node.

We start the process by considering the first node in W, let this node be node k. For this node, we consider all other nodes in W which are compatible with node k, but only if we introduce a delay of $1 \leq q \leq q_k^{max}$ minutes, and we store these nodes in a list W^*. Note that if node k is compatible with node $i \in W$ without a delay, this node is already added to the network, so we can skip this possibility. If node k is compatible with node $i \in W^*$, with a delay of q minutes, we add a copy of node k to the network and denote it by n_k^q. If this node (delay possibility) has already been added to the network, we do not add it again, such that each delay possibility is only added once. If all nodes in W^* are considered, we remove node k from the list W, and add the delayed nodes to the list W. We also empty list W^*. Finally, W is resorted and we restart the process again by consider the first node in W. The process ends if there are no nodes left in W. As stated in van Lieshout et al. (2018), by introducing delay opportunities in this manner, the network is only expanded if it leads to new possibilities such that the network does not become too large to solve.

We let the vehicle network with delay possibilities be denoted by $G_e = (\mathcal{V}_e,\ A_e)$, and all tasks (delayed and not) by the set N_e. All recovery routes for vehicle v are still denoted by $\mathcal{R}^v$. Note that we may remove task nodes (from set N_e) which can **only** be done by a vehicle **or only** be done by a driver, as for each task both a vehicle and (at least) one driver are required to perform the task. Furthermore, we let $N_e(k)$ be the set of delayed and not delayed nodes corresponding to task $k \in N$. The set of delay arcs associated with task $k \in N$ is defined by $A_e(k) = \{(i,j) | i \in N_e(k) \text{ and } (i,j) \in A_e\}$. The driver network, for driver $d \in D$, with retiming is given by $G_e^d = (\mathcal{V}_e^d, A_e^d)$, where $\mathcal{V}_e^d = s^d \cup S^V \cup N_e \cup T^V \cup t^d$. Now that a driver network is established for each driver, we can find all feasible paths within the network, which corresponds to feasible recovery duties. We present the set of recovery duties for driver $d \in D$ still by $\mathcal{R}^d$.

4.2 Mathematical Formulation

In this section, we provide the mathematical formulation of the integrated vehicle and crew rescheduling problem with retiming. We use the information given in the previous sections, and as stated earlier, we heavily rely on the formulations given by van Lieshout et al. (2018), van Dockum (2018) and van Meer et al. (2019).

In order to formulate the VCRSP, we define the following sets:

- N set of all tasks consisting of cut trips, and all other, not affected, trips, both splitted by relief locations and locations at which detour can be taken,
- N_e set of all tasks including delay possibilities,
- $N_e(k)$ set of all delay copies of task $k \in N$,
- A_e set of all arcs over which vehicles and/or drivers can operate,
- $A_e(k)$ set of all delay arcs associated with task $k \in N$,

- D set of all drivers,
- $\mathcal{R}^d$ set of all recovery duties of driver $d \in D$.
- V set of all vehicles,
- $\mathcal{R}^v$ set of all recovery routes of vehicle $v \in V$.

Furthermore, we define the following parameters:

- For all tasks $i \in N$, we let
  - $Q_i \in \mathbb{N}$, number of scheduled stops at which the vehicle must stop in task i.
- For all recovery duties $\delta \in \mathcal{R}^d$ of driver $d \in D$, we let
  - $a^d_{i\delta} = \begin{cases} 1, & \text{if task } i \in N_e \text{ is included in recovery duty } \delta \in \mathcal{R}^d, \\ 0, & \text{otherwise.} \end{cases}$.
  - $b^d_{i\delta} = \begin{cases} 1, & \text{if task } i \in N \text{ is included in recovery duty } \delta \in \mathcal{R}^d, \\ 0, & \text{otherwise.} \end{cases}$.
- For all recovery routes $\delta \in \mathcal{R}^v$ of vehicle $v \in V$, we let
  - $a^v_{i\delta} = \begin{cases} 1, & \text{if task } i \in N_e \text{ is included in recovery route } \delta \in \mathcal{R}^v, \\ 0, & \text{otherwise.} \end{cases}$.
  - $b^v_{i\delta} = \begin{cases} 1, & \text{if task } i \in N \text{ is included in recovery route } \delta \in \mathcal{R}^v, \\ 0, & \text{otherwise.} \end{cases}$.

We define the following variables:

- For all tasks $i \in N$, we let
  - $z_i = \begin{cases} 1, & \text{if task } i \text{ is cancelled,} \\ 0, & \text{otherwise.} \end{cases}$.
- For all drivers $d \in D$, we let
  - $y^d_\delta = \begin{cases} 1, & \text{if recovery duty } \delta \in \mathcal{R}^d \text{ is chosen as recovery duty for driver } d, \\ 0, & \text{otherwise.} \end{cases}$.
  - $e^D_d = \begin{cases} 1, & \text{if for driver } d \text{ no recovery duty can be found,} \\ 0, & \text{otherwise.} \end{cases}$.
- For all vehicles $v \in V$, we let
  - $x^v_\delta = \begin{cases} 1, & \text{if recovery route } \delta \in \mathcal{R}^v \text{ is chosen as recovery route for vehicle } v, \\ 0, & \text{otherwise.} \end{cases}$.
  - $e^V_v = \begin{cases} 1, & \text{if for vehicle } v \text{ no recovery route can be found,} \\ 0, & \text{otherwise.} \end{cases}$.

To take into account passengers, vehicles and crew members, we use task, vehicle and crew penalties by creating the recovery timetable such that changes to the original timetable are minimized. In case no feasible route or duty can be found for a vehicle or driver respectively, we set the penalty cost to a large integer M. The task cancellation penalty is p^1 for each missed scheduled stop within the task (but only if the task is an in-service trip). A penalty of p^M is incurred for every minute delayed for each scheduled stop. Furthermore, a vehicle penalty of p_W^V is thus incurred for every new piece of work compared to the original route of the vehicle. The crew penalties are p_W^D for every new piece of work compared to the original duty and p^O for every minute of overtime compared to the original planned duty end time. We let f_δ^d be the penalty cost of recovery duty δ for driver d. That is,

$$f_\delta^d = p_W^D w_\delta^d + p^O o_\delta^d, \qquad \forall \delta \in \mathcal{R}^d,\ \forall d \in D, \tag{4.2}$$

with w_δ^d the number of new pieces of work compared to the original duty and o_δ^d the number of minutes of overtime compared to the original planned end time of the duty. We let g_δ^v be the penalty cost of recovery route δ for vehicle v, which can be computed in a similar way as done for the recovery duties of the drivers (but without overtime), by

$$g_\delta^v = p_W^V w_\delta^v + p^M tot_\delta^{delay}, \qquad \forall \delta \in \mathcal{R}^v,\ \forall v \in V, \tag{4.3}$$

with w_δ^v the number of new tasks compared to the original route of the vehicle and tot_δ^{delay} the total delay in minutes over all tasks in the recovery route, compared to the original planned starting times of the tasks.

Note that the task delay penalty of p^M per minute is taken into account on the arcs of the vehicles. Note also that, because we know the original timetable, we can include the penalty costs on the arcs of the duty graph/route graph for each driver/vehicle as well. Namely, if an arc from i to j is taken, a penalty is incurred if task j was not part of the original duty/route, and 0 otherwise. To account for the overtime, we assign to each arc connected to the end node of driver d a penalty of p^O times the number of minutes of overtime. To account for delay, we assign to each arc connected to a task with delay, a penalty of p^M times the number of minutes delay. If a feasible path from the start node to the end node of the driver/vehicle is found, we can compute the total cost of the recovery duty/route by adding all penalty costs over the arcs.

The mathematical formulation of the VCRSP including retiming is given by:

$$\text{min.} \sum_{k \in N} p^1 Q_k z_k + \sum_{v \in V}(M e_v^V + \sum_{\delta \in \mathcal{R}^v} g_\delta^v x_\delta^v) + \sum_{d \in D}(M e_d^D + \sum_{\delta \in \mathcal{R}^d} f_\delta^d y_\delta^d) \tag{4.4}$$

subject to

$$\sum_{v \in V} \sum_{\delta \in \mathcal{R}^v} b_{k\delta}^v x_\delta^v + z_k \geq 1 \qquad \forall k \in N \tag{4.5}$$

$$|D|(1 - z_k) - \sum_{d \in D} \sum_{\delta \in \mathcal{R}^d} b_{k\delta}^d y_\delta^d \geq 0 \qquad \forall k \in N \tag{4.6}$$

$$\sum_{d \in D} \sum_{\delta \in \mathcal{R}^d} a_{i\delta}^d y_\delta^d - \sum_{v \in V} \sum_{\delta \in \mathcal{R}^v} a_{i\delta}^v x_\delta^v \geq 0 \qquad \forall i \in N_e \tag{4.7}$$

$$|D| \sum_{v \in V} \sum_{\delta \in \mathcal{R}^v} a_{i\delta}^v x_\delta^v - \sum_{d \in D} \sum_{\delta \in \mathcal{R}^d} a_{i\delta}^d y_\delta^d \geq 0 \qquad \forall i \in N_e \tag{4.8}$$

$$1 - \sum_{v \in V} \sum_{\delta \in \mathcal{R}^v} a_{i\delta}^v x_\delta^v \geq 0 \qquad \forall i \in N_e \tag{4.9}$$

$$\sum_{\delta \in \mathcal{R}^v} x_\delta^v + e_v^V = 1 \qquad \forall v \in V \qquad (4.10)$$

$$\sum_{\delta \in \mathcal{R}^d} y_\delta^d + e_d^D = 1 \qquad \forall d \in D \qquad (4.11)$$

$$z_k \in \mathbb{B} \qquad \forall k \in N \qquad (4.12)$$

$$e_v^V \in \mathbb{B} \qquad \forall v \in V \qquad (4.13)$$

$$x_\delta^v \in \mathbb{B} \qquad \forall \delta \in \mathcal{R}^v,\ \forall v \in V \qquad (4.14)$$

$$e_d^D \in \mathbb{B} \qquad \forall d \in D \qquad (4.15)$$

$$y_\delta^d \in \mathbb{B} \qquad \forall \delta \in \mathcal{R}^d,\ \forall d \in D \qquad (4.16)$$

The objective function (4.4) minimizes the total deviation from the original timetable by minimizing the penalty cost of missing one stop, plus the penalty cost for the recovery route (including penalties for delay) of all vehicles and plus the penalty cost for the recovery duties of all drivers. The latter is based on the penalty costs for every new piece of work compared to the original duty and for every minute of overtime compared to the original planned duty end time.

Constraints (4.5) make sure that each (delayed) task is either cancelled or executed by a vehicle. Note that multiple vehicles may perform the same task, as the delayed versions of this task are scheduled on different moments in time. Constraints (4.6) ensure that if a task is cancelled, no drivers can be assigned to the task.

Constraints (4.7) make sure that each (delayed) task is executed by more drivers than vehicles. Note that this constraint is necessary as the two previous constraints do not take into account different delay possibilities of tasks. For example, by only using the previous two constraints, it could happen that a driver is assigned to task k with 2 minutes delay, while the vehicle which should be operated by the driver is assigned to task k without any delay. This is not allowed as the driver and the vehicle both must be assigned to the same task with the same amount of delay. The same holds for constraints (4.8), which ensure that if a (delayed) task is not executed by a vehicle, it can also not be executed by a driver. Constraints (4.9) make sure that each (delayed) task is executed by at most one vehicle. This is necessary as vehicles cannot be operating the same track at the same time.

Furthermore, constraints (4.10) and (4.11) make sure that each driver and each vehicle have exactly one recovery route or duty, if a feasible route or duty can be found. Lastly, the restrictions on the variables are given by constraints (4.12) - (4.16).

Chapter 5

Solution Approach

In this chapter we discuss the solution approach of creating a recovery timetable based on a given disruption in the original timetable. The goal is to find an optimal solution (recovery timetable) with the least number of cancelled or delayed trips, while at the same time, the deviation from the original timetable must also be minimized. However, due to the complexity of the problem, it is not realistic to achieve an optimal recovery timetable (in reasonable time). The complexity of the problem is due to the fact that very many recovery duties and routes can be found for both drivers and vehicles. As we also include delay possibilities into the task list, the number of possible routes and duties increases even more. Creating all feasible routes and duties, and then solving the problem to optimality, will take too much time. Therefore, we propose an alternative approach to solve the vehicle and crew rescheduling problem with retiming.

In the next section, we provide an outline of our solution approach. Thereafter, we explain each step of the solution approach in detail in the Sections 5.2 up to 5.5. Finally, in Section 5.6, we discuss different methods to improve the computation time of the solution method.

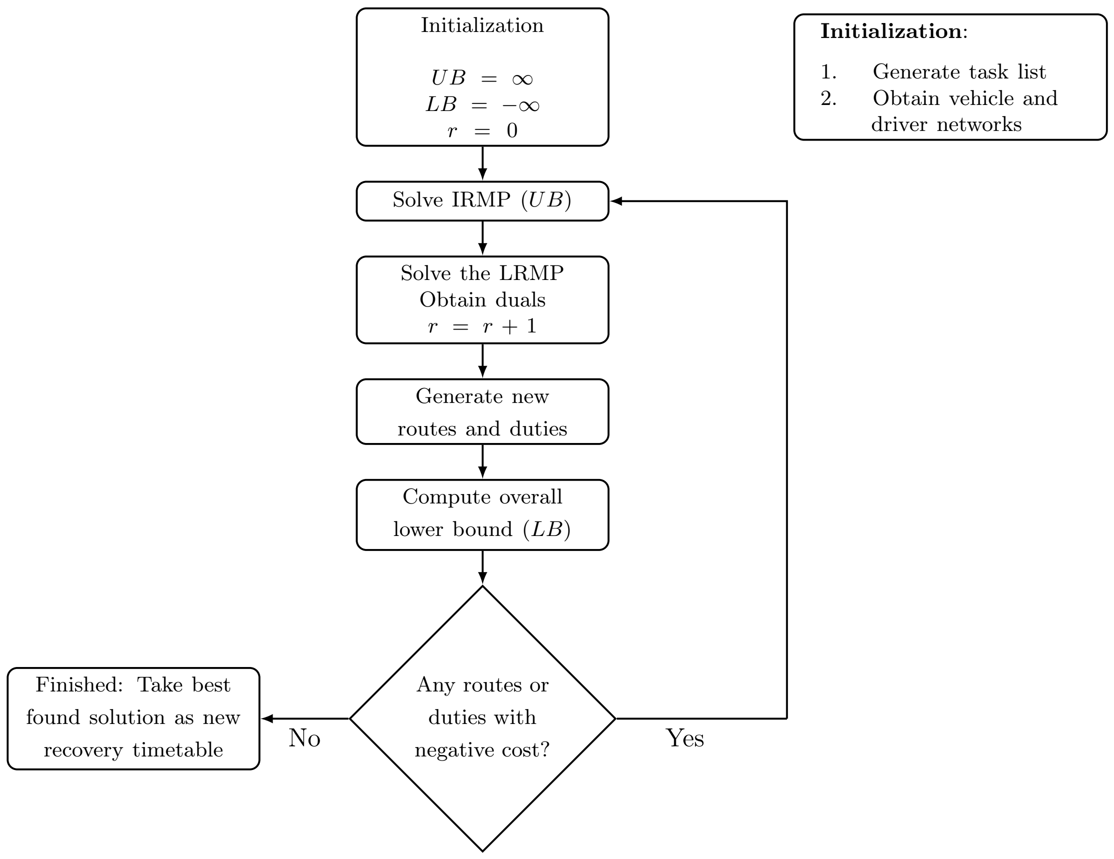

Initialization
$UB = \infty$
$LB = -\infty$
$r = 0$
Initialization:
1. Generate task list
2. Obtain vehicle and driver networks
Solve IRMP (UB)
Solve the LRMP
Obtain duals
$r = r + 1$
Generate new routes and duties
Compute overall lower bound (LB)
Any routes or duties with negative cost?
Yes
No
Finished: Take best found solution as new recovery timetable

Figure 5.1: Flowchart of the solution process

5.1 Outline

In most literature, the VCRSP is solved sequentially by first establishing a solution (timetable) for the vehicle rescheduling problem and thereafter solving the crew rescheduling problem using the fixed timetable obtained from the first step. This has as disadvantage that the solution from the vehicle rescheduling problem, may be infeasible for the crew rescheduling problem, or that better solutions may exist. To overcome this disadvantage, we propose an integrated solution approach to solve the VCRSP. Note that previous research conducted by students from the Erasmus University (see papers van Dockum (2018), Abouelrous et al. (2019), Kunst et al. (2019), Blokland et al. (2019), van de Pol et al. (2019) and van Meer et al. (2019)) already propose (different) sequential approaches. By combining methods from their research, the integrated approach can easily be developed. The flowchart shown in Figure 5.1 presents the process of our integrated solution method to solve the VCRSP. In this section, we only discuss each step briefly while in the following sections each step will be discussed in more detail.

We start the solution method by initializing the process. In the initialization step, the task list is generated, delayed tasks and possible detours are introduced and also the vehicle networks and driver networks are set up based on the task list for each of the vehicles and drivers. Furthermore, we also initialize the parameters needed for the solution approach.

After initializing the process, we solve the integer restricted master problem (IRMP) of the VCRSP. This provides us with a new upper bound for the overall problem. If the upper bound is better than the upper bound found so far, we update the best found upper bound to this upper bound. The formulation of the restricted master problem (RMP) will be discussed in Section 5.3. Once we solve the RMP, not all feasible recovery routes and recovery duties are included for each of the vehicles and drivers, as generating all these feasible recovery routes and duties will take too much time. This is also the reason why this version of the VCRSP is called ‘restricted’: not all the possibilities are added. Note that generating all feasible recovery routes and duties is not done in the initialization step, but only recovery **networks** are created from which these recovery routes and duties can be obtained. Therefore, we can only solve a restricted version of the VCRSP and not the complete version.

In addition of solving the IRMP, we solve the linear restricted master problem (LRMP). The solution of the LRMP can be interpreted as follows. For those tasks (partly) cancelled or delayed in the optimal solution of the RMP, we can find a better solution by adding new recovery routes or duties for those vehicles and drivers having these tasks in their recovery network.

After solving the LRMP, we check if we are able to create new recovery routes and duties by solving the pricing problem. By solving the pricing problem, we try to find new recovery routes for the vehicles and recovery duties for the drivers, covering tasks that were (partly) cancelled or delayed in the solution obtained from solving the IRMP. In other words, we try to find recovery routes and duties with negative reduced cost. The process of solving the pricing problem will be further explained in Section 5.4.

The next step of the solution method is to compute the overall lower bound of the problem. This is done by combining the solution found for the LRMP and the solution found for the pricing problem. How we compute the overall lower bound will be discussed in Section 5.5.

If any new recovery routes or recovery duties are found by solving the pricing problem, we add them to the current set of recovery routes and duties if they are not already present in the RMP, and we move on to the next step in the solution process by again solving the IRMP. If there are no new recovery routes or recovery duties with negative reduced cost, we have finished the solution process and we take the best found solution as new recovery timetable.

5.2 Initialization

Before we are able to start the solution process, we first have to set up the task list, as explained in Section 4.1.1, based on the given disruption in the current timetable. After creating the task list, we determine the start and end locations of the vehicles and drivers for the recovery period. The recovery period is defined from the start of the disruption up to a certain moment after the disruption. The latter can be until the end of the day, however in Section 5.6.3, we will also discuss other methods to set the end time of the recovery period.

Besides the start and end nodes, we also determine the original route and duty of each vehicle and driver by assigning each task in the task list to its original vehicle and driver. Thereafter, we add detour possibilities, and also delay possibilities by applying the algorithm explained in Section 4.1.4, such that only delay possibilities of tasks are added to the task list if these delay possibilities create new connections within the networks of vehicles and/or drivers. This is because, it is unnecessary to add a delay possibility of a task if this task is connected to the same tasks as the task without delay possibility. In this manner, we only expand the network if this may lead to better solutions for the VCRSP.

Now that all tasks, possible delays and detours are known, we set up the networks for each vehicle and for each driver, such that each task in the network of a vehicle or a driver can be reached by the vehicle or the driver (meaning that a path from the source to this task to the sink exists). Note that we may remove delay tasks which can not be performed by any vehicle or driver because each task needs to be assigned to at least one vehicle and one driver before it can be executed.

5.3 Restricted Master Problem

In the best scenario, we could solve the VCRSP as formulated by equations (4.4) - (4.16) to optimality by first generating all feasible recovery routes and duties. Unfortunately, the number of feasible routes and duties within the recovery period is extremely large. Even if it is possible to generate all possibilities, it will still not be possible to obtain an optimal solution from these possibilities in reasonable time. Among others, van Meer et al. (2019) and van Dockum (2018) propose column generation to overcome this problem. The idea of column generation is to solve a restricted version of the VCRSP, in which only a subset of all possible recovery routes and duties are added. Let $K^v \subset \mathcal{R}^v$ and $K^d \subset \mathcal{R}^d$ denote all routes and duties available at the time we need to compute either the integer version or the linear relaxation version of the restricted master problem. The integer restricted master problem (IRMP) is formulated by

$$\text{min.} \sum_{k \in N} p^1 q_k z_k + \sum_{v \in V} (M e_v^V + \sum_{\delta \in K^v} g_\delta^v x_\delta^v) + \sum_{d \in D} (M e_d^D + \sum_{\delta \in K^d} f_\delta^d y_\delta^d) \tag{5.1}$$

subject to

$$\sum_{v \in V} \sum_{\delta \in K^v} b_{k\delta}^v x_\delta^v + z_k \geq 1 \qquad \forall k \in N \tag{5.2}$$

$$|D|(1 - z_k) - \sum_{d \in D} \sum_{\delta \in K^d} b_{k\delta}^d y_\delta^d \geq 0 \qquad \forall k \in N \tag{5.3}$$

$$\sum_{d \in D} \sum_{\delta \in K^d} a_{i\delta}^d y_\delta^d - \sum_{v \in V} \sum_{\delta \in K^v} a_{i\delta}^v x_\delta^v \geq 0 \qquad \forall i \in N_e \tag{5.4}$$

$$|D| \sum_{v \in V} \sum_{\delta \in K^v} a_{i\delta}^v x_\delta^v - \sum_{d \in D} \sum_{\delta \in K^d} a_{i\delta}^d y_\delta^d \geq 0 \qquad \forall i \in N_e \tag{5.5}$$

$$1 - \sum_{v \in V} \sum_{\delta \in K^v} a_{i\delta}^v x_\delta^v \geq 0 \qquad \forall i \in N_e \tag{5.6}$$

$$\sum_{\delta \in K^v} x_\delta^v + e_v^V = 1 \qquad \forall v \in V \tag{5.7}$$

$$\sum_{\delta \in K^d} y_\delta^d + e_d^D = 1 \qquad \forall d \in D \tag{5.8}$$

$$z_k \in \mathbb{B} \qquad \forall k \in N \tag{5.9}$$

$$e_v^V \in \mathbb{B} \qquad \forall v \in V \tag{5.10}$$

$$x_\delta^v \in \mathbb{B} \qquad \forall \delta \in K^v,\ \forall v \in V \tag{5.11}$$

$$e_d^D \in \mathbb{B} \qquad \forall d \in D \tag{5.12}$$

$$y_\delta^d \in \mathbb{B} \qquad \forall \delta \in K^d,\ \forall d \in D. \tag{5.13}$$

The linear restricted master problem (LRMP) is formulated by the same objective function and constraints (5.2) - (5.8), and adds the constraints

$$z_k \geq 0 \qquad \forall k \in N, \tag{5.14}$$

$$e_v^V \geq 0 \qquad \forall v \in V, \tag{5.15}$$

$$x_\delta^v \geq 0 \qquad \forall \delta \in \mathcal{K}^v,\ \forall v \in V, \tag{5.16}$$

$$e_d^D \geq 0 \qquad \forall d \in D, \tag{5.17}$$

$$y_\delta^d \geq 0 \qquad \forall \delta \in \mathcal{K}^d,\ \forall d \in D. \tag{5.18}$$

We let the objective value obtained from solving the LRMP be denoted by LB_{LRMP}. A first feasible solution for the LRMP would be that all tasks are left unassigned. Finding good recovery routes and duties, using the dual values obtained from solving the LRMP to optimality, is called the pricing problem.

5.4 Pricing Problem

The pricing problem is used to find new recovery routes for vehicles and recovery duties for drivers. For this, we need the dual values obtained from solving the LRMP to optimality using the current sets of duties and routes. We let the dual variables be λ_k, for all $k \in N$, ϕ_k, for all $k \in N$, μ_i, for all $i \in N_e$, γ_i, for all $i \in N_e$, and α_i, for all $i \in N_e$, for each of the constraints (5.2), (5.3), (5.4), (5.5) and (5.6), respectively. Furthermore, we let π^v for all $v \in V$ and π^d for all $d \in D$ be the dual variables for constraints (5.7) and (5.8), respectively. Using these dual variables, we can compute the reduced cost of a recovery route $\delta \in \mathcal{R}^v$ for vehicle $v \in V$ by

$$g_\delta^v - \sum_{k \in N} \lambda_k b_{k\delta}^v + \sum_{i \in N_e} \mu_i a_{i\delta}^v - |D| \sum_{i \in N_e} \gamma_i a_{i\delta}^v + \sum_{i \in N_e} \alpha_i a_{i\delta}^v - \pi^v, \tag{5.19}$$

and the reduced cost of a recovery duty $\delta \in \mathcal{R}^d$ for driver $d \in D$ can be computed by

$$f_\delta^d + \sum_{k \in N} \phi_k b_{k\delta}^d - \sum_{i \in N_e} \mu_i a_{i\delta}^d + \sum_{i \in N_e} \gamma_i a_{i\delta}^d - \pi^d. \tag{5.20}$$

The pricing problem then corresponds to finding recovery routes and recovery duties for each vehicle and driver, with (the most) negative reduced cost. Note that for each vehicle and each driver, the pricing problem can be solved independently from the other vehicles and drivers. By putting the penalty cost and the dual values on the arcs, the pricing problems can be solved by solving a shortest path problem from the source of the vehicle/driver to their sink.

The most used algorithm to solve the shortest path problem in a weighted graph is *Dijkstra's algorithm.* The algorithm finds the shortest path between all nodes in the network. Thus in our case, we would use this algorithm to find the shortest path between the start and end node of the vehicle/driver.

In order to find this path, each node in the network of the vehicle/driver is visited. For the vehicles, the path always corresponds to a feasible route as no restrictions are given. However, for the drivers, not all paths from the start node to any other node in the network, will correspond to a feasible duty. This is due to the labour rules specified in Section 2.2.2.

To overcome this problem, Potthoff et al. (2010) and van Dockum (2018) propose using a resource constrained shortest path problem. The resource constrained shortest path problem basically extends *Dijkstra's algorithm* by not only considering the costs on the arcs, but also the resources consumed on that arc. The resources corresponding to an arc from task i to task j, are whether or not a break can be taken if these tasks are executed directly after each other, the driving time between these tasks and the duty length between these tasks. Note that the driving time and the duty length are equal if no break is possible between these two tasks. Also note that the driving time and the duty length include the duration of task j. Using this algorithm, we find the shortest feasible path(s) between the start node (s) and end node (t) of a driver. To be able to fully account for all labour rules, we let the start node contain information about the number of breaks before the start node of the driver, the duty length and the driving time. In a similar way, we let the end node contain information about the number of breaks, duty length and driving time after the end node of the driver.

Solving the Pricing Problem

In the existing literature, different rules exist for adding new routes and duties to the RMP after solving the pricing problem. One may add all routes and duties with negative reduced cost found by solving the (resource constrained) shortest path problem. However, we could also only add the routes and duties with the most negative reduced cost. Finding all paths with negative reduced cost in each iteration will take too much time. However, adding only the route or duty with the most negative reduced cost will most definitely result in an increase of the total computation time of the solution process. This is because a slightly lower reduced cost route or duty may turn out to be the route or duty with the most negative reduced cost in (one of) the next iterations. To overcome this issue, Potthoff et al. (2010) and van Dockum (2018) propose using a labeling procedure in combination with the (resource constrained) shortest path problem.

The labeling procedure extends the (resource constrained) shortest path problem by not only considering the driving time, number of breaks and duty length on the arcs, but also the reduced cost. At each node i (except for the sink of the vehicle), we check for all paths leading to node i which of these paths dominates the other paths. For example, let p_1 and p_2 be two paths leading to node i, with reduced cost r_1 and r_2 respectively. If it holds that $r_1 < r_2$, then path p_1 dominates p_2, and we will no longer consider path p_2 as possible path for reaching node i, as it is better to follow path p_1. As we visit each node in topological order, the last node to be considered will be the sink node of the vehicle or driver.

For the sink node of the vehicle, we do not perform any dominance checks on the paths leading to the sink node, such that we back-trace the paths with negative reduced cost at the sink node, which together form the set of all found recovery paths during this particular iteration of the solution process. For the driver we do perform a dominance check at the sink node, because otherwise too many recovery duties are added in each iteration. This is because there are far more possibilities to form new recovery duties than recovery routes. Also, because only the routes of the vehicles take into account the delay penalty, many duties of the drivers, which do differ from driving time, still have the same cost (and reduced cost) as the same tasks are performed but not the same delayed versions of the tasks. The latter does also apply for each node i in the network of the drivers, and not only for the sink nodes of the drivers. To take this into account, we further extend the labeling procedure by considering the driving time and the number of breaks up to node i for the drivers, besides the reduced cost.

It thus may be the case for the drivers, that the path with the lowest reduced cost up to task i is not necessarily better than another path with higher (or the same) reduced cost but with more breaks, or with the same or more breaks and a higher total driving time. At node i we are not sure if continuing with the dominated path up to node i still leads to the lowest reduced cost over all possible paths. Namely, all paths leading to node i could have negative reduced cost if they reach the sink node such that all of them are promising. By using the labeling procedure discussed so far, we would only consider the path with the most negative reduced cost up to node i. However, continuing this particular path to the sink node may not be possible as it may turn out that before reaching node i a break was necessary to make the path from the source of the driver, to the sink of the driver, feasible. This would imply that it may be the case that we do not find the recovery duty with the lowest reduced cost, because we did not further pursue other possibilities.

Another possibility we would like to pursue is a path leading to node i with the same or more breaks, but a higher driving time than another path leading to node i. Namely, the number of breaks of a path could be the same for path p_1 and p_2, and p_1 has slightly lower reduced cost, but if p_2 does visit more stops, we still want to pursue this possibility. Namely, if the paths have the same breaks but differ from driving time, this would mean that the driver is unnecessary idle. To make sure that we always find the best recovery duty for each driver, we use the following dominance rule once we solve the pricing problem for the drivers: *Path p_1 dominates path p_2 if path p_1 has lower reduced cost* ***and*** *path p_2 does not have more breaks than path p_1* ***and*** *path p_2 does not have less breaks than path p_1 while the total driving time of p_2 is higher than the total driving time of p_1.*

5.5 Overall Lower Bound

In our solution method, not all feasible routes and duties are known once we solve the IRMP and the LRMP. Because of this, we are not able to use the objective value of the LRMP (LB_{LRMP}) as overall lower bound for the problem. It is possible that better routes and duties may be found in the next iterations of the solution method, which results in a lower objective value than currently found for the LRMP. This thus implies that the objective value obtained from solving the LRMP is only a lower bound to the IRMP of the current iteration. Fortunately, Huisman et al. (2005) propose a method to deal with this issue. We use their ideas in our solution method to compute an overall lower bound for the problem.

The reason for using an overall lower bound is because of the necessity to obtain a recovery timetable in a short amount of time. If we are able to compare the upper bound with an overall lower bound, we may terminate the solution method if the difference is small (e.g. less than 1%). The LRMP would provide an overall lower bound if all routes and duties are known and included in the sets $\mathcal{R}^v$ and $\mathcal{R}^d$, respectively. Let $K^v \subset \mathcal{R}^v$ and $K^d \subset \mathcal{R}^d$ denote all routes and duties available at the time we need to compute a lower bound for the overall problem. Assume for now that all routes and duties with negative reduced cost are known. If we denote the set of routes with negative reduced costs in $\mathcal{R}^v \setminus K^v$ by $\hat{R}^v$, it holds that,

$$\min\{0, \min_{\delta\in\hat{R}^v}\{g^v_\delta - \sum_{k\in N}\lambda_i b^v_{k\delta} + \sum_{i\in N_e}\mu_i a^v_{i\delta} - |D|\sum_{i\in N_e}\gamma_i a^v_{i\delta} + \sum_{i\in N_e}\alpha_i a^v_{i\delta} - \pi^v\}\}$$

is the maximum improvement for vehicle v achievable in the next iteration of the solution method, and if we denote the set of duties with negative reduced cost in $\mathcal{R}^d \setminus K^d$ by $\hat{R}^d$, it holds that,

$$\min\{0, \min_{\delta\in\hat{R}^d}\{f^d_\delta + \sum_{k\in N}\phi_i b^d_{k\delta} - \sum_{i\in N_e}\mu_i a^d_{i\delta} + \sum_{i\in N_e}\gamma_i a^d_{i\delta} - \pi^d\}\}$$

is the maximum improvement for driver d achievable in the next iteration of the solution method.

The above two problems are thus basically to find for each vehicle and driver the recovery route or duty with the most negative reduced cost, over all routes and duties not yet added to the master problem. Note that finding the route or duty with the most negative reduced cost, over all routes and duties not yet included in the master problem, is exactly the same as solving the pricing problems. If we add the expected improvement on the objective value which still can be made by adding more routes or duties in next iterations, to the lower bound of the current iteration, we can obtain an overall lower bound to the problem. If the expected improvement is greater than or equal to zero, no more routes or duties can be found in next iterations for this particular vehicle or driver, which improve the overall objective value. The overall lower bound (LB) of the problem can thus be computed by

$$\begin{aligned} LB = LB_{LRMP} + \sum_{v \in V} \min\{0, \min_{\delta \in \hat{R}^v} \{g^v_\delta - \sum_{k \in N} \lambda_i b^v_{k\delta} + \sum_{i \in N_e} \mu_i a^v_{i\delta} - |D| \sum_{i \in N_e} \gamma_i a^v_{i\delta} + \sum_{i \in N_e} \alpha_i a^v_{i\delta} - \pi^v\}\} \\ + \sum_{d \in D} \min\{0, \min_{\delta \in \hat{R}^d} \{f^d_\delta + \sum_{k \in N} \phi_i b^d_{k\delta} - \sum_{i \in N_e} \mu_i a^d_{i\delta} + \sum_{i \in N_e} \gamma_i a^d_{i\delta} - \pi^d\}\}. \end{aligned}$$

Note that this means that: $LB \leq LB_{LRMP} \leq UB$.

5.6 Improving Computation Time

Up to this point, we have discussed the solution method for obtaining a recovery timetable based on the disruption of the original timetable. For the vehicle and crew rescheduling problem, it is not only important to obtain a feasible recovery timetable, but also to obtain a feasible recovery timetable in a very short amount of time. This is necessary due to the fact that the changes made in the recovery timetable have to be known to the drivers before they are able to adapt their duties. Because the solution method discussed so far may not be able to find solutions very quickly, we propose different methods to improve the computation time in this section.

In the initialization of our solution process, no recovery routes or duties are added for the vehicles and the drivers, but only networks from which these routes and duties can be obtained. Therefore, it can take a very long time before even a feasible solution is found, in which each driver has a recovery duty and each vehicle a recovery route. To improve the computation time, we propose a pre-solving method. After the pre-solving method, we continue with the solution process as has been discussed in the previous sections.

Another way to improve the computation time, is to make smart use of the fact that the recovery timetable may also not deviate too much from the original timetable. This can be done by excluding those parts of the original timetable for which no changes are needed or expected to be needed. Note that this means that we may not find the optimal recovery timetable for all kinds of disruptions, as not all possibilities are included. We can exclude parts of the original timetable by, for example, introducing a *neighbourhood of vehicles and drivers*. A neighbourhood of vehicles and drivers only contains those vehicles and drivers for which we expect that changes in their route or duty must be made in order to obtain a feasible recovery timetable. If we expect no changes, we can exclude them from the list of vehicles and drivers for which changes in the original timetable are allowed. Another possibility to reduce the computation time is to choose *recovery period* such that the size of the instance is reduced. A recovery period is defined such that only within this period, changes to the original timetable are allowed.

The computation time can also be reduced by only solving the IRMP at certain moments during the solution process, instead of solving it at the start of every iteration. Solving the IRMP does require a lot of time, while a better solution will not be found in each iteration. Finally, the most commonly used way to reduce computation time is to set some stopping criteria. In the remainder of this section we discuss the implementation of each of these possibilities to improve the computation time of the solution method.

5.6.1 Pre-solving method

After initializing the solution process with the task list, delay and detour possibilities, and the recovery networks, we would like to start at the first iteration of our solution approach and continue until no further improvements can be found. If no recovery duties or routes are added before starting the solution process, no tasks can be executed. Because this means that each task is cancelled, all tasks will get a very high dual value (which means that it is highly recommended to let any of the drivers of vehicles perform this task). High dual values result in longer computation times of the pricing problems, as more promising recovery duties and routes can be found (in the first iterations, almost every feasible path from source to sink is considered promising). It may be the case that not all these duties and routes are added, due to the labeling procedure, however, most connections in the network are still explored, which takes a lot of time. In the pre-solving method, we try to overcome this by a greedy approach for solving the pricing problem.

Before the start of the pre-solving method, we check for each vehicle and driver whether its/their original route or duty is still feasible. If this is the case, we add the original route or duty as possible recovery route or duty. Note that this does not mean that for all those drivers and vehicles, their original duty or route can be executed. For example, let drivers A and B both be assigned to vehicle 1, in that order, and consider the following situation.

If a disruption occurs such that driver A is delayed with vehicle 1, the recovery network of driver A and vehicle 1 will not contain all their original tasks (here, we do not mean delayed versions of the original tasks), as these can no longer be executed. However, for driver B, all original tasks will still be present in the recovery network, whilst vehicle 1 may not be able to arrive on time at the relief location where the change of driver takes place. This thus means that for driver A and vehicle 1, we do not add their original duty and route. However, for driver B, the original duty is still added. The reason that we do add the original duty of driver B is because we are not fully able to check for each vehicle and driver if such a situation as in the example will occur, as multiple drivers may be assigned to a single vehicle.

After initializing the pre-solving method, we basically execute our solution approach but with one important difference. Namely, we only solve the pricing problem for those drivers and vehicles for which no recovery duty or route has been selected as recovery duty or route, after solving the IRMP. This is done until the overall lower bound for this approach exceeds 0, because then we are interested in better results, or if the IRMP can find a feasible recovery duty or route for every driver and vehicle. If one of these stop criteria is met, we continue with the solution approach as has been explained in the previous sections. The benefit of the pre-solving method is thus faster convergence to a feasible solution, which leads to a reduction of the total computation time of the solution method.

5.6.2 Neighbourhood of Vehicles & Drivers

Another way to reduce the computation time is to exclude drivers and vehicles for which definitely no changes are needed in their duties or routes. To make sure that we only include the vehicles and drivers for which we expect that it is necessary to make a recovery route/duty, we apply the following algorithm to form a neighbourhood of vehicles and drivers.

Let Q be the set of vehicles passing the disruption in the original timetable during the recovery period. For those vehicles, we search for all drivers who operate these vehicle during the recovery period. Let all those drivers be included in the set P. We then perform the following steps repeatedly until no more vehicles can be added to the set Q **and** no more drivers can be added to the set P. First, we add all vehicles operated by drivers from set P to the set Q. Then, we add all drivers who operate vehicles from the set Q to the set P. If new vehicles or new drivers are added to either one of the sets, we go back to the first step.

Note that we may loose optimality by creating a neighbourhood of vehicles and drivers. However, creating a neighbourhood has a large positive effect on the computation time. Also note that, if we introduce a neighbourhood in combination with a recovery period, only drivers and vehicles are added to the neighbourhood which are active during the recovery period. Thus, vehicles and drivers either starting their route or duty after the end of the recovery period, or ending their route or duty before the start of the recovery period, are not added to the neighbourhood.

5.6.3 Recovery period

To even further reduce the computation time, we also propose to use a recovery period. Introducing a recovery period reduces the size of the task list (and thus also the number of delay possibilities and possible detours) such that less potential recovery timetables can be created. Another benefit is that we hopefully exclude drivers and vehicles for which no changes in their duties or routes are needed, because they start after the disruption can been solved.

A recovery period is defined as a period in which each task of the task list is allowed to be assigned to another vehicle or driver than it was originally assigned to. The recovery period must be at least as large as the duration of the disruption, however, in almost all cases that the disruption affects the timetable, more time is needed. There is thus a trade-off between choosing a large recovery period with (most likely) better solutions against a high computation time, or a small recovery period, risking infeasibility for one of more vehicles and drivers but with a low computation time. The goal is to find a recovery period somewhere in between.

We decided to test different approaches for choosing a (fixed) window as recovery period. Our final choice will be based on the results of experimenting with the different approaches on some test cases. This will be discussed later in Chapter 7. The different approaches are as follows:

1. A fixed window (F) as recovery period, starting after the disruption.
2. A window dependent on the disruption.
3. A combination of approach 1 and 2.

Note that outside of the recovery period the penalty cost is thus equal to 0. However, even if we find the optimal solution within the recovery period, it may still not be the overall optimal solution. This is because, excluding parts of the problem reduces the number of possibilities for forming new recovery routes and duties within the recovery period. Thus, we may loose optimality by introducing a recovery period.

In our first approach, we choose a fixed window after the disruption as recovery period. For the remainder of this thesis, the fixed part of the recovery period will be denoted by F given in minutes. A fixed window will probably work well for small disruptions, because if we choose the window large enough, there should always be enough time to reschedule the vehicles and drivers such that a feasible recovery timetable can be obtained. However, choosing the window large enough is difficult as the impact of the disruption on the original timetable is hard to predict beforehand. Therefore, the recovery period of our second approach, depends on the disruption.

For this, we checked the idle time in the original route of each vehicle which is affected by the disruption. After each original trip, there may be some time left for a vehicle before it needs to start another trip. If not enough idle time is present on the route of the vehicle, the recovery period must be extended until enough idle time is present for each vehicle to cover the disruption, as otherwise, the vehicle is not able to reach its destination in time. We thus need to search for the first start time of a trip at which the disruption has no longer an affect. After checking this for each vehicle, we take the maximum start time as end time of the recovery period.

We expect that this method works well for most disruptions, however it may still be the case that the recovery period is too short. Namely, it can be that even though all vehicles are able to reach their destination, some drivers may not be able to reach their destinations due to the fact that drivers need to take breaks and they are restricted by the labour rules. Therefore, our final method is to combine the previous two approaches. For this, we first apply the second method to find the latest starting time of a task which is used as end time of the recovery period, and then, we add a fixed period such that always enough time is given to create a feasible recovery timetable in which all drivers and vehicles have a recovery duty or route, respectively.

5.6.4 Solving the IRMP

During each iteration of the solution method, most time is spend on solving the IRMP. However, not in each iteration, a better solution is found for the IRMP. Therefore, we may reduce the number of times we solve the IRMP during the solution process.

To reduce the number of times the IRMP is solved, we only solve the IRMP if a better overall lower bound is found. We choose this criteria because for large instances, the duration of an iteration does take more time than for a smaller instance, such that a criteria like *solve the IRMP every s seconds*, does sometimes result in still solving the IRMP every iteration, especially for large instances. Also, for small instances, it may take far more time than needed before the solution is found, because we have to wait s seconds each time after we solve the IRMP, before we are again allowed to solve the IRMP. Furthermore, a criteria like *solve the IRMP every n iterations* neither result in a faster computation time for all instances, because for small instances, we would like to choose n small (because more improvement is made in each iteration leading to a possibly better solution to the IRMP), while for large instances, we would like to choose n large.

As initial strategy we therefore solve the IRMP each time a better overall lower bound is found during the solution process, because this works for both small and large instances. In Chapter 7, we will however also propose another strategy in solving the IRMP during the solution process, which is based on some of the other results obtained in that chapter.

Note that we always solve the IRMP in the first iteration of the solution process, as an upper bound still needs to be determined at this point. Also note that we always solve the IRMP before terminating the solution process. In this manner, we always find the best solution of the current routes and duties included in the IRMP.

5.6.5 Stopping Criteria

Finally, the most commonly used way to reduce the computation time is to implement some stopping criteria. In our case, we stop the solution process if the overall lower bound is equal to the upper bound, which means that an optimal solution is found. If it takes too much time to find the optimal solution or to prove that the optimal solution is the optimal solution (it may happen that the solution method has a hard time to improve the overall lower bound) we terminate the solution process after x minutes, with

$$x = \max\{20, \textit{duration of disruption (in minutes)}\}.$$

Such that a solution is always obtained within the duration of the disruption, and within 20 minutes. Unfortunately, it may happen that even after x minutes, a feasible solution is not found. In that case, we can choose to continue the solution process until the first feasible solution is obtained, or we can manually stop the solution process and obtain the best solution found so far. Besides the time limitation, we propose another stopping criteria based on the gap between the overall lower bound and the value obtained by solving the LRMP in Chapter 7.

Chapter 6

Data Description

In this chapter, we discuss the data used to test our solution method. We first provide more insight in the original timetable. Thereafter, we provide the parameter values and the penalty cost used for obtaining the results of the different test cases. Finally in the last section, we describe the different test instances (disruptions in the original timetable) which will be used to test our solution approach.

6.1 Original Timetable

The solution method is tested using an original timetable provided by the RET of a normal week day. In this original timetable, 112 vehicles operate during the day performing a total of 1,834 trips consisting of a total of 47,862 trip points. Furthermore, 203 drivers have a duty during this day, of which 17 have a split duty. The original duties of the drivers consist of a total of 662 work pieces. Each driver operates at least 1 different vehicle during his/her duty and at most 5 different vehicles. On average, each driver operates 2.88 different vehicles. For each vehicle, at least 1 different driver is scheduled to operate the vehicle during the day, and at most 11 different drivers. On average, for each vehicle 5.21 different drivers are scheduled to operate the vehicle, during the day. Furthermore, each driver operates a vehicle for at least 21 consecutive minutes and at most 306 consecutive minutes.

During the day, the number of vehicles actively operating trips differs, as during rush hours, more vehicles need to be used because the lines operate with a higher frequency. As a result, the number of active drivers also differs during the day. This results in less breaks during the rush hours and also more starts and finishes of duties during off-rush hours. Figures 6.1 and 6.2 show the number of active vehicles and drivers during the day, as well as the number of drivers on breaks and the total number of drivers present at each moment of the day.

As can be seen from the figures, around 4:30 the first vehicles and drivers start operating and around 2:00 the next day, all vehicles are back at the depot and all drivers have finished their duty. Most breaks are taken around 12:00, and at each moment in time, the number of active drivers is at least equal to the number of active vehicles.

Once a disruption occurs, we expect that it would have a larger impact on the planned breaks of the drivers, if the disruption takes place around 12:00. This means that it is likely that drivers, affected by the disruption, may need to skip their break and have it at another moment and/or place. However, as can be seen from the figures, around 20:00, also a large percentage of the number of present drivers take a break (at some point this is even around 20.0% of the present drivers). As also many drivers end their duty around this time (this can be seen from the number of active vehicles during this time), we expect that a disruption between 19:00 and 20:00 would also have a larger impact on the planned breaks of the drivers as well as the possible overtime needed for the drivers to return to their end location.

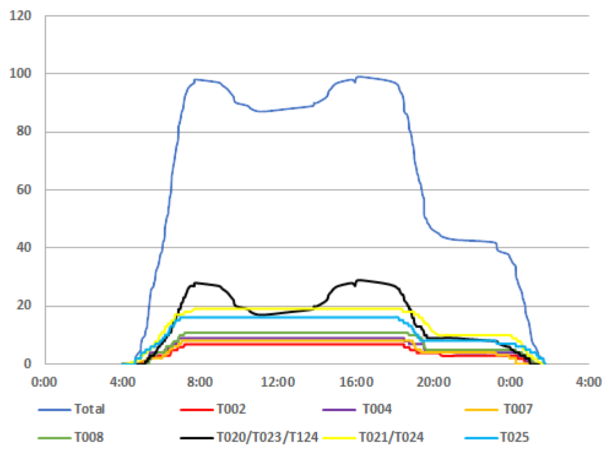


Figure 6.1: Active vehicles

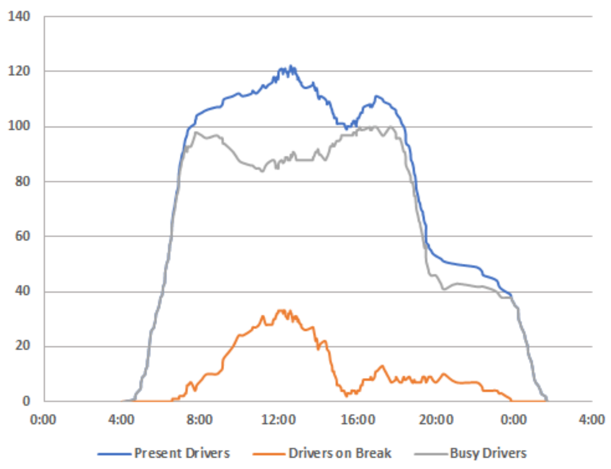


Figure 6.2: Drivers

If more vehicles are active, we expect more changes to the original routes of the vehicles in the recovery timetable. Figure 6.1 also shows the number of active vehicles on each (group of) lines during the day. Some lines are grouped because the same vehicle operates on both lines in the original timetable. This would for example mean that it is likely that a disruption on line T020 also affects vehicles operating lines T023 and T124, as these lines are grouped. As more vehicles operate on these lines, and on lines T021 and T024, we expect that disruptions occurring on these lines have a larger impact on the original timetable and thus requires more changes. For line T002 we expect less needed changes if a disruption occurs as during the day the least number of vehicles need to operate simultaneously on this line.

Note furthermore, as can be seen from Table 6.1, that all lines have some overlapping tracks with other lines. A track is defined as the rail segment between two stops. This again indicates that a disruption occurring on for example line T008 could also impact all other lines except for line T002, and the more overlapping tracks a line has with other lines, the more likely it becomes that more changes are needed to the routes of the vehicles using the tracks, to overcome the disruption. Note again that for line T002 not much correlation exist with the other lines, as line T002 only shares 11 tracks with line T020 and no tracks at all with other lines.

Table 6.1: Overlapping track segments

Line	Number of overlapping tracks with line ...									
	T002	T004	T007	T008	T020	T023	T124	T021	T024	T025
T002	39	-	-	-	11	-	-	-	-	-
T004	-	60	1	10	-	-	-	-	-	2
T007	-	1	46	10	-	-	-	-	-	3
T008	-	10	10	66	6	7	2	4	4	6
T020	11	-	-	6	38	14	3	5	5	17
T023	-	-	-	7	14	72	3	22	22	12
T124	-	-	-	2	3	3	15	15	15	2
T021	-	-	-	4	5	22	15	80	75	4
T024	-	-	-	4	5	22	15	75	88	4
T025	-	2	3	6	17	12	2	4	4	64

This table states for each line, the number of overlapping tracks with each of the other lines. A track is defined as the rail segment between two stops.

Up to this point, we described the expectations of the impact of a disruption at different moments in time on the number of vehicles and the planned breaks of drivers, and the propagation of disruptions of one line to other lines. However, recently the RET also introduced crew interlining, which caused a new relationship between the vehicles, drivers and the lines.

Before interlining was introduced, each driver operated on a single line during the day, while now, drivers may operate on different lines during the day. Interlining has as advantage that the breaks could be scheduled more efficiently, and the idle time of the vehicles and drivers could be reduced. However, this advantage becomes a disadvantage when a disruption occurs. Namely, the probability that a disruption on a line also propagates to other lines increases. This is because there is less room for rescheduling as not many drivers are having a break, while they do not necessarily need a break (in other words, the slack time within their duty has decreased).

Table 6.2 shows for each line, if any drivers of this line also operates on one of the other lines. As can be seen, drivers who operate on one of the lines T002 or T025 do not operate on any other lines than this line during their duty. Therefore, we expect that disruptions occurring on these lines do not necessarily propagate towards other lines, as a results of interlining. However, for the other lines, there is much interlining during the day. This indicates that disruptions occurring on one of the lines in one of the clusters ({T004, T007, T008, T021, T024} and {T020, T023, T124}) may also propagate towards the other lines in the cluster.

Table 6.2: Crew interlining

Line	Drivers also operating line ...									
	T002	T004	T007	T008	T020	T023	T124	T021	T024	T025
T002	Yes	-	-	-	-	-	-	-	-	-
T004	-	Yes	Yes	Yes	-	-	-	Yes	Yes	-
T007	-	Yes	Yes	Yes	-	-	-	Yes	Yes	-
T008	-	Yes	Yes	Yes	-	-	-	Yes	Yes	-
T020	-	-	-	-	Yes	Yes	Yes	-	-	-
T023	-	-	-	-	Yes	Yes	-	-	-	-
T124	-	-	-	-	Yes	-	Yes	-	-	-
T021	-	Yes	Yes	Yes	-	-	-	Yes	Yes	-
T024	-	Yes	Yes	Yes	-	-	-	Yes	Yes	-
T025	-	-	-	-	-	-	-	-	-	Yes

This table states for each line, if drivers operating this line also operate any of the other lines, during his/her duty.

The main purpose of stating above expectations is to better understand why a disruption (at the same time and with the same duration) could have a very different impact on the size of the instance and the outcome, if the location of the disruption differs. The same holds for disruptions taking place at the same location and for the same duration, while they do not occur at the same time. By discussing the results of the test cases, we will see if our expectations meet with reality.

6.2 Parameters: Labour Rules & Vehicle Rules

In the original timetable, we assume that each driver has a duty which adheres the labour rules. If it turns out that this is not the case, it may also not be possible to find a feasible recovery timetable if a disruption occurs. This should be kept in mind if we interpret the results. Note furthermore that this also holds for the vehicles, for which we assume that their original route adheres the vehicles rules.

To initialize the parameters for the labour rules and vehicle rules for our research, we have used the following values, which are summarized in Table 6.3.

The duty of a driver requires a minimum of 2 (b_{min}) breaks, with a minimal duration of 15 minutes (l^b_{min}), if the duty is at least 6 hours (h^{duty}_{min}). The working time between 2 breaks (or the start or the end of a duty and a break) may not exceed 4 hours and 15 minutes (h^{work}_{max}).

Furthermore, the maximum allowed overtime for each driver at the end of his/her duty is set to 60 minutes (o^D_{max}). The travel time for a driver between any two relief locations is set to 45 minutes (T_{travel}). To make sure that the network of each driver remains small, and the driver remains active during its duty, we set the maximum allowed waiting time at all stops equal to 60 minutes (w^{MAX}).

For the vehicles rules, we use the following parameter values. We set the maximum idle time at stops which are not at relief locations, equal to 15 minutes (w_i^{max}), for stops at relief locations we set the maximum idle time to 60 minutes (w^{MAX}). Furthermore, each vehicle may return up to 60 minutes (o^V_{max}) later to the depot than originally planned.

Labour Rules	
b_{min}	2
l^b_{min}	15 min.
h^{duty}_{min}	6 hours
h^{work}_{max}	4 hours and 15 min.
o^D_{max}	60 min.
w^{MAX}	60 min.
T_{travel}	45 min.
Vehicle Rules	
w_i^{max}	15 min.*
w^{MAX}	60 min.
o^V_{max}	60 min.

* Not at relief locations.

Table 6.3: Parameter values

Finally, note that we do not set any maximum allowed delay at each stop (q_i^{max}). This is because if we set a maximum of for example 10 minutes, while the disruption lasts 20 minutes, we may not even be able to find a feasible recovery timetable. Also, we only create delay possibilities if they make new connections within the network, so this keeps the number of delay possibilities already small while maintaining a feasible recovery timetable.

6.3 Penalty Cost

For the recovery timetable, we want the solution to contain as few as possible cancelled tasks, while at the same time the recovery duties and the recovery routes should not deviate too much from the original timetable. To be able to achieve this, we have set the penalty cost as follows.

We set the penalty for not having any route or duty (M) equal to 10^7. This penalty should be set very high, as the recovery timetable is not feasible if we do not find a recovery route or duty for each vehicle and driver. We let the penalty cost for cancelling a stop (p_1) be equal to 1000 per stop. The delay penalty is determined based on the penalty cost for cancelling a stop. Namely, we set the delay penalty (p^M) to 50 per stop per minute, such that an indifference exist between cancelling a stop or delaying a stop with 20 minutes. Note that this does not mean that delays of more than 20 minutes are never chosen in a recovery timetable. This is because a delay can be high for the first task of a recovery route or duty, but lower (until reaching no delay) towards the end of the recovery route or duty. Therefore, it can still be better to start with (for example) a delay of 30 minutes, instead of cancelling large parts of the original route or duty.

The penalty for every new visited stop compared to the original route of the vehicle (p^V_W) is set to 100 per stop. Furthermore, the penalty for every new visited stop compared to the original duty of the driver (p^D_W) is set to 200 per stop and the overtime penalty (p^O) to 10 per minute.

6.4 Test Instances: Disruptions of Original Timetable

To test our solution method, we disrupt the current timetable at two different points in the network. We choose to disrupt the timetable for instance 1) between *Maashaven* and *Brielselaan* and for instance 2) between *Leuvehaven* and *Wilhelminaplein.* For both instances, we let the disruption takes places in both directions between the locations. The first instance is chosen because between these locations, only line 2 operates and no interlining occurs with other lines. Also, this line shares the least number of tracks with other lines. The second instance is chosen because this is one of the more difficult kinds of daily disruptions the RET currently needs to solve. Also, multiple lines visit the disruption location and interlining occurs on these lines.

Besides the different locations, we can also choose to disrupt the timetable at different moments in time. The instance size will change if the duration of the disruption differs, or the time at which the disruption occurs differs. The latter is because, as explained earlier, the number of vehicles and drivers operating during the day differs at each moment of the day. We test the influence of these two factors on the quality of the solution provided by the solution approach and the computation time in Section 8.2.2. For all other sections, the instances can be summarized by:

I1D20: Disruption of 20 minutes at 12:00 between *Maashaven* and *Brielselaan*

- Isolated line (T002) with no interlining.

I2D20: Disruption of 20 minutes at 12:00 between *Leuvehaven* and *Wilhelminaplein*

- Technical problems at the Erasmus Bridge resulting in a blockage of the bridge. The disruption directly affects line T020, T023 and T025, as these lines pass the disruption. Also, interlining occurs on lines T020 and T023.

In the remainder of this thesis, we refer to one of the above disruptions by 'IxDy', where x presents the instance number and y presents the duration of the disruption.

Chapter 7

Algorithm Settings & Tuning

In this chapter, we perform some experiments with the different methods to reduce the computation time of the solution method. The different methods result in different ways in a reduction of the computation time, either indirectly (by reducing the size of the instance) or directly (by introduction the pre-solving method, or by decreasing the number of times the IRMP is solved during the solution process). For each of these methods, the impact on the computation time will be shown for the test instances I1D20 and I2D20 in Sections 7.1 and 7.2. Finally, in Section 7.3, we summarize our final algorithm settings and provide some recommendations for further research. To test our solution approach, we implemented the solution method in *Java*. Furthermore, we made use of *CPLEX Version 12.8* to solve the integer restricted master problem (IRMP) and the linear restricted master problem (LRMP).

7.1 Instance Size

The computation time indirectly decreases if we reduce the size of the instance. This can be done by the introduction of a neighbourhood or by the introduction of a recovery period. We show the impact of the introduction of a neighbourhood on the size of the instance in this section. Furthermore, we also discuss the impact on the size of the instance by experimenting with different variations for determining the end time of the recovery period.

Note that we may loose optimality or feasibility by the introduction of a neighbourhood or a recovery period. This is because less vehicles and drivers means less rescheduling possibilities. Loss of optimality is not necessarily an issue if this saves a lot of computation time without a large difference in the solution quality. However, infeasibility does cause problems if the solution needs to be used as recovery timetable. Both loss of optimality and loss of feasibility should be kept in mind by the introduction of the methods. We distinguish between two sorts of infeasibilities. Namely, the infeasibility of an instance and the infeasibility of a solution.

The *infeasibility of an instance* means that not a single path from source to sink can be found for at least one driver or one vehicle. This results in an empty recovery network. In the initialization step of the solution process, we guarantee feasibility of the instance, if no recovery period is used. However, if a recovery period is used, it may be that this sort of infeasibility occurs because either a vehicle or a driver is not able to reach its sink node. In that case, we should extend the recovery period. The other sort of infeasibility is the *infeasibility of the solution.* If this occurs, all vehicles and drivers do have a recovery network, and paths from source to sink for all drivers and vehicles can be found, but there is no combination of recovery routes and duties which forms a feasible timetable. A feasible timetable is created if each driver has a recovery duty and each vehicle a recovery route. Infeasibility can occur because the recovery duties must also satisfy the labour rules. Another cause of infeasibility is the fact that even though a path from source to sink for a vehicle could exist, this path may not be executable as this combination of tasks cannot be done by any (combination) of the drivers.

7.1.1 Neighbourhood of Vehicles & Drivers

Our first method to reduce the computation time is to introduce a neighbourhood of vehicles and drivers. The benefit of this method is that we decrease the size of the instance such that less possibilities need to be considered which results in a lower computation time. To decrease the size of the instance, we exclude vehicles and drivers for which we do not expect that any changes to the original route are necessary, because they do not pass the disruption location and are not related to any of the vehicles or drivers that do pass the disruption location. If less vehicles and drivers are included in the problem, the number of tasks included in the task list decrease, which result in less delay possibilities, possible detours and connections between tasks in the recovery networks.

The impact of the introduction of a neighbourhood of vehicles and drivers on the size of the instance is shown in Table 7.1 for instances I1D20 and I2D20. Note that at this point, no recovery period is used, such that the instances include all tasks from the start time of the disruption, up to the end of the day. Table 7.1 presents, for each instance, with and without a neighbourhood, the total number of stops ($\#S$) that must be visited in the recovery period, the total number of delayed stops ($\#S^D$) that are created as delay possibilities, the total number of possible created detours ($\#O$), the total number of drivers ($\#D$) included in the instance and total number of vehicles ($\#V$) included in the instance. Note that we do not mention the size of the task list (N), nor the size of the delay task list ($N_e \setminus N$), because each of these tasks includes a different number of stops that must be visited if this task is executed. As the penalty cost depends on the total number of stops included in each task, mentioning the total number of stops included in all tasks of the (delay) task list provides more inside in the size of the instance.

Table 7.1: Impact introduction of neighbourhood

	I1D20					I2D20				
	$\#S$	$\#S^D$	$\#O$	$\#D$	$\#V$	$\#S$	$\#S^D$	$\#O$	$\#D$	$\#V$
Without Neighbourhood	28,947	118	70	194	101	28,947	1,079	997	194	101
With Neighbourhood	2,174	118	70	15	7	6,288	855	279	48	24
Reduction (%)	-92.5	0.0	0.0	-92.3	-93.1	-78.3	-25.0	-72.0	-75.3	-76.2

This table shows the impact on the size of the instance by introducing a neighbourhood of vehicles and drivers, for instances I1D20 and I2D20.

The results show that without using a neighbourhood, both instances contain the same number of stops that must be visited, and the same number of drivers and vehicles who/which operate between 12:00 and the end of the day, which is as expected. The total number of delay possibilities and possible detours increase if more lines cross the disruption location (I2D20). This is also as expected because if more lines cross the disruption, more vehicles are affected by the disruption.

The results further show that for both instances, huge reductions (92.5% and 78.3%) in the total number of stops that must be visited ($\#S$) are made by the introduction of a neighbourhood. Note that the reduction in the number of drivers and vehicles included in the instance, is reduced by around the same percentage. This is because both a vehicle and a driver are needed for a single task, so by excluding vehicles and drivers, we should expect the same decrease in percentage in the number of tasks (and thus stops that need to be visited). The reduction is larger for I1D20 because far less vehicles and drivers are affected by the disruption occurring on an isolated line instead of at a location where vehicles operate multiple lines, and also drivers interline between different vehicles of different lines. Furthermore, the number of delay possibilities remains the same compared to without the neighbourhood, while a reduction of 25.0% occurs for I2D20.

I1D20: Disruption of 20 minutes at 12:00 between *Maashaven* and *Brielselaan*
I2D20: Disruption of 20 minutes at 12:00 between *Leuvehaven* and *Wilhelminaplein*

The number of possible detours does also not change for instance I1D20 if we introduce a neighbourhood, while the reduction is 72.0% for instance I2D20. The reason for no improvement in the reduction of the delay and detour possibilities is that for I1D20, the disruption occurs on an isolated line. Therefore, without the introduction of a neighbourhood, all delay possibilities are created from tasks of this line, and possible detours are also created from tasks of this line to other tasks of this particular line, as only these connections form new possibilities. Note that therefore, loss of optimality by creating a neighbourhood will not occur for instance I1D20. However, there is a chance that loss of optimality for instance I2D20 does occur. Unfortunately, we cannot determine if this is true based on the information provided so far. For this, we need to compare the solutions of solving the instances with and without a neighbourhood.

Table 7.2 states the upper bound (UB), the best found overall lower bound (LB), the time at which a first feasible solution is found (T_1), the time at which the best solution is found (T^*) and the total computation time (T_{tot}), all in seconds, for both instances, with and without the introduction of a neighbourhood.

Table 7.2: Results introduction of neighbourhood

	I1D20					I2D20				
	UB	LB	T_1	T^*	T_{tot}	UB	LB	T_1	T^*	T_{tot}
Neighbourhood			(sec.)	(sec.)	(sec.)			(sec.)	(sec.)	(sec.)
Without	$2.968 \cdot 10^9$	$-3.148 \cdot 10^7$	-	-	>1,200	$2.968 \cdot 10^9$	$-3.739 \cdot 10^8$	-	-	>1,200
With	45,730	18,764	59	142	>1,200	625,551,400	$-592,832$	-	603	>1,200

This table states the results obtained after 20 minutes of solving the instances I1D20 and I2D20 with and without a neighbourhood.

Without the introduction of a neighbourhood, the results show that for both instances, not even a single driver or vehicle could be assigned a recovery route or duty during the 20 minutes of computation time. This can be seen from the fact that both upper bound values are the same. Note that the best lower bound found after 20 minutes is lower for instance I2D20 than for I1D20, which is as expected because the instance is larger such that less improvement can be made within the same time.

By the introduction of a neighbourhood, the computation time decreases significantly, because a feasible solution can be obtained within 59 seconds for instance I1D20, compared to not at all in case no neighbourhood is used. For instance I2D20, at least some drivers and vehicles have been given a recovery duty or route. However, even with the introduction of a neighbourhood, a feasible solution cannot be found within 20 minutes. For instance I1D20, the best found solution differs with 59.0% from the best found lower bound. Note however that the first feasible solution and the best found solution for the upper bound for this instance are obtained early in the solution process, while thereafter, no improvement is made until the solution process is terminated.

Finally note that, with the introduction of a neighbourhood, infeasibility of the instance does not occur for these instances, as each driver and each vehicle in the instance has a recovery network for which holds that a path from source to sink exist. We cannot prove infeasibility of the solution (for I2D20) or the loss of optimality, as no solution can be found without the use of the neighbourhood, and with the use of a neighbourhood, a feasible solution could still not be obtained for instance I2D20. However, because for our thesis, a solution has to be found in a very short amount of time, we still decided to use a neighbourhood regardless the chance on loss of optimality. In the remainder of this thesis, all results are obtained with the use of a neighbourhood of vehicles and drivers.

I1D20: Disruption of 20 minutes at 12:00 between *Maashaven* and *Brielselaan*
I2D20: Disruption of 20 minutes at 12:00 between *Leuvehaven* and *Wilhelminaplein*

7.1.2 Recovery Period

The second approach to reduce the computation time is by introducing a recovery period. By introducing a recovery period, we set an end time for the window in which tasks of the task list are allowed to be delayed, cancelled or assigned to other vehicles and drivers. This leads to less nodes in the networks of the drivers and vehicles, such that the pricing problem can be solved faster as less nodes, and connections between nodes, need to be considered. Also, the number of drivers and vehicles included in the problem usually reduce if an end time is set, as some of them may start their duty/route after the end of the recovery period.

To understand the impact of introducing a recovery period, we test different approaches (M1, M2 and M3) with different values for the fixed part of the recovery period (F in minutes). M1 corresponds to the approach in which only a fixed window (after the disruption) is used as recovery period. M2 corresponds to the approach in which a the recovery period is dependent on the disruption. The combination of the two approaches is denoted by M3.

For M1, we let the fixed part of the recovery period F, be equal to 60, 120 and 180 minutes. The values of F are based on the minimum break time for drivers, the maximum overtime for drivers and the default travel time between relief locations. For example, we make sure that for M1 the fixed part of the recovery period is at least equal to the maximum allowed overtime for a driver. However, sometimes this may not even be enough, as breaks need to take place due to the restriction on the maximum driving time of the drivers and the fact that drivers need to end at their originally planned end location in time. The latter may result in drivers needing to travel from relief location to relief location, which requires time. Therefore, we have also set F equal to larger values (120 and 180 minutes) to account for these events.

For M3, we let F be equal to 30, 60 and 90. These values of F are also based on the minimum break time for drivers and the default travel time between relief locations. However, the values do not have to be as high as those for M1, because the recovery period already includes enough time for the vehicles to arrive in time at their end locations. Therefore, we have set the values of F such that for the first case (30 minutes) extra time is given to account for at least two extra breaks for each driver, during the recovery period. For the second (60 minutes) extra time is given to account for at least two extra breaks for each driver, or for at least one break and one travel piece for each driver. For the third case (90 minutes), enough extra time is given for at least two extra breaks and a travel piece, or two travel pieces, needed for a driver.

Table 7.3 presents for each of these variations the end time of the disruption window (W_e), and all other information about the instance (similar as for the introduction of the neighbourhood), for both instances. Note that we only mention the end time of the recovery period (or disruption window), because the start time of the recovery period is equal to the start time of the disruption (for both instances 12:00).

For all methods, each instance contains less visited stops than compared to not using a recovery period (for reference, see results in Table 7.1). For method M1, also the number of vehicles and/or drivers included in the instance has reduced. This does however not hold for all variations of the methods M2 and M3, for both instances. The reason for this is that the end time of the recovery period, is later than the start times of the duties of the drivers and routes of the vehicles included in the neighbourhood, for those instances. This indicates that using a recovery period is useful for reducing the tasks, (and sometimes also the detour and delay possibilities) but not necessarily for reducing the number of vehicles and drivers included in the instance.

I1D20: Disruption of 20 minutes at 12:00 between *Maashaven* and *Brielselaan*

I2D20: Disruption of 20 minutes at 12:00 between *Leuvehaven* and *Wilhelminaplein*

Table 7.3: Impact introduction of recovery period

	I1D20						I2D20					
	W_e	$\#S$	$\#S^D$	$\#O$	$\#D$	$\#V$	W_e	$\#S$	$\#S^D$	$\#O$	$\#D$	$\#V$
M1F60	13:20	195	77	47	7	5	13:20	561	131	160	21	16
M1F120	14:20	326	93	56	8	5	14:20	1,118	423	235	29	20
M1F180	15:20	569	98	70	10	6	15:20	1,862	536	251	33	22
M2F0	14:52	493	98	70	9	6	18:51	4,289	855	276	48	24
M3F30	15:22	586	98	70	10	6	19:21	4,565	855	277	48	24
M3F60	15:52	679	98	70	11	6	19:51	4,715	855	277	48	24
M3F90	16:22	789	98	70	12	6	20:21	4,853	855	277	48	24

This table shows the impact on the size of the instance by introducing a recovery period, for instances I1D20 and I2D20 (with use of neighbourhood).

As holds for the introduction of the neighbourhood, we may loose optimality or infeasibility of the solution, or even worse, infeasibility of the instance. Recall that infeasibility of the solution means that each driver and vehicle has a recovery network, and recovery duties and routes can be obtained, but a feasible combination of these duties and routes cannot be found. Infeasibility of the instance means that one (or more) driver(s) and/or one (or more) vehicle(s) do not even have a recovery network, such that no path from source to sink can be found.

Table 7.4 states results obtained after 20 minutes of solving the instances I1D20 and I2D20 with a neighbourhood and different methods to determine the end time of the recovery period. For instance I1D20, the same solution was found for each method, all within 23 seconds of total computation time. This indicates that using a neighbourhood in combination with a recovery period works well for solving disruptions on isolated lines. Infeasibility of the instance occurred for instance I2D20 using method M1 with $F = 60$ and $F = 120$ minutes. This is not a surprise, as it is very unlikely that, after a disruption of 20 minutes, the original timetable can again be executed after only such short amount of time, especially if disruptions also affect drivers and vehicles not even passing the disruption. Note that the loss of feasibility is also expected if we take a look at the end time of the recovery window for method M2, for these instances. Namely, if the end time of the recovery period much earlier than the end time of the estimated earliest time for which all vehicles are able to reach their destination (the end time used in method M2), it may be that a vehicle passing the disruption is not able to reach its destination in time.

Table 7.4: Results introduction of recovery period

	I1D20					I2D20				
	UB	LB	T_1 (sec.)	T^* (sec.)	T_{tot} (sec.)	UB	LB	T_1 (sec.)	T^* (sec.)	T_{tot} (sec.)
M1F60	26,950	26,950	0	0	1	x	x	x	x	x
M1F120	26,950	26,950	1	2	4	x	x	x	x	x
M1F180	26,950	26,950	3	6	18	324,250	324,250	26	52	214
M2F0	26,950	26,950	3	3	14	275,570	101,473	1,124	1,198	>1,200
M3F30	26,950	26,950	2	3	15	71,219,590	-847,003	-	1,200	>1,200
M3F60	26,950	26,950	2	5	23	272,660	241,181	404	1,035	>1,200
M3F90	26,950	26,950	3	6	23	563,943,455	-909,853	-	448	>1,200

This table states the results obtained after 20 minutes of solving the instances I1D20 and I2D20 with a neighbourhood and different methods to determine the end time of the recovery period.

Loss of optimality occurred for I2D20 using method M1 with $F = 180$ minutes, as (for example) the best found solution for M2 has a lower objective value than was found as optimal value when using method M1 with $F = 180$ minutes. This may again be due to the fact that the end time of the recovery period for this method (15:20), does not come close to the estimated end time needed for the recovery period (18:51).

I1D20: Disruption of 20 minutes at 12:00 between *Maashaven* and *Brielselaan*
I2D20: Disruption of 20 minutes at 12:00 between *Leuvehaven* and *Wilhelminaplein*

For M2 and M3 with $F = 60$, feasible solutions could be found, however this was not possible for M3 with $F = 30$ and M3 with $F = 90$. Because M2 includes less or as much possibilities as M3 with $F = 30$, we do not believe that infeasibility occurs for this method, but rather that too much time is spend on finding even a feasible solution. This can also be seen from the first time a feasible solution is found for M2, which is close to the end of the recovery period.

The reason that M3 with $F = 60$ finds a feasible solution within less time is probably due to the fact that more cancellations can take place if more recovery time is given, such that more feasible solutions exist if there are more of these possibilities. This suggest that providing more recovery time results in a reduction of the computation time because a feasible solution can more easily be found. However, this is not always true, as can be seen from the results of method M3 with $F = 90$ for instance I2D20. Namely, if too much recovery time is given, the purpose of the recovery period is no longer feasible, as not even for most drivers and vehicles a feasible recovery duty or route can be found, which was already the case if we did not use any recovery period.

In conclusion, introducing a recovery period reduces the total computation time. Because the computation time reduces, more improvement can be made within the same set time limit of 20 minutes. This leads to obtaining optimal solutions for small instances and (almost always) feasible solutions for larger instances, while without a recovery period, this could not be done. For instance I1D20, the results show that introducing a fixed recovery period (M1) could work if we choose the window large enough, however it can be hard to predict how much time is needed. The latter is shown by the results for I2D20, as for M1 with $F = \{60, 120\}$ minutes a feasible solution could not be found and optimality was lost for M1 with $F = 180$ minutes. Therefore, M1 should not be used as method to determine the end time of the recovery period, as this is not guaranteed to work for all sorts of disruptions. For the M2, M3 with $F = 30$ minutes and M3 with $F = 60$ minutes, no final conclusion can yet be made. This is because for instance I1D20, the solution method could quickly solve the instance to optimality. However, for instance I2D20, we did not find any optimal solution, such that we could not compare the impact of setting a larger or shorter recovery period, on the quality of the solution. Therefore, in the next section, we still analyze both instances for M2, M3 with $F = 30$ minutes and M3 with $F = 60$ minutes. Note that we do not further analyze M3 with $F = 90$ minutes, because almost no improvement of using this method was found compared to not using a recovery period at all for instance I2D20.

7.2 Acceleration of Computation Time

In Chapter 5, we propose to use a pre-solving method to obtain faster convergence to a feasible solution. Consequently, this leads to a reduction of the total computation time of the solution method. In this section, we test the use of the pre-solving method on instances I2D20, as for I1D20, optimal solutions can already be found within a short amount of time. Besides the use of the pre-solving method to reduce the total computation time, we also try to reduce the computation time by using another strategy in the number of times the IRMP is solved during the solution process. Furthermore, we propose an extra stop criterion for early termination of the solution method while maintaining a good quality solution. Note that all results presented in this section are obtained with the use of a neighbourhood of vehicles and drivers.

For practical usage, it would be beneficial to at least obtain a feasible solution within a short amount of time. If a feasible solution is known, the optimal recovery timetable will not differ for most drivers and vehicles included in the instances. Unfortunately, even with the use of a neighbourhood and recovery period, finding a feasible solution can still take a lot of time for larger instances such as I2D20. By using the pre-solving method, we hope that a feasible solution can be found sooner.

I1D20: Disruption of 20 minutes at 12:00 between *Maashaven* and *Brielselaan*
I2D20: Disruption of 20 minutes at 12:00 between *Leuvehaven* and *Wilhelminaplein*

Table 7.5 shows for instance I2D20, and methods M2, M3F30 and M3F60, the results obtained by the solution method with the use of the pre-solving method (and neighbourhood). The time until a feasible solution is found has decreased for all methods, however, the total computation time still exceeds 20 minutes. Besides the fact that feasible solution can now be obtained for all three methods within a short amount of time, we also obtain closer to optimal solutions than compared to the outcomes without the use of the pre-solving method. Namely, for all methods, the upper bound and best found lower bound do not differ much (at most 12% for M3F30).

Table 7.5: Results introduction of pre-solving method (I2D20)

	UB	LB	T_1	T^*	T_{tot}
			(sec.)	(sec.)	(sec.)
M2F0	272,970	249,394	529	1,173	>1,200
M3F30	272,660	240,178	608	951	>1,200
M3F60	272,170	241,306	245	1,200	>1,200

This table states the results obtained after 20 minutes of solving instance I2D20 with a neighbourhood and a pre-solving method, for 3 methods to determine the end time of the recovery period.

The reduction of the computation time by the implementation of the pre-solving method is mostly caused by solving the pricing problem for less drivers and vehicles during the first iterations. This is because we exclude those drivers and vehicles for which already a recovery route or duty could be found in the IRMP. The other factor reducing the computation time is that after the pre-solving method, the overall lower bound is higher at the start of the solution process, than is the case if no pre-solving method is used. Note that we solve the IRMP every time a new best lower bound is found. Therefore, the IRMP is initially solved less often if the pre-solving method is used, however, if a feasible solution is found, many small improvements of the overall lower bound occur, after a while. Consequently, the IRMP is solved much more often during the solution process itself, if the pre-solving method is used, because more improvement can be found within the same time in the lower bound. Because the latter does not necessarily mean that also an improvement in the upper bound can be made in the next iteration, we could save time in the solution process by solving the IRMP even less often. To test this, we run the same methods again using the pre-solving method, but now we only solve the IRMP during the pre-solving method, if the best found lower bound during the solution process exceeds 0 for the first time, and at the end of the solution method if no recovery routes or duties can be found with negative reduced cost or the time limit of 20 minutes is reached. Table 7.6 states the results.

Table 7.6: Results reducing number of times solving IRMP (I2D20)

	UB	LB	T_1	T^*	T_{tot}	UB_1
			(sec.)	(sec.)	(sec.)	
M2F0	272,170	263,093	105	1,200	>1,200	295,950
M3F30	272,170	258,312	131	1,200	>1,200	273,315
M3F60	272,170	251,911	215	1,200	>1,200	276,460

This table states the results obtained after 20 minutes of solving instance I2D20 with a new strategy to reduce the number of times the IRMP is solved.

The results show that by using this strategy, we can obtain feasible solutions within 4 minutes for all methods. A first feasible solution is found much faster if we solve the IRMP less often, because the overall lower bound exceeds 0 much earlier in the solution process. Also, the first found feasible solution (UB_1) does not differ much from the best found solution for each method (at most 8% for method M2F0). This indicates that even for difficult disruptions, affecting 48 drivers and 24 vehicles within a recovery period of 7.5 hours, the solution method could be used to reschedule vehicles and drivers by using the best found solution within the maximum set computation time.

I2D20: Disruption of 20 minutes at 12:00 between *Leuvehaven* and *Wilhelminaplein*

Note furthermore that also the difference between the upper bound and the best found lower bound (now at most 7% for method M3F60) for each method has reduced compared to solving the IRMP much more often during the solution process. However, we could still not prove optimality for any of the methods. The latter is probably due to the commonly known problem of column generation, which is the tailing-off effect. The tailing-off effect means that towards the end of the solution process (if no time limit is used), it takes very many iterations to close the gap between LB and LB_{LRMP}, while an optimal solution for the integer master problem can already be found.

If the tailing-off effect occurs, we could manually terminate the solution process and take the best found solution as recovery timetable. To see how much the results would differ if we terminated the process earlier, we again run instance I2D20, but now terminate the solution process after 5, 10 and 15 minutes. The results are presented by Table 7.7. Because we only solve the IRMP if we terminate the solution process after the set time limit, we cannot stop the solution method if the gap between the upper bound and the overall lower bound is less than a certain percentage. Therefore, we also report the best found value of the LMRP at the end of the solution process. If we compare the difference between the best found over all lower bound (at the end of the solution process) and the best obtained objective value for the LRMP, we may find a percentage after which we expect no further improvements in the upper bound. Note that this is because of the following relationship between the objective values: $LB \leq LB_{LRMP} \leq UB$.

Table 7.7: Results early termination of solution process (I2D20)

	x (min.)	UB	LB_{LRMP}	LB	$\frac{UB-LB}{UB} \cdot 100\%$	$\frac{LB_{LRMP}-LB}{LB_{LRMP}} \cdot 100\%$
M2	5	**274,120**	271,670	206,763	24.57	23.89
	10	272,170	272,170	251,494	7.60	7.60
	15	272,170	272,170	259,323	4.72	4.72
	20	272,170	271,170	*263,093*	3.33	2.98
M2F30	5	**273,070**	273,070	192,268	29.59	29.59
	10	273,070	270,633	232,993	14.68	13.91
	15	272,660	270,115	252,402	7.43	6.56
	20	272,170	269,915	*258,312*	5.09	4.30
M3F60	5	**273,070**	272,070	143,459	47.46	47.27
	10	272,660	270,640	223,675	17.97	17.35
	15	272,170	269,915	246,960	9.26	8.50
	20	272,170	269,735	*251,911*	7.44	6.61

This table states the results obtained after x minutes of solving instance I2D20 with a new strategy to reduce the number of times the IRMP is solved.

The results for which the upper bound is equal to the best known upper bound for the instance is coloured in green. Note that for all methods (expect for M2F30 for which the results are coloured in orange), this solution is found if the gap between the LRMP value and the overall lower bound is less than 10.00%. For M2F30, the upper bound does not differ much (only 7.43%) from the best found lower bound if we terminate the solution process after 15 minutes. In fact, the outcome (272,660) only differs 0.18% compared to the best found upper bound for the instance (272,170). Furthermore, note that for each method, the difference between the best found lower bound after 20 minutes (results in bold), and the upper bound found after 5 minutes (results in italics), is at most 7.75% for M3F60. This suggests that an extra stop criterion could be used for early termination of the solution process, because good quality feasible solutions can already be obtained within 5 minutes. As the optimality gap between the upper bound and the overall lower bound ($\frac{UB-LB}{UB} \cdot 100\%$) can only be determined after termination of the solution process, we base the new stop criterion on the gap between the LRMP value and the overall lower bound ($\frac{LB_{LRMP}-LB}{LB_{LRMP}} \cdot 100\%$). First, however, we take a closer look at the differences in outcome for the different methods to determine the end time of the recovery period.

I2D20: Disruption of 20 minutes at 12:00 between *Leuvehaven* and *Wilhelminaplein*

Notice that the upper bound solutions provided for M3F60 are not better than those obtained for M2 (all best found upper bounds are equal to 272,170). Also, the highest overall lower bound for M3F60 (in italics) only differs with 7.44% from the best found upper bound for all methods. We therefore suspect that no better solutions can be found by using M3F60 instead of M2, while solutions can be obtained much faster by using M2. This suggest that the estimated end time computed by M2 as end time for the recovery period, can be used as final method to solve the disruptions.

Note that if it turns out that, for other disruptions than tested in this thesis, not enough recovery time is given, the recovery period can always be extended accordingly. Because our final choice is to use method M2 to determine the end time of the recovery period, we base our new stop criterion on the results for this method. The extra stop criterion for the solution method is: *Stop the solution process if the difference between the solution value obtained from solving the LRMP and the overall lower bound is less than 10.0%, and the computation time exceeds 5 minutes.* Up to 5 minutes, we are willing to wait for possible better solutions.

7.3 Final Settings & Recommendations

In the previous sections we showed that the introduction of a neighbourhood of vehicles and drivers and of a recovery period reduces the size of the instance which result in a significant improvement in the computation time. The improvement is mostly noticeable for a disruption occurring on an isolated line. Furthermore, we also showed that the pre-solving method can reduce the time until a feasible solution is found as well as the use of a different strategy in the number of times the IRMP is solved during the solution process.

For the remainder of this thesis, we therefore use a neighbourhood and a pre-solving method to obtain the results. Also, we use M2 to determine the end time of the recovery period, as this method showed no loss of optimality nor feasibility for both instances. Finally, as been concluded in the previous section, we only solve the IRMP at the start of the solution process, if the overall lower bound exceeds 0 for the first time in the solution process, during the pre-solving method or if one of the following stop criteria is met:

- No recovery routes or duties with negative reduced cost can be found.
- The total computation time exceeds the maximum time limit, which is the minimum of the duration of the disruption and 20 minutes.
- The difference between the solution value obtained from solving the LRMP and the overall lower bound is less than 10.0%, and the computation time exceeds 5 minutes.

Note that much more improvement can still be made in the solution method, but optimizing the solution method to its full potential is outside the scope of this thesis. For further research, we however recommend to explore one or more of the possible improvements stated in Chapter 10 (see also Chapter 3 for an extensive literature review).

Chapter 8

Results

In this chapter, we discuss the final results of our solution approach to solve the disruptions of the test cases. For this, we use the final parameter settings as stated in the previous chapter, and the penalty cost and the parameter values for the labour rules and vehicle rules as stated in Chapter 6. This chapter is organized as follows. In Section 8.1, we discuss the final results obtained for instances I1D20 and I2D20. We focus on the solution process itself, as well as the outcome of the solution. Furthermore, in Section 8.2, we perform some sensitivity analysis on the penalty cost. Also, we test our solution approach for the sensitivity of changing the start time of the disruption and the duration of the disruption.

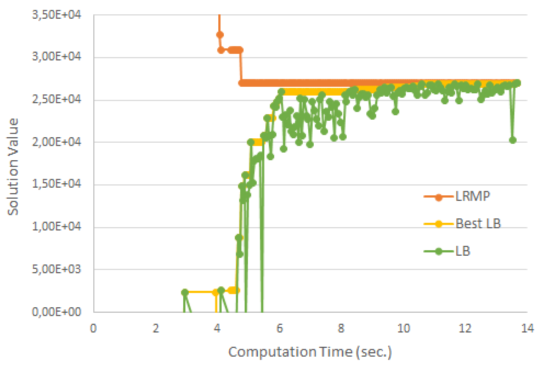
8.1 Final Results

In this section, we discuss the results obtained for instances I1D20 and I2D20. The solution process starts with the initialization phase after a disruption occurs in the original timetable. For instance I1D20, the initialization phase takes 0.5 seconds and for instance I2D20, the task list, delay possibilities, possible detours and recovery networks are set-up within 2.5 seconds. After the initialization phase, the process begins with the pre-solving method, which lasts 2.9 seconds for instance I1D20 and 19.6 seconds for instance I2D20. Thereafter, the solution method continues until one of the stop criteria is met. For instance I1D20, 10.8 seconds after the pre-solving method, no new recovery routes and duties with negative reduced cost could be found and the solution process terminated. For instance I2D20, the solution process terminated 8 minutes and 7.2 seconds after the pre-solving method, because at time, the gap between the best found overall lower bound and the solution value of the LRMP became less than 10.0%.

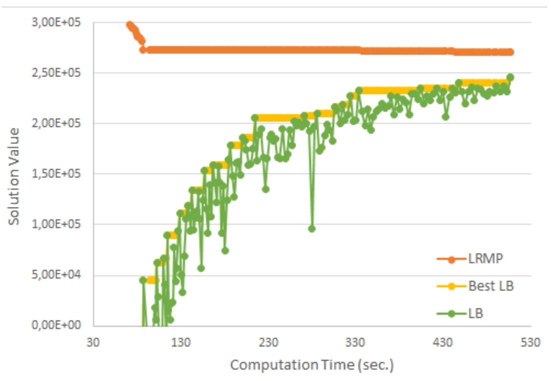
Figures 8.1a and 8.1b show for instances I1D20 and I2D20 respectively, the solution value of the LRMP, the best found overall lower bound and the lower bound of each iteration, against the computation time. In total, the solution process for I1D20 consisted of 163 iterations (of which 19 were performed in the pre-solving method) and the solution process for I1D20 consisted of 213 iterations (of which 13 were performed in the pre-solving method). In the last iteration, 1,471 recovery duties and 729 recovery routes were included in the master problem for instance I1D20. For instance I2D20, this were 14,429 recovery duties and 4,297 recovery routes.

The figures show that towards the end of the solution process, the tailing-off effect occurs for this instance I1D20 and not for I2D20, as we terminated I2D20 the moment that the gap between the best found overall lower bound and the solution value of the LRMP became less than 10.0%. We did not terminate instance I1D20 earlier, while the solution was probably already found after 6 seconds. This is because we are willing to wait up to 5 minutes to be sure that no better solution can be found.

I1D20: Disruption of 20 minutes at 12:00 between *Maashaven* and *Brielselaan*
I2D20: Disruption of 20 minutes at 12:00 between *Leuvehaven* and *Wilhelminaplein*



(a) I1D20



(b) I2D20

Figure 8.1: Solution process

Up to this point, we focused on the computation time of the solution method rather than the outcome in terms of cancelled stops, delayed stops, and the deviation from the original timetable. For instance I1D20 and I2D20, the outcome of the solution is presented by some statistics in Table 8.1. Note that the outcome of the solution approach is a recovery timetable, for which it is hard to present the difference with respect to the original timetable in full extend, if multiple drivers have a recovery duty and multiple vehicles have a recovery route (as is the case for I2D20). Therefore, we only discuss the most important statistics of the recovery timetable.

Table 8.1: Statistics of obtained recovery timetable for I1D20 and I2D20

	Cancelled Stops		Delayed Stops		Delay/ Delayed Stop	Recovery Duty		Recovery Route		Overtime Driver	
	(#)	(%)*	(#)	(%)*	(min.)	(#)	(%)*	(#)	(%)*	(#)	(min./driver)
I1D20	12	2.43	36	7.30	8	1	11.11	3	66.67	0	0
I2D20	36	0.83	582	13.57	7	8	16.67	17	70.83	8	43

This table states some statistics of the recovery timetable obtained by the solution approach, for instances I1D20 and I2D20.
* The percentage is taken from the total number of stops, drivers and vehicles of the instance (see Table 7.3 for M2).

The results show that more rescheduling is needed for I2D20 compared to for I1D20, as more stops need to be cancelled or delayed, which results in more recovery duties and recovery routes. Note that more vehicles need a recovery route than drivers need a recovery duty. This seems strange, as drivers are needed to drive the vehicles, such that changes in the route of the vehicle should also change the duty of the driver. However, recall that the delay penalty is not taken into account for the drivers, which means that if they execute only delayed tasks of their original duty, it is not considered a recovery duty, as the penalty cost for the duty equals 0. So only if the driver performs tasks originally performed by another driver, or if the driver has to work overtime, the duty is considered a recovery duty.

The latter also explains the low percentage of drivers who need a recovery duty in the neighbourhood. The percentage is much higher for the vehicles, indicating that we formed a strong neighbourhood. However, less than 20% of the drivers need a recovery duty, for both instances, which suggest that we could exclude more drivers from the neighbourhood. If we however add those duties for which the penalty cost is 0 and some tasks of the duty are delayed, to the recovery duty already needed for instance I1D20, we get that a total of 4 recovery duties is needed, which brings the percentage to 44.44% of the drivers. This indicates that more drivers were needed to form the recovery timetable than initially thought. If we would like to shrink the neighbourhood, we should introduce different start times and end times for drivers and vehicles, such that drivers and vehicles performing (for example) only some tasks at the end of the neighbourhood are already excluded as their original duty and route will most likely not change.

I1D20: Disruption of 20 minutes at 12:00 between *Maashaven* and *Brielselaan*
I2D20: Disruption of 20 minutes at 12:00 between *Leuvehaven* and *Wilhelminaplein*

Note furthermore that the delay per delayed stop is higher for instance I1D20, which could be explained by the fact that the disruption for this instance occurred on a line which is executed on a lower frequency compared to for the lines on which the disruption occurs in instance I2D20. Therefore, less vehicles and drivers are able to cover for cancellations or delays for instance I1D20, which result in a higher average delay and a higher percentage of cancelled tasks. All conclusions about the outcome should however be interpreted with care because the instances differ in size and many factors could influence the solution outcome. As been discussed, these are factors such as the amount in which interlining occurs within the instance and the number of drivers which have a break or end their duty within the recovery period.

Finally, the results show that only for instance I2D20 drivers need to work overtime. The average overtime per driver with overtime is even 43 minutes. By inspecting the instance, we saw that this is the case because multiple drivers suffer from the disruption right before they end their duty, while this is only the case for a single driver in instance I1D20. For instance I1D20, a detour is taken such that the driver is still arriving on time at its destination and no overtime is needed. Another possibility for this driver (and corresponding vehicle) was to delay its original tasks, which would lead to some overtime for the driver. In the next section, we perform some sensitivity analysis to see if this solution (or other solutions) may be preferred if other values for the penalty cost are used.

8.2 Sensitivity Analysis

In this section, we test the sensitivity of the outcome of our solution approach if we change the penalty cost, the objective function, the duration of the disruption or the start time of the disruption. Note that we do not test the robustness of the outcome by changing any of the parameters, because most labour rules and vehicle rules cannot be changed. Therefore, changing the parameter values from the initial values is not interesting for our research.

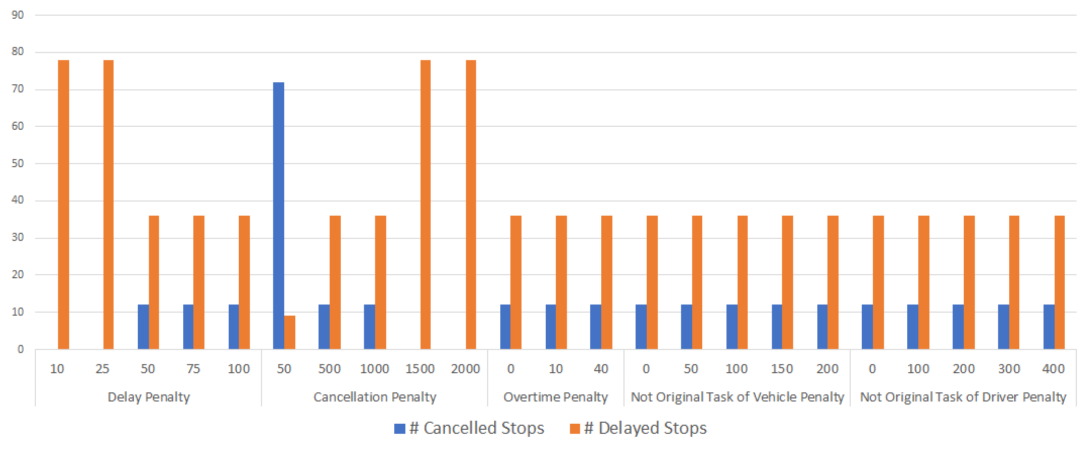
8.2.1 Penalty Cost

The sensitivity on the outcome of our solution approach can be tested in two ways for the penalty cost. Namely, we can change the current penalty cost to different values to test if other recovery timetables are preferred as a consequence of the change. Besides that, we could also make changes in the objective function.

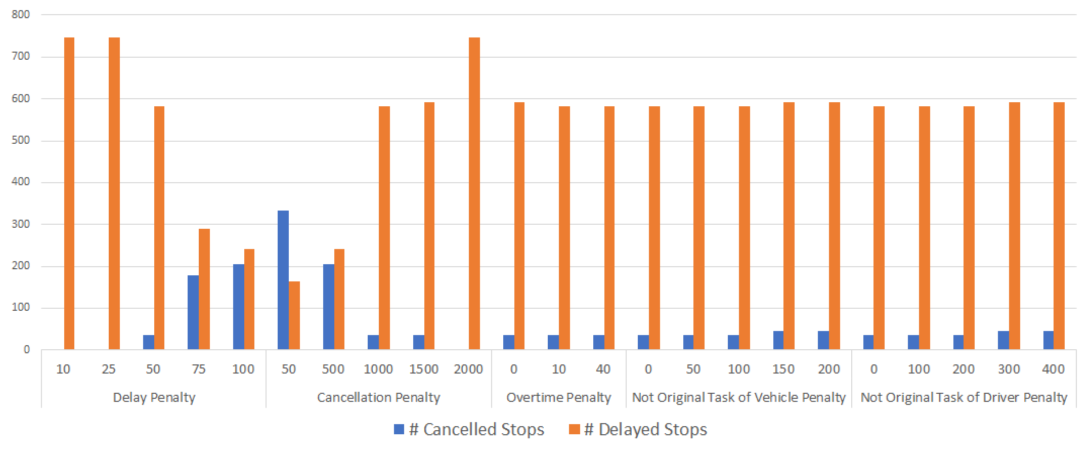
Changing the current penalty cost

In our solution method, we use penalties to make sure that changes to the original timetable are minimized such that it causes the least amount of inconveniences for the drivers and the passengers. By setting the penalty cost, we however made the assumption that this leads to the best recovery timetable, while it could be that changing the penalty cost leads to better solutions from a drivers and passengers perspective. For example, using the penalty cost as done so far, 12 stops are cancelled for instance I1D20, while we already saw that it is possible to form a recovery timetable without the cancellation of any stops. For our research, we do not make any recommendations about which values for the penalty cost should be used, as we believe that this should be further investigated based on preferences of the traffic controllers of the RET. For now, the purpose is to show that different solutions exist by changing the penalty cost and solving the instances I1D20 and I2D20 for these different values. By obtaining the results, we made only a single change in the penalty cost at the time, such that the other penalty costs are set (back) to their initial values, for each run. The results are presented by Figures 8.2, 8.3, 8.4 and 8.5.

I1D20: Disruption of 20 minutes at 12:00 between *Maashaven* and *Brielselaan*
I2D20: Disruption of 20 minutes at 12:00 between *Leuvehaven* and *Wilhelminaplein*



(a) I1D20



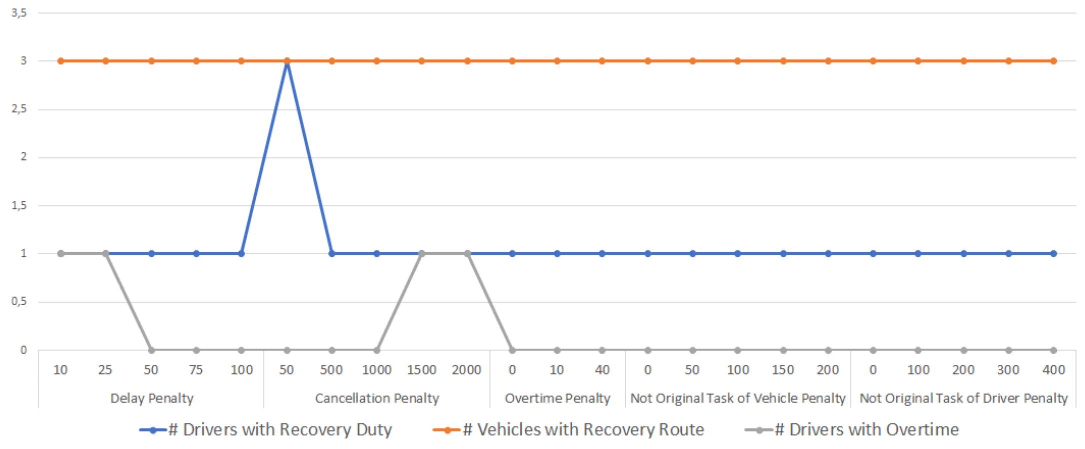
(b) I2D20

Figure 8.2: Number of cancelled & delayed stops

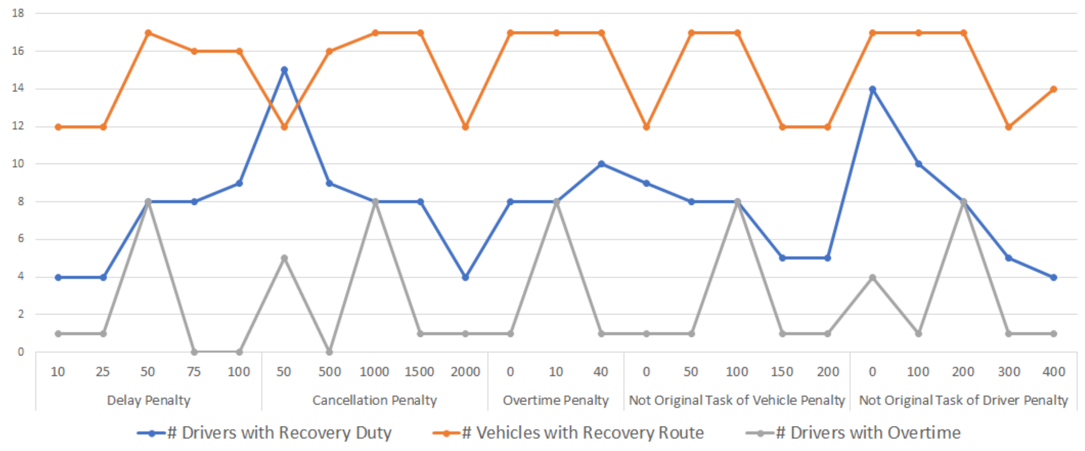
For the delay penalty (p^M), which is initially set to 50 per stop per minute, we change the value to 10, 25, 75 and 100 per stop per minute. The cancellation penalty (p^1), which is initially 1000 per stop, is changed to 50, 500, 1500 and 2000 per stop. For the overtime penalty (p^O) (initially 10 per minute) we change the value to 0 per minute and 40 per minute. The penalty given per stop in a task not included in the original route of the vehicle (p_W^V), or not included in the original duty of a driver (p_W^D), which are initially set to 100 and 200 per stop, are changed to 0, 50, 150 and 200 per stop, and 0, 100, 300 and 400 per stop, respectively.

The figures show that for a higher delay penalty, the recovery timetable contains less delayed stops, more cancelled stops, and more recovery routes and duties. Despite that more drivers need a recovery duty, the number of drivers which have to work overtime decrease if the delay penalty is higher. This is probably due to the fact that drivers (and vehicles) take detours to skip delay tasks. By skipping parts of the original route of the vehicle, the vehicle (and thus also the driver) is more likely to arrive on time at its destination, which consequently results in less overtime. Note furthermore that for instance I2D20 not only the number of delayed stops decreases if the delay penalty is higher, but also the average delay per stop, while this increases for I1D20. Both differences are however very small, and a real cause for these opposite results cannot be found.

I1D20: Disruption of 20 minutes at 12:00 between *Maashaven* and *Brielselaan*
I2D20: Disruption of 20 minutes at 12:00 between *Leuvehaven* and *Wilhelminaplein*



(a) I1D20



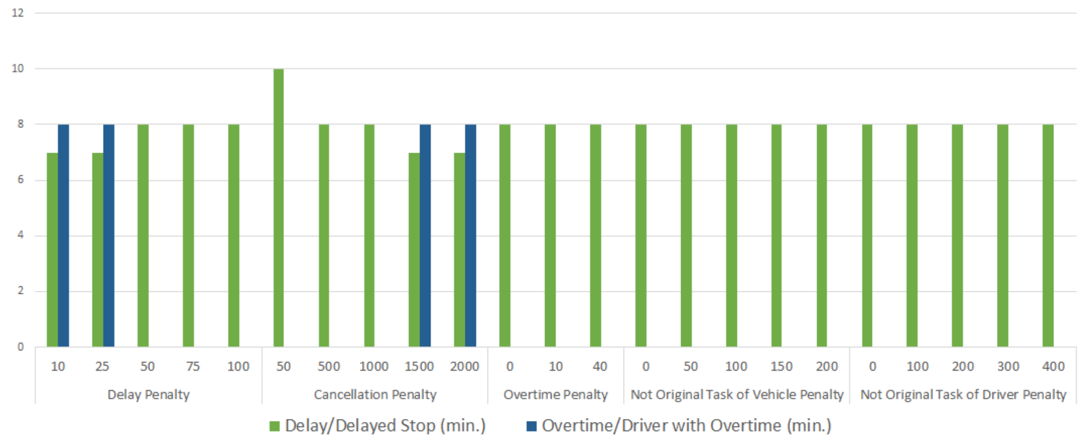
(b) I2D20

Figure 8.3: Number of recovery routes & duties

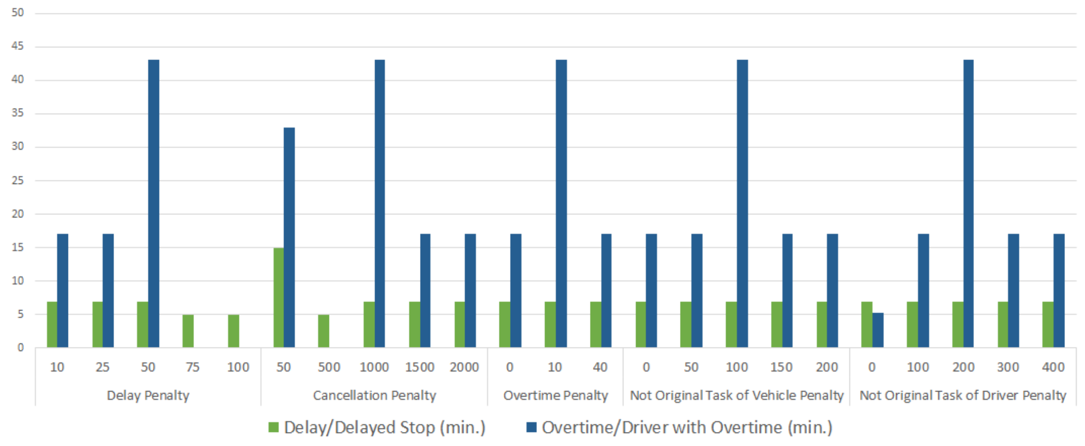
Changing the penalty for the cancellation of stops results for both instances in the most differences in the outcome of the recovery timetable, compared to the runs for which one of the other penalty cost is changed. As expected, the lower the penalty for the cancellation of a stop, the higher the number of cancellations. We also see that the opposite occurs for the number of delayed stops, which is expected as the disruption can either be solved by canceling or delaying a part of the original route of the vehicle. Note that the number of recovery duties increase if the cancellation penalty is lower, which is again due to the fact that a recovery duty is not considered a recovery duty if the driver has to visit the same stops as originally planned even if parts of the duty are delayed, but it is considered a recovery duty if a detour must be taken, overtime is needed or the driver does not perform the same tasks as originally planned.

Changing the other penalty costs does not result in different outcomes for instance I1D20. For instance I2D20, only small (or also no) differences occur for the number of cancelled stops, the number of delayed stops and the average delay per delayed stop. The outcome does differ much for instance I2D20 in terms of the number of drivers with a recovery duty (incl. overtime) and vehicles with a recovery route. Most of these results are completely in accordance with our expectations: the higher the penalty for a certain change from the original timetable, the less occurrence of this sort of change in the recovery timetable, and vise versa.

I1D20: Disruption of 20 minutes at 12:00 between *Maashaven* and *Brielselaan*
I2D20: Disruption of 20 minutes at 12:00 between *Leuvehaven* and *Wilhelminaplein*



(a) I1D20



(b) I2D20

Figure 8.4: Average delay & average overtime (min.)

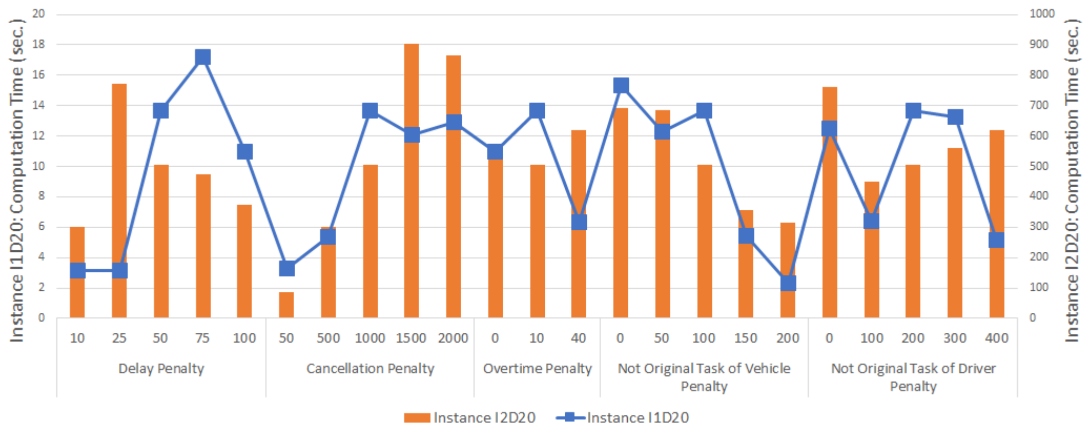


Figure 8.5: Computation time (sec.)

Besides changes in the outcome of the solution, we also noticed differences in the computation time of the solution approach. The most significant change occurred by changing the cancellation penalty to 50. Only for this change in penalty cost, the computation time for instance I1D20 is less than 5 minutes, which means that the solution method ended because no more recovery duties or routes could be found with negative reduced cost. Furthermore, another significant improvement in the computation time occurred if the penalty given per stop in a task not included in the original route of the vehicle was set to a higher value.

I1D20: Disruption of 20 minutes at 12:00 between *Maashaven* and *Brielselaan*
I2D20: Disruption of 20 minutes at 12:00 between *Leuvehaven* and *Wilhelminaplein*

Changing the objective function

Besides changing the penalty cost, we could also change the objective function. Because many changes are possible, and some of them require a lot of changes to implementation of the solution approach, we only discuss some of the promising changes to the objective function without showing any experiments with these ideas.

In our solution approach, the penalty for canceling a task and delaying a task, are multiplied by the number of stops that need to be visited if the task is executed. However, a task with less stops may be evenly important for obtaining a(n) (optimal) solution, as a task with many stops. Therefore, it is interesting to see if changing this in the objective function results in improvements of the solution. Another possibility is to introduce a penalty which takes into account the number of consecutive times a certain line does not visit a stop in the recovery timetable, due to cancellations of tasks. This would make it less likely that passengers at a certain stop, have to wait a very long time before the first vehicle visits this stop after a disruption. Yet another possibility is to penalize the event that a driver has to operate a vehicle other than originally planned.

Furthermore, the delay penalty is only taken into account on the arcs in the network of the vehicles. This is because multiple drivers may execute a single task, such that the delay penalty would be accounted for twice (or more) if we take the delay penalty into account on the arcs of the drivers. Note however that during the pricing problem for the drivers, each (delayed) task is considered the same as its (delayed) copies. This means that the version of the same task with less delay is not more promising than the one with more delay. Therefore, if not both options are explored, it takes a longer time before the right tasks are executed by the driver. Also, even though we may penalize a delay twice (or more) if multiple drivers execute the same task, this does not necessarily mean that this could lead to less interesting solutions. Namely, if multiple drivers execute the same (delayed) task, another task originally performed by one of these drivers may no longer be executed, because in the original timetable not a single task is assigned to multiple drivers.

Finally, it may be interesting to occasionally allow for violation of the labour rules. During our solution approach, we ignore the maximum driving time regulation if a driver is no longer able to adhere this rule. We could however also penalize every minute that the driver does exceed his maximum driving time, such that the violation is minimized. Also, for larger disruptions it may be necessary to allow violation of the break rules, or the restriction on the maximum overtime. In these cases we also would like to minimize the violation. Note that violating the labour rules is not recommended, however if one minute extra overtime could lead to a much better recovery timetables, it may be worthwhile to further analyze this possibility.

8.2.2 Disruption

In this section, we discuss the final two factors influencing the quality of the solution approach, besides the factors discussed in the previous sections and the location of the disruption. The final two factors are the duration of the disruption and the time at which the disruption occurs. The latter is because the number of vehicles and drivers operating during the day differs. We discuss the impact on the size of the instance, the outcome and the computation time of the solution method, by disrupting the timetable at the same locations as done so far, however, we change the start times and the durations of the disruptions. Note that we already discussed the influence of the location of the disruption on the solution method, as up to this point, we compared instance I1D20 with instance I2D20, for which hold that the location differs. From this, we concluded that the more isolated the location and the less interlining occurs on lines passing the location, the faster the solution approach, and the better the final outcome (if the solution process is not terminated before the maximum allowed computation time).

The following extra instances are created to test the solution method:

I1: Disruption between *Maashaven* and *Brielselaan*
- I1.1: Start of disruption at 12:00 (already used before as I1(D20)).
- I1.2: Start of disruption at 8:00.
- I1.3: Start of disruption at 19:00.

I2: Disruption between *Leuvehaven* and *Wilhelminaplein*
- I2.1: Start of disruption at 12:00 (already used before I2(D20)).
- I2.2: Start of disruption at 8:00.
- I2.3: Start of disruption at 19:00.

Table 8.2 shows for each instance, the duration of the disruption (Dur) in minutes, the recovery window of the disruption, the number of stops that must be visited ($\#S$), the number of delay possibilities ($\#S^D$) and the possible detours ($\#O$), the number of drivers ($\#D$) and the number of vehicles ($\#V$) included in the instance. As can be seen, the recovery window extends quickly if the duration of the disruption increases. Also, if the disruption takes place in the morning, the recovery window is on average larger than for the other start times. This again suggest that setting different end times and start times for drivers in the neighbourhood should be considered in further research. Moreover, in practice, we do not prefer to obtain a recovery timetable for a disruption of 30 minutes in the morning, in which after 16:00 (maybe even sooner) still changes in the recovery duties and routes are needed, if not absolutely necessary. This supports the recommendation. The instance sizes further show a huge increase in the number of delay possibilities and the possible detours if the duration of the disruption increases. Also, we see that the longer the duration of the disruption, the more drivers and vehicles are included in the instance.

Table 8.2: Introduction extra instances: instance size

	Dur (min.)	$Window$	$\#S$	$\#S^D$	$\#O$	$\#D$	$\#V$
	10	12:00 - 13:32	83	5	5	3	2
I1.1	20	12:00 - 14:52	493	98	70	9	6
	30	12:00 - 16:46	863	268	141	12	6
	10	08:00 - 10:00	281	32	36	6	5
I1.2	20	08:00 - 11:00	525	165	111	8	6
	30	08:00 - 12:10	877	332	196	10	7
	10	19:00 - 22:05	296	12	11	4	3
I1.3	20	19:00 - 23:33	462	96	26	4	3
	30	19:00 - 01:01	601	182	31	4	3
	10	12:00 - 15:44	1,802	281	70	32	18
I2.1	20	12:00 - 18:51	4,289	855	276	48	24
	30	12:00 - 21:50	5,619	1,841	466	52	26
	10	08:00 - 12:12	2,326	425	316	30	22
I2.2	20	08:00 - 14:44	4,340	1,131	689	43	27
	30	08:00 - 17:59	7,255	2,380	1266	63	33
	10	19:00 - 22:56	1,083	161	32	15	13
I2.3	20	19:00 - 00:36	1,581	346	72	15	13
	30	19:00 - 01:27	2,078	670	191	17	15

This table shows the size of the instances created to test the influence of the start time and the duration of the disruption on the solution method.

The results obtained by the solution method, using the final parameter settings as stated in the previous chapter, and the penalty cost and the parameter values for the labour rules and vehicle rules as stated in Chapter 6, are presented in Table 8.3. The results show that if the disruption lasts 20 minutes, more stops are executed later than originally planned, than would be the case if the disruption only lasts for 10 minutes. If the duration of the disruption equals 30 minutes, the number of delayed stops decreases for most instances compared to the solution for the instances for which the disruption lasts 20 minutes.

We suspect that this is the case because of the frequency of the lines. For example, if a line operates with a frequency of 10 minutes, a task with a delay of 30 minutes will not be chosen if not absolutely necessary, as there are better options to choose from (such as, for example, the next task starting on that location with a delay of 20 minutes, or the task thereafter with a delay 10 minutes). The reason for less delayed stops if the duration increases can also be that the penalty cost are set such that a delay of more than 20 minutes is penalized heavier than the cancellation of that task. Furthermore, the results show that for disruptions with a longer duration, more vehicles need a recovery route, while this is not necessarily the case for the drivers. We do unfortunately notice that even if the disruption is only 10 minutes longer, the number of recovery duties needed to overcome the disruption changes for (almost) every instance. This indicates that if the solution method is used in practice, fairly accurate estimations for the duration of the disruption must be made to make sure that the recovery timetable can be executed. Note that overestimation of the duration is not recommended, as this leads to more cancelled stops.

Table 8.3: Results extra instances

	Dur	Cancelled Stops	Delayed Stops	Delay/ Delayed Stop	Recovery Duty	Recovery Duty with Overtime	Recovery Route	Overtime/ Driver	Computation Time
	(min.)	(#)	(#)	(min.)	(#)	(#)	(#)	(min.)	(sec.)
	10	0	5	7	0	0	1	0	0
I1.1	20	12	36	8	1	0	3	0	14
	30	24	121	9	4	1	5	15	250
	10	0	17	3	0	0	2	0	1
I1.2	20	27	59	3	2	0	4	0	85
	30	70	38	5	4	0	5	0	743
	10	0	12	2	0	0	1	0	0
I1.3	20	12	30	9	1	0	2	0	2
	30	64	21	10	3	0	3	0	6
	10	3	215	4	2	1	6	57	13
I2.1	20	36	582	7	8	8	17	43	507
	30	306	377	8	14	3	19	21	903
	10	3	344	4	2	1	10	4	130
I2.2	20	192	493	7	8	2	15	8	906
	30	x	x	x	x	x	x	x	1,200
	10	0	107	5	3	3	6	6	1
I2.3	20	0	298	9	3	3	9	16	14
	30	175	130	10	12	4	12	20	23

This table shows the results for each instance with a different start time or duration for the disruption.

The results further show that the impact of a disruption on the original timetable is not as large in the evening as would be in the morning. This is mostly because less drivers and vehicles are still active in the evening, but also because less breaks take place such that it is possible to fully delay the original route of the vehicle (and thus the duties of the driver). Notice that this does result in more overtime for the drivers at the end of the day (if the disruption occurs between Leuvenhaven and Wilhelminaplein). Lastly, note that for I2.3 with a duration of 30 minutes, many stops are cancelled at the end of the day. At this moment, we do not apply any policy for cancelling trips which take place at the end of the day. It could be that, for example, the last two trips are never allowed to be cancelled, or that within the last hour at least two trips need to be performed of each line. This should be kept in mind for further research.

Finally, notice that if the size of the instance increases, the solution method does have a much harder time in finding feasible, and optimal, solutions. In case of I2.2 with a duration of 30 minutes, we could not find any feasible solution within 20 minutes. This while for all variations of instance 1, a (close to) optimal solution could be found within 20 minutes, and for most of these variations even within a couple of seconds. This suggest that improvement must be made in either the computation time of the solution method, or in reducing the size of the instance or the number of connections within the networks of the vehicles and drivers.

Chapter 9

Discussion

The goal of this thesis is to find a solution approach that quickly solves the integrated vehicle and crew rescheduling problem with delay possibilities, such that it could support the traffic controllers of the RET by solving real life disruptions in the tram network. In the previous chapters, we explained our solution approach and we discussed the results. The results indicated that the solution method works, especially for small disruptions. However, the solution method also has some limitations. We will state the limitations of the solution method in Section 9.1. In Section 9.2, we discuss if and how the proposed solution method could be used in practice.

9.1 Limitations

The results presented in the previous chapters show that relatively small disruptions can be solved to optimality within reasonable time. Unfortunately, it is hard to predict if a disruption is small, because the impact on the original timetable differs if the location of the disruption, the duration of the disruption or the start time or the disruption is different. Therefore, a limitation of our solution approach is that we are unable to predict the time needed for the solution approach to at least obtain a feasible solution, and preferably an optimal solution. Especially for large disruptions, a feasible solution may not even be found within 20 minutes.

Another limitation of the solution approach is the fact that we cannot guarantee optimality for the overall problem, because we need to exclude drivers and vehicles such that at least a feasible solution can be obtained in reasonable time (introduction of a neighbourhood). By introducing a recovery period, the solution approach can also not guarantee feasibility if the recovery period is chosen too short. Note that loss of optimality is not an issue if it saves a lot of computation time while maintaining a good quality solution, as we are mostly interested in a good feasible solution. Also, if loss of feasibility occurs, the lower bound of the solution will quickly increase (and may even exceed the high penalty set for not having a recovery route or duty in the recovery timetable). If this is noticed early in the solution process, it is recommended to terminate the solution process and extend the recovery period.

Furthermore, the solution approach is also limited by the used penalty costs and the parameter values. The limitation of the penalty costs is that we may not have chosen the correct values to fully capture our preference of the recovery timetable. Many (almost similar) solutions exist, such that the solution process may have a hard time to find the optimal solution. Changing the penalty cost (or objective function) could improve the computation time, such that feasible solutions can be obtained earlier in the solution process, and optimal solutions may even be found within the maximum computation time. Another limitation is created by some of the parameter values, or more specifically, the strictness in satisfying the rules related to these parameter values. For example, a solution with one minute more overtime than the maximum allowed overtime will not be considered, while this may prevent cancellation of multiple trips.

Moreover, at this moment, we also do not apply any rules related to the number of trips that are allowed to be cancelled within a certain time frame. It may be that the recovery timetable obtained from our solution approach, cancels all trips at the end of the day. This will not be done in practice due to passengers waiting at these stops, which need to be able to continue their journey. A similar argument holds for consecutive trips on a single line during the day.

Finally, the greatest limitation of our solution approach is the computation time (for larger instances). However, as stated in Section 7.3, optimizing the solution method for the fastest computation time for all sorts of disruptions is outside the scope of this thesis.

9.2 Practical Use of Solution Method

The purpose of our solution method is that it should support the traffic controllers of the RET by providing a recovery timetable which reschedules the drivers and vehicles, to account for the occurred disruption. At this moment, the traffic controllers of the RET do not make use of any optimization tool at all to solve the disruptions. This can lead to violation of the labour rules, as well as canceling or delaying trips while this may be unnecessary. Also, many vehicle and drivers could be affected by a disruption as many duties pass by the same locations and disrupted duties may propagate through the consecutive duties, because of crew interlining. This makes solving a large disruption almost impossible for the traffic controllers of the RET, such that a disruption in the morning might still have a disruptive effect in the afternoon.

We believe that the implementation of our solution method in a decision support tool could definitely be used to solve small disruptions, and to even obtain optimal solutions for these disruptions. For larger disruptions, we believe that even though optimal solutions may not be obtained, feasible solutions are already close to optimal solutions such that the traffic controllers could adjust the obtained recovery timetable based on their own knowledge if necessary. Furthermore, even if a feasible solution is not obtained, it could be that this is only for some of the drivers, such that the traffic controllers could still manually reschedule those drivers for which the solution approach could not find any recovery duty. Note that this can also be done (some time) after the initialization phase for relatively smaller disruptions, as we exclude drivers and vehicles for which no changes in the routes or duties are allowed, by the introduction of a neighbourhood or a recovery period.

Chapter 10

Recommendations for Further Research

In the previous section we stated some limitations of our solution method. Therefore, in this section, we make some suggestions for further research to improve the solution method.

Note that the greatest limitation of our solution approach is the computation time (especially for large disruptions). In recent years, many research has been done to improve computation time of column generation. For example, Huisman et al. (2005) propose to use column generation in combination with Lagrange relaxation and a sub-gradient optimization to overcome the tailing-off effect. Another way to reduce the computation time is to implement dynamic constraint and/or variable aggregation as proposed by Elhallaoui et al. (2008). Using such an implementation means that the task list, delay possibilities and detour possibilities are first merged to larger tasks (if possible), after the initialization phase. The tasks are merged if we expect that the same driver and vehicle are still performing this task in the optimal recovery timetable. If a task is cancelled, only then the task is split (and also other tasks in the neighbourhood of this task) into smaller pieces or allowed to be delayed (depending on the implementation of the method). Towards the end of the process, more and more possibilities are added until all possibilities can be chosen as part of a recovery duty or route, such that optimality is guaranteed. Another implementation could be to ignore all (or a part of) the constraints from the master problem, and slowly obeying more and more constraints towards the end of the solution process until all constraints are satisfied. This makes solving the master problem easy during the beginning of the solution process, which saves a lot of computation time.

Note that besides improving the solution approach for faster convergence, it may also be interesting to further improve the initialization phase. By analyzing the network structure of the drivers and vehicles we may exclude connections which are not promising at all. If less connections between tasks need to be considered, it could lead to less possibilities for new recovery routes and duties, which consequently results in a reduction of the computation time of the pricing problem. Also, at this moment we set the same end time and start time for each vehicle and driver (if they start before and end after the recovery period), while it may be convenient to set different end times (and/or start times) for each driver and vehicle based on its relation towards the disruption, such that the instance possibly includes less drivers and vehicles and thus less tasks. For example, a driver starting almost at the end of the recovery period will most likely not change its duty. In this case, it is better to exclude this driver from the neighbourhood.

Besides these recommendations, we also recommend to explore the possibility of using a sequential approach for larger instances. This is already done by van Meer et al. (2019), for the same data as used in our thesis, and they obtained good results (larger disruptions can be solved within 20 minutes). The main issue with a sequential approach is the fact that the solution obtained for the vehicle rescheduling problem may not be feasible for the crew rescheduling problem.

Therefore, we suggest to make it possible that the solution obtained for the vehicle rescheduling problem can still be changed after solving the crew rescheduling problem. If no better solutions can be found, and the computation time has not exceeded the maximum computation time, we could still use our solution method to possibly obtain even a better solution, or to prove that the found solution is the optimal solution.

The other recommendations are mainly focused on the practical use of the solution method. Namely, at this point, the recovery timetable is only allowed to deviate from the original timetable from the moment the disruption occurs. It may however be interesting to see if better solutions can be obtained if the recovery timetable is allowed to deviate earlier than the disruption has occurred. So, for example, it may be that a traffic controller suspects at 11:00 a disruption at 12:00 for 20 minutes, because of some inside information about an event taking place during that time at that location. Instead of changing the recovery routes and duties of the vehicles and drivers at the moment the disruption occurs, less cancellations (and delays) are probably necessary if we are allowed to make changes earlier.

Also, the solution method does not consider the time needed to come up with the recovery timetable. This means that if a disruption occurs at 12:00, the recovery timetable may change a duty or route such that at 12:01 already deviation is needed from the original timetable. If the total computation time exceeds for example 10 minutes, the traffic controllers of the RET are not able to notify the driver of the change in time such that the driver may be at a totally different location at 12:10 than was expected. Consequently, the recovery timetable cannot be used as drivers and vehicles are not at the specified start locations, as used in the solution method, anymore. For further research, we recommend to include (an estimation of) the computation time needed to solve the disruption, such that within this period, drivers and vehicles are not allowed to perform tasks from other drivers or vehicles. Another possibility is to make use of online data, which is updated throughout the solution process. Besides, we could also very quickly construct a solution for the first x minutes after the disruption such that thereafter, we have more time to solve the disruption from this moment up to the end of the recovery period.

Lastly, the solution method also showed some limitations by changing the penalty cost, parameter values and the start time and duration of the disruption. In our research, we only presented this for some changes, which showed that different solutions for the same instance can be obtained, and that the computation time of the solution method is highly dependent on changes of one of the factors (some more than others). Therefore, we recommend to make a simulation tool to test the robustness on the data, which may be used to select a specific solution approach for every possible disruption. We also recommend to ask advice about the preferences for different recovery timetables from the traffic controllers of the RET, to obtain the best recovery timetable from all feasible recovery timetables.

Chapter 11

Concluding Remarks

The goal of this thesis is to develop a decision support system to support the traffic controllers at the RET. The tool must be able to reschedule vehicles and crew members the moment that a disruption occurs in the original planned timetable, in a very short amount of time. As previous research already focused on sequential approaches to solve the vehicle and crew rescheduling problem, this thesis focuses on the development of an integrated approach. Since many duties consist of trips that need to be performed on different vehicles (and multiple lines), a disruption occurring on a single line affecting a single vehicle/driver can propagate to consecutive duties, and vehicles and lines not even passing the disruption.

To solve the integrated vehicle and crew rescheduling problem with retiming, we used a column generation approach in which the columns correspond to recovery routes and duties. These columns are found by solving a (resource constrained) shortest path problem in combination with a labeling procedure. It turned out that the computation time was still too long and therefore, we also proposed to use a neighbourhood of vehicles and drivers and a recovery period in which tasks are allowed to be changed. Furthermore, for faster convergence to a feasible solution, we implement a pre-solving method and to only solve the IRMP during certain moments in the solution process, instead of every time a better overall lower bound is found.

The results showed that huge improvements were made in the reduction of the instance size by introducing a neighbourhood of vehicles and drivers for which changes in the original route and duty are allowed. This consequently led to a huge improvement in the computation time. Furthermore, this was also the case for the introduction of a recovery period. The reduction was more noticeable for small instances, for which less vehicles and drivers are affected by the disruption. For larger disruptions (disruptions for which more vehicles and drivers are affected, and crew interlining occurs in the duties of the drivers), the solution method had a hard time in even finding feasible solutions. Fortunately, the pre-solving method, solving the IRMP only a couple of times during the solution method, and an extra stop criteria based on the gap between the objective value of the LRMP and the overall lower bound of the instance, reduced the computation time for larger disruptions such that at least a feasible solution can be obtained (if the disruption is not still too large).

Using the final parameter settings, we saw that small disruptions can be solved to optimality within a very short amount of time, and for most larger instances, also feasible solutions can quickly be obtained. The results showed that changing the penalty cost can change the computation time of the solution method as well as the outcome of the solution method. We did not make any recommendations for which penalty cost to use, as we believe that this should be further discussed with the traffic controllers of the RET, based on their preferences for different timetables. Note that we also encountered different computation times and outcomes by making changes in the duration of the disruption and the start time of the disruption.

This brings us by the limitations of the solution method. Namely, for practical use, accurate estimations for the duration of the disruption and the start time of the disruption must be made in order to find the best possible recovery timetable. Overestimation of the duration may for example lead to cancellation or delaying some stops, while this is not necessary. Also, the fact that changing the penalty cost leads to different outcomes indicates that preferences for the different recovery timetables should be further analyzed, as we may not have captured all preferences at this point. Moreover, another limitation of our solution method is that we are not able to guarantee optimality of the overall problem, as we make use of a neighbourhood and a recovery period. The latter may even lead to infeasibility of the created instance, however this can easily be solved by extending the recovery period. The greatest limitation of our solution method is the computation time for larger instances, as for some, not even feasible solutions can be obtained within the maximum computation time of 20 minutes, while for small instances, optimal results can be obtained within a few seconds. We do however see much more potential for the solution method if other (extra) methods to reduce the computation time are used.

In conclusion, improvement can still be made in (mostly the computation time of) the solution method by exploring the recommendations. However, despite the limitations, the solution method can be used to solve small disruptions, and it is able to support the traffic controllers of the RET for large disruptions, affecting many vehicles and drivers.

Bibliography

Abouelrous, A., de Pater, I., Vrakidis, P., and van den Puttelaar, R. (2019). A multi-depot vehicle and crew rescheduling approach for dynamic disruption management in a tram network.

Bettinelli, A., Santini, A., and Vigo, D. (2017). A real-time conflict solution algorithm for the train rescheduling problem. *Transportation Research Part B: Methodological*, 106:237–265.

Blokland, R., Salem, R., Stuyling de Lange, V., and Verstraete, M. (2019). Real-time vehicle rescheduling for a disrupted tram network.

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.

Dávid, B. and Balogh, J. (2016). An algorithmic framework for real-time rescheduling in public bus transportation. *matcos*, 29:7.

Dávid, B. and Krész, M. (2014). A model and fast heuristics for the multiple depot bus rescheduling problem. In *10th international conference on the practice and theory of automated timetabling (PATAT)*, pages 128–141. sn.

Dávid, B. and Krész, M. (2017). The dynamic vehicle rescheduling problem. *Central European Journal of Operations Research*, 25(4):809–830.

Elhallaoui, I., Desaulniers, G., Metrane, A., and Soumis, F. (2008). Bi-dynamic constraint aggregation and subproblem reduction. *Computers & Operations Research*, 35(5):1713–1724.

Huisman, D. (2004). *Integrated and dynamic vehicle and crew scheduling.* Number 325.

Huisman, D., Freling, R., and Wagelmans, A. P. (2004). A robust solution approach to the dynamic vehicle scheduling problem. *Transportation Science*, 38(4):447–458.

Huisman, D., Jans, R., Peeters, M., and Wagelmans, A. P. (2005). Combining column generation and lagrangian relaxation. pages 247–270.

Kiefer, A., Kritzinger, S., and Doerner, K. F. (2016). Disruption management for the viennese public transport provider. *Public Transport*, 8(2):161–183.

Kunst, N., Meijer, B., Smidt, K., and Witsen, O. (2019). Two-phase approach for disruption management in the ret tram network.

Li, J.-Q., Borenstein, D., and Mirchandani, P. B. (2007a). A decision support system for the single-depot vehicle rescheduling problem. *Computers & Operations Research*, 34(4):1008–1032.

Li, J.-Q., Mirchandani, P. B., and Borenstein, D. (2007b). The vehicle rescheduling problem: Model and algorithms. *Networks: An International Journal*, 50(3):211–229.

Li, J.-Q., Mirchandani, P. B., and Borenstein, D. (2009). A lagrangian heuristic for the real-time vehicle rescheduling problem. *Transportation Research Part E: Logistics and Transportation Review*, 45(3):419–433.

Lückerath, D., Ullrich, O., and Speckenmeyer, E. (2013). Applicability of rescheduling strategies in tram networks. *Proceedings of ASIMWorkshop STS/GMMS 2013*.

Malucelli, F. and Tresoldi, E. (2019). Delay and disruption management in local public transportation via real-time vehicle and crew re-scheduling: a case study. *Public Transport*, 11(1):1–25.

Meng, L. and Zhou, X. (2014). Simultaneous train rerouting and rescheduling on an n-track network: A model reformulation with network-based cumulative flow variables. *Transportation Research Part B: Methodological*, 67:208–234.

Potthoff, D. (2010). *Railway crew rescheduling: Novel approaches and extensions.* Number EPS-2010-210-LIS.

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

van de Pol, B., Mariman, Y., Rutgers, N., and Ettema, M. (2019). Vehicle and crew rescheduling in case of disruptions in a tram network.

van Dockum, T. (2018). Crew rescheduling in the tram network of rotterdam.

van Lieshout, R., Mulder, J., and Huisman, D. (2018). The vehicle rescheduling problem with retiming. *Computers & Operations Research*, 96:131–140.

van Meer, F., van Paassen, M., Chinnasamy, S., and Stoop, T. (2019). A new approach to disruption management at ret.

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.

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.

Yu, G. and Qi, X. (2004). *Disruption management: framework, models and applications.* World Scientific.

List of Abbreviations

RET	Rotterdamse Elektrische Tram
MRDH	Metropoolregio Rotterdam Den Haag
VRSP	Vehicle Rescheduling Problem
CRSP	Crew Rescheduling Problem
VCRSP	(Combined or Integrated) Vehicle and Crew Rescheduling Problem (with Retiming)
RMP	Restricted Master Problem
IRMP	Integer Restricted Master Problem
LRMP	Linear Restricted Master Problem

List of Symbols

Labour rules for the crew members:

b_{min}	minimum number of breaks
l^{b}_{min}	minimum duration of a break
h^{duty}_{min}	minimum duration of a duty which must include the minimum number of breaks
h^{work}_{max}	maximum allowed working time (consecutive driving time) between two breaks
h^{duty}_{max}	maximum allowed duty length including break time
o^{D}_{max}	maximum time a driver is allowed to work overtime
T_{travel}	default travel time between two relief locations for a driver

Vehicle rules:

o^{V}_{max}	total minutes a vehicle is allowed to arrive at the depot later than originally planned

Vehicle and Driver Network:

N	set of all tasks consisting of (cut)trips, splitted by relief locations and locations at which detour can be taken
V	set of all vehicles
$S^V = \cup_{v\in V}\ s^v$	set of all source nodes s^v of vehicles $v \in V$
$T^V = \cup_{v\in V}\ t^v$	set of all sink nodes t^v of vehicles $v \in V$
st_i	start time of node i
et_i	end time of node i
tt_{ij}	travel time from node i to node j
sl_{ij}	slack time between node i and j
w^{max}_i	maximum allowed waiting time at node i
w^{MAX}	maximum allowed waiting time at all locations
$G = (\mathcal{V}, A)$	the vehicle network without retiming, where $\mathcal{V} = S^V \cup N \cup T^V$ and A represents the set of all arcs over which the vehicles can travel
$\mathcal{R}^v$	set of all vehicle routes of vehicle $v \in V$
D	set of all drivers
$S^D = \cup_{d\in D}\ s^d$	set of all source nodes s^d of drivers $d \in D$
$T^D = \cup_{d\in D}\ t^d$	set of all sink nodes t^d of drivers $d \in D$
D^s	set of all drivers with a split duty
$R = \cup_{d\in D}\ r^d$	set of all rest nodes r^d of drivers $d \in D^s$
$G^d = (\mathcal{V}^d, A^d)$	the network of driver $d \in D$ without retiming, where $\mathcal{V}^d = s^d \cup S^V \cup N \cup T^V \cup t^d$ and A^d represents the set of all arcs over which driver d can travel
$\mathcal{R}^d$	set of all recovery duties of driver $d \in D$

Introducing retiming:

q_i^{max}	maximum allowed delay for node i
N_e	set of all tasks including delay possibilities
$N_e(k)$	set of all (delay) tasks of task $k \in N$
$G_e = (\mathcal{V}_e, A_e)$	vehicle network with delay possibilities, where $\mathcal{V}_e = S^V \cup N_e \cup T^V$ and A represents the set of all arcs over which the vehicles can travel
$G_e^d = (\mathcal{V}_e^d, A_e^d)$	the network of driver $d \in D$ with delay possibilities, where $\mathcal{V}_e^d = s^d \cup S^V \cup N_e \cup T^V \cup t^d$ and A^d represents the set of all arcs over which driver d can travel

Mathematical formulation of the VCRSP

We define the following sets:

N	set of all tasks consisting of (cut)trips, splitted by relief locations and locations at which detour can be taken
N_e	set of all tasks including delay possibilities
$N_e(k)$	set of all delay copies of task $k \in N$
A_e	set of all arcs over which vehicles and/or drivers can operate
$A_e(k)$	set of all delay arcs associated with task $k \in N$
D	set of all drivers
$\mathcal{R}^d$	set of all driver duties of driver $d \in D$
V	set of all vehicles
$\mathcal{R}^v$	set of all recovery routes of vehicle $v \in V$

We define the following parameters:

$Q_i \in \mathbb{N}$	number of scheduled stops at which the vehicle must stop for task $i \in N$
$a_{i\delta}^d \in \mathbb{B}$	indicator if task $i \in N_e$ is included in recovery duty $\delta \in \mathcal{R}^d$ of driver $d \in D$
$b_{i\delta}^d \in \mathbb{B}$	indicator if task $i \in N$ is included in recovery duty $\delta \in \mathcal{R}^d$ of driver $d \in D$
$a_{i\delta}^v \in \mathbb{B}$	indicator if task $i \in N_e$ is included in recovery route $\delta \in \mathcal{R}^v$ of vehicle $v \in V$
$b_{i\delta}^v \in \mathbb{B}$	indicator if task $i \in N$ is included in recovery route $\delta \in \mathcal{R}^v$ of vehicle $v \in V$

We define the following variables:

$z_i \in \mathbb{B}$	indicator if task $i \in N$ is cancelled
$y_\delta^d \in \mathbb{B}$	indicator if recovery duty $\delta \in \mathcal{R}^d$ is chosen as recovery duty for driver $d \in D$
$e_d^D \in \mathbb{B}$	indicator if for driver $d \in D$ no recovery duty can be found
$x_\delta^v \in \mathbb{B}$	indicator if recovery route $\delta \in \mathcal{R}^v$ is chosen as recovery route for vehicle $v \in V$
$e_v^V \in \mathbb{B}$	indicator if for vehicle $v \in V$ no recovery route can be found

We define the following penalties:

M	penalty given to a vehicle or driver if the vehicle or driver does not have any recovery route or duty in the recovery timetable
p^1	task cancellation penalty for each missed scheduled stop
p^M	penalty for every minute delay for each stop
p_W^V	penalty for every new visited stop compared to the original route, for a vehicle
p_W^D	penalty for every new visited stop compared to the original duty, for a driver
p^O	penalty for every minute of overtime compared to the original planned end time of the duty of a driver
f_δ^d	total penalty cost of recovery duty δ, for driver d
g_δ^v	total penalty cost of recovery route δ, for vehicle v

Restricted Master Problem:

In case of (I)RMP: All sets, parameters, variables and penalties used to formulate VCRSP

In case of LRMP: (I)RMP but with relaxation of the variables

K^v	set of recovery routes of vehicle $v \in V$ included in the master problem
K^d	set of recovery duties of driver $d \in D$ included in the master problem
LB_{LRMP}	objective value obtained from solving the LRMP

Pricing Problem:

λ_k	the dual variable for task $k \in N$, of the constraints (5.2)
ϕ_k	the dual variable for task $k \in N$, of the constraints (5.3)
μ_i	the dual variable for task $i \in N_e$, of the constraints (5.4)
γ_i	the dual variable for task $i \in N_e$, of the constraints (5.5)
α_i	the dual variable for task $i \in N_e$, of the constraints (5.6)
π^v	the dual variable for vehicle $v \in V$, of the constraints (5.7)
π^d	the dual variable for driver $d \in D$, of the constraints (5.8)

Overall Lower Bound:

K^v	set of recovery routes of vehicle $v \in V$ included in the master problem
K^d	set of recovery duties of driver $d \in D$ included in the master problem
$\hat{R}^v$	set of recovery routes of vehicle $v \in V$ **not** included in the master problem, with negative reduced cost
$\hat{R}^d$	set of recovery duties of driver $d \in D$ **not** included in the master problem, with negative reduced cost
LB_{LRMP}	objective value obtained from solving the LRMP
LB	overall lower bound of the VCRSP or the (I)RMP

Improving Computation Time:

F	fixed part of the recovery period
x	maximum computation time of the solution method (in minutes)