Master Thesis Operations Research and Quantitative Logistiscs

Robust railway crew schedules

September 7, 2010

Author:
Dennis Vlugt
- 302463 -

Supervisor:
Dr. D. Huisman
Co-reader:
Prof. dr. A.P.M. Wagelmans

Abstract

On an intensively used network as the Dutch railway network, the problem of crew rescheduling is hard. Currently at NS, when disruptions occur there is no quantitative support for dispatchers to reschedule the duties. However, the manual approach of the dispatchers is not effective nor efficient. Therefore, operations researchers developed several methods that can support dispatchers in decision making. One of them, an approach using Column Generation, is very promising and is extensively tested in this thesis together with an heuristic approach that mimics the manual approach of the dispatchers and an agent-based approach.

To test the three methods several disruption cases are constructed and combined with the schedules, either with or without reserve duties. After the comparison of the methods, the set of schedules is extended with some additional schedules. One of them is a schedule that is constructed as emergency schedule that can be used during extreme situations. Besides schedules, also different sets of stand-by duties are proposed. These new sets can be divided into two categories, where one is based on a location perspective and the other is based on a time perspective. Finally, we extended the scenarios such that a more realistic presentation of the practical situation, where multiple serious disruptions a day occur, is obtained.

From the analysis it becomes clear the method based on Column Generation is in general the best approach. However, the agent-based approach performs particular well on smaller instances, sometimes even better than the Column Generation approach. The heuristic approach does not perform well. Only rescheduling a schedule like the emergency schedule with the heuristic provides reasonable results. Apart from the emergency schedule there is very little difference in robustness between the schedules for all rescheduling methods. Robustness of a schedule is measured as the way schedules can be adapted after disruptions. Reserves make the process of rescheduling more easy, but the Column Generation approach performs also good without reserves. So the use of a good rescheduling method seems to contribute more to a robust railway crew schedule than changing the properties of a schedule.

Acknowledgements

This thesis has been written in order to finish the Master Operations Research and Quantitative Logistics (specialization of Econometrics and Management Science) at Erasmus University Rotterdam. This thesis is a result of a six months internship at NS, where I worked at the Innovation department of NSR (a division of NS). I enjoyed working there and I would like to thank the persons that helped and supported me with this thesis.

First of all, I would like to thank Dennis Huisman. Dennis arranged my internship at NS. Furthermore, his suggestions helped me with improving my work and at the same time he provided space to set up my own research. This contributed a lot into keeping me motivated to the end. I was very pleased to have Dennis as my supervisor. I would also like to thank my co-reader, Albert Wagelmans, for taking time to read and judge my work.

Then, I would like to thank NS to provide me an internship in a nice and learningful environment. I want to thank my colleagues at the Innovation department for having a great time. Especially, I would like to thank Daniel Potthoff, who seriously contributed to this work by helping me out with preparing the methods he developed such that I could use them for the experiments I designed. Furthermore, I thank Pieter-Jan Fioole for his support with the agent model. I am grateful to Erwin Abbink and Twan Dollevoet, because they constructed the schedules used for this thesis.

My thanks go out to Joris for his comments and, last but not least, I would like thank Sietske for her interest and support and for letting her critical eye go over this thesis.

Contents

1 Introduction

Many Europeans travel frequently by train, either to commute or in their leisure time. Therefore, the operational performance of railway systems is often discussed in the public debate. Travelers expect to arrive at a specific time at their destination. If they travel by rail, they expect to arrive more or less at the time published in the timetable. However, unforeseen events often take place, which cause delays or even cancellations of trains. As a result, passengers arrive later than expected at their final destinations. Due to missed connections, the delay of a passenger can be even much larger than the delays of the individual trains. To deal with these kinds of problems, railway operators can use mathematical models to support the decision making.

Netherlands Railways (in Dutch: Nederlandse Spoorwegen or NS) is a good example of a company that successfully uses Operations Research (OR) techniques in order to solve numerous problems in passenger railway transportation. One of the successful applications led to the introduction of a completely new timetable on the Dutch railway network in 2006, for which the NS recently won the Franz Edelman Award ([14]). But constructing a timetable is not the only operational planning problem that has to be overcome. Given a timetable, consecutively rolling stock and crew have to be scheduled ([10]). For constructing the annual crew schedule plan NS uses the solver LUCIA, which is part of the CREWS package, in which several OR techniques are implemented ([1] and [13] discuss the crew scheduling problem at NS in more detail).

Besides solving planning problems, also on the day of operation serious challenges have to be overcome. For example, disruptions can have huge influence on the execution of the schedules by causing (huge) delays. In case of disruptions, dispatchers sometimes are under large pressure, since they need to determine a new schedule for the train drivers in a short amount of time. Sometimes this is impossible, with an extreme case in December 2009 where the Dutch railway network incurred a lot of disruptions due to the weather circumstances, leading to many canceled trains. During the whole period, dispatchers were far behind in rescheduling the crew, which made the situation even worse. Especially at these moments, the use of innovative approaches to make quick decisions is required.

Researchers developed several approaches for rescheduling the crew within in a short amount of time. In this thesis we will analyze the different methods for rescheduling, especially their effect on the robustness of the crew schedules. The robustness of the crew schedules is a very important factor for NS. The robustness of a crew schedule can be interpreted in two ways [12]. First, a schedule is seen as robust when delays to a certain level are absorbed without necessity of rescheduling. These delays are a result of little disruptions or are a result of delays caused by disruptions that impose delays elsewhere in the schedule (so called secondary delays or propagation of delays). The way delays are propagated through the schedule is called the absorption ability of the crew schedule. Second, when rescheduling is necessary, a crew schedule is seen as robust if the schedule can (easily) be adapted to the new situation caused by disruptions. These disruptions concern mostly severe disruptions that affect the normal timetable for some amount of time, mostly longer than 30 minutes.

At NS there is currently little insight in the robustness of crew schedules and the corresponding factors that determine whether a schedule is robust or not. The goal of this thesis is to analyze the performance of the different rescheduling methods given a number of disruption scenarios and crew schedules. Furthermore, after gaining insight in what determines the robustness of a schedule, our final goal is to find robust schedules, preferably without changing the rules of the currently implemented production plan.

Besides having a robust crew schedule for the normal situation on the day of operation, dispatchers are also interested in finding an extremely robust schedule that could be implemented when there are large problems, for example during the winter of 2009, on the railway network. In such a schedule, drivers should be assigned to a small part of the railway network and execute the same task during his or her whole duty. Our goal is to find out how expensive such a schedule will be and whether it really makes a difference for the rescheduling process or not.

The outline of this thesis is as follows. First, in chapter 2, we describe in detail the way NS currently solves crew scheduling problems. Besides that, the rescheduling approaches developed at NS are introduced. In chapter 3 there is a review of the available literature. Chapter 4 deals with the design of the experiments for comparing the methods. In chapter 5 the results are presented. With the information of chapter 5 more schedules and scenarios to test the schedules are introduced in chapter 6. The results obtained with these schedules are outlined in chapter 7. Finally, in chapter 8 we end with conclusions and we point out some directions for further research.

2 Problem description

In this thesis we will focus on the situation at NS, the main railway operator in the Netherlands, having the exclusive right to operate passenger trains on the so-called Dutch Main Railway Network until 2015. First we will discuss the way NS is dealing with crew scheduling, followed by the problem of rescheduling.

2.1 Crew scheduling at NS

NS operates a set of lines, where a *line* is designed as a *route* between a start and an end station. A line has a number of intermediate stops and is operated with a certain frequency, e.g. once or twice per hour. To illustrate a line, the route of the 500 intercity line from Groningen (Gn) to The Hague (Gvc) with stops in Assen (Asn), Zwolle (Zl), Amersfoort (Amf), Utrecht (Ut) and Gouda (Gd) is shown in Figure 2.1, together with the whole network NS operates.

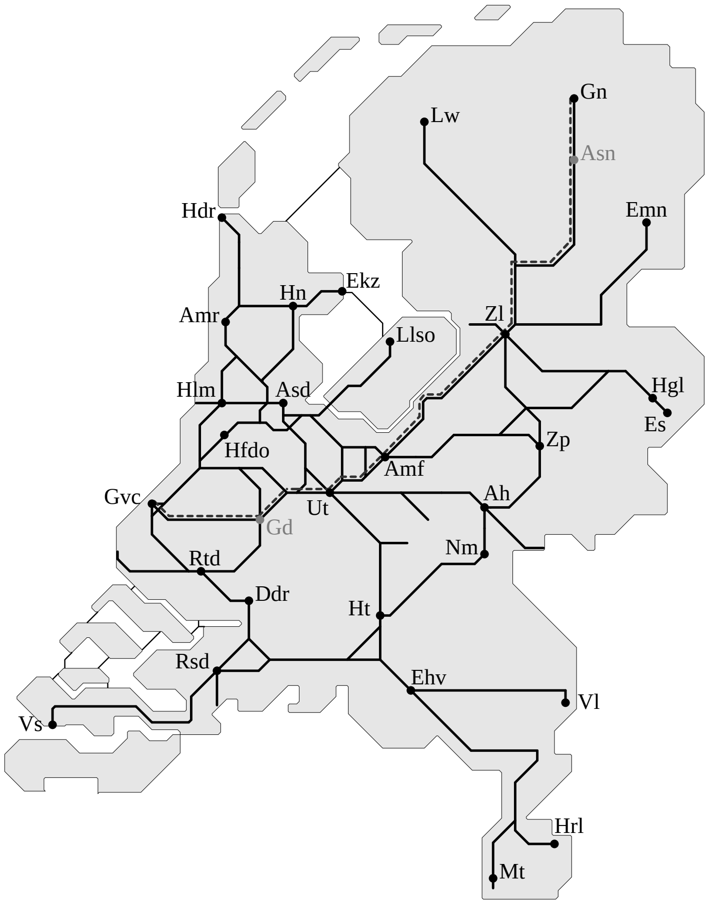


Figure 2.1: The network on which NS operates passenger trains (in 2007). The dashed line shows the 500 intercity line from Groningen to The Hague with its five intermediate stops. All black points correspond to crew bases, while grey points correspond to 'regular' stations.

Every train on every line needs a driver and a number of conductors, depending on for example the length of the train. This means that crew planning can only be done when the timetable and the rolling stock schedule is determined. Constructing a timetable and scheduling the rolling stock are problems that will not be considered in this thesis. For planning the crew, there is a division in two problems, namely the *Crew Scheduling Problem* (CSP) and the *Crew Rostering Problem* (CRP). The rostering problem is executed at the crew bases. Since it is not executed centrally like the CSP, we will not consider the CRP anymore.

Crew scheduling at NS is complex; on a weekday, about 5,000 timetabled trips are scheduled. A *trip* is a train operating on a line between a start and end location having a certain start and end time. For the operation of these trips, about 2,800 rolling stock carriages are used, and there are about 3,000 drivers and 3,500 conductors employed. There are 29 crew bases across the country from which the crew members operate from (see Figure 2.1). Each crew member has to perform tasks. A *task* is the smallest amount of work that has to be assigned to one driver and starts and ends at a *relief location*, which is either a crew base or another location where a change of a driver is allowed. At most relief locations it possible for the crew to have a meal break. Besides trips or parts of trips, a task can also be a passenger task, a shunt task or a task where the driver has to walk or taxi from one station to another. A passenger task means that a driver travels as passenger on a certain train. Then, a sequence of tasks, possibly interrupted by breaks, is called a *duty*. In this thesis we will only focus on train drivers, as the (re-)scheduling of conductors is done similarly to the (re-)scheduling of drivers.

Each day, about 1,000 duties are carried out by the drivers. At any moment in time, the number of active driver duties at that moment is about 300. But not only the size of the problem makes crew scheduling hard, also the rules that the crew schedules have to satisfy influence the complexity of the problem.

For the duties there are some hard and soft constraints that have to be satisfied. For example, a duty can at most last 9:30 hours and each duty that lasts longer than 5:30 hours has to contain a (meal) break of at least 30 minutes. In some cases the break time has to be at least 32, 35 or 40 minutes. Furthermore, there are constraints concerning the maximum working hours until and after the break and concerning the minimum connection time. Another rule is that a duty starts with a sign-on time, depending on the type of the first task, and ends with a sign-off time at the base of the crew member. Soft constraints concern constraints at crew base level. An example of such a constraint is that per crew base at most 5% of the duties can be longer than 9:00 hours.

In 2001 the drivers and conductors were dissatisfied with the structure of their duties, which led to nation wide strikes. The production plan implemented back then was called 'Destination: Customer' (see also [13]). As the name states, the plan was focused more on satisfying the customer than on satisfying the employees of NS. In this production plan the repetitiveness of the duties was unacceptable for the drivers. NS had to change their production model into a model that both satisfied the drivers and conductors, and at the same time supported an increment of the punctuality, efficiency and robustness of the railway services (that is, satisfying the customer). The alternative production model, called 'Sharing Sweet & Sour', aims at a fair division of the sweet and sour amounts of work among the crew depots. 'Sweet' represents the variation in the routes and the train series as well as the work on Intercity trains, while 'Sour' mainly represents the work on lines with a lot of passenger aggression and the work on the older rolling stock units. The production model imposed additional constraints in order to achieve a fair division of the 'Sweet & Sour' among the crew depots. For example, there have to be enough variation in the duties, that is, the

Repetition-in-Duty (RID) has to at most equal to a predefined value. The RID of a certain duty is defined as (see also [1])

$$RID_d = \frac{\text{number of routes in duty } d}{\text{number of different routes in duty } d}.$$

Furthermore, there is a lower bound for the number of routes per crew depot and there is an upper bound for the percentage of work per crew depot on lines with a lot of passenger aggression. Concepts as the RID lead to complex constraints at the crew base level, which makes the whole production model mathematically demanding. More details about the production model 'Sharing Sweet & Sour' can be found in [1]. In December 2010 an adapted version of 'Sharing Sweet & Sour' will be implemented.

In the system used at NS for crew scheduling, the problem is defined as a set covering model. Since there are a lot of extra constraints compared to a standard *set covering problem* (SCP), the following SCP with additional constraints for crew scheduling is introduced.

(CSP):

$$\min \sum_{d \in D} c_d x_d \tag{2.1}$$

$$\text{s.t.} \sum_{d \in D} a_{t,d} x_d \geq 1 \qquad \forall t \in T \tag{2.2}$$

$$l_r \leq \sum_{d \in D} b_{r,d} x_d \leq u_r \qquad \forall r \in R \tag{2.3}$$

$$x_d \in \{0, 1\} \qquad \forall d \in D \tag{2.4}$$

Here we use the notation $t = 1, \ldots, T$ for the tasks to be covered, $d = 1, \ldots, D$ for the potential feasible duties and $r = 1, \ldots, R$ for the additional restrictions to be satisfied. Furthermore, the binary decision variable represents

$$x_d = \begin{cases} 1 & \text{if duty } d \text{ is selected in the final solution,} \\ 0 & \text{otherwise.} \end{cases}$$

c_d equals the costs against which a duty can be used in the solution. In the binary matrix $a_{t,d}$, each row represents a task, each column represents a feasible duty and $a_{t,d} = 1$ if and only if task t is covered by duty d (and $a_{t,d} = 0$ otherwise). The parameters l_r and u_r represent respectively the lower bound and upper bound for restriction r.

The objective is to minimize the costs (2.1). Constraints (2.2) indicate that every task is covered by at least one duty. In practice, each train can only be operated by one driver. When more drivers are on the same train, one of the duties contains a 'real working' task and the others contain a passenger task. Rules for individual duties are not in the additional constraints (2.3), but are used at the generation of the set D. So the restrictions (2.3) represent constraints at crew base level, which we discussed before.

The complete set of feasible duties is usually extremely large, making a priori enumeration of all feasible duties impossible. Therefore, Column Generation is applied so that only promising duties are generated during the solution process. For the generated duties a heuristic is applied that evaluates the estimated value of the duty for the solution, based on dual information obtained by applying Lagrangian relaxation and subgradient optimization to (2.1)-(2.4). Next, based on the dual information, duties from the generated duties are selected heuristically and added to the solution (see also [13] and [1]).

2.2 Crew rescheduling at NS

Unfortunately, on the day of operation, the constructed schedule cannot always be executed as planned. Trains do not always run on time due to unexpected events. Examples are infrastructure malfunctions, rolling stock break downs, accidents, and weather conditions. Such events are called disruptions. We define a *disruption* as an event or a series of events that causes conflicts in the planned resource schedules for rolling stock. The Dutch railway network encounters approximately, per day, 17 disruptions related to the infrastructure with an average duration of 1.8 hours. About 35% of these infrastructure related disruptions are due to technical failures, while another 35% is related to third parties (e.g. accidents with other traffic). Next to the disruptions leading to infrastructure failures, there are also disruptions caused by the operators. The main reasons for the latter are passengers causing longer dwell times, rolling stock problems and delayed crew members. The proportion between the disruptions caused by the operators and the infrastructure is roughly 50-50 in the Netherlands. On average 18% of the disruptions are large disruption having a duration of at least 3 hours. This means that on a regular day on average 3 large disruptions occur.

Of course, infrastructure managers and operators try to avoid disruptions. Unfortunately, many of them are hard to influence. Therefore, it is very important to limit the consequences of these disruptions. A very common problem in railways is that, due to the strong interdependencies in the railway network and due to cost efficient resource schedules, disruptions are very likely to spread over the network in space and time. This well-known phenomenon is called *knock-on effect* or *propagation of delays*. The key to a good performance of railways is to limit the knock-on effect and thereby to limit the impact of single disruptions. Therefore, operating plans should be robust and effective disruption management is required.

Consider the same 500 intercity line between Groningen and The Hague from the previous section. Figure 2.2 shows a situation with a disruption at Beilen. The disruption causes a complete blockage between Beilen and Hoogeveen in both directions. Ermergency scenarios state how to deal with such a scenario. In this case, the emergency scenario states that the 500 line from Groningen to The Hague drives to Assen and turns there in order to get back to Groningen. In other direction, the train drives to Hoogeveen and is turned there. The duties that are affected by these changes in the timetable have to be rescheduled.

The way operators deal with disruptions is called *disruption management*. Railway disruption management is discussed in more detail in [12] and [17]. Like said before, within NS, several approaches for rescheduling are developed. The three approaches we will focus on in this thesis are:

- 2-phase repeated shortest path problem with resource constraints (2P-RSPPRC)
- Column generation with dynamic duty selection (CGDDS)
- Actor-agent application for train driver rescheduling (AAATDR)

The 2P-RSPPRC approach

The 2P-RSPPRC approach is introduced by [17]. It is a heuristic method that tries to mimic the current way dispatchers manually reschedule the crew duties. From interviews

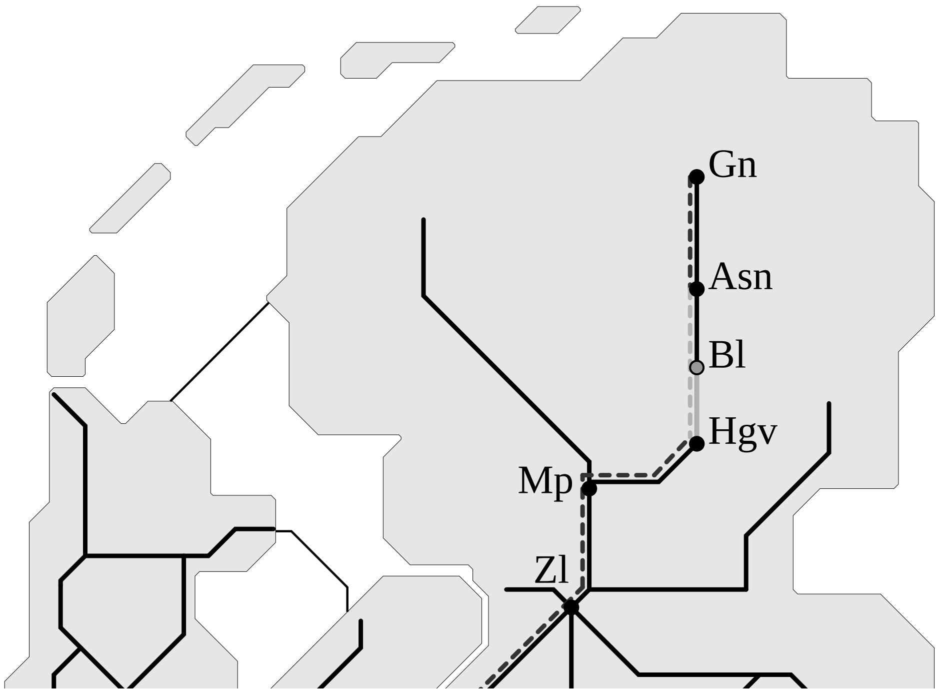


Figure 2.2: The northern part of the network on which NS operates passenger trains. There is a disruption at Beilen (grey dot) causing a complete blockage between Beilen (Bl) and Hoogeveen (Hgv) in both directions (light grey). The dashed line shows the 500 intercity line from Groningen to The Hague and the emergency scenario states that the train does not drive between Assen (Asn) and Hoogeveen (Hgv) (indicated with a light grey dashed line).

with dispatchers of NS it turned out that their manual rescheduling roughly follows a two phase approach. The first phase consists of constructing a feasible crew schedule in a greedy way. When constructing these feasible completions they try to use parts of the infeasible duties to cover as many tasks as possible. When there are tasks left uncovered in the feasible completion for every duty found by the dispatchers, they try to cover these by utilizing the reserve duties. At the end of the first phase they have a feasible replacement duty for every original duty and possibly a number of uncovered tasks. The second phase is based on improving the schedule by resolving uncovered tasks one by one. When assigning a task to a duty, the dispatchers try to avoid newly uncovered tasks if possible. If this turns out to be impossible, a new feasible completion that covers the uncovered task under consideration is only accepted under a certain condition. The condition indicates that only an uncovered task is accepted when this latter task starts later than the task selected before. In that way the dispatchers can move the problems to a later point in time, which provides the dispatchers more time to resolve the problem later on.

In the 2P-RSPPRC approach, the feasible completions that are considered are computed as solutions to auxiliary *shortest path problems with resource constraints* (SPPRC) on a weighted directed acyclic graph. The length of a path indicates the attractiveness of a feasible completion and is dependent on the phase and the uncovered task in the focus. The resource constraints guarantee feasibility of each replacement duty with respect to the rules we discussed in section 2.1. A detailed description of all the steps in the method can be found in

[17].

The CGDDS approach

We will now introduce a set covering formulation of the *operational crew rescheduling problem* (OCRSP) as introduced by [16]. Let N be the set of tasks which have not started at the time of rescheduling, where for every $i \in N$ we have ds_i the departure station, dt_i the departure time, as_i the arrival station and at_i the arrival time. Furthermore, $\Delta = \Delta_A \cup \Delta_R$ is the set of unfinished duties with Δ_A the active and Δ_R the reserve duties, where for every $\delta \in \Delta$ we have cs_δ, which represents the station where the original duty is at the time of rescheduling or the arrival station of the task performed by the driver at the time of rescheduling, and b_δ, the crew base where the original duty starts and ends. Finally, we have the set K^δ containing all feasible completions for original duty $\delta \in \Delta$. The crew rescheduling problem is then given by

(OCRSP):

$$\min \sum_{\delta \in \Delta} \sum_{k \in K^\delta} c_k^\delta x_k^\delta + \sum_{i \in N} f_i y_i \tag{2.5}$$

$$\text{s.t.} \sum_{\delta \in \Delta} \sum_{k \in K^\delta} a_{i,k}^\delta x_k^\delta \geq 1 \qquad \forall i \in N \tag{2.6}$$

$$\sum_{k \in K^\delta} x_k^\delta = 1 \qquad \forall \delta \in \Delta \tag{2.7}$$

$$x_k^\delta, y_i \in \{0,1\} \qquad \forall \delta \in \Delta, \forall k \in K^\delta, \forall i \in N \tag{2.8}$$

Here, $a_{i,k}^\delta$ is a binary parameter indicating if task i is covered by feasible completion k or not. c_k^δ is the cost of feasible completion k for original duty δ. The cost is equal to zero if the duty is not modified. Otherwise it is the sum of the costs for changing a duty, the cost for taxis and the penalties for short connection times and overtime. The parameter f_i is the cost for canceling task i. x_k^δ and y_i are binary variables respectively corresponding to the completions of duty δ and indicating whether task i is canceled or not.

In the problem, constraints (2.6) make sure that every task is either covered by a feasible completion or is canceled. Constraints (2.7) ensure that every original duty is assigned exactly once to a feasible completion. Recall from Section 2.1 that there are a lot of duties. Since a disruption occurs only in a part of the country, it seems highly unlikely that duties that cover only tasks elsewhere in the country will be modified. Therefore, [16] decided to only consider a core problem containing only a subset of the original duties and tasks. Such a core problem is given by a subset $\bar{\Delta}$ of the original duties and a subset $\bar{N}$ of the tasks. Given $\bar{\Delta}$, $\bar{N}$ contains the tasks that are covered by at least one $\delta \in \bar{\Delta}$ plus the tasks uncovered in the current solution. The subproblem is now given by (2.5)-(2.8), where all sets are replaced by subsets. To solve the subproblem, a Lagrangian heuristic similar to the one proposed by [11] is used. Since the number of feasible completions for every driver can still be very large, the Lagrangian relaxation is combined with column generation. For every *restricted master problem* (RMP) a subgradient optimization procedure is applied. In order to determine whether the outcome of the Lagrangian subproblem is a good approximation, or if the outcome can potentially be improved by adding feasible solution to the RMP, a pricing problem for every original duty

$\delta \in \bar{\Delta}$ is solved. The pricing problems are modeled as a SPPRC. When the subgradient method terminates, a greedy procedure is used to find feasible solutions to the core problem.

Since only a subset is considered, it can very well be the case that in the solution there is a list of uncovered tasks. If that is the case, an improvement step is executed by considering a neighbourhood. In the neighbourhood, duties that possibly can cover the uncovered task are selected and for this new set the same method as described above is applied. More details on the whole procedure can be found in [16] and [17].

The AAATDR approach

A totally different approach for rescheduling is introduced by [2]. The method proposed is based on multi-agent techniques. In the model, three types of agents are present. There are driver-agents, route-analyzer-agents (RAA) and network-agents. When a driver-agent is affected by a disruption, the driver becomes a team leader and by using heuristics, driver-agents that have a high probability of improving the solution are selected to join the team. The process of exchanging duties depends on a scoreboard mechanism. The *scoreboard* keeps track of the cost of the exchange configuration. When the conflicts are resolved without generating new conflicts, the costs of the exchange configuration is compared with the scoreboard and the scoreboard is updated when the costs are lower.

However, tasks cannot be exchanged when the constraints are not satisfied. If a driver-agent wants to exchange duties, its requests for route calculations first has to be handled by an RAA. A request consist of the current duty of the requesting driver-agent and one or more tasks to take over from another driver-agent. Based on the restrictions we mentioned before, the RAA can answer either *feasible*, *conditional feasible* or *infeasible*. *Feasible* means that the tasks to take over can be added to the driver-agent's duty without imposing a new conflict. If a request is *conditional feasible*, the tasks to take over can be added to the driver-agent's duty but leads to a new conflict. This means that the taking over leads to tasks that have to be taken over by another driver-agent. If the driver-agent cannot take tasks over at all, the answer will be *infeasible*.

Finally, there are network-agents that maintain knowledge of the current timetable, including disruptions and delays. The goal of the network-agents is to maintain as much of the original duty of the driver-agent as possible and to let driver-agents arrive at the destination (i.e. the location of the conflict) as soon as possible. Network-agents do the route calculations, based on the actual timetable, by using a shortest-path algorithm. The network-agents return the adjustments that have to be made in the duty. The results generated by the multi-agent system can be analyzed by the process manager and the dispatchers. They can accept the outcomes or let the system generate new solutions.

2.3 Robust crew schedules

Recall that there are two measures of robustness, namely:

- The way a schedule can absorb delays, caused by little disruptions, without necessity of rescheduling (the absorption ability of a schedule).
- The way a schedule can be adapted when rescheduling is necessary (the flexibility of a schedule).

Both aspects are important for this thesis, but the main focus will be on the flexibility of a schedule.

With the crew scheduling method described in (2.2) we can create several crew schedules and analyze their robustness, mainly in terms of flexibility. We would like to know what parameter settings produce the best schedules, based on their flexibility. This means that we try to point out the indicators that influence the robustness of a schedule. In case of large disruptions, we have to do rescheduling. Of course, we would like to see what the effect of each method is on the robustness of the different crew schedules. Furthermore, we would like to have an overview of the differences between the three rescheduling approaches in terms of their performance. The performance is measured by the number of shifted tasks compared to the old schedule, the amount of overtime in the new schedule and the computational effort made to obtain the new schedule. Besides that, we also keep track of the (additional) operational costs imposed by each of the method.

The main research question is "Which factors determine whether a schedule is robust or not and how does the rescheduling methods contribute to the robustness of schedules?". This means that our focus will be twofolded. On the one hand we try to find robust schedules against low additional costs, and on the other hand we will also consider the differences between the rescheduling methods. To do so, we need to generate cases which have to be a good representation of problems that occur on the Dutch railway network. A case in combination with a schedule for which rescheduling is necessary, is called a *scenario*. For the constructed scenarios, generated with several cases and schedules, we will do rescheduling using different methods and we will consider the differences between the methods. A more in depth analysis will be done only with CGDDS, as this method is going to be introduced in practice at NS. We will combine some cases, since in reality also multiple disruptions per day can occur, and we will introduce some additional schedules such that we can test them for robustness. Combined cases for a specific schedule will be called an *instance*.

3 Literature review

There is very little literature available on robustness of crew schedules. The literature that is available, concerns mostly research done for the airline industry. The main reason that railway crew scheduling gets less attention is that operational costs caused by disruptions are much lower compared to the costs in the airline industry. Besides that, there is limited competition between train operators. We will consider two types of literature, based on the two robustness measures mentioned before. First, we review literature where research on the reduction of delay propagation is discussed, followed by the discussion of available work on flexibility of schedules. In both sections there is a distinction between train crew schedules and airline crew schedules.

3.1 Reduction of delay propagation

3.1.1 Delay propagation in train crew schedules

Delay propagation in train crew schedules is not often discussed in the literature. There is a thesis work executed at the Erasmus University Rotterdam. [20] wrote a bachelor thesis on finding a robustness measure for different NS crew schedules. To determine how delays knock on through the schedule, a Linear Programming model that measures the mean delay is solved. Using the mean delay as a measure, different schedules can be compared. In this thesis work only delays caused by small disruptions are considered. The conclusion is that there is little difference between crew schedules that are based on different transfer times. So varying transfer times have little impact on the robustness of the schedules.

3.1.2 Delay propagation in airline crew schedules

In the airline industry a CSP is the assignment of flights to so called *trips* (or *pairings*) against minimal costs. A *trip* consists of consecutive flights starting and ending at the crew base. The CSP is formulated as a *set partitioning problem* (SPP). The problem of rostering is assigning each trip to a crew member. We will discuss the methods that focus on reducing the propagation of delays through the airline crew schedules and we discuss the methods that focus on making the process of rescheduling more easy.

[8] describe a bicriteria optimization approach for reducing the propagation of delays through the airline crew schedule. In the model a second objective, with respect to propagation of delays through the crew schedule, is added to the traditional SPP. For every possible consecutive flights in a trip, a parameter that measures to what extent a delay of the first flight works through on the second flight is added. For each trip a non-robustness parameter is defined which is equal to the sum of the parameters of all consecutive flights within the trip. The second objective of the bicriteria model is the minimization of the non-robustness of the trips. In the model, instead of a search for optimal solutions there is a search for Pareto optimal solutions. A solution is called Pareto optimal, if there is no other solution which is at least as good with respect to both objectives and strictly better with respect to one objective. The problem is solved using an elastic version of the ϵ-constraint method. With this technique one objective is handled and the others are transformed into constraints with upper bound ϵ on their values. By using different values for ϵ all Pareto optimal solutions can be generated. Ehrgott and Ryan transform the objective with respect to costs into a constraint, such that the price of robustness is easy to specify. The bicriteria optimization

model is implemented into a crew scheduling system and is tested on historic data. The results indicate that a significant reduction of propagation of delays through the schedule is possible with little increase in costs.

Another research for reducing propagation of delays through the airline crew schedules is executed by [22]. They formulate an two-phase stochastic optimization model, which considers interaction between personnel, equipment and flights under disruptions. The authors developed a *branching* algorithm to identify costly scheduling decisions and to generate an alternative crew schedule. This is an iterative process, where initially a crew schedule is generated based on a standard SPP. Next the planning is evaluated under stochastic disruptions in a *recourse* model. In this model, the total propagation of delays through the schedule is evaluated. In this way, the two consecutive flights that probably produce the highest propagation of delays to other flights are identified. Then, the identified flights are neglected in the remaining *branching* steps. This means that in every iteration there is a feasible and more robust crew schedule found, since certain pairs of flights are eliminated from the set of possible trips.

A very recent work on minimizing propagated delay through airline schedules is executed by [5]. The authors propose an approach for integrating the problems of aircraft routing and crew pairing, while finding a minimal propagated delay cost solution. They use two acyclic graphs, one for the aircraft routing and one for the crew pairing problem, where the nodes correspond to flights and the arcs correspond to possible feasible connections between flight nodes. The propagated delay at a node is calculated by inductively applying formulas to calculate propagated delay along paths in both the aircraft connection network and crew connection network. To find the minimizing path, pricing problems are solved. In the pricing problem for finding an aircraft path, the propagated delay from the crew is taken into account and in the pricing problem for finding a crew path, the propagated delay from the aircraft is taken into account. This makes sure that in each node there is a good representation of the real delay. The authors propose some algorithms solving the integrated problem iteratively by linking the outputs of one problem to the other. For solving the pricing problems, two *label setting* algorithms are implemented. It is shown by testing several instances that the new approach clearly improves the amount of propagated delay.

There is other literature available where the minimization of delay propagation is incorporated in determining the airline crew schedule. [15] developed a multi-objective genetic algorithm for robust crew scheduling. They improved both the robustness and operational cost of an existing crew schedule. However, computation times are quite long; the system required 90 hours to solve a problem of 441 flights with 126 crew members. A similar approach, but involving all scheduling sub-problems of an airline, is done by [3]. Here also a multi-objective approach is used, but now there is a trade-off between *reliability* (the schedule its ability to absorb the effects of minor stochastic influences) and *flexibility* (related to the number of recovery options available to reduce the effects of a disruption), where both indicators are seen as robustness measures. The search methodology used is based on a hybridization of genetic algorithms with local search, also referred to as memetic algorithms. When there are multiple local search operators involved, the method is called a multi-meme memetic algorithm. For KLM Royal Dutch Airlines a large scale simulation study was undertaken to quantify the influence of the robustness objectives on the operational performance of the schedules. After a sensitivity analysis, the authors concluded that the influence of the schedule its reliability is dominant and that increased flexibility could improve the operational performance.

3.2 Flexibility of schedules

3.2.1 Flexible train crew schedules

The concept of *move-up crews*, introduced in the airline industry (see the discussion of [18] in the next section), applied to train crew scheduling is explored by [9]. They base their work on the rescheduling strategy imposed by [18] and focus on the combinatorial problems that occur when that strategy is applied to the railways. Those problems concern determining crew schedules with the maximum number of possible *crew swaps*. At the railways, a *crew swap* is possible when the concerning train drivers belong to the same crew base and the swap does not lead to violation of the constraints (like rules for breaks, maximal working hours and the licenses of drivers to drive certain rolling stock types on certain parts of the railway network). The problem is decomposed into three different types of a *set cover with pairwise prizes* (SCPP) problem, where for every duty costs, and for every pair of duties gains that are related to crew swapping possibilities, are defined. One variant is the SCPP *trade-off cost* (SCPP-tc) problem. In this problem a set cover which minimizes the difference between costs and revenues is defined. The authors show that the SCPP-tc problem can be transformed into a maximum profit subgraph (MPS) problem in a graph $G(V, E)$ by assuming that every introduced subgraph is a set cover of the set of duties. In this graph, every node corresponds to a duty and there exists an arc between two nodes if a crew swap is possible between the duties. The MPS problem can be solved within polynomial time. G can be transformed into a bipartite graph containing a maximum weight independent set. Such a set consists of independent (not connected through an arc) nodes in a graph with maximum weight and can be found in a bipartite graph with a *maximum flow* algorithm. Since $|V|$ can be very large for problem instances at the railways, the bipartite preflow-push algorithm would take too much calculation time. The authors therefore discuss an algorithm that approaches the bipartite preflow-push algorithm, but has a calculation time of $O(|E| + |V|log|V|)$ instead of $O(|V|^2|E|)$ for the bipartite preflow-push algorithm. The solutions produced by the algorithm are within 200% of the optimal solution. For solving the SCPP-tc problem, the authors present a heuristic. This heuristic adds duties found with the algorithm to the initial solution of the CSP. In this way, the possible number of crew swaps in the solution is increased. So far, there are no computational experiments executed with the heuristic.

Another work on the flexibility of train crew schedules is a master thesis by [21]. Also [21] is a research on NS crew schedules. But unlike [20], [21] considers scenarios with large disruptions that actually took place, and uses for rescheduling a version of CGDDS available at that time. The scenarios and rescheduling method are tested on four different driving duty schedules combined with four different numbers of reserve duties, resulting in 16 different crew schedules for one day. Three of the four driving duty schedules are based on 'Sharing Sweet & Sour', but with different transfer times. The fourth schedule is based on the old production plan of NS which led to nation wide strikes of the crew members. One reserve duty set contains no reserve duties, one reserve duty set is based on an actual set and the other two contain an amount of reserve duties between zero and the actual set. On each schedule, 50 different scenarios (five situations with two different lengths, starting at five different times) are tested. It is concluded that longer transfer times have a positive effect on the flexibility of the schedule, but only to a limited extent. Furthermore, the old production model performs better than the model 'Sharing Sweet & Sour' in terms of robustness, but limit the propagation of delays through the network less than expected. Finally, the research shows that also without or with less reserve duties good results can be obtained by using CGDDS. However, this is based on scenarios where only one disruption occurs on the day of operation.

3.2.2 Flexible airline crew schedules

[18] describe a method to improve the flexibility of airline crew schedules. The method that is developed is based on the concept of *move-up crews*. A *move-up crew* consist of crew members that, after disruptions, take over flights of delayed crew and vice versa. Such a solution is relatively easy and cheap, but only possible if both crews are assigned to the same crew base and if a crew swap does not lead to violation of constraints. Since a crew schedule with a lot of move-up crews probably is more flexible due to many *crew swapping* possibilities, there has to be a trade off between minimization of the costs and the maximization of the number of move-up crews. In the research first a crew schedule is determined based on a standard SPP. Then a crew schedule has to be constructed where the number of move-up crews is maximized and the costs are within a certain range from the optimal costs (determined with standard SPP). This problem is formulated as an *integer programming problem* and solved with an algorithm that makes use of a combination of column generation and Lagrangean relaxation. By using a rescheduling module, the traditional and robust planning are compared based on operational costs, the number of canceled flights and the number of used reserves following from disruptions. The results indicate that robustness leads to lower operational costs. However, using a too large range from the optimal costs will give no operational advantage anymore. Also the number of move-up crews per flight has to be limited.

Also [19] make use of the concept of *move-up crews*. They use it in order to determine the demand for reserve capacity caused by disruptions in daily operations in terms of so called *open-time trips*. Trips that are not assigned to 'regular' crew in the planning phase due to conflicts of the rosters with for example individual holidays and in the operational phase due to disruptions, are called *open-time trips*. The idea of the authors is to improve the employability of reserve crew compared to the current scheduling by defining demand in terms of open-time trips. They do that by defining a SCP two times for two different phases, where phase A assigns all open-time trips to a collection of *reserve duty periods* (sequences of trips that are feasible for the scheduling constraints) and phase B generates a roster by assigning the reserve duty periods to a reserve crew member. After testing seven problem instances the authors conclude that the proposed model improves the employability of the reserve crew. Furthermore, the model indicates that during problem instances less reserve crew is required.

3.2.3 Flexible airline schedules

Finally, a research for more efficient airline schedules that can be recovered more easily is executed by [7]. This work is not explicitly on crew schedules, but according to the authors the methodology can be extended such that also crews are considered and so the work can be relevant for this thesis. For the planning phase, the *Maintenance Routing Problem* (MRP) is solved in order to find a feasible route for each aircraft and a departure time for each flight, where the loss of revenue compared to a desired schedule is minimized. When a disruption occurs, rescheduling is done by solving the *Aircraft Recovery Problem* (ARP). The MRP and ARP algorithms are quite similar except for the difference in the specification of the constraint specific networks and its cost structure. The algorithms are solved using Column Generation. More details about these algorithms can be found in [6]. In order to make the original airline schedule more easy to reschedule, *Uncertainty Features* (UFs) are used to reformulate the MRP. The authors selected four UFs based on their potential to increase robustness and recoverability and based on the implications on the algorithm. Increasing idle time of planes, increasing minimal idle time of planes, increasing the number of plane crossings and increasing passengers connection time correspond to the four UFs, leading to four MRP algorithms. Several disruption scenarios are implemented and the rescheduling is done with the ARP algorithms as described in [6]. Experiments show that increasing the idle time improves the recoverability of a schedule the most. There is a large reduction in recovery costs, while there is only a small increase in lost revenues.

4 Experimental design for comparing the rescheduling methods

In this section we describe how we design the computational experiments for our analysis of the methods. First, we discuss the choice of the different crew schedules we will compare. Next, we have to choose several cases with disruptions in different parts of the country occurring at different moments of the day. The rescheduling methods have some different goals and parameters. So in order to make a fair comparison, we have to make a choice for the goals and parameters of the methods such that they are comparable.

4.1 Crew schedules

The crew schedules we analyze are all based on the timetable of a regular day in 2007, namely the 19th of June. The base schedule is the schedule that was actually executed on that day and is based on the production model 'Sharing Sweet & Sour'. The schedules from [21] are also available for this research, so it seems straightforward to analyze those schedules again, but now making use of other rescheduling methods and a more recent version of the CGDDS model. The base set consist of four plans. The first plan is the basic plan that normally is executed on that day. This plan is based on the concept 'Sharing Sweet & Sour' and according to that concept the drivers should have at least 20 minutes transfer time to go from one train to another. The second plan is based on the same concept, but now the transfer time is shorter, namely 15 minutes. The third plan is again based on 'Sharing Sweet & Sour', but for this plan the minimal transfer time is reduced to 10 minutes. Finally, the fourth plan is based on the older production plan 'Destination: Customer' and the transfer time for drivers is at least 20 minutes in this plan. Each of these four plans can be combined with different numbers of reserves duties. We can use the plans without reserve duties (R0) and with 84 reserve duties (R1). Those 84 reserve duties are the same as they would be on a regular day of operation at NS. An overview of all the schedules and the number of duties the schedule contains can be found in Table 4.1.

Table 4.1: Overview of the different schedules

Schedule	Duties	Description
$SS_{20}R_0$	891	'Sharing Sweet & Sour' with 20 minutes transfer time (the basic plan, without reserves).
$SS_{20}R_1$	975	The basic plan, reserves included.
$SS_{15}R_0$	865	'Sharing Sweet & Sour' with 15 minutes transfer time (without reserves).
$SS_{15}R_1$	949	Again 15 minutes transfer time, reserves included.
$SS_{10}R_0$	861	'Sharing Sweet & Sour' with 10 minutes transfer time (without reserves).
$SS_{10}R_1$	945	Again 10 minutes transfer time, reserves included.
$DC_{20}R_0$	884	'Destination: Customer', with 20 minutes transfer time (without reserves).
$DC_{20}R_1$	968	'Destination: Customer', reserves included.

4.2 Disruption cases

To define representative cases, we consider cases throughout the country. Some occurred in practice and for which log files are available such that the actual situation can exactly be represented. Other disruptions are arbitrarily chosen and constructed based on an arbitrarily chosen emergency scenario. It seems realistic to not only consider complete blockages, but also cases where a reduced number of trains can be operated. Furthermore, it seems fair to consider cases that affect many duties, as well as cases that have a smaller impact on the schedule. We chose to model the cases where there were complete blockages at Abcoude (Ac), Alkmaar (Amr), Almelo (Aml), Beilen (Bl), Heerlen (Hrl), 's Hertogenbosch (Ht), Lelystad (Lls) and Schiphol (Shl), and the cases with disruptions causing a reduction in the number of trains at Rotterdam (Rtd) and Zoetermeer (Ztm). All cases have a duration of at least three hours. The disruption around Abcoude and 's Hertogenbosch involve heavily used routes and affect between 40 and 60 original duties, while the involved routes in the cases Beilen, Heerlen and Lelystad are not so heavily used. The disruptions at Alkmaar and Almelo involve routes that are both not heavily nor little used routes (medium). Schiphol involves heavily used routes, but the number of affected duties is not as large as for the cases Abcoude and 's Hertogenbosch. Rotterdam and Zoetermeer are also on heavily used routes, but in these cases a reduced number of trains can be operated. If we model these cases on two moments of the day (mostly morning and afternoon) and combine them with the schedules mentioned before, we come to a total of 80 scenarios. An overview of the cases is given in Table 4.2.

Table 4.2: Overview of the different cases

Location	ID	Start time	End time	Type
Abcoude	Ac_1	11:07	14:07	two sided blockage
Abcoude	Ac_2	16:37	19:37	two sided blockage
Alkmaar (Alkmaar Noord)	Amr_1	07:07	11:07	two sided blockage
Alkmaar (Alkmaar Noord)	Amr_2	19:07	23:07	two sided blockage
Almelo (Wierden)	Aml_1	09:29	12:29	two sided blockage
Almelo (Wierden)	Aml_2	14:59	17:59	two sided blockage
Beilen	Bl_1	07:10	10:10	two sided blockage
Beilen	Bl_2	16:10	19:10	two sided blockage
Heerlen (Hoensbroek)	Hrl_1	07:18	11:18	two sided blockage
Heerlen (Hoensbroek)	Hrl_2	16:18	20:18	two sided blockage
's Hertogenbosch (Vught)	Ht_1	08:00	11:00	two sided blockage
's Hertogenbosch (Vught)	Ht_2	15:30	18:30	two sided blockage
Lelystad	Lls_1	03:52	06:52	two sided blockage
Lelystad	Lls_2	12:52	15:52	two sided blockage
Rotterdam (Rotterdam Lombardijen)	Rtd_1	12:40	15:40	reduced number of trains
Rotterdam (Rotterdam Lombardijen)	Rtd_2	17:10	20:10	reduced number of trains
Schiphol (Amsterdam Lelylaan)	Shl_1	08:29	11:29	two sided blockage
Schiphol (Amsterdam Lelylaan)	Shl_2	13:59	16:59	two sided blockage
Zoetermeer	Ztm_1	07:59	10:59	reduced number of trains
Zoetermeer	Ztm_2	11:29	14:29	reduced number of trains

However, implementing a scenario is not that straightforward. Patterns that are in line with the actual or chosen emergency plan have to be constructed. Furthermore, the rolling stock circulation has to be respected. For example, we cannot add trains for which no rolling stock is available. Another aspect that has to be considered is the transfer stations. Sometimes we have to create tasks that consist of the driving of multiple trains instead of the driving of only one train. This occurs for example when, according to the emergency scenario, a train is turned at a station which is not a transfer station. Also, a case can be different for each schedule. We illustrate the implementation of a scenario an example.

We will consider the earlier mentioned case of Beilen, where at 07:10 a disruption (e.g. accident with a person) takes place causing a complete blockage in both directions. At that moment, the emergency scenario becomes active. The train lines on this route are shown in Figure 4.1.

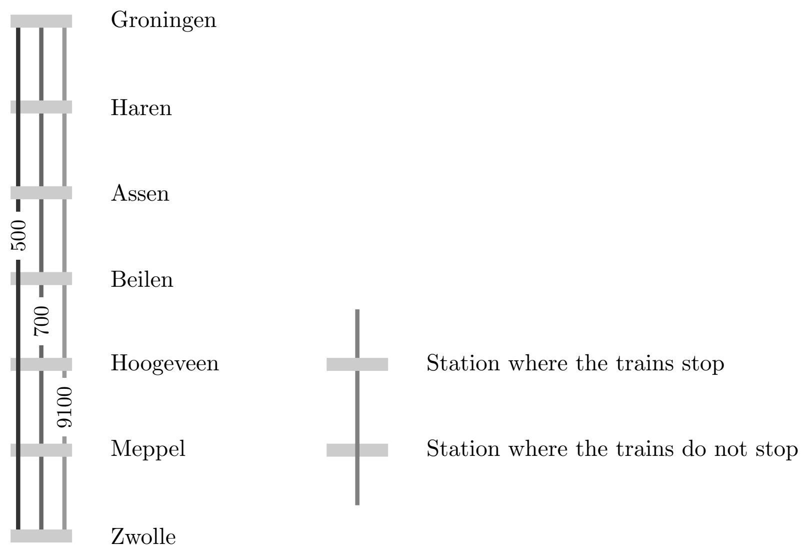

Groningen
Haren
Assen
Beilen
Hoogeveen
Meppel
Zwolle
500
700
9100
Station where the trains stop
Station where the trains do not stop

Figure 4.1: The lines operated between Groningen (Gn) and Zwolle (Zl).

The earlier mentioned 500-line (intercity) between Groningen and The Hague, the 700-line (intercity) between Groningen and Schiphol, and the 9100-line (regional) from Groningen to Zwolle are all train lines that use the route with a frequency of once per hour. The scenario states that the 500-line has to be turned at Assen (Asn). The 500-line in the other direction (Zwolle to Groningen) has to be turned at Hoogeveen (Hgv) according to the emergency plan. The same holds for the 700 series. Furthermore, the 9100 line from Groningen to Zwolle is turned at station Beilen (Bl) and the 9100 in other direction is turned at Meppel (Mp). This pattern is clearly represented by the time-space diagram of the timetable around Beilen in Figure 4.2.

Like said before, the intercity lines 500 and 700 are turned in Assen and Hoogeveen depending on the direction the train comes from. But Assen and Hoogeveen are both not a transfer station. This imposes that the new driver tasks have to be from Groningen to Assen and back (indicated by Groningen - Groningen) and from Zwolle to Hoogeveen and back (indicated by Zwolle - Zwolle). So not only the trains have to turn, the drivers on that train need to drive the same train in other direction. The same is true for drivers of the regional trains on the 9100-line; their tasks are now given by the sequence Groningen - Beilen - Groningen (task Groningen - Groningen) and for the other direction by Zwolle - Meppel - Zwolle (task Zwolle - Zwolle).

We will consider a driver duty that is affected by the blockage at Beilen. Figure 4.3.a shows the original duty of the driver, starting at 7:00 in Groningen. His or her first task belongs to the 700-line and consist of driving an intercity train from Groningen to Zwolle. However, due to the disruption at Beilen, the train has to be turned at Assen according to the emergency scenario. This implies that the driver also has to return to Groningen, as Assen is not a transfer station. Since the original schedule cannot be executed anymore, rescheduling has to be done. For example, the driver could perform the task Groningen - Groningen (driving

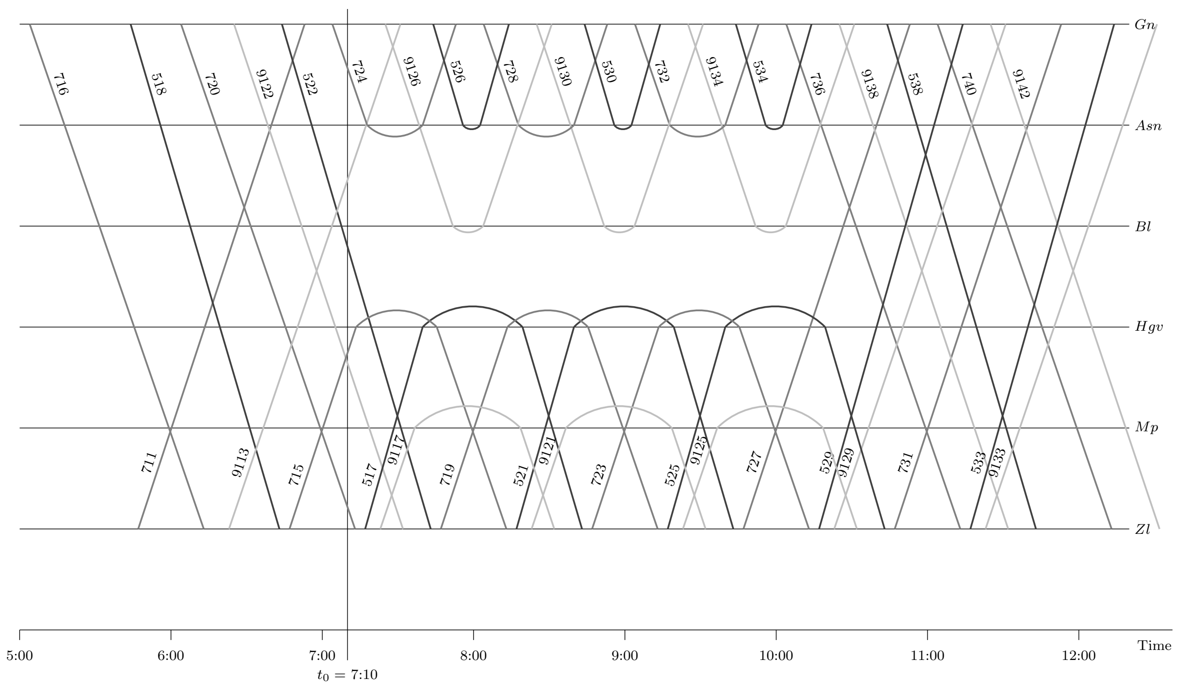


Figure 4.2: The adapted timetable between Groningen (Gn) and Zwolle (Zl).

from Groningen to Assen and back) and make use of a taxi to get to Zwolle. From there he or she performs some other tasks and when the driver gets back in Amersfoort around 13:00, he or she can execute the last two tasks that are also in his or her original schedule. This completion is represented in Figure 4.3.b. Another example of a completion is Figure 4.3.c, where the driver has to perform the new tasks Groningen - Groningen until the disruption is over. This example implies that the driver ends his or her duty one hour later than the original schedule, but that is allowed.

4.3 Fair comparison of the rescheduling methods

By testing some of the larger scenarios mentioned in the previous section, it became clear that the at NS available application of the agent-based model is not suitable to be used in practice. The rescheduling process took sometimes up to several hours of computation time. This would imply that a blockage has already ended at the time the new schedule is finished. To make the method somewhat comparable to the other two rescheduling methods, at least in terms of computation time, the procedure has to be accelerated. This can be done by cutting off a large part of the search tree. To that extent, we make sure that the procedure does not check paths in the tree that can possibly give a small improvement, but only the paths that possibly give a large improvement. Evidently, this will make the overall solution worse, as some solutions with higher costs will be accepted. Still, it is an essential adjustment to make our analysis meaningful. And even with this adjustment, as we will see later on, for some harder cases the computation times are still unacceptable.

When comparing AAATDR with respect to the other two approaches, it becomes clear that some constraints that are implemented as hard rules in the 2P-RSPPRC and CGDDS approach are implemented as soft constraints in the multi-agent system. In AAATDR it is

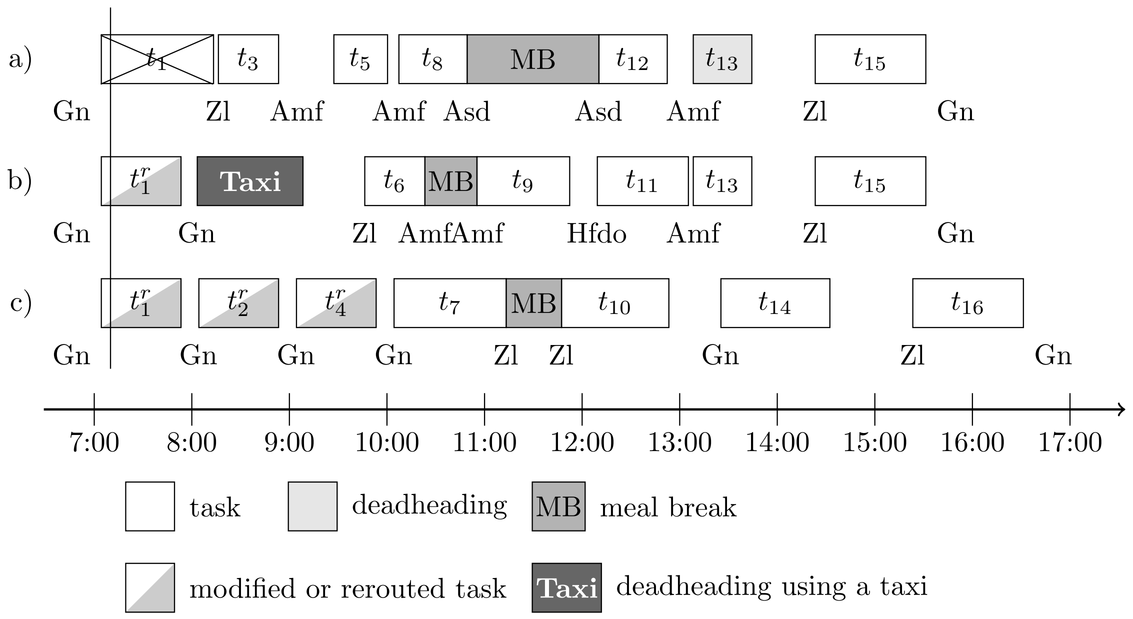


Figure 4.3: Examples of feasible completions for an affected original duty from crew base Groningen (Gn).

possible to modify a duty against a certain penalty, such that the duty is still longer than 5.5 hours, but contains no break anymore. Furthermore, against another penalty it also possible to let a duty end later than the maximum working time subscribed by the Collective Employment Agreement (in Dutch: CAO). It is not possible to make these constraints hard without changing the structure of AAATDR. However, we can set the penalties arbitrarily high. Unfortunately, we cannot make sure that the model never chooses to incur the high penalties. That is due to the the structure of the model. The model is very eager to resolve each conflict in the schedule. It will only decide to leave tasks uncovered if there are zero propositions of other agents to resolve the conflict. However, in most cases there are agents that propose a solution. Even if these costs are higher than the costs for not covering a task it will accept the proposition, since the model prefers to do rescheduling above leaving tasks uncovered.

Another important aspect is the goals of the different approaches. 2P-RSPPRC and CGDDS have the same direction, namely to reduce the number of changes between duties as much as possible. AAATDR has a different scope, namely to keep the amount of overtime as small as possible. To adjust all methods for making it possible to use both directions would be too time consuming. Unfortunately, 2P-RSPPRC cannot be modified such that it has the same direction as AAATDR. So either 2P-RSPPRC or AAATDR can be compared with CGDDS under equal parameters, as CGDDS can take both the direction of 2P-RSPPRC and AAATDR. This means that we have to make a distinction in comparisons: We compare AAATDR with CGDDS and CGDDS with 2P-RSPPRC.

Related to the goals of the methods are the parameter settings. Tables 4.3.a and 4.3.b provide an overview of the settings that will be used for comparison of CGDDS with AAATDR and for comparison of CGDDS with 2P-RSPPRC, respectively.

The first three parameters in Table 4.3.a are quite straightforward. The fourth and fifth

Table 4.3.a: Parameter settings for comparing CGDDS with AAATDR

Parameter	Value
Costs for not covering a task	20.000
Costs for a taxi	3.000
Costs per minute overtime	5
Increase in overtime costs after interval	5
Interval after which the overtime costs increase	30 minutes

Table 4.3.b: Parameter settings for comparing CGDDS with 2P-RSPPRC

Parameter	Value
Costs for not covering a task starting at location A and ending at location B	20.000
Costs for not covering a task starting at location A and ending at location A	3.000
Costs for not covering a reserve or an education task	250
Costs for modifying a duty	400
Costs for covering a task that is currently assigned to another duty	50
Costs for using a transfer that did not appear in any planned duty	1
Costs for a taxi that takes the driver over the disruption	1.000
Costs for any other taxi that takes the driver to his/her crew base	3.000

parameter need some explanation. The function for overtime is a piecewise linear function, where after each interval the costs increase by 5 units. So if a duty has for example 50 minutes overtime in the solution, and the interval equals 30 minutes and the costs equal 5, it means that the costs for overtime are 350 (30 minutes times 5 plus 20 minutes times 10). The values in the table are chosen such that there is a reasonable difference between the parameters; some solutions are clearly preferred above others. Furthermore, after some experiments we concluded that both methods can easily deal with these parameter settings.

Table 4.3.b shows the parameter settings for comparing CGDDS with 2P-RSPPRC. These settings are the same as used in [16] and [17]. For phase 2 of 2P-RSPPRC we use the second settings (indicated in [17] by $SET2$). As we can see in Table 4.3.b, there are some other parameters compared to Table 4.3.a. Now, there is a differentiation in costs for uncovered tasks; it is more expensive to not cover a task that goes from station A to station B than to not cover a task that starts and ends at the same station. Furthermore, there are some parameters relating to the modification of duties and there are two different types of costs for the use of taxis.

5 Computational evaluation of the rescheduling methods

In our analysis, we will compare CGDDS and AAATDR on the following statistics: The number of modified duties (MD), the number of used reserves (UR), the number of uncovered tasks (UT), the number of taxi trips (TT), the number of duties with overtime (OD), the total amount of overtime in minutes (TO), the total costs (Costs) and the computation time in minutes (CT). For each scenario we will also present the number of affected duties (AD). CGDDS and 2P-RSPPRC are compared on less statistics. We decided to test the methods for all schedules schedules presented in Table 4.1; four variants without reserve duties and all four variants with all reserve duties. Combined with the 20 cases this results into 80 scenarios to run for each method. We ran all scenarios on a *Intel Xeon X5472* processor with 3.25 GB RAM clocked at 2.99 GHz.

For some scenarios, the actor-agent application required a lot of computation time. However, rescheduling has to be done within short amount of time; it makes no sense to obtain a solution that states that a driver's duty should have been changed 30 minutes ago. We decided to set the maximum computation time at 20 minutes (actually this is still quite a long time, but this way we are sure that we remain with a significant number of results). So runs that take longer than 20 minutes are aborted. This implies that for some scenarios no results are available. The complete table of the results can be found in the Appendix, Table A.1.a and A.1.b. In this chapter we will only display and analyze some of the most important results. After the analysis of CGDDS and AAATDR, we present some results for 2P-RSPPRC. If we talk about robust schedules, we mean that the schedule is of robustness type 2, namely if the schedule can be easily adapted in case of a disruption or not.

5.1 Comparing CGDDS with AAATDR

Before considering the results in detail for the two methods, we will first present some general results in Table 5.1. In the previous section we presented the parameter settings. Since MD and UR are not affected by the current parameter settings and so the methods can adjust as many duties and use as many reserves as needed, they are separated from the other statistics.

Table 5.1: Average performance of CGDDS and AAATDR based on all 20 cases and 4 schedules

Model	Reserves	MD	UR	UT	TT	OD	TO	Costs	CT
CGDDS*	No	27	0	1.78	3	8	231	44,748	0:20
CGDDS	No	44	0	2.8	3	16	526	69,037	1:07
CGDDS	Yes	71	28	0.54	2	4	133	18,614	2:11
AAATDR*	No	39	0	1.2	2	3	366	37,512	4:50
AAATDR	No	-	-	-	-	-	-	-	-
AAATDR	Yes	57	12	0.79	5	3	192	35,097	4:59

* Averages determined with half of the scenarios (40 instead of 80 scenarios).

First of all, we have to mention that in Table 5.1 there are two rows without reserves for both methods. This distinction is made since for AAATDR without reserves half of the scenarios took more than 20 minutes to solve and therefore no results are available. The scenarios with the cases at Abcoude, Alkmaar, 's Hertogenbosch, Rotterdam and Schiphol are not incorporated in the average. This means that the averages represented in the table are not representative; each Abcoude, ’s Hertogenbosch, Alkmaar, Rotterdam and Schiphol scenario would increase the average computation time a lot, just like all the other means (because at the moment of termination of those runs all the values were already higher than the here presented means). But this is also true for CGDDS, so in this way we can still compare the methods. For CGDDS all scenarios could be solved, so we also present the averages over all scenarios.

Concerning half of the scenarios without reserves, AAATDR is performing better than CGDDS in terms of costs. This is related to the lower average number of uncovered tasks and the lower number of used taxis in the solution. Also the number of duties with overtime is lower, but the total overtime in minutes is higher. With AAATDR on average more duties are modified and on average much more computation time is required than with CGDDS.

For the case when reserves are used we can compare all scenarios, since the agent application also could solve them all within 20 minutes. Overall seen, CGDDS clearly performs better than the actor-agent application. The total costs are 19,616 compared to 35,097 for the multi-agent system. Furthermore, CGDDS is on average more than two times faster than the actor-agent application. With CGDDS less tasks are left uncovered, although the difference is small. There is a large difference in average taxi usage: CGDDS uses on average 2 taxis, while AAATDR uses 5 taxis. This is possibly caused by the limited possibilities of CGDDS to use taxis, because if a task is concerned that is not the last task of the driver’s duty, only taxis that take the driver over the disruption can be used. Also, only when a task is the last task (taking the driver back to its crew base) any taxi drive is allowed. AAATDR can use all taxi drives and it does not matter in which part of the duty the task is. On the other hand, CGDDS modifies more duties and uses more reserves. Also, in CGDDS solutions there are more duties with overtime. But although there are more duties with overtime, the total overtime is lower compared to the multi-agent system. Although we do not have complete results in Table 5.1, it seems that AAATDR is performing well on smaller instances, but worse on larger instances.

In Table 5.2 we consider the overall performance of each schedule for both methods. Again due to missing results for Abcoude, ’s Hertogenbosch, Alkmaar, Rotterdam and Schiphol, we present for both methods an extra row for averages over only half of the scenarios.

Considering only the costs in Table 5.2, the schedule based on ’Sharing Sweet & Sour’ with 10 minutes transfer time and reserves is the most robust when CGDDS is used, followed very closely by ’Destination:Customer’ with reserves included. The difference is however negligible small. That ’Destination: Customer’ performs well is just like we would expect; it is designed to be as robust as possible. That ’Sharing Sweet & Sour’ with 10 minutes transfer time performs even better and of course also better than the other ’Sharing Sweet & Sour’ schedules is on the first sight quite remarkable. One would expect that the plans with longer transfer times are easier to reschedule. That still holds when we consider the schedules without reserves; $SS_{10}R_0$ is one of the least robust schedules. But in combination with reserves, a tight schedule (in terms of short transfer times) seems to improve robustness.

’Sharing Sweet & Sour’ with 20 and 15 minutes transfer times and reserves included perform worse than DC_{20} and SS_{10}, but the difference is small. Between SS_{20} and SS_{15} there is almost no difference in robustness. Without reserves, the difference is larger, but for the 10 smaller scenarios SS_{15} is actually more robust than both SS_{20} and SS_{10}. This indicates, compared to the other schedules, that SS_{15} is harder to reschedule when combined with the larger cases. Later on, when we consider the scenarios in more detail we will analyze why that is the case.

Comparing the plans with and without reserves, we notice that reserves are the most useful for the plan with 10 minutes transfer time. For this plan the most improvement is gained by using reserves (roughly 77% cost reduction). Besides SS_{10}, reserves also have a large impact on the results of SS_{15} and DC_{20} (74% and 73% respectively). The least improvement is obtained for SS_{20}, where the costs are reduced with 67%. To achieve those reductions, SS_{15} uses the most reserves on average (29) and DC_{20} the least reserves (27).

We notice that with CGDDS, with or without reserves, there are always some uncovered tasks. In most scenarios there are some tasks that start quickly after the occurrence of the disruption at stations without reserves and can therefore not be covered. We will illustrate this later on. Another reason could be the size of the core problem in CGDDS. Using a larger core problem will make conflict solving easier but will at the same time increase the computation time.

For AAATDR we see more or less the same pattern. Only now 'Destination: Customer' with reserves is the most robust plan (considering costs and uncovered tasks). Also, $SS_{10}R_1$ is the most robust plan of the plans based on 'Sharing Sweet & Sour' and $SS_{15}R_1$ is the least robust schedule. Considering the plans without reserve (for half of the instances), we notice that SS_{10} is performing well, even better than the same plan including stand-by duties. However, the larger instances are not included so it is not fair to compare both schedules.

The number of uncovered tasks with AAATDR is never higher than with CGDDS when reserves are not included. When reserves can be used, AAATDR leaves slightly more tasks uncovered, but modifies less duties, uses less reserves and produces less duties with overtime. However the total overtime is higher, so the average overtime per duty is much higher. Besides, the multi-agent system uses more taxis and needs more time to find the solution.

As we will also see later on, the figures in Table 5.2 do not necessarily represent the results for each scenario individually. Therefore, we introduce Table 5.3 in which in two parts (without and with reserves) for each schedule the percentage that it is the most robust schedule (ranked first) to that it is the least robust schedule (ranked fourth) is represented. For example, for $SS_{20}R_1$, the value for share ranked first of 15% for CGDDS means of all 20 scenarios it is ranked as the most robust schedule 3 times.

The table shows that individual rankings are not consistent with the overall results. For example, $SS_{10}R_0$ is for 60% (40% ranked first, 20% ranked second) of the scenarios the most robust or second most robust schedule, while in the overall statistics $SS_{10}R_0$ is only more robust than $SS_{15}R_0$. This must indicate that the 60% of scenarios that $SS_{10}R_0$ performs well are small scenarios and the 40% that it performs not so well are large scenarios with a high numer of uncovered tasks and high correspondingcosts. Considering the total table in the Appendix, this is indeed the case. In four of the five times that $SS_{10}R_0$ is the least robust schedule, the costs are very high. Compared to other schedules, the difference in costs can be up to almost 200,000. These differences have of course a huge impact on the averages.

This is also true for other schedules in Table 5.3. It shows us that we have to be careful with overall conclusions. The averages of each schedule can be seriously affected by one or more scenarios. Since all cases are not evenly likely to occur, it seems appropriate to analyze the *performance profile* of the different schedules. The performance profile was introduced by [4] in order to analyze the performance of optimization software. For our purpose the solvers s will be replaced by the schedules and the problems p are going to be replaced by the cases. Then the profile for each schedule and case is given by

$$P\left(r_{s,p} \leq \tau : 1 \leq s \leq n_s\right).$$

Table 5.2: Average performance of each schedule for CGDDS (C) and the AAATDR (A)

Schedule	AD	MD		UR		UT		TT		OD		TO		Costs		CT	
		C	A	C	A	C	A	C	A	C	A	C	A	C	A	C	A
$SS_{20}R_0$*	18	27	40	0	0	2.2	1.4	3	2	8	3	223	362	54,036	40,874	0:17	4:33
$SS_{20}R_0$	26	46	-	0	-	2.5	-	3	-	15	-	478	-	62,547	-	1:16	-
$SS_{20}R_1$	26	72	59	28	11	0.6	0.8	3	6	4	2	149	168	20,500	35,091	2:10	5:35
$SS_{15}R_0$*	17	27	41	0	0	1.6	1.5	3	2	7	4	222	426	41,588	46,379	0:20	5:12
$SS_{15}R_0$	25	42	-	0	-	3.3	-	4	-	15	-	506	-	80,359	-	1:06	-
$SS_{15}R_1$	25	72	58	29	12	0.6	1	3	5	4	2	135	175	20,788	38,906	1:56	4:56
$SS_{10}R_0$*	17	26	40	0	0	1.8	0.9	2	1	9	3	259	320	43,899	26,467	0:20	5:01
$SS_{10}R_0$	25	45	-	0	-	2.95	-	3	-	19	-	633	-	71,994	-	1:08	-
$SS_{10}R_1$	25	71	57	28	12	0.5	0.7	2	5	4	3	127	227	16,572	34,054	2:52	4:51
$DC_{20}R_0$*	16	30	37	0	0	1.5	1	3	2	7	3	218	366	39,471	36,331	0:23	4:50
$DC_{20}R_0$	24	43	-	0	-	2.45	-	3	-	15	-	488	-	61,248	-	0:58	-
$DC_{20}R_1$	24	68	55	27	11	0.45	0.65	2	6	4	3	122	198	16,598	32,336	1:45	4:33

* Averages determined with the available results (10 instead of 20 scenarios).

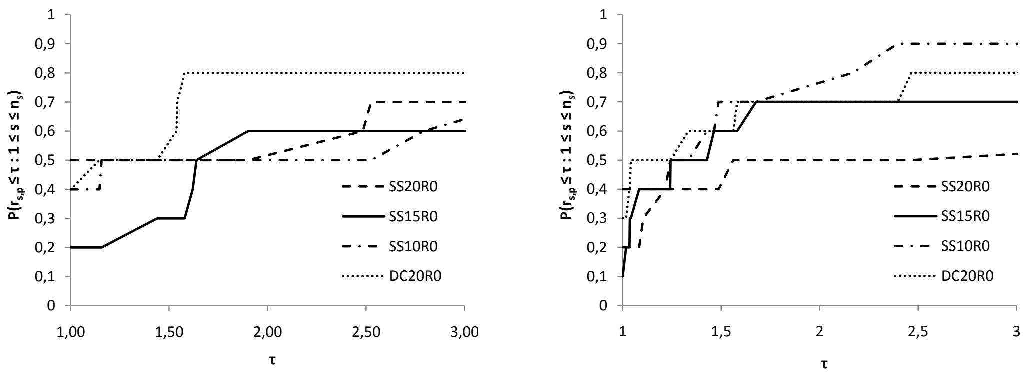
Table 5.3: Robustness ranking of the schedules for CGDDS (C) and AAATDR (A)

Schedule	% 1st		% 2nd		% 3rd		% 4th	
	C	A	C	A	C	A	C	A
$SS_{20}R_0$*	20	50	20	0	20	20	40	30
$SS_{15}R_0$*	10	20	40	0	20	30	30	50
$SS_{10}R_0$*	40	40	30	40	10	10	20	10
$DC_{20}R_0$*	30	40	10	30	50	20	10	10
$SS_{20}R_0$	25	-	35	-	15	-	25	-
$SS_{15}R_0$	15	-	20	-	35	-	30	-
$SS_{10}R_0$	40	-	20	-	15	-	25	-
$DC_{20}R_0$	20	-	25	-	35	-	20	-
$SS_{20}R_1$	15	40	15	15	40	20	30	25
$SS_{15}R_1$	30	40	25	15	15	25	30	20
$SS_{10}R_1$	30	50	35	25	15	10	20	15
$DC_{20}R_1$	35	55	25	10	20	20	20	15

* Percentages determined with half of the results (10 instead of 20 scenarios).

It is the probability that the schedule's solution is within a factor τ of the best found solution for the same case. $r_{s,p}$ represents the value of the solution obtained by using schedule s for case p divided by the best found solution for case p. Furthermore, n_s equals the number of cases solved with schedule s. When τ increases the probability will go to 1 for all schedules. This imposes that it is preferred to have high probabilities for low τ's.

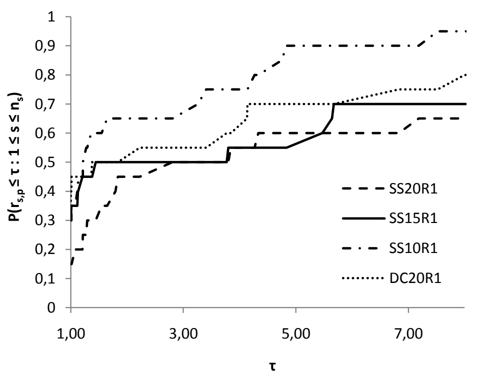
Figure 5.1.a shows the performance profile when CGDDS and AAATDR are used for each schedule without reserve duties and Figure 5.1.b shows the profile when CGDDS and AAATDR are used for rescheduling the same schedules with reserves. Both profiles are constructed with respect to the total costs of rescheduling. Some profiles do not show the probabilities for large τ, because otherwise the figures would become unclear.



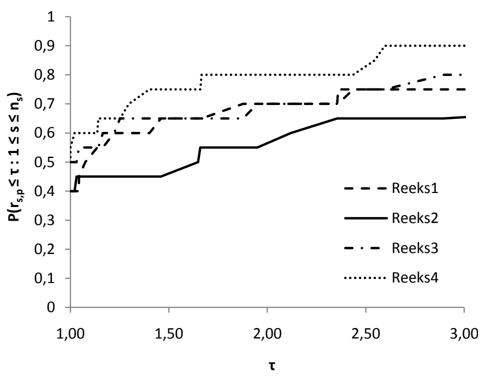
CGDDS AAATDR

Figure 5.1.a: Performance profiles for CGDDS and AAATDR for the four basic schedules without reserves

Interestingly, the schedules without reserves rescheduled with CGDDS do not show a clear distinction (Figure 5.1.a). The most robust schedule seems to be SS_{10}, as the probability that



CGDDS



AAATDR

Figure 5.1.b: Performance profiles for CGDDS and AAATDR for the four basic schedules with reserves

it is within a factor 2.40 times the best solution is 0.9 and with probability 1 it is within a factor 6.60 times the best solution. The other schedules their profile have more or less the same development, only SS_{20} stays somewhat behind. The probability that a solution obtained with SS_{20} is within a factor 7.25 is only 0.7. For AAATDR it is immediately clear that best schedule without reserves is DC_{20}, as with probability 0.8 a solution is within 1.54 times the best solution. Furthermore, SS_{20} is also doing quite good, while the other two schedules leave somewhat behind.

In Figure 5.1.b we see that with CGDDS SS_{10} with reserves is the most robust schedule. A solution is with probability 0.9 within 4.83 times the best solution. SS_{20} and SS_{15} are less robust, where some scenarios have a high factor compared to the best solution. For AAATDR DC_{20} is the most robust schedule and SS_{15} the least robust schedule.

So far we only considered the schedules. As shown by Figures 5.1.a and 5.1.b for some scenarios the best solution can be multiple times better than other solution. In the next sections we will, for both methods, consider the harder scenarios and evaluate them in more detail in order to find out why a schedule is less robust than other schedules.

5.2 CGDDS in detail

To keep things clear, we mention again that a *case* is a disruption at a specific place, whereas a *scenario* is a schedule with a set of reserve duties combined with a case. Although a case is for each schedule on the same place and has the same corresponding emergency scenario, it affects each schedule differently resulting in different scenarios.

For CGDDS the hardest scenarios are scenarios with the Abcoude cases, the Alkmaar cases, the Schiphol cases and scenarios with the second Lelystad case. This is remarkable, as we concluded earlier that most of these cases do not cause that many affected duties. One would expect scenarios with the 's Hertogenbosch cases also to be hard. In terms of computation time for scenarios with the second 's Hertogenbosch case that is true; the scenarios containing that case required relatively a lot of computational effort. But considering the solutions, the scenarios containing that case seem not to be that hard.

For the scenarios with the Abcoude cases, we see large difference in the number of uncovered tasks between scenarios where no reserves are used and the scenarios where all reserves are used; reserve duties make it (much) easier to solve the scenarios. The tasks that even with the help of reserve duties cannot be covered, are mostly tasks starting in Gouda within 20 minutes after the disruption has started. If there are more drivers needed than arriving in Gouda, it takes some time before reserves from Utrecht, The Hague or Rotterdam are at Gouda. So if no other regular duties can be modified such that the tasks starting in Gouda can be covered, we remain with uncovered tasks. Table 5.4 shows the results for Abcoude.

The scenarios containing the first Abcoude case are solvable within reasonable amount of time (up to roughly 3 minutes). The most robust schedule seems to be $SS_{10}R_1$, since it has no uncovered tasks and therefore the lowest costs. However on the criteria of overtime, $SS_{15}R_1$ performs better. Still it is the least robust schedule due to the highest number of uncovered tasks. And due to the uncovered tasks, the costs are roughly 40,000 higher than the costs for $SS_{10}R_1$. Such differences of course influence the averages discussed earlier.

Furthermore, the table shows that the afternoon Abcoude case (Ac_2) for the crew schedule based on 10 minutes transfer time including stand-by duties is the hardest to solve in terms of computational effort; it took more than 12 minutes. Also both SS_{20} schedules required a lot of computation time. This probably related to the high number of affected duties in combination with a high number of modified duties. In the second scenario (with reserve duties) also reserve duties are added to the core problem, which makes the problem even larger and therefore it takes longer to solve it. Although the computation times are long for SS_{10} and SS_{20}, both schedules perform well in terms of robustness. SS_{15} is clearly the least robust schedule in this scenario. Again there is a large difference in costs that influences the averages. For DC_{20} reserve duties turn out to be only useful to reduce the number of duties with overtime and the total overtime. To the question which specific reserve duties are needed for these scenarios there is no answer, since (almost) all at the time of the scenario available and even some duties starting later on the day are used, no matter where the reserve duty starts.

Table 5.4: Results CGDDS for Abcoude (Ac)

Scenario	AD	MD	UR	UT	TT	OD	TO	Costs	CT
$Ac_1SS_{20}R_0$	50	94	0	3	5	41	1,380	83,585	2:14
$Ac_1SS_{20}R_1$	50	131	40	1	4	11	400	33,900	3:07
$Ac_1SS_{15}R_0$	46	87	0	7	3	52	2,025	162,425	1:22
$Ac_1SS_{15}R_1$	46	132	48	2	2	13	378	48,230	3:05
$Ac_1SS_{10}R_0$	49	100	0	3	2	65	2,509	82,315	1:56
$Ac_1SS_{10}R_1$	49	128	49	0	2	17	489	8,815	2:39
$Ac_1DC_{20}R_0$	46	83	0	4	8	53	2,181	117,560	1:38
$Ac_1DC_{20}R_1$	46	112	39	1	5	11	387	36,435	1:49
$Ac_2SS_{20}R_0$	56	112	0	1	1	21	494	26,005	7:34
$Ac_2SS_{20}R_1$	56	139	40	0	1	4	134	3,875	11:17
$Ac_2SS_{15}R_0$	48	99	0	4	4	29	842	95,765	3:31
$Ac_2SS_{15}R_1$	48	138	44	2	4	8	353	52,840	6:47
$Ac_2SS_{10}R_0$	47	104	0	1	2	42	1,112	32,470	4:22
$Ac_2SS_{10}R_1$	47	117	40	0	2	3	94	6,380	12:18
$Ac_2DC_{20}R_0$	49	86	0	1	2	48	707	29,945	2:14
$Ac_2DC_{20}R_1$	49	137	42	1	2	5	116	26,470	4:35

Next we will consider the results of the Alkmaar scenarios, which are shown in Table 5.5. Without reserves, the number of uncovered tasks are quite high. But there is a lot of improvement when reserve duties can be used. Still there are a lot of reserve duties used for some scenarios, but for example the scenario $Amr_2SS_{15}R_1$ and $Amr_2DC_{20}R_1$ require somewhat less stand-by duties. For these scenarios we see specific reserve usage, namely only duties that start at crew bases that are on the same lines as station Alkmaar is on. That imposes that mostly reserve duties starting at Amsterdam, Utrecht, Arnhem, Nijmegen, 's Hertogenbosch and to less extend at The Hague and Rotterdam, are used in the solution.

Table 5.5: Results CGDDS for Alkmaar (Amr)

Scenario	AD	MD	UR	UT	TT	OD	TO	Costs	CT
$Amr_1SS_{20}R_0$	23	53	0	4	1	23	892	91,960	1:04
$Amr_1SS_{20}R_1$	23	88	41	1	2	0	0	26,000	1:17
$Amr_1SS_{15}R_0$	25	38	0	7	1	15	600	152,845	0:31
$Amr_1SS_{15}R_1$	25	60	25	1	2	1	41	26,260	0:46
$Amr_1SS_{10}R_0$	24	50	0	7	3	31	1,158	159,550	0:50
$Amr_1SS_{10}R_1$	24	81	36	1	3	2	6	29,030	1:25
$Amr_1DC_{20}R_0$	27	47	0	5	2	25	903	114,985	1:06
$Amr_1DC_{20}R_1$	27	74	32	1	1	2	20	26,010	1:11
$Amr_2SS_{20}R_0$	18	43	0	7	2	8	650	150,100	0:13
$Amr_2SS_{20}R_1$	18	61	28	0	4	4	290	33,865	0:17
$Amr_2SS_{15}R_0$	19	31	0	9	4	6	549	195,420	0:17
$Amr_2SS_{15}R_1$	19	53	21	0	6	8	146	18,880	0:14
$Amr_2SS_{10}R_0$	22	44	0	5	3	6	399	111,240	0:14
$Amr_2SS_{10}R_1$	22	75	29	1	1	3	181	24,085	0:49
$Amr_2DC_{20}R_0$	19	30	0	9	5	6	347	197,035	0:15
$Amr_2DC_{20}R_1$	19	52	19	1	7	5	220	42,040	0:14

For Schiphol more or less the same as for Alkmaar holds. When there are no reserves there are a lot of uncovered tasks leading to very high costs. But when reserve duties can be used, solving the scenario becomes much easier. This is shown by Table 5.6. The tasks that still cannot be covered mostly concern tasks starting in Leiden and Hoorn. Again we come to the same conclusions as before, as the tasks are at the start of the scenario and both stations have little or no reserves at all (Hoorn and Leiden respectively). Again we see more used reserve duties at stations that are on the same lines as Leiden and Schiphol, so Amsterdam and Utrecht are used a lot, but also stations in southern direction up to Roosendaal.

Besides Abcoude, Alkmaar and Schiphol, CGDDS is clearly having some difficulties with the Lelystad cases, especially scenarios containing the afternoon situation. Table 5.7 provides the results for the Lelystad afternoon case. In the scenarios without reserves there are cases where up to 7 tasks are left uncovered ($Lls_2SS_{10}R_0$). The same scenario, but with reserve duties, is solved by CGDDS with still leaving 2 tasks uncovered. For all scenarios, the problematic tasks are tasks from Almere to Amsterdam and tasks from Almere to Hoofddorp, both at the beginning of the scenario. For Almere the same analysis is true as for Gouda in the Abcoude scenario. A clear pattern in reserve duties can be found for $SS_{20}R_1$ and $SS_{10}R_1$, as the usage is low for these scenarios. Clearly most used reserves start in Amsterdam and Utrecht.

There are no cases that are consistently (over all scenarios) easy solve with CGDDS. Only the Beilen, Heerlen, Rotterdam and Zoetermeer scenarios seem to be more easy than others, but even for those scenarios there are some schedules that produce uncovered tasks, even when reserves are used. Only in 4 cases (with reserves) a scenario is solved against 0 costs.

Table 5.6: Results CGDDS for Schiphol (Shl)

Scenario	AD	MD	UR	UT	TT	OD	TO	Costs	CT
$Shl_1SS_{20}R_0$	28	41	0	2	4	21	827	60,350	1:04
$Shl_1SS_{20}R_1$	28	63	27	0	3	8	277	10,675	1:51
$Shl_1SS_{15}R_0$	27	42	0	6	2	18	666	130,390	0:31
$Shl_1SS_{15}R_1$	27	64	24	0	2	3	138	6,890	0:23
$Shl_1SS_{10}R_0$	31	39	0	11	6	16	582	241,625	0:50
$Shl_1SS_{10}R_1$	31	80	30	1	4	6	153	32,715	0:34
$Shl_1DC_{20}R_0$	27	39	0	3	3	14	421	71,700	1:06
$Shl_1DC_{20}R_1$	27	54	19	0	3	3	72	9,495	0:19
$Shl_2SS_{20}R_0$	30	71	0	6	4	21	651	135,875	0:13
$Shl_2SS_{20}R_1$	30	76	26	0	3	4	126	9,715	0:45
$Shl_2SS_{15}R_0$	29	45	0	10	9	11	385	229,140	0:17
$Shl_2SS_{15}R_1$	29	65	27	2	5	3	81	55,120	0:50
$Shl_2SS_{10}R_0$	31	41	0	11	7	24	667	245,210	0:14
$Shl_2SS_{10}R_1$	31	72	26	3	4	6	170	73,075	1:11
$Shl_2DC_{20}R_0$	26	42	0	3	1	16	554	66,565	0:15
$Shl_2DC_{20}R_1$	26	55	21	2	0	2	44	40,220	1:20

Table 5.7: Results CGDDS for the afternoon case of Lelystad (Lls_2)

Scenario	AD	MD	UR	UT	TT	OD	TO	Costs	CT
$Lls_2SS_{20}R_0$	13	26	0	5	6	13	284	129,680	0:09
$Lls_2SS_{20}R_1$	13	40	13	2	6	7	240	59,280	0:25
$Lls_2SS_{15}R_0$	12	23	0	3	5	10	289	76,700	0:11
$Lls_2SS_{15}R_1$	12	43	19	1	5	7	260	36,370	1:18
$Lls_2SS_{10}R_0$	11	24	0	7	8	12	397	166,315	0:14
$Lls_2SS_{10}R_1$	11	38	12	1	6	10	234	39,220	0:23
$Lls_2DC_{20}R_0$	12	31	0	5	5	14	526	121,240	0:23
$Lls_2DC_{20}R_1$	12	66	26	1	4	6	92	32,270	0:37

Overall we notice that the reserve usage is high, but that is not strange as there are no costs related to using (modifying) a reserve duty. If we take a closer look at the scenarios, we see that for the most scenarios almost all at that time available reserves are used, also from crew bases that are not within proximity of the place where the disruption occurred. For other scenarios there is pattern that can be discovered, namely reserve duties at stations that are on the same lines as the point where the disruptions take place are used more frequently than stand-by duties starting at other stations. This is a quite logical result.

Besides reserve usage, also the number of modified duties is quite high. This is not remarkable, as there is no penalty on modifying duties. The model only considers overtime, uncovered tasks and taxis, and if those can be reduced by modifying a lot of duties, including reserve duties, CGDDS will do so as there are no additional costs.

5.3 AAATDR in detail

Before discussing the results we have to mention that the agent model can produce different results in a new run. This means that the results in the table are not consistent; if we would run all scenarios again we could get some different results. The reason for that is the communication process of the model. The communication between the different parts is not ordered. So in different runs different orders of incoming messages can occur, leading to another order of message processing and therefore possibly leading to different results. After some testing we noticed that the differences are not significant, so there is a high probability that the conclusions will remain the same if the tests are executed multiple times.

The actor-agent application is clearly having some difficulties with the scenarios CGDDS also having difficulties with. For these scenarios the costs can be very high, due to more uncovered tasks compared to CGDDS. The difference in performance is made on these scenarios, as for smaller cases AAATDR is not necessarily worse. The agent system uses less reserves, but on the other hand it uses more taxis and in many scenarios more overtime. The uncovered tasks with AAATDR occur mostly for the same reasons as described for CGDDS.

Besides the harder cases for CGDDS, Abcoude, Alkmaar, Schiphol and Lelystad afternoon, the actor-agent application is also having some problems with solving the Lelystad morning case. We will consider the scenarios corresponding to this case in more detail. Table 5.8 represents the results for the morning case of Lelystad (Lls_1).

Table 5.8: Results actor-agent application for the morning case of Lelystad (Lls_1)

Scenario	AD	MD	UR	UT	TT	OD	TO	Costs	CT
$\text{Lls}_1\text{SS}_{20}\text{R}_0$	11	51	0	4	9	14	1,900	141,310	10:39
$\text{Lls}_1\text{SS}_{20}\text{R}_1$	11	26	6	7	9	2	287	162,285	4:36
$\text{Lls}_1\text{SS}_{15}\text{R}_0$	10	30	0	9	8	7	864	231,705	12:03
$\text{Lls}_1\text{SS}_{15}\text{R}_1$	10	24	5	3	6	4	252	86,440	3:53
$\text{Lls}_1\text{SS}_{10}\text{R}_0$	11	26	0	7	4	4	450	163,590	3:42
$\text{Lls}_1\text{SS}_{10}\text{R}_1$	11	28	8	4	2	4	399	90,190	3:24
$\text{Lls}_1\text{DC}_{20}\text{R}_0$	10	39	0	7	8	7	780	217,325	12:21
$\text{Lls}_1\text{DC}_{20}\text{R}_1$	10	27	5	6	7	4	218	144,045	4:38

Actually, that the morning situation in Table 5.8 cannot be solved easily is not very surprising. It is not strange that the model is having problems with the first couple of conflicts in those scenarios as there are no other (or very little) drivers available to take over tasks. This is still true for the scenarios with reserves, as there are no reserves around Lelystad and Almere. The nearest location with reserve duties is Amsterdam. Also, it is early in the morning when the scenario starts, so there is a little number of drivers in the system. Apparently using taxis would take too much time, so the tasks are left uncovered.

The scenarios that are somewhat easier to solve with CGDDS are also easy to solve with the actor-agent application. As said before, the model even performs better on these 'easy' scenarios. And besides Beilen, Heerlen, Rotterdam and Zoetermeer, the agent system has also less difficulties with the Almelo scenarios. In total there are 27 scenarios that are solved against 0 costs, all for the mentioned scenarios.

Regarding the scenarios where reserves are used, we notice that the agent application uses not that many reserve duties for rescheduling (especially compared to CGDDS). In most cases the reserve duties that are nearest to the place where the disruption occurred are all used. Besides those duties, we see that the multi-agent system uses several other duties that are further away. These duties are in most cases in 'second range' and sometimes in 'third range'. So when the reserve closest to the disruption are used, the reserves second closest are considered. For example, the scenario with a disruption at 's Hertogenbosch first uses reserve duties from 's Hertogenbosch and Eindhoven. However, at any point in time at most 3 reserve duties are available at both stations. The model therefore also considers reserve duties at Utrecht, Nijmegen, Roosendaal and even the third range stations Arnhem, Amersfoort and Amsterdam. Finally, a general observation is that reserve duties from Utrecht, Amsterdam and The Hague to less extent, are used in many scenarios. This can be explained by the fact that at these crew bases most reserve duties are available. Also these stations are the most central stations in the network, so for many scenarios these stations are at most within third or second range.

An overall conclusion is that the number of uncovered tasks is low when using this application. Only the Lelystad morning and Alkmaar afternoon scenario turn out to be hard if we consider the uncovered tasks. As a result of the low number of uncovered tasks, the overtime and number of used taxis are relatively high. This result is not strange, since canceling tasks is the last option in the model. Like said before, if the conflict can be resolved, the model will go for that solution.

Finally, we can conclude that the agent application does not work well in all cases. Some of the strange results we obtained can be explained by the heuristics that are behind the model. Using greedy heuristics does often not result in an optimal or near-optimal solution and can even result in very bad solutions. Comparing the solutions of AAATDR with solutions of other approaches can give us an indication of how the heuristic performs.

5.4 Comparing CGDDS with 2P-RSPPRC

For comparing 2P-RSPPRC with CGDDS we will present less statistics and make a less deep analysis than we did so far. The main reason for that is that the comparison of 2P-RSPPRC with CGDDS is already done by [17]. In that research, some of the same cases as presented in this thesis are used and analyzed. We are mainly interested in robustness differences of the schedules for both methods, so the statistics that are the most interesting to us are the non-covered A-B tasks (UT A-B), non-covered A-A tasks (UT A-A) and the total costs (costs). Furthermore, the calculation time (CT) is presented in order to get an indication of how fast this approach is compared to CGDDS. The results are summarized in Table 5.9. The complete results can be found in the Appendix (Table A.2.a and A.2.b).

Table 5.9: Average performance of each schedule for CGDDS (C) and the 2P-RSPPRC (P)

Schedule	AD	UT A-B		UT A-A		Costs		CT	
		C	P	C	P	C	P	C	P
$SS_{20}R_0$	27	0.7	13.2	2.25	3.5	47,817	287,522	0:48	1:09
$SS_{20}R_1$	27	0.35	2.1	0.3	0.75	29,988	77,197	0:53	0:39
$SS_{15}R_0$	26	0.65	16.65	4.25	4.7	52,916	358,985	0:51	1:18
$SS_{15}R_1$	26	0.3	2.5	0.3	1.1	28,807	84,607	0:46	0:34
$SS_{10}R_0$	26	0.7	16.6	3.4	5.25	50,340	358,226	0:57	1:17
$SS_{10}R_1$	26	0.3	2.4	0.3	1	27,199	81,084	0:55	0:39
$DC_{20}R_0$	25	0.8	15.47	2.55	4.42	48,905	334,109	0:49	1:28
$DC_{20}R_1$	25	0.35	1.68	0.05	1	27,612	69,590	0:44	0:32

We do not display the results, but now that the modified duties and UR can only increase against costs (UR imposes costs as the reserve duties is modified and contains tasks from other duties), there are much less modified duties and there is a lower reserve usage compared to the same statistics for the comparison of CGDDS with AAATDR.

If we compare the results for CGDDS with the results of CGDDS in Table 5.2, we see that the costs decreased when no reserve duties are available, while the costs increased when reserve duties are available. That is not strange, as the costs for canceling a task were 20,000, no matter if it concerns an A-B task or A-A task. Therefore rescheduling schedules without stand-by duties is less expensive. For the schedules with reserve duties the number of uncovered tasks is so small, that the difference in costs is low. However, in the new situation CGDDS has more cost parameters. For example, there are costs for modifying duties. These additional costs cause an increase in the total costs compared to the costs in Table 5.2. The robustness of the schedules is similar to the CGDDS results in table 5.2. Without reserves, SS_{20} and DC_{20} are the most robust schedules. Again a combination of a tight schedule and reserves improves the robustness. Just like before, the differences between schedules is very small.

As shown in Table 5.9, 2P-RSPPRC requires little computation time to solve the scenarios, but more than CGDDS. Just like with CGDDS, reserves improve the robustness of the schedules a lot. However, the differences between with and without reserves are much larger, where the largest improvement is even 82% (for DC_{20}). There is not a clear pattern in reserve usage; it looks like all at the start of the disruption available stand-by duties are used in the solution. But even with reserves, the number of uncovered A-B tasks and costs are higher than when rescheduling is done without reserves and using CGDDS. So we find the same result as [17]; CGDDS clearly outperforms 2P-RSPPRC.

Just like when using the other rescheduling methods, DC_{20} seems to be a robust schedule when rescheduling is done with 2P-RSPPRC. However, without reserves SS_{20} is the most robust schedule and both with and without reserves SS_{15} is the least robust schedule. So the differences are again small and it is hard to point out the most robust schedule, but it seems that more using more slack in the schedules is preferred when using 2P-RSPPRC. Again Abcoude, Alkmaar, Lelystad and Schiphol cases are difficult. In addition also Rotterdam and to less extent 's Hertogenbosch turn out to be hard cases. We saw in the previous section that these harder instances can have a huge impact on the averages. Besides, the cases are still not evenly likely to occur, so to analyze what the most robust schedules are when using 2P-RSPPRC we again consider the performance profiles of the schedules. Figure 5.2 presents the performance profiles for the schedules both without and with reserves.

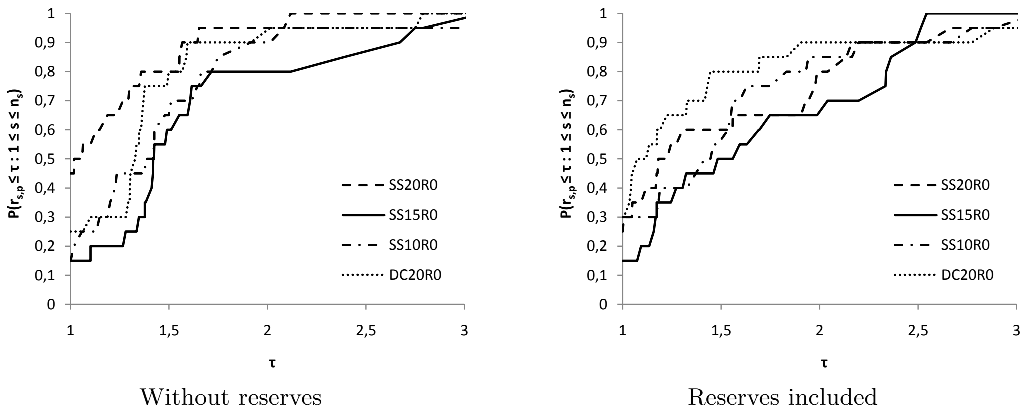


Figure 5.2: Performance profiles for 2P-RSPPRC schedules without and with reserves

When the schedules are not combined with reserves, SS_{20} and to less extent DC_{20} are the most robust schedules. SS_{10} and especially SS_{15} have lower probabilities to be within a low factor of the best solution. When reserves can be used, DC_{20} is the most robust plan when using 2P-RSPPRC for rescheduling. Compared to the performance profiles in Figure 5.1.a and 5.1.b, we see that the performance profiles in Figure 5.2 have higher probabilities for lower τ's. That is because we are dealing with higher costs. For example, if with CGDDS there is one schedule that only has some costs for overtime, while another schedule has costs for an uncovered tasks, there is a large factor involved. With 2P-RSPPRC, there is a much smaller factor involved by comparing a schedule with for example 8 and a schedule with for example 10 uncovered tasks. However, the true difference in costs for these examples is larger with 2P-RSPPRC. Therefore we cannot conclude that the difference in robustness between schedules when using 2P-RSPPRC is smaller only based on the τ's.

6 Robust schedules

So far we only considered schedules that are used in earlier research. It is also interesting to consider other schedules. However, there is a limited number of variations possible due to all the restrictions of the production plan 'Sharing Sweet & Sour'. Still, there are some modifications that can be done to obtain numerous new schedules that possibly affect the rescheduling process. In this chapter we will introduce one new schedule that is based on 'Sharing Sweet & Sour', but now it is assumed that drivers have complete route knowledge. We will also present a schedule requested by dispatchers that possibly can be used under extreme circumstances. We will also introduce various new sets of stand-by duties. Since we noticed in the previous chapter that the differences in robustness between the schedules are very small, we think that the stand-by duties have a larger impact on robustness compared to the use of different schedules. Finally, we also consider a more realistic situation where multiple cases a day occur.

6.1 Extended experiments

In the previous chapter we showed that CGDDS clearly outperforms the other two approaches. So it would be reasonable to test new scenarios only with CGDDS. However, CGDDS has not yet been implemented at NS and rescheduling is still done manually. So from a practical point of view, it is interesting to also consider the 2P-RSPPRC approach. By doing so, we can analyze if a proposed schedule that is doing well when using CGDDS for rescheduling is also an improvement when it is rescheduled using the current, less intelligent way of rescheduling. So all the proposed adjustments in this section will be analyzed using two rescheduling methods.

6.1.1 Schedules

Making a new schedule is a time consuming process. Given that, and given the limited number of possibilities we have if we want to keep the production plan 'Sharing Sweet & Sour', we decided to generate only two new schedules. In the first new schedule, indicated by SS'_{20}, we keep all the basic rules as in the normal schedule with 20 minutes transfer time. Only the route knowledge is now not an issue anymore, that is, we assume that all drivers have the knowledge to drive all lines operated by NS. Besides that, we add some additional relief stations such that drivers can transfer at more stations. Since the problem turned out to become too large to solve when all stations are relief stations, we decided to only add stations that connect multiple lines. The number of duties without reserve duties is for SS'_{20} 868. This is in the same order of quantities as schedules in the base set.

The second schedule, indicated by DC'_{20}, is based on an idea to have a robust schedule that can be used during extreme situations, like the winter period in 2009. In this schedule, drivers can only drive trains from their crew base to another crew base or relief station and back. Until the meal break the duty consists of the same tasks. After the meal break a driver can drive on the same route or the driver can drive up and down between two other stations (of which one is his or her crew base). So actually this schedule is an extreme form of 'Destination: Customer', as drivers are assigned to a line and do not leave that line during their duty (only possibly after the meal break). But it is more extreme, as the driver can be assigned to only a part of that line. For example, in the schedule based on 'Destination: Customer' a driver can start in Groningen and drive a train on the 500-line to The Hague and another train on the same line back to Groningen. For DC'$_{20}$ this is not possible as the distance between Groningen and The Hague is too large. In this example the driver would still drive trains on the 500-line, but now only up to Zwolle and not further. This plan is not directly seen as a competitor in our search to robust plans in terms of the practical situation. The reason why this schedule is adopted in our analysis is that dispatchers are searching for a robust schedule that can be implemented during a period of (extreme) disruptions, like the winter period of 2009. The number of duties, without reserves, is 1125 for this schedule. This makes the schedule already before analysis impossible to implement in practice. So many extra duties makes the schedule way too expensive. But even besides the costs the schedule cannot be implement, as the number of duties in Utrecht is almost doubled. In practice this cannot be achieved.

6.1.2 Reserve duties

So far, we only considered schedules without reserves (R_0) and with all the reserves as scheduled on a regular day at NS (R_1). In the previous chapter we noticed that many reserves are used when solving the instances with CGDDS. However, now that we are going to penalize the modification of duties (as will be explained later on), we expect to use less reserves (as shown in [17] and [21]). So it is interesting to see whether it is worthwhile to use less reserves. For that purpose we can use half of the reserves (R_2). This set of stand-by duties is derived from R_1 according to a certain philosophy, which is explained in [21].

To analyze the importance of reserve crew, we make a set of reserve duties that are only stand-by in the busiest parts of the country. Therefore, we make a set with the same 84 stand-by duties as in R_1, but now the reserves are all located in Utrecht and Amsterdam (R_3). The main reason for this is to check if it matters where the reserves are located. So the question is whether this set of stand-by duties produces results that are closer to R_0 or to R_1. Furthermore, we possibly can validate the proposition that cases that are not in proximity of crew bases will have the same number of uncovered tasks in the first period no matter where the reserves are located. Finally, R_3 can indicate whether the first period with uncovered tasks is longer when reserves are located further away from the scenario than the period when reserves are nearby.

Optimizing the number of reserve duties, at which time they should be scheduled and at which crew base they should be located is very hard. Especially because one can never know where and when a disruption will take place and at what time. For that purpose a simulation model could be used with for each part of the track certain probabilities for a disruption to occur. We do not have a simulation model nor information about the probability a disruption will occur at a certain place in the railway network.

What we can do is analyzing the current set of reserve duties and try to find some improvements. We already shifted reserve duties to the two largest stations. However, since Utrecht and Amsterdam are very busy stations there are a lot of possibilities to travel to and from the stations, so if there is a disruption drivers will never be stuck due to the many travel possibilities. Also, most duties start at the crew bases Utrecht and Amsterdam and a lot of drivers pass those stations. This means that there are always possibilities to exchange tasks between duties.

Outside the Randstad, drivers can more easily get stuck on one side of the disruption. Furthermore, there are a lot of drivers who travel from outside into the Randstad and back. A result of this is that crew bases at the edges of the Randstad, for example at 's Hertogenbosch and Zwolle, have a relatively low number of duties. It seems reasonable to locate more reserves at those places, as both the Randstad and the outer areas can easily be reached. But if something occurs between the station on the edge of the Randstad and a station in the corner of the railway network, the corner is isolated from the rest of the railway network. Therefore also reserve duties at the corner stations are needed. Figure 6.1 shows the crew bases with the original number of reserve duties at those places (R_1), together with the number of stand-by duties of the set that we propose. This set will be indicated with R_4.

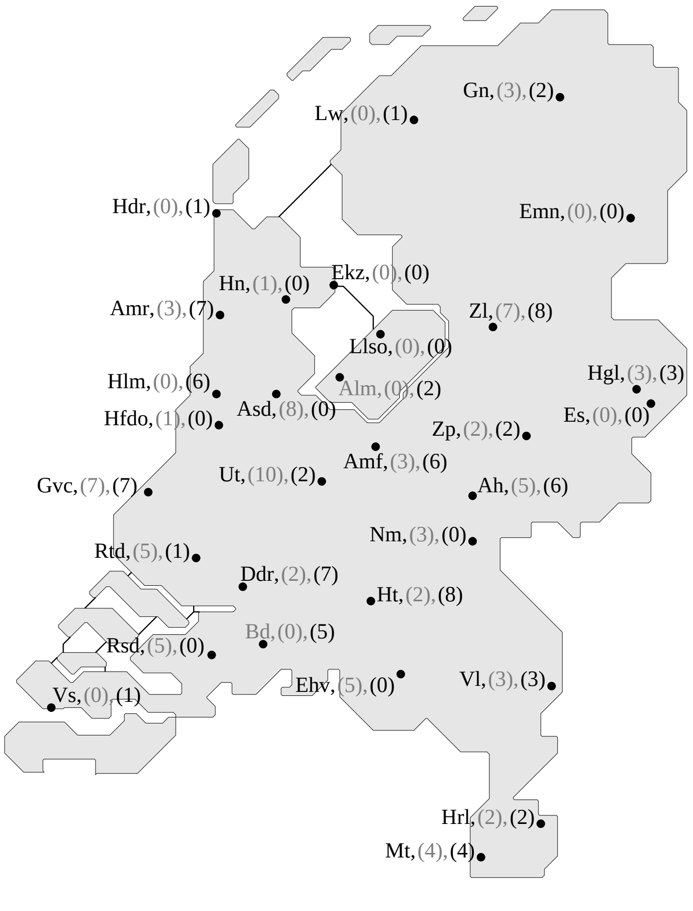


Figure 6.1: Number of reserve duties at each crew base, both for the regular situation (grey number) and the new situation (black number). Grey stations are not a crew base.

In the figure we see that at Utrecht and Rotterdam still some reserves are available. That is because we did not change the reserve duties that take place during the night. Furthermore we propose two new stations, that are not crew bases, where also reserves are available in the new situation.

Besides a place perspective we can also consider the reserve duties from a time perspective. Figure 6.2 shows for the set of reserve duties R_1 how many reserves are active at any point in time. Besides a line for the number of reserve duties at each half hour on a day, there is also a line for available reserve duties when the duties last one hour longer. We also present this line since all duties are allowed to be extended by at most one hour when rescheduling is necessary.

Figure 6.2 illustrates that the duties are not equally distributed over time. The distribution

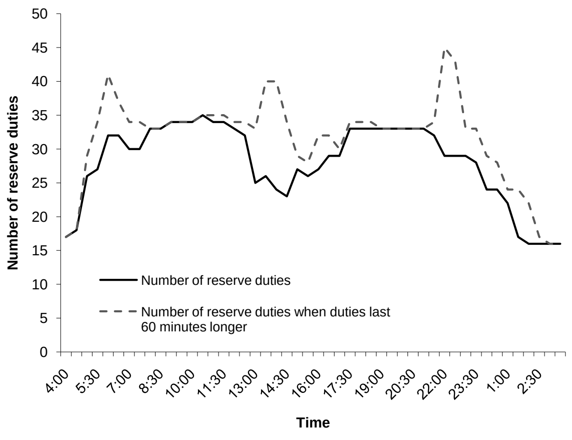


Figure 6.2: Number of active reserve duties for each half hour on a regular day at NS

of the stand-by duties is clearly related to the peak hours; most duties are available during the morning and the end of the afternoon/beginning of the evening. Such a distribution seems to make sense, since during peak hours there are some extra trains. Also, canceled trains due to uncovered tasks have a larger impact during rush hour than during a less busy period. But on the other hand, outside the peak hours there are not strictly less drivers active, except during the night. From that point of view it seems reasonable to have a more smooth distribution of active duties. It is not possible to obtain a completely smooth distribution, because we also have to account for the distribution of active stand-by duties at each crew base individually. Figure 6.3.a shows a smoother distribution of active duties without changing the number of reserve duties at each crew base compared to the duties in Figure 6.2. This set of stand-by duties will be indicated with R_5.

Considering in Figure 6.2 the active duties when we account for the possible extra 60 minutes, we notice a contrary effect. The peaks of active duties now lie outside the busiest hours. The reason for that is that at crew bases many stand-by duties start around the same time and when those duties are finished, new duties start at that moment. So by adding one hour, there is no transition anymore resulting in a peak starting at the moment the duties would normally change. If we smooth the distribution of active duties and still account for the same number of reserve duties at each crew base as in Figure 6.2 is the case, we come to the distribution as shown in Figure 6.3.b. If we then extract one hour from each duty we come to a new set, which we will call R_6.

6.2 Multiple cases

Testing a rescheduling method based on the assumption that there is only one disruption per day is not representative for the practical situation. Like said before, the Dutch railway network incurs on average three large disruptions per day. Currently, only CGDDS is capable of handling multiple instances. Besides, even when the other methods could deal with multiple instances, especially using the actor-agent application would be very time consuming. In case of rescheduling, it is often desirable in practice to modify a limited number of duties. For CGDDS we can set parameters with respect to the modification of duties and therewith minimize the total number of modified duties. Like said before, the actor-agent application cannot go into this direction. Also, as we showed in the previous section, CGDDS performs

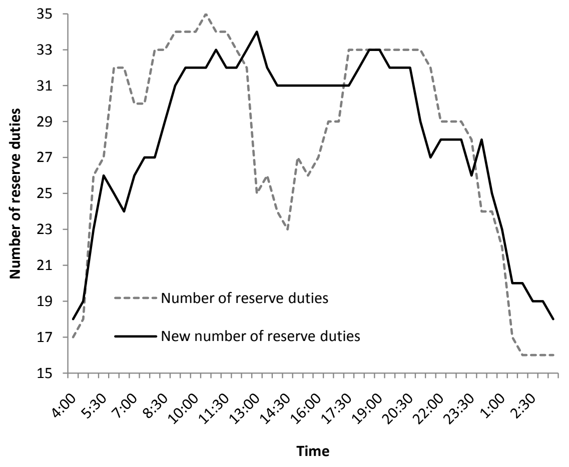


Figure 6.3.a: Smoothened number of active reserve duties for each half hour on a regular day at NS

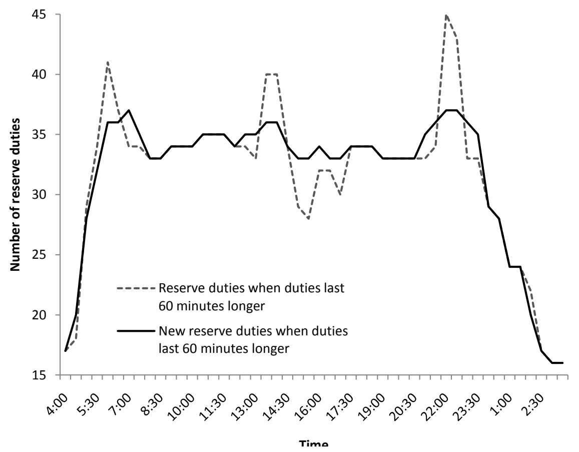


Figure 6.3.b: Smoothened number of active reserve duties for each half hour on a regular day at NS when duties last 60 minutes longer

better than the other two methods. Finally, since at NS CGDDS is going to be used in practice, we decided to do the remaining of our analysis only with CGDDS.

Case combinations

If we want to analyze the robustness of the schedules with respect to the practical situation, we need to model multiple cases per day. So when the schedule has been modified for one case, the adapted schedule has to be stored and used when the next case occurs. In that way we can keep on modifying adapted schedules, just as is done in practice.

Since on average three disruptions per day occur, we will create mostly instances with three disruptions. With the 20 cases introduced in Chapter 4 we can construct a lot of combinations. An overview of the 12 combinations that are going to be used is shown in Table 6.1. Details about place specific cases can be found in Table 4.2.

As we can see in Table 6.1, there two instances that contain four cases and two instances that contain two cases. All other instances contain 3 cases. These combinations indicate that

Table 6.1: Overview of the different case combinations

Combination	Cases
C_{01}	Lls_1, Ztm_1, Rtd_1, Hrl_2
C_{02}	Lls_1, Shl_1, Shl_2, Rtd_2
C_{03}	Ac_1, Ht_2
C_{04}	Amr_1, Aml_2
C_{05}	Bl_1, Ztm_2, Rtd_2
C_{06}	Hrl_1, Rtd_1, Ac_2
C_{07}	Ht_1, Lls_2, Amr_2
C_{08}	Aml_1, Lls_2, Bl_2
C_{09}	Ztm_1, Ac_1, Aml_2
C_{10}	Bl_1, Ht_2, Amr_2
C_{11}	Hrl_1, Ztm_2, Bl_2
C_{12}	Ht_1, Shl_2, Rtd_2

for each schedule 32 unique scenarios will be constructed (36 minus 4, as Lls_1, Bl_1, Hrl_1 and Ht_1 occur two times as the first case in the instances and so their results will be the same).

Parameter settings

All the settings for CGDDS will be the same as used for the comparison of CGDDS and 2P-RSPPRC, except for two parameters relating to overtime that were not yet implemented. All parameters and their values are represented by Table 6.2.

Table 6.2: Parameter settings for CGDDS

Parameter	Value
Costs for not covering a task starting at location A and ending at location B	20.000
Costs for not covering a task starting at location A and ending at location A	3.000
Costs for not covering a reserve or an education task	250
Costs for modifying a duty	400
Costs for covering a task that is currently assigned to another duty	50
Costs for using a transfer that did not appear in any planned duty	1
Costs for a taxi that takes the driver over the disruption	1.000
Costs for any other taxi that takes the driver to his/her crew base	3.000
Costs per minute overtime	5
Increase in overtime costs after interval	5
Interval after which the overtime costs increase	30 minutes

7 Computational evaluation of the schedules

The schedules will be evaluated based on the same statistics as presented before. Again, all scenarios and instances were solved on a *Intel Xeon X5472* processor with 3.25 GB RAM clocked at 2.99 GHz. All scenarios and instances could be solved within reasonable calculation time. First we consider results for CGDDS and 2P-RSPPRC for the new schedules in combination with the in Chapter 4 introduced cases. Thereafter we will present and discuss the results of CGDDS for the instances, where all schedules are tested for multiple cases per day.

7.1 New schedules

First we consider the results for the in the previous chapter introduced schedules and reserve duties sets. Again all 20 cases are used and a summary of the results is presented in Table 7.1.

First we will consider the results for CGDDS in Table 7.1. By using half of the reserves (R_2) we see the same structure in robustness as when using all reserves. In fact, the costs are just little higher due to a little higher average of uncovered tasks, but the difference is small. When all reserves are located in Utrecht and Amsterdam (R_3), there is still a large improvement compared to the same schedules without reserves. It seems to be not a very good idea, because when half of the reserves are spread across the railway network there are less uncovered tasks when there are two times more reserves located in only the busiest part of the country. For most schedules it is slightly better to locate the reserves at the edges of the Randstad and in the corners of the country (R_4). But also this set of stand-by duties leads to less robust schedules than using the set with only half of the reserve duties (R_2). Dividing the stand-by duties more smoothly over time leads to a slight improvement in robustness compared to using the basic set of reserve duties. However, it depends on the schedule whether it is better to use R_5 or R_6.

Considering the new schedules, SS'$_{20}$ and DC'$_{20}$, it is clear that SS'$_{20}$ is not a more robust schedule than the four schedules we already analyzed. It is even the least robust schedule due to that there are slightly more uncovered tasks. Complete route knowledge cannot make schedules less robust, which must mean that using more relief stations, leading to a schedule with more transfers, makes the process of rescheduling more complex. Contrary, DC'$_{20}$ is a very robust schedule. That is not surprising, as there is much more slack and much more duties in this plan. The high costs of implementing such a plan can be somewhat reduced by cancelling the reserve duties, as the gain of using reserve duties for rescheduling is very small.

For 2P-RSPPRC there is a higher increase in costs when using only half of the reserve duties (R_2) instead of all reserves. Also shifting the reserve duties among crew bases (R_3 and R_4) is not contributing to the robustness of the schedules; the costs are for many schedules higher than when the schedule is combined with R_2. Just like with CGDDS, smoothing improves the robustness of the schedules and it depends on the schedule whether the improvement is made by using R_5 or R_6.

With respect to new schedules, we conclude from Table 7.1 that also when using the 2P-RSPPRC approach for rescheduling SS'$_{20}$ is the least robust schedule. Furthermore, DC'$_{20}$ is a very robust schedule. But even such a robust schedule with reserves is harder to reschedule when using the 2P-RSPPRC approach than when the rescheduling of regular schedules with reserves is done with CGDDS.

Table 7.1: Average performance of each schedule for CGDDS (C) and 2P-RSPPRC (P)

Schedule	AD	UT A-B		UT A-A		Costs		CT	
		C	P	C	P	C	P	C	P
$SS_{20}R_2$	27	0.4	3.75	0.4	1.2	32,232	103,411	0:48	0:45
$SS_{20}R_3$	27	0.5	3.7	0.6	1.4	35,403	104,436	1:03	0:59
$SS_{20}R_4$	27	0.45	3.9	0.3	0.95	33,246	115,385	0:46	1:05
$SS_{20}R_5$	27	0.35	1.55	0.3	0.7	29,988	66,994	0:48	0:35
$SS_{20}R_6$	27	0.35	1.75	0.3	0.7	29,938	70,178	0:49	0:36
$SS_{15}R_2$	26	0.3	4.25	0.6	1.75	30,032	113,618	0:41	0:47
$SS_{15}R_3$	26	0.4	4.15	0.75	1.8	32,810	111,570	0:58	0:59
$SS_{15}R_4$	26	0.4	4.15	0.4	1.6	32,178	120,894	0:45	1:03
$SS_{15}R_5$	26	0.35	1.8	0.3	0.95	29,576	70,889	0:44	0:30
$SS_{15}R_6$	26	0.3	2.15	0.3	1.05	28,618	77,306	0:42	0:33
$SS_{10}R_2$	26	0.25	4.65	0.45	1.8	27,909	120,530	0:44	0:46
$SS_{10}R_3$	26	0.25	3.75	0.6	1.95	28,315	102,705	0:59	1:08
$SS_{10}R_4$	26	0.35	4.2	0.5	1.5	29,981	119,890	0:49	1:09
$SS_{10}R_5$	26	0.3	2.1	0.35	1.15	27,204	76,007	0:46	0:36
$SS_{10}R_6$	26	0.25	1.95	0.3	1.15	26,124	72,420	0:48	0:36
$DC_{20}R_2$	25	0.35	3.35	0.25	1.6	28,647	95,284	0:39	0:45
$DC_{20}R_3$	25	0.5	4.55	0.35	1.25	32,937	119,467	0:55	1:12
$DC_{20}R_4$	25	0.4	4.15	0.25	1.2	30,287	120,470	0:44	1:13
$DC_{20}R_5$	25	0.35	1.7	0.05	0.8	27,397	69,294	0:37	0:31
$DC_{20}R_6$	25	0.35	1.6	0.05	0.95	27,430	67,288	0:39	0:30
$SS'_{20}R_0$	27	1.95	23.3	2.65	3.4	76,736	488,866	2:26	2:41
$SS'_{20}R_1$	27	0.6	3.05	0.1	1.1	34,885	96,549	1:59	1:14
$SS'_{20}R_2$	27	0.55	5.5	0.25	1.6	35,135	139,280	1:48	1:23
$SS'_{20}R_3$	27	0.8	6	0.55	1.35	40,504	148,740	2:37	2:01
$SS'_{20}R_4$	27	0.75	4.75	0.1	1.1	38,622	120,470	1:53	1:13
$SS'_{20}R_5$	27	0.6	3	0.1	0.8	34,312	95,491	1:49	1:07
$SS'_{20}R_6$	27	0.6	2.9	0.2	1	35,070	93,266	1:58	1:11
$DC'_{20}R_0$	20	0.15	2.55	0.4	1.25	17,108	61,674	0:47	0:59
$DC'_{20}R_1$	20	0.15	0.35	0	0.25	14,956	34,882	0:54	0:48
$DC'_{20}R_2$	20	0.15	0.75	0.1	0.35	15,271	34,695	0:51	0:46
$DC'_{20}R_3$	20	0.15	0.85	0.25	0.7	16,136	40,023	0:59	1:04
$DC'_{20}R_4$	20	0.15	0.05	0.1	0.3	15,526	49,901	0:53	1:12
$DC'_{20}R_5$	20	0.15	0.4	0	0.2	15,074	36,165	0:52	0:48
$DC'_{20}R_6$	20	0.15	0.4	0	0.15	14,911	35,644	0:50	0:49

7.2 Multiple cases

For the tests with multiple cases will consider the average performance of the 6 crew schedules combined with the 7 different reserve duty sets. The statistics are represented by Table 7.2. The presented statistics in the table are averages based on the 12 instances. The result of one instance is the sum of the results of all scenarios in that instance (recall Table 6.1). We have to mention that the number of affected duties can differ for each schedule when different reserve duty sets are used. This can be the case when for example for the rescheduling of a scenario, tasks that are affected in the next scenario, are assigned to reserve duties.

Compared to the one-day scenarios, the results for the four initial schedules have not changed. Still, DC_{20} and SS_{10} including reserve duties seem to be robust schedules and SS_{20} with reserves is the least robust schedule of the four initial schedules. Without stand-by duties, SS_{20} and DC_{20} are the most robust plans. The differences are still small, but for schedules based on 'Sharing Sweet & Sour' it seems that also for multiple cases a day a tight schedule combined with reserve duties is favourable. Furthermore, from Table 7.2 it becomes clear that SS'_{20} is the least robust and DC'_{20} is a very robust schedule. The difference in robustness of DC'_{20} compared to the other schedule has become even larger. Still the costs of such a plan can be somewhat reduced as it is not necessary to use reserves.

The effect of using different sets of reserve duties is still small. Positioning reserves at stations outside the Randstad (R_4) seems to be not a very good idea. Also the best option is to smooth the number of active reserve duties by combining either schedules with R_5 or R_6. Finally, also for multiple cases a day the difference between using all reserves (R_1), only half (R_2) or reserves only in Utrecht and Amsterdam (R_3) is remarkably small. This indicates that reserves do not necessarily have to be spread all over the country. It is important to have enough stand-by duties available at the right stations.

Throughout this thesis we mentioned the little differences in robustness between the schedules, especially when advanced methods are used for rescheduling. To find out whether the differences in costs are significant or not, we use One-Way ANOVA. One-Way ANOVA allows us to test whether the mean costs of the schedules are equal or not, where the null hypothesis is that the means are equal. We will perform this test for the schedules with the regular set of reserve duties (R_1) and DC'_{20} will be excluded from the test. At a significance level of 0.05 we do not reject the null hypothesis ($F = 2.078$, $p = 0.096$). This indicates that there is no significant difference between mean total costs for the schedules SS_{20}, SS_{15}, SS_{10}, DC_{20} and SS'_{20}.

However, this ANOVA analysis is based on only 12 observations per schedule, namely the costs for the 12 instances. We can also consider the cases individually, so that we do not have the summed costs to obtain the results for one instance anymore. So in this case we have 36 observations per schedule. Performing again the One-way ANOVA analysis we come to the same conclusion, namely that we do not reject the null hypothesis ($F = 1.799$, $p = 0.131$).

Table 7.2: Average performance of each schedule

Schedule	AD	MD	UR	UT A-B	UT A-A	TT	OD	TO	Costs	CT
$SS_{20}R_0$	80	111	0	1.83	7.17	10	73	2,730	151,377	2:46
$SS_{20}R_1$	80	103	18	1.42	0.92	8	39	1,203	106,366	2:44
$SS_{20}R_2$	79	105	17	1.58	1.5	9	43	1,391	115,203	2:28
$SS_{20}R_3$	80	104	17	1.33	1.75	9	37	1,222	110,035	2:57
$SS_{20}R_4$	80	104	17	1.58	0.92	8	41	1,368	112,573	2:52
$SS_{20}R_5$	80	100	17	1.08	0.92	8	36	1,097	97,616	2:36
$SS_{20}R_6$	80	104	19	1.17	0.92	8	37	1,156	100,941	2:52
$SS_{15}R_0$	76	103	0	2	11.83	10	74	2,804	163,802	2:33
$SS_{15}R_1$	77	100	19	0.67	1	8	36	1,044	89,158	2:27
$SS_{15}R_2$	77	100	17	0.92	1.92	9	42	1,303	97,546	2:17
$SS_{15}R_3$	76	103	20	0.92	1.75	10	36	1,085	99,879	2:59
$SS_{15}R_4$	77	102	18	0.92	1.33	10	40	1,226	98,780	2:36
$SS_{15}R_5$	77	100	20	0.58	1.17	9	33	948	89,534	2:16
$SS_{15}R_6$	76	100	20	0.58	1	9	33	987	87,549	2:38
$SS_{10}R_0$	75	101	0	2.73	11.91	7	83	3,072	175,290	2:41
$SS_{10}R_1$	76	101	20	0.91	1.09	6	37	1,109	89,453	2:30
$SS_{10}R_2$	76	100	18	1.18	1.55	7	40	1,248	98,394	2:15
$SS_{10}R_3$	76	98	17	1	1.45	7	36	1,081	94,628	2:55
$SS_{10}R_4$	76	103	19	1.27	1.55	6	43	1,299	101,566	2:31
$SS_{10}R_5$	76	101	20	1.18	1.09	7	33	950	95,728	2:31
$SS_{10}R_6$	76	100	19	1.18	1	6	34	946	94,155	2:44
$DC_{20}R_0$	73	100	0	2.42	8.58	9	67	2,420	159,544	2:35
$DC_{20}R_1$	73	93	17	0.92	0.33	8	34	918	86,751	2:23
$DC_{20}R_2$	73	94	15	0.92	1	8	39	1,189	91,795	2:17
$DC_{20}R_3$	73	95	15	1.42	1.33	9	34	1,030	104,927	3:10
$DC_{20}R_4$	73	96	17	1.42	0.92	8	37	1,156	103,726	2:39
$DC_{20}R_5$	73	93	17	0.75	0.5	8	30	852	84,663	2:19
$DC_{20}R_6$	73	92	16	1.08	0.25	8	30	884	90,094	2:34
$SS'_{20}R_0$	77	113	0	5.58	12.25	7	79	2,999	249,212	7:18
$SS'_{20}R_1$	78	106	22	2.33	0.67	7	33	1,123	125,919	5:18
$SS'_{20}R_2$	78	106	21	3	1	7	35	1,234	142,243	5:00
$SS'_{20}R_3$	78	102	18	2.92	1.58	7	33	1,123	140,120	7:15
$SS'_{20}R_4$	78	105	21	3.25	0.67	7	35	1,171	145,908	6:08
$SS'_{20}R_5$	78	104	22	2.75	0.5	6	30	1,005	132,591	5:48
$SS'_{20}R_6$	78	105	23	2.58	0.67	6	32	1,064	129,958	6:00
$DC'_{20}R_0$	58	64	0	0.58	1.92	6	21	703	58,589	2:50
$DC'_{20}R_1$	58	65	6	0.67	0	4	12	431	52,229	2:53
$DC'_{20}R_2$	58	64	4	0.67	0.33	4	15	527	53,041	2:42
$DC'_{20}R_3$	58	63	2	0.58	0.75	5	15	500	53,473	3:13
$DC'_{20}R_4$	58	67	5	0.58	0.17	4	19	666	53,204	3:13
$DC'_{20}R_5$	58	65	6	0.58	0	5	11	390	50,518	2:57
$DC'_{20}R_6$	58	65	5	0.58	0	4	12	420	49,654	3:18

8 Conclusions, limitations and directions for further research

In this final section, we will return to our main research question and present our conclusions. Thereafter the limitations of this research are discussed, followed by some suggestions for further research.

8.1 Conclusions

Every day the Dutch railway network incurs several serious disruptions. Due to disruptions the timetable has to be adapted and as a result of that drivers cannot execute their scheduled duties anymore. Therefore, during a disruption the crew schedule has to be modified. Rescheduling NS crew schedules is complex, but on the same time has to be done quickly such that changes can be communicated fast. At NS, the current practice of rescheduling is based on a manual procedure executed by dispatchers. However, practical situations showed that sometimes the problem of rescheduling becomes too complex for dispatchers, leading to even more delays. An extreme situation occurred during the winter of 2009, where due to weather circumstances the entire railway network was affected. The dispatchers could not deal with these problems and were far behind in the rescheduling process, making the situation even worse.

Because rescheduling is so complex, researchers developed several rescheduling methods in order to have quantitative support for taking decisions. At NS mainly two methods are developed. One of them is based on Column Generation, while the other is based on a multi-agent system. Besides the two methods, the way dispatchers do rescheduling is mimicked by a heuristic based on an approach consisting of two phases.

The robustness of a schedule can contribute in making the process of rescheduling more easy. Therefore it is interesting to know for NS which factors determine whether a schedule is robust or not. Robustness of a schedule is often measured in two ways, namely:

- The way a schedule can absorb delays, caused by little disruptions, without necessity of rescheduling (the absorption ability of a schedule).
- The way a schedule can be adapted when rescheduling is necessary (the flexibility of a schedule).

This thesis focused mainly on the last measure. In order to test the flexibility of a schedule several disruption scenarios had to be constructed. The scenarios were arbitrarily selected throughout the whole railway network NS operates upon. First, rescheduling was done with all three methods for several schedules, either with or without reserves. Results show that there are some harder scenarios and less hard scenarios, but overall it is clear that CGDDS is the best method to do rescheduling. For all rescheduling methods, it seems that a schedule with reserve duties based on an older production plan is the easiest to adapt, together with a schedule with reserves based on the current implemented production plan at NS, but with a short transfer time for drivers to go from one train to another. In advance, one would assume that a schedule with more slack, so with longer transfer times, would be more robust than schedules with little slack. That is indeed the case when rescheduling methods cannot use stand-by duties and when the heuristic is used. When reserve duties and a more advanced rescheduling method are used, a schedule with short transfer times is more robust. However, the differences are very small and seem to be not true over all individual scenarios. In fact the differences are most of the times caused by only one or two cases, where for example a schedule with a duty starting at the right time at the right place in order to cover some tasks on that line performs much better than a schedule that has to use duties from elsewhere to cover the same tasks.

The comparison of the methods was based on the assumption there is only one disruption a day. Since that is not the case in reality, we combined some scenarios such that we had instances with on average three scenarios. The only method that is far enough developed in order to deal with multiple scenarios is the method based on Column Generation. It turns out that the method is also performing well when multiple disruptions a day occur.

Furthermore, to the set of schedules two schedules were added, of which one is a schedule that could be implemented during an extreme situation. In this schedule, drivers only can do tasks on small parts of the railway network. Results show that such a schedule is indeed very robust. That is, the schedule can be easily adapted during rescheduling, even with the simple heuristic method. However, such a schedule is very expensive to implement in practice.

Besides new schedules, also some new sets of reserve duties were introduced. We see that some improvements can be made, but the differences are only minor. Adapting schedules with a good rescheduling method like the method based on Column Generation already is not hard when using less reserve duties or even no reserve duties at all.

Overall, we see in all our tests differences between the schedules are small. In fact, when comparing the mean of the total costs of schedules, except the mean costs for the emergency schedule, there is no significant difference. The factor of importance is the reserve duties; schedules become much more flexible when stand-by duties are available. Only an extreme schedule like the emergency schedule can improve robustness significantly. The method that contributes most the robustness is the approach based on Column Generation. It performs much better than the heuristic that mimics the current way of rescheduling done by dispatchers. So quantitative support for decision making is a requirement to achieve more robustness.

8.2 Limitations

All methods used in this thesis have different parameter settings and underlying assumptions. Therefore, comparison of the methods is never completely fair. But even when comparison is not completely fair, we can get an indication of how well each method performs. The goal of this thesis is not to provide a theoretical comparison, but is more focused on how useful the methods are in practice.

Second, the cases chosen for the disruption can be unrealistic. Some are based on cases that occurred in practice, but others are arbitrarily chosen, just as the underlying emergency scenario. It could be that in reality such a scenario would never be used.

The tests in the last part of the thesis are based on the assumption that on average three large disruptions a day occur. In reality, that is indeed the case. But besides large disruption, also smaller disruptions take place every day. This is not incorporated in our tests.

8.3 Directions for further research

Finally, we point out some issues for future research. First, there was not enough time to construct a variant of DC'_{20} that is less expensive. So we also could not test how robust such a schedule is compared to the other schedules. Like said before, DC'_{20} will never be implemented at NS, also not during an extreme situation. Therefore it would be interesting to find out whether there is a similar plan that can be used in practice, without losing too much robustness of the very robust schedule DC'$_{20}$. Of course, such a schedule has to be significantly more robust than the schedules based on 'Sharing Sweet & Sour'.

Second, in this thesis we showed that there is little difference in robustness between the schedules, except when using a more extreme crew schedule. This does not mean that there is no other schedule that is significantly more robust than the basic crew schedule. In future research, also crew schedules with different sign-on and sign-off times could be considered. Increased sign-on and sign-off times will lead to more available stand-by drivers for rescheduling. However, to test such a schedule, CGDDS will have to be modified such that duties of drivers that are, at the moment of rescheduling, in their sign-on or sign-off time are selected in the core problem. Currently, that is not the case for CGDDS.

Other crew schedules that could be interesting to consider are schedules with less relief stations, as we saw that using more relief stations did not lead to a more robust schedule. Furthermore, schedules constructed by using different crew bases and even more or less crew bases could also have an effect on the robustness.

Besides changing only some parameters for crew scheduling, also a completely new scheduling approach could be considered. Literature on crew scheduling in the airline industry introduces some approaches where during the scheduling the robustness of the schedule is taken into account. Examples of these approaches are *bicriteria optimization*, *stochastic programming* and the use of *move-up crews*. However, it is the question whether these approaches are also beneficial for the railways, since the operational costs caused by disruptions is much lower compared to the costs in the airline industry.

Concerning the reserve duties, there are some options not covered in this thesis. In one of the chapters we mentioned that for some scenarios there was specific use of reserve duties that were on the same line as the disruption was. In further research it could be an option to consider this phenomenon in more detail and try to find a set of stand-by duties accounting for this.

Finally, we came to the conclusion that robustness can be improved by having a more smooth distribution of the number of active duties on each moment of the day. However, there was no advanced method behind the smoothening; we just shifted some duties until we found a reasonable distribution without changing the original structure too much. It would be interesting to try to obtain a more smooth distribution, perhaps by creating a Linear Programming model or by creating an algorithm, and to find out whether the resulting set of reserve duties can improve robustness even more.

References

[1] Abbink, E.J.W., Fischetti, M., Kroon, L.G., Timmer, G. and Vromans, M.J.C.M. (2005). Reinventing Crew scheduling at Netherlands Railways. *Interfaces*, 35, (5), 393-401.

[2] Abbink, E.J.W., Mobach, D.G.A., Fiole, P.-J., Kroon, L.G., van der Heijden, E.H.T. and Wijngaards, N.J.E. (2009). Train Driver Rescheduling Applying Actor-Agent Techniques. Technical Report.

[3] Burke, E.K., de Causmaecker, P., de Maere, G., Mulder, J., Pealinck, M., vanden Berghe, G. (2010). A Multi-Objective Approach for Robust Airline Scheduling. *Computers & Operations Research*, 37, (5), 822-832.

[4] Dolan, E.D. and Moré, J.J. (2002). Benchmarking Optimization Software with Performance Profiles. *Mathematical Programming*, 91, (2), 201-213.

[5] Dunbar, M., Froyland, G. and Wu, C.-L. (2010). Robust Airline Schedule Planning: Minimizing Propagated Delay in an Integrated Routing and Crewing Framework. Technical Report.

[6] Eggenberg, N., Salani, M. and Bierlaire, M. (2010). Constraint-Specific Recovery Network for Solving Airline Recovery Problems. *Computers & Operations Research*, 37, (6), 1014-1026.

[7] Eggenberg, N., Salani, M. and Bierlaire, M. (2010). Robust and Recoverable Maintenance Routing Schedules. Technical Report TRANSP-OR 100115. Transport and Mobility Labaratory, ENAC, EPFL.

[8] Ehrgott, M. and Ryan, D. M. (2002). Constructing Robust Crew Schedules with Bicriteria Optimization. *Journal of Multi-criteria Decision Analysis*, 11, 139-150.

[9] Flier, H., Gaurav, A. and Nunkesser, M. (2008). Combinatorial Aspects of Move-up Crews. Technical Report ARRIVAL-TR-0073. ARRIVAL Project.

[10] Huisman, D., Kroon, L.G., Lentink, R.M. and Vromans, M.J.C.M. (2005). Operations Research in Passenger Railway Transportation. *Statistica Neerlandica*, 59, (4), 467-497.

[11] Huisman, D. (2007). A Column Generation Approach to Solve the Crew Re-scheduling Problem. *European Journal of Operational Research*, 180, 163-173.

[12] Jespersen-Groth, J., Potthoff, D., Clausen, J., Huisman, D., Kroon, L.G., Maróti, G. and Nielsen, M.N. (2009). Disruption Management in Passenger Railway Transportation. *Robust and Online Large-Scale Optimization*, 399-421. Springer, New York.

[13] Kroon, L.G. and Fischetti, M. (2001). Crew Scheduling for Netherlands Railways "Destination: Customer". *Computer Aided Scheduling of Public Transport*, 181-201. Springer, Berlin.

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

[15] Lee, L.H., Lee, C.U., Tan, Y.P. (2007). A Multi-Objective Genetic Algorithm for Robust Flight Scheduling Using Simulation. *European Journal of Operational Research*, 177, (3), 1948-1968.

[16] Potthoff, D., Huisman, D. and Desaulniers, D. (2010). Column Generation with Dynamic Duty Selection for Railway Crew Rescheduling. *Transportation Science*, DOI: 10.1287/trsc.1100.0322.

[17] Potthoff, D. (2010). Railway Crew Rescheduling: Novel Approaches and Extensions. PhD Thesis, ERIM.

[18] Shebalov, S. and Klabjan, D. (2006). Robust Airline Crew Pairing: Move-up Crews. *Transportation Science*, 40, (3), 300-312.

[19] Sohoni, M.G., Johnson, E.L. and Bailey, T.G (2006). Operational Airline Reserve Crew Planning. *Journal of Scheduling*, 9, 203-211.

[20] Tan, C. (2007). Het ontwikkelen van een robuustheidmaatstaf voor NS personeelsplannen [Dutch; Developing a robustness measure for NS crew schedules]. Bachelor Thesis, Erasmus University Rotterdam.

[21] de Vries, J. (2010). Bijstuurbaarheid van de generieke personeelsplanning [Dutch; Operatability of the generic crew schedule]. Master Thesis, University of Amsterdam.

[22] Yen, J.W. and Birge, J.R. (2006). A Stochastic Programming Approach to the Airline Crew Scheduling Problem. *Transportation Science*, 40, (1), 3-14.

A Appendix

A.1 CGDDS versus AAATDR

Table A.1.a: Overview of the scenarios tested with CGDDS (C) and AAATDR (A) - No reserves

Scenario	AD	MD		UR		UT		TT		OD		TO		Costs		CT	
		C	A	C	A	C	A	C	A	C	A	C	A	C	A	C	A
$Ac_1SS_{20}R_0$	50	94	-	0	0	3	-	5	-	41	-	1380	-	83585	-	2:14	-
$Ac_1SS_{15}R_0$	46	87	-	0	0	7	-	3	-	52	-	2025	-	162425	-	1:22	-
$Ac_1SS_{10}R_0$	49	100	-	0	0	3	-	2	-	65	-	2509	-	82315	-	1:56	-
$Ac_1DC_{20}R_0$	46	83	-	0	0	4	-	8	-	53	-	2181	-	117560	-	1:38	-
$Ac_2SS_{20}R_0$	56	112	-	0	0	1	-	1	-	21	-	494	-	26005	-	7:34	-
$Ac_2SS_{15}R_0$	48	99	-	0	0	4	-	4		29	-	842	-	95765	-	3:31	-
$Ac_2SS_{10}R_0$	47	104	-	0	0	1	-	2	-	42	-	1112	-	32470	-	4:22	-
$Ac_2DC_{20}R_0$	49	86	-	0	0	1	-	2	-	28	-	707	-	29945	-	2:14	-
$Amr_1SS_{20}R_0$	23	53	-	0	0	4	-	1	-	23	-	892	-	91960	-	1:03	-
$Amr_1SS_{15}R_0$	25	38	-	0	0	7	-	1	-	15	-	600	-	152845	-	0:31	-
$Amr_1SS_{10}R_0$	24	50	-	0	0	7	-	3	-	31	-	1158	-	159550	-	0:50	-
$Amr_1DC_{20}R_0$	27	47	-	0	0	5	-	2	-	25	-	903	-	114985	-	1:06	-
$Amr_2SS_{20}R_0$	18	43	-	0	0	7	-	2	-	20	-	650	-	150100	-	0:13	-
$Amr_2SS_{15}R_0$	19	31	-	0	0	9	-	4	-	16	-	549	-	195420	-	0:17	-
$Amr_2SS_{10}R_0$	22	44	-	0	0	5	-	3	-	15	-	399	-	111240	-	0:14	-
$Amr_2DC_{20}R_0$	19	30	-	0	0	9	-	5	-	12	-	347	-	197035	-	0:15	-
$Aml_1SS_{20}R_0$	23	36	59	0	0	2	0	5	0	15	1	588	1	57920	5	0:27	5:24
$Aml_1SS_{15}R_0$	23	38	76	0	0	3	1	7	0	21	5	726	416	84805	25320	0:14	11:09
$Aml_1SS_{10}R_0$	21	29	58	0	0	3	0	3	0	12	3	434	274	71860	3770	0:15	6:39
$Aml_1DC_{20}R_0$	23	47	51	0	0	3	0	5	0	16	2	481	220	77230	2600	0:22	4:07
$Aml_2SS_{20}R_0$	25	29	70	0	0	8	1	7	1	8	1	245	95	181985	24000	0:46	5:12
$Aml_2SS_{15}R_0$	23	42	60	0	0	1	1	5	2	14	1	516	1	37905	26005	1:35	4:54
$Aml_2SS_{10}R_0$	21	28	66	0	0	2	0	4	0	12	2	382	193	54370	2095	0:44	5:56
$Aml_2DC_{20}R_0$	26	35	65	0	0	1	0	5	0	7	1	346	2	36560	10	0:38	5:17
$Bl_1SS_{20}R_0$	14	19	25	0	0	1	1	2	1	3	2	112	462	26105	35555	0:07	3:52
$Bl_1SS_{15}R_0$	15	19	28	0	0	1	0	2	2	4	4	199	774	26665	22805	0:04	1:58
$Bl_1SS_{10}R_0$	13	31	28	0	0	1	0	0	2	9	4	237	457	21445	14075	0:10	6:43
$Bl_1DC_{20}R_0$	15	26	31	0	0	2	0	4	2	7	2	237	578	52880	22220	0:09	2:49

$Bl_2SS_{20}R_0$	12	18	21	0	0	2	1	3	0	7	1	200	71	49440	20615	0:05	2:38
$Bl_2SS_{15}R_0$	13	23	20	0	0	1	0	1	2	3	4	70	493	23350	14620	0:07	1:02
$Bl_2SS_{10}R_0$	12	23	28	0	0	0	0	0	0	7	2	161	219	955	2970	0:06	3:40
$Bl_2DC_{20}R_0$	14	32	25	0	0	1	0	2	0	8	2	132	120	26720	900	0:09	3:06
$Hrl_1SS_{20}R_0$	11	13	21	0	0	0	0	1	0	2	1	32	150	3160	2250	0:02	1:24
$Hrl_1SS_{15}R_0$	12	14	28	0	0	0	0	0	0	1	2	41	228	260	3240	0:03	2:07
$Hrl_1SS_{10}R_0$	11	12	21	0	0	0	0	0	0	2	1	31	150	155	2250	0:02	0:56
$Hrl_1DC_{20}R_0$	9	11	20	0	0	0	0	1	0	1	1	19	161	3095	2580	0:01	1:30
$Hrl_2SS_{20}R_0$	14	18	30	0	0	0	0	1	0	3	2	97	215	3535	3350	0:02	2:56
$Hrl_2SS_{15}R_0$	12	18	32	0	0	0	0	1	0	3	3	51	144	3255	995	0:04	3:29
$Hrl_2SS_{10}R_0$	14	16	32	0	0	0	0	1	0	3	2	40	202	3200	2780	0:03	3:05
$Hrl_2DC_{20}R_0$	13	15	30	0	0	0	0	1	0	3	2	116	270	3310	4910	0:02	4:55
$Ht_1SS_{20}R_0$	43	71	-	0	0	1	-	7	-	26	-	781	-	45880	-	1:26	-
$Ht_1SS_{15}R_0$	43	64	-	0	0	0	-	5	-	23	-	750	-	19770	-	1:31	-
$Ht_1SS_{10}R_0$	42	71	-	0	0	0	-	7	-	33	-	1001	-	27330	-	1:23	-
$Ht_1DC_{20}R_0$	37	64	-	0	0	1	-	4	-	13	-	385	-	33945	-	1:00	-
$Ht_2SS_{20}R_0$	43	73	-	0	0	2	-	6	-	13	-	284	-	59510	-	3:29	-
$Ht_2SS_{15}R_0$	46	70	-	0	0	5	-	17	-	22	-	555	-	152675	-	3:20	-
$Ht_2SS_{10}R_0$	45	73	-	0	0	0	-	7	-	22	-	741	-	25545	-	4:00	-
$Ht_2DC_{20}R_0$	42	68	-	0	0	1	-	5	-	17	-	477	-	37880	-	2:43	-
$Lls_1SS_{20}R_0$	11	24	51	0	0	3	4	1	9	6	14	164	1900	66895	141310	0:15	10:39
$Lls_1SS_{15}R_0$	10	14	30	0	0	3	9	4	8	3	7	89	864	72450	231705	0:11	12:03
$Lls_1SS_{10}R_0$	11	16	26	0	0	4	7	5	4	4	4	92	450	95520	163590	0:18	3:42
$Lls_1DC_{20}R_0$	10	27	39	0	0	3	7	2	8	4	7	108	780	69655	217325	0:30	12:21
$Lls_2SS_{20}R_0$	13	26	49	0	0	5	7	6	10	13	6	284	729	129680	181650	0:09	7:13
$Lls_2SS_{15}R_0$	12	23	57	0	0	3	4	5	10	10	10	289	1343	76700	139090	0:11	7:06
$Lls_2SS_{10}R_0$	11	24	54	0	0	7	2	8	7	12	15	397	1252	166315	73125	0:14	11:14
$Lls_2DC_{20}R_0$	12	31	41	0	0	5	3	5	10	14	16	526	1424	121240	112760	0:23	6:00
$Rtd_1SS_{20}R_0$	36	45	-	0	0	2	-	2	-	19	-	714	-	50340	-	1:47	-
$Rtd_1SS_{15}R_0$	31	47	-	0	0	1	-	1	-	27	-	958	-	29305	-	3:26	-
$Rtd_1SS_{10}R_0$	30	53	-	0	0	3	-	1	-	30	-	1303	-	71780	-	2:21	-
$Rtd_1DC_{20}R_0$	31	45	-	0	0	6	-	3	-	29	-	922	-	133860	-	2:25	-

Rtd$_2$SS$_{20}$R$_0$	39	59	-	0	0	0	-	1	-	23	-	651	-	6970	-	3:46	-
Rtd$_2$SS$_{15}$R$_0$	35	59	-	0	0	1	-	0	-	19	-	556	-	23570	-	3:32	-
Rtd$_2$SS$_{10}$R$_0$	35	52	-	0	0	0	-	0	-	22	-	599	-	3825	-	3:24	-
Rtd$_2$DC$_{20}$R$_0$	32	53		0	0	1	-	1	-	23	-	677	-	26780	-	3:17	-
Shl$_1$SS$_{20}$R$_0$	28	41	-	0	0	2	-	4	-	21	-	827	-	60350	-	0:08	-
Shl$_1$SS$_{15}$R$_0$	27	42	-	0	0	6	-	2	-	18	-	666	-	130390	-	0:24	-
Shl$_1$SS$_{10}$R$_0$	31	39	-	0	0	11	-	6	-	16	-	582	-	241625	-	0:24	-
Shl$_1$DC$_{20}$R$_0$	27	39	-	0	0	3	-	3	-	14	-	421	-	71700	-	0:19	-
Shl$_2$SS$_{20}$R$_0$	30	71	-	0	0	6	-	4	-	21	-	651	-	135875	-	0:50	-
Shl$_2$SS$_{15}$R$_0$	29	45	-	0	0	10	-	9	-	11	-	385	-	229140	-	0:38	-
Shl$_2$SS$_{10}$R$_0$	31	41	-	0	0	11	-	7	-	24	-	667	-	245210	-	0:39	-
Shl$_2$DC$_{20}$R$_0$	26	42	-	0	0	3	-	1	-	16	-	554	-	66565	-	0:35	-
Ztm$_1$SS$_{20}$R$_0$	23	50	33	0	0	0	0	1	0	13	0	283	0	4660	0	0:40	2:00
Ztm$_1$SS$_{15}$R$_0$	22	38	35	0	0	2	0	1	0	5	1	134	1	43810	5	0:28	4:56
Ztm$_1$SS$_{10}$R$_0$	24	42	44	0	0	0	0	0	0	4	0	121	0	755	0	1:04	5:23
Ztm$_1$DC$_{20}$R$_0$	17	42	33	0	0	0	0	0	0	2	0	55	0	315	0	1:05	2:56
Ztm$_2$SS$_{20}$R$_0$	24	32	45	0	0	1	0	2	0	8	0	165	0	26815	0	0:14	4:11
Ztm$_2$SS$_{15}$R$_0$	24	38	44	0	0	2	0	2	0	5	0	109	0	46680	0	0:27	3:21
Ztm$_2$SS$_{10}$R$_0$	23	41	42	0	0	1	0	0	0	22	1	699	3	24415	15	0:20	2:50
Ztm$_2$DC$_{20}$R$_0$	19	34	31	0	0	0	0	1	0	5	0	161	0	3700	0	0:28	2:19

Table A.1.b: Overview of the scenarios tested with CGDDS (C) and AAATDR (A) - Reserves

Scenario	AD	MD		UR		UT		TT		OD		TO		Costs		CT	
		C	A	C	A	C	A	C	A	C	A	C	A	C	A	C	A
Ac$_1$SS$_{20}$R$_1$	50	131	95	40	18	1	0	4	8	11	2	400	69	33900	24505	3:07	9:05
Ac$_1$SS$_{15}$R$_1$	46	132	82	48	18	2	0	2	7	13	2	378	125	48230	22005	3:05	6:13
Ac$_1$SS$_{10}$R$_1$	49	128	87	49	18	0	0	2	3	17	5	489	129	8815	10385	2:39	7:41
Ac$_1$DC$_{20}$R$_1$	46	112	90	39	22	1	0	5	7	11	7	387	447	36435	26435	1:49	10:55
Ac$_2$SS$_{20}$R$_1$	56	139	99	40	19	0	0	1	10	4	5	134	196	3875	37820	11:17	7:48
Ac$_2$SS$_{15}$R$_1$	48	138	98	44	20	2	2	4	13	8	6	353	321	52840	85360	6:47	8:03

$Ac_2SS_{10}R_1$	47	117	94	40	22	0	0	2	10	3	4	94	289	6380	36245	12:18	7:34
$Ac_2DC_{20}R_1$	49	137	77	42	17	1	0	2	14	5	8	116	507	26470	47025	4:35	7:23
$Amr_1SS_{20}R_1$	23	88	115	41	23	1	1	2	17	0	6	0	414	26000	78460	1:17	13:02
$Amr_1SS_{15}R_1$	25	60	90	25	20	1	0	2	19	1	4	41	551	26260	71155	0:46	8:14
$Amr_1SS_{10}R_1$	24	81	85	36	20	1	0	3	19	2	5	6	646	29030	69015	1:25	7:30
$Amr_1DC_{20}R_1$	27	74	90	32	21	1	0	1	19	2	9	20	790	26010	70510	1:11	8:01
$Amr_2SS_{20}R_1$	18	61	69	28	12	0	4	4	17	8	11	290	840	33865	138895	0:17	5:33
$Amr_2SS_{15}R_1$	19	53	74	21	14	0	8	6	16	6	13	146	969	18880	229005	0:14	5:52
$Amr_2SS_{10}R_1$	22	75	67	29	13	1	6	1	18	6	10	181	541	24085	161845	0:49	4:59
$Amr_2DC_{20}R_1$	19	52	65	19	13	1	7	7	16	6	11	220	643	42040	194625	0:14	6:10
$Aml_1SS_{20}R_1$	23	56	55	27	7	0	0	4	0	3	0	202	0	12000	0	0:24	3:14
$Aml_1SS_{15}R_1$	23	66	56	32	10	0	0	3	0	5	1	189	71	9285	615	0:23	3:32
$Aml_1SS_{10}R_1$	21	56	58	22	7	1	0	3	0	5	1	198	97	30290	1040	0:27	3:48
$Aml_1DC_{20}R_1$	23	62	53	28	5	1	0	5	0	3	0	138	0	35000	0	0:27	2:43
$Aml_2SS_{20}R_1$	25	72	63	31	12	4	0	4	2	4	0	97	0	92025	6000	1:40	3:49
$Aml_2SS_{15}R_1$	23	62	66	27	13	0	0	3	0	3	0	167	0	9750	0	1:15	3:00
$Aml_2SS_{10}R_1$	21	51	49	24	11	1	0	1	0	0	0	0	0	23000	0	0:59	2:40
$Aml_2DC_{20}R_1$	26	63	66	24	14	0	0	2	0	3	0	220	0	6750	0	1:54	3:04
$Bl_1SS_{20}R_1$	14	45	21	23	5	0	0	1	0	1	1	91	202	3920	6255	0:13	0:49
$Bl_1SS_{15}R_1$	15	45	20	22	5	0	0	1	0	1	1	91	202	3920	6255	0:09	0:54
$Bl_1SS_{10}R_1$	13	49	19	24	4	0	0	0	0	1	0	39	0	240	0	0:41	1:14
$Bl_1DC_{20}R_1$	15	39	20	19	3	0	0	2	1	3	2	131	179	6045	5190	0:11	1:10
$Bl_2SS_{20}R_1$	12	42	20	20	4	0	0	3	1	2	1	122	71	9940	3615	0:08	0:46
$Bl_2SS_{15}R_1$	13	46	21	24	4	0	0	1	0	0	1	0	80	3000	750	0:14	0:57
$Bl_2SS_{10}R_1$	12	42	21	17	5	0	0	0	0	0	1	0	159	0	2520	0:15	1:10
$Bl_2DC_{20}R_1$	14	49	22	25	5	0	0	1	0	0	0	0	0	3000	0	0:14	1:03
$Hrl_1SS_{20}R_1$	11	21	24	9	4	0	0	0	0	1	0	30	0	150	0	0:04	1:05
$Hrl_1SS_{15}R_1$	12	29	26	13	5	0	0	0	0	1	0	41	0	260	0	0:09	1:18
$Hrl_1SS_{10}R_1$	11	25	24	13	5	0	0	0	0	0	0	0	0	0	0	0:06	0:33
$Hrl_1DC_{20}R_1$	9	20	23	10	4	0	0	0	0	0	0	0	0	0	0	0:03	0:56
$Hrl_2SS_{20}R_1$	14	29	32	11	5	0	0	1	0	1	0	38	0	3230	0	0:07	1:13
$Hrl_2SS_{15}R_1$	12	30	28	14	3	0	0	1	0	0	0	0	0	3000	0	0:07	1:05

$Hrl_2SS_{10}R_1$	14	29	32	13	4	0	0	1	0	1	0	7	0	3035	0	0:07	1:12
$Hrl_2DC_{20}R_1$	13	28	30	13	5	0	0	1	0	2	0	73	0	3040	0	0:05	1:19
$Ht_1SS_{20}R_1$	43	110	70	40	19	0	0	5	0	5	3	164	277	15940	6335	1:40	4:44
$Ht_1SS_{15}R_1$	43	120	74	44	19	0	0	5	1	6	1	196	112	15835	4340	2:41	5:13
$Ht_1SS_{10}R_1$	42	110	71	42	19	0	0	4	1	11	3	414	233	14815	5435	1:53	5:23
$Ht_1DC_{20}R_1$	37	102	68	37	18	0	0	4	3	6	2	297	154	13120	11270	2:37	4:26
$Ht_2SS_{20}R_1$	43	111	76	37	18	1	0	3	0	5	1	75	28	26420	140	5:22	4:37
$Ht_2SS_{15}R_1$	46	120	72	42	19	1	0	11	1	9	0	311	0	53105	3000	8:39	4:55
$Ht_2SS_{10}R_1$	45	111	71	41	16	0	0	4	1	5	0	182	0	12810	3000	8:02	5:09
$Ht_2DC_{20}R_1$	42	101	75	35	18	0	0	3	2	5	2	129	131	9410	7065	6:03	4:25
$Lls_1SS_{20}R_1$	11	30	26	12	6	1	7	2	9	0	2	0	287	26000	162285	1:13	4:36
$Lls_1SS_{15}R_1$	10	47	24	19	5	0	3	2	6	0	4	0	252	6000	86440	0:40	3:53
$Lls_1SS_{10}R_1$	11	32	27	14	8	1	4	3	2	0	4	0	399	29000	90190	5:23	3:24
$Lls_1DC_{20}R_1$	10	39	27	10	5	0	6	2	7	0	4	0	218	6000	144015	2:13	4:38
$Lls_2SS_{20}R_1$	13	40	47	13	11	2	3	6	13	7	5	240	363	59280	102840	0:25	6:39
$Lls_2SS_{15}R_1$	12	43	37	19	7	1	3	5	6	7	3	260	272	36370	81680	1:18	2:44
$Lls_2SS_{10}R_1$	11	38	41	12	9	1	0	6	5	10	5	234	296	39220	57755	0:23	3:56
$Lls_2DC_{20}R_1$	12	66	34	26	8	1	0	4	6	6	3	92	218	32270	19945	0:37	3:05
$Rtd_1SS_{20}R_1$	36	87	66	31	11	0	0	2	0	6	1	293	4	7545	20	3:47	5:42
$Rtd_1SS_{15}R_1$	31	75	65	28	11	0	0	0	0	3	0	150	0	1050	0	3:22	4:42
$Rtd_1SS_{10}R_1$	30	66	58	26	12	0	0	1	0	7	1	213	4	4480	20	2:10	5:48
$Rtd_1DC_{20}R_1$	31	72	58	28	11	1	0	3	0	7	0	267	0	29785	0	3:25	4:23
$Rtd_2SS_{20}R_1$	39	79	65	28	13	0	0	1	0	7	0	286	0	4925	0	6:56	5:07
$Rtd_2SS_{15}R_1$	35	78	67	26	11	0	1	0	1	4	0	110	0	665	23000	4:51	4:27
$Rtd_2SS_{10}R_1$	35	92	53	33	12	0	0	0	0	2	0	35	0	175	0	7:36	4:07
$Rtd_2DC_{20}R_1$	32	70	59	28	14	0	0	1	0	3	0	88	0	3275	0	4:57	4:21
$Shl_1SS_{20}R_1$	28	63	87	27	17	0	0	3	12	8	4	277	365	10675	42925	1:51	12:54
$Shl_1SS_{15}R_1$	27	64	84	24	16	0	0	2	13	3	4	138	132	6890	39930	0:23	11:37
$Shl_1SS_{10}R_1$	31	80	91	30	20	1	1	4	21	6	7	153	727	32715	97265	0:34	12:54
$Shl_1DC_{20}R_1$	27	54	90	19	17	0	0	3	15	3	2	72	79	9495	45575	0:19	9:12
$Shl_2SS_{20}R_1$	30	76	91	26	16	0	1	3	23	4	2	126	233	9715	91725	0:45	18:18
$Shl_2SS_{15}R_1$	29	65	104	27	20	2	3	5	16	3	3	81	403	55120	124555	0:50	16:52

$Shl_2SS_{10}R_1$	31	72	103	26	19	3	3	4	25	6	13	170	1020	73075	146360	1:11	13:55
$Shl_2DC_{20}R_1$	26	55	87	21	17	2	0	0	23	2	6	44	585	40220	75070	1:20	11:47
$Ztm_1SS_{20}R_1$	23	89	31	40	3	0	0	1	0	1	0	27	0	3135	0	1:54	1:07
$Ztm_1SS_{15}R_1$	22	76	36	33	6	1	0	1	0	1	0	29	0	23145	0	1:42	1:43
$Ztm_1SS_{10}R_1$	24	80	41	36	7	0	0	0	0	1	0	60	0	450	0	2:24	1:16
$Ztm_1DC_{20}R_1$	17	87	26	37	3	0	0	0	0	0	0	0	0	0	0	1:57	0:54
$Ztm_2SS_{20}R_1$	24	75	35	34	5	1	0	2	0	4	0	94	0	26460	0	0:43	1:38
$Ztm_2SS_{15}R_1$	24	87	43	39	9	2	0	1	0	1	2	16	4	43080	20	1:23	3:33
$Ztm_2SS_{10}R_1$	23	77	41	35	6	0	0	0	0	1	0	60	0	450	0	0:54	3:04
$Ztm_2DC_{20}R_1$	19	77	31	40	7	0	0	1	0	4	0	141	0	3600	0	0:42	1:13

A.2 CGDDS versus 2P-RSPPRC

Table A.2.a: Overview of the scenarios tested with CGDDS (C) and 2P-RSPPRC (P) - No reserves

Scenario	AD	UT A-B		UT A-A		Costs		CT	
		C	P	C	P	C	P	C	P
ss20ac1	50	0	28	3	7	63090	602349	1:36	2:10
ss15ac1	46	1	38	9	12	96776	811438	1:21	3:05
ss10ac1	49	1	34	3	10	75352	723539	1:38	2:37
dcbk20ac1	46	0	30	4	14	62560	663199	1:05	2:34
ss20ac2	56	1	25	0	6	63385	535714	3:51	2:30
ss15ac2	48	2	27	3	8	102534	590125	3:03	2:19
ss10ac2	47	1	28	0	12	63802	614556	2:58	2:11
dcbk20ac2	49	1	33	0	10	63910	711590	1:43	2:21
ss20amr1	23	1	15	7	6	72108	326731	0:42	1:02
ss15amr1	25	1	22	7	5	69245	463791	0:31	1:54
ss10amr1	24	1	20	8	13	81579	451436	0:52	1:38
dcbk20amr1	27	1	24	6	9	74338	518760	1:03	2:31
ss20amr2	18	0	11	7	4	52468	240246	0:24	0:17
ss15amr2	19	1	14	11	6	75114	307505	0:13	0:24
ss10amr2	22	0	11	7	7	41469	254154	0:12	0:16
dcbk20amr2	19	2	15	10	6	100949	330923	0:18	0:36
ss20aml1	23	0	15	1	2	30841	320110	0:13	1:04
ss15aml1	23	0	13	3	5	39689	288359	0:13	0:59
ss10aml1	21	1	9	2	5	43821	204235	0:09	0:49
dcbk20aml1	23	2	12	3	5	73624	272792	0:14	1:16
ss20aml2	25	4	18	6	5	131635	400230	0:46	1:52
ss15aml2	23	0	14	2	7	30170	316176	0:35	1:46
ss10aml2	21	0	15	3	4	30575	321490	0:40	1:02
dcbk20aml2	26	0	19	2	4	29232	411525	0:36	1:25
ss20bl1	14	0	5	1	0	13538	105098	0:04	0:14
ss15bl1	15	0	7	2	4	17842	156544	0:06	0:18
ss10bl1	13	0	7	1	2	17067	149600	0:06	0:26
dcbk20bl1	15	0	7	2	4	19096	157504	0:06	0:40
ss20bl2	12	0	3	2	4	17241	74673	0:04	0:09
ss15bl2	13	0	7	2	2	15640	148947	0:08	0:23
ss10bl2	12	0	2	0	2	10546	48076	0:04	0:09
dcbk20bl2	14	0	6	1	3	17053	134144	0:06	0:15
ss20hrl1	11	0	2	0	2	6768	49389	0:01	0:34
ss15hrl1	12	0	5	0	1	6471	104377	0:03	0:47
ss10hrl1	11	0	2	0	1	5870	43671	0:01	0:23
dcbk20hrl1	9	0	4	0	2	6717	87924	0:01	0:33
ss20hrl2	14	0	7	0	0	12882	144130	0:04	0:23
ss15hrl2	12	0	7	0	1	9178	149520	0:03	0:32
ss10hrl2	14	0	8	0	1	9426	166939	0:03	0:41
dcbk20hrl2	13	0	4	0	0	11272	87173	0:02	0:23

ss20ht1	43	0	13	1	7	41746	300859	1:01	1:50
ss15ht1	43	0	18	0	9	33356	407100	0:54	1:21
ss10ht1	42	0	20	0	9	39161	443189	1:09	1:50
dcbk20ht1	37	0	10	1	12	32499	253381	0:42	1:25
ss20ht2	43	0	13	2	9	42835	311135	1:56	1:22
ss15ht2	46	0	19	8	12	75456	443436	2:41	2:48
ss10ht2	45	0	16	1	13	49058	384137	3:22	2:00
dcbk20ht2	42	0	20	3	2	47292	424188	2:16	1:49
ss20lls1	11	2	12	1	0	59544	247131	0:16	0:30
ss15lls1	10	2	9	2	1	59072	190130	0:16	0:16
ss10lls1	11	2	11	3	3	63943	233281	0:17	0:41
dcbk20lls1	10	3	12	0	0	72677	245834	0:33	0:42
ss20lls2	13	3	9	2	0	87043	194600	0:08	0:41
ss15lls2	12	2	13	4	0	72402	271332	0:14	0:38
ss10lls2	11	3	15	8	2	109119	317995	0:25	0:36
dcbk20lls2	12	2	9	3	0	74715	191289	0:19	0:38
ss20rtd1	36	2	18	2	4	76751	389185	1:17	1:51
ss15rtd1	31	0	30	1	6	28635	628637	2:27	2:47
ss10rtd1	30	0	27	5	2	48270	554725	2:14	2:10
dcbk20rtd1	31	3	29	8	7	120204	620230	1:49	3:25
ss20rtd2	39	0	25	0	2	30115	518958	2:33	2:13
ss15rtd2	35	0	23	2	5	35378	488004	1:51	1:56
ss10rtd2	35	0	35	0	4	29161	722247	2:50	3:14
dcbk20rtd2	32	1	31	1	1	55416	636355	3:52	4:03
ss20shl1	28	0	9	3	3	36517	204321	0:08	0:38
ss15shl1	27	0	15	10	3	62296	317168	0:25	0:48
ss10shl1	31	3	20	12	3	130343	425726	0:26	0:53
dcbk20shl1	27	0	13	4	2	42750	275401	0:19	0:54
ss20shl2	30	1	18	6	5	76768	394449	0:25	1:38
ss15shl2	29	3	26	17	2	165442	543435	1:13	1:13
ss10shl2	31	2	28	13	5	118960	595845	0:53	2:09
dcbk20shl2	26	1	26	3	4	51177	540384	0:32	2:15
ss20ztm1	23	0	11	0	1	20828	231981	0:21	1:20
ss15ztm1	22	0	14	1	2	19968	292755	0:22	1:05
ss10ztm1	24	0	8	0	5	13698	180112	0:22	0:46
dcbk20ztm1	17	0	5	0	2	9833	109558	0:22	1:12
ss20ztm2	24	0	5	1	3	20243	119054	0:07	0:36
ss15ztm2	24	1	11	1	3	43658	240961	0:20	0:46
ss10ztm2	23	0	16	2	3	25572	332202	0:20	1:03
dcbk20ztm2	19	0	4	0	1	12780	87638	0:13	0:32

Table A.2.b: Overview of the scenarios tested with CGDDS (C) and 2P-RSPPRC (P) - Reserves

Scenario	AD	UT A-B		UT A-A		Costs		CT	
		C	P	C	P	C	P	C	P
ss20ac1	50	0	7	1	1	45659	183665	1:15	1:51

ss15ac1	46	0	9	1	3	41185	222431	1:01	1:47
ss10ac1	49	0	7	0	1	36666	174381	1:38	1:55
dcbk20ac1	46	0	2	0	4	40907	95170	1:02	1:02
ss20ac2	56	0	1	0	0	38397	47236	4:45	1:24
ss15ac2	48	1	4	2	1	68659	117486	2:43	1:15
ss10ac2	47	0	3	0	1	35068	91804	3:17	1:10
dcbk20ac2	49	1	3	0	0	56315	90000	1:31	0:51
ss20amr1	23	1	2	0	1	39059	74722	0:51	0:25
ss15amr1	25	1	2	0	2	40717	81026	0:18	0:32
ss10amr1	24	1	1	0	3	41512	63719	0:34	0:25
dcbk20amr1	27	1	2	0	4	42871	91159	0:59	0:30
ss20amr2	18	0	2	2	4	26952	71409	0:09	0:09
ss15amr2	19	0	0	0	5	23454	36116	0:08	0:11
ss10amr2	22	0	5	2	4	24418	136073	0:22	0:14
dcbk20amr2	19	1	1	1	5	47504	61182	0:13	0:11
ss20aml1	23	0	1	0	1	20653	63027	0:12	0:28
ss15aml1	23	0	2	0	0	21555	73925	0:10	0:20
ss10aml1	21	1	3	0	1	37861	97715	0:17	0:31
dcbk20aml1	23	1	1	0	2	43710	67640	0:13	0:16
ss20aml2	25	3	5	1	1	91128	142789	1:06	0:38
ss15aml2	23	0	2	0	0	19857	73094	0:25	0:30
ss10aml2	21	0	2	1	1	17702	66965	0:34	0:17
dcbk20aml2	26	0	1	0	1	21019	45850	0:31	0:18
ss20bl1	14	0	0	0	0	9280	29364	0:06	0:02
ss15bl1	15	0	1	0	1	9936	49665	0:04	0:03
ss10bl1	13	0	1	0	0	8520	45062	0:06	0:06
dcbk20bl1	15	0	0	0	0	11184	30365	0:05	0:02
ss20bl2	12	0	1	0	0	9727	36642	0:04	0:02
ss15bl2	13	0	1	0	0	8530	34953	0:04	0:02
ss10bl2	12	0	0	0	0	6826	13746	0:05	0:07
dcbk20bl2	14	0	1	0	0	9227	39658	0:04	0:02
ss20hrl1	11	0	0	0	0	6116	31487	0:03	0:44
ss15hrl1	12	0	0	0	0	6373	30844	0:08	0:09
ss10hrl1	11	0	0	0	0	5617	26635	0:02	0:15
dcbk20hrl1	9	0	0	0	0	5262	31285	0:02	0:34
ss20hrl2	14	0	0	0	0	10622	19326	0:03	0:06
ss15hrl2	12	0	0	0	0	8022	21131	0:04	0:03
ss10hrl2	14	0	0	0	0	8824	23003	0:06	0:20
dcbk20hrl2	13	0	0	0	0	10822	23680	0:03	0:03
ss20ht1	43	0	2	0	2	32606	90796	0:49	1:25
ss15ht1	43	0	1	0	1	31713	67750	1:25	0:49
ss10ht1	42	0	0	0	2	32159	45692	1:02	0:43
dcbk20ht1	37	0	1	0	1	27398	66030	0:58	0:37
ss20ht2	43	0	0	1	1	32307	40865	2:26	0:44
ss15ht2	46	0	2	1	5	47416	95410	2:46	1:01
ss10ht2	45	0	0	0	2	33461	48075	3:08	0:50
dcbk20ht2	42	0	1	0	1	31196	54207	2:01	0:38

ss20lls1	11	1	3	0	0	30175	97688	0:23	0:30
ss15lls1	10	0	4	0	0	13684	113398	0:13	0:19
ss10lls1	11	1	6	0	0	30831	151251	0:19	0:44
dcbk20lls1	10	0	3	0	0	11530	97099	0:18	0:34
ss20lls2	13	2	4	0	0	61438	121793	0:11	0:32
ss15lls2	12	1	4	0	0	38647	116042	0:13	0:34
ss10lls2	11	1	6	0	0	42300	159159	0:12	0:20
dcbk20lls2	12	1	4	0	0	39390	121002	0:15	0:36
ss20rtd1	36	0	4	0	0	27183	121913	1:35	1:09
ss15rtd1	31	0	3	0	1	21475	96923	2:01	1:19
ss10rtd1	30	0	1	0	1	21080	55471	1:27	1:11
dcbk20rtd1	31	1	8	0	0	49619	203646	2:00	1:33
ss20rtd2	39	0	4	0	1	25126	107833	2:36	0:57
ss15rtd2	35	0	2	0	0	22442	66731	2:26	0:51
ss10rtd2	35	0	4	0	1	22787	109231	3:21	0:58
dcbk20rtd2	32	0	1	0	0	24929	50487	2:53	0:47
ss20shl1	28	0	2	0	1	25102	80719	0:08	0:14
ss15shl1	27	0	2	0	0	22519	72151	0:11	0:32
ss10shl1	31	1	3	1	1	52526	102207	0:19	0:34
dcbk20shl1	27	0	2	0	0	21952	75474	0:08	0:56
ss20shl2	30	0	3	0	1	32031	100161	0:16	0:42
ss15shl2	29	2	7	0	2	70623	181463	0:27	0:44
ss10shl2	31	1	4	2	2	59779	125167	0:32	0:49
dcbk20shl2	26	1	2	0	2	36304	76803	0:42	0:30
ss20ztm1	23	0	0	0	0	15953	29097	0:29	0:35
ss15ztm1	22	0	1	1	0	17388	47682	0:16	0:12
ss10ztm1	24	0	2	0	0	12947	64792	0:37	0:29
dcbk20ztm1	17	0	0	0	0	8976	23368	0:26	0:25
ss20ztm2	24	0	0	1	1	20243	33258	0:13	0:13
ss15ztm2	24	1	1	1	1	41942	53627	0:20	0:17
ss10ztm2	23	0	0	0	0	13091	21329	0:18	0:20
dcbk20ztm2	19	0	0	0	0	12130	24209	0:10	0:06