Column generation with dynamic duty selection for railway crew rescheduling

Daniel Potthoff[1*], Dennis Huisman[1,2], Guy Desaulniers[3]

[1] Econometric Institute and ECOPT,
Erasmus University Rotterdam, P.O. Box 1738, NL-3000 DR Rotterdam,
The Netherlands
[2] Department of Logistics, Netherlands Railways,
P.O. Box 2025, NL-3500 HA Utrecht,
The Netherlands
[3] GERAD and École Polytechnique de Montréal
C.P. 6079, Succursale Centre-ville
Montréal (Québec) Canada, H3C 3AT

Econometric Institute Report EI 2008-28

December 19, 2008

Abstract

The Dutch railway network experiences about three large disruptions per day on average. In this paper, we present an algorithm to reschedule the crews when such a disruption occurs. The algorithm is based on column generation techniques combined with Lagrangian heuristics. Since the number of duties is very large in practical instances, we first define a core problem of tractable size. If some tasks remain uncovered in the solution of the core problem, we perform a neighborhood exploration to improve the solution. Computational experiments with real-life instances show that our method is capable of producing good solutions within a couple of minutes of computation time.

1 Introduction

During the last decades, Operations Research (OR) methods have been successfully applied to solve numerous problems in passenger railway transportation. The best example is probably the introduction of a completely new timetable on the Dutch railway network in 2006, for

*Corresponding author. E-mail: potthoff@ese.eur.nl, Phone: +31 10 4088940, Fax: +31 10 4089162

which Netherlands Railways recently won the Franz Edelman Award (Kroon et al. (2009)). As in this example, OR methods have mainly been used to solve planning problems (see Huisman et al. (2005); Caprara et al. (2007) for recent surveys). However, on the day of operations, there are also serious challenges to solve. Huge delays are one of the most annoying experiences for railway passengers and have a significant impact on passenger satisfaction. Such delays mainly occur during and after disruptions.

On the day of operations, unforeseen events disturb the smooth completion of the plans. Events like infrastructure malfunctions, accidents or rolling stock breakdowns lead to a temporarily reduced capacity or a complete blockage of a certain route. In the Netherlands, there are on average three complete blockages of a route per day. As a consequence, some trains cannot run as planned; they are delayed or even canceled. Delayed and/or canceled trains could make the underlying rolling stock and/or crew schedules infeasible. In such a disrupted situation, the timetable and the resource schedules need to be modified. Railway disruption management is defined in Jespersen-Groth et al. (2007) as the process of finding a new timetable by rerouting, delaying or canceling trains and rescheduling the resources such that the new timetable is compatible with the resource schedules. Usually, the recovery of the timetable, the rolling stock schedule, and the crew schedule is performed in a sequential way. In the first step, a new timetable is proposed. This is often based on the estimated duration of the disruption. Secondly, the rolling stock is rescheduled in a way that all trips from the modified timetable are covered (see Nielsen (2008)). Finally, given the modified timetable and rolling stock schedule, the crew is rescheduled. Iterations between the three steps may be necessary when it is not possible to assign resources to one of the trips from the proposed timetable. Of course, several disruptions can happen on the same day. Moreover, the estimation of the disruption's duration can be wrong. In this case, an additional recovery is required. All together, the whole process is very dynamic in practice.

One of the most challenging problems in railway disruption management is to reschedule the crews (drivers and conductors). The reason is that these decisions have to be made quickly, while one has to deal with a large number of crews. For instance, the crew schedule of the largest passenger railway operator in the Netherlands, Netherlands Railways (NS), contains around 1,000 duties for drivers on an average workday.

In this paper, we will present an innovative approach to reschedule crews at the moment a disruption occurs and the changes in the timetable and rolling stock schedule are given. We call this problem the operational crew rescheduling problem (OCRSP).

The contribution of this paper is twofold. We propose a new algorithm that first applies a heuristic based on ideas from Huisman (2007) for crew rescheduling to a subset of the duties. However, our algorithm dynamically selects a new subset of duties if this seems promising. This is our first contribution and this idea can be applied to many other domains of rescheduling as well. Our second contribution lies in the fact that we test our methods on real-life data of NS and we show that we can find good solutions in a reasonable amount

of time. As a result, the methods that we propose can lay the foundations for algorithmic decision support for dispatchers in the operations control centers of NS.

The remainder of this paper is organized as follows. We give a more formal problem description of the OCRSP in Section 2. A brief literature overview on crew rescheduling is given in Section 3. A mathematical formulation and the outline of our solution approach is presented in Section 4. In Section 5, we present a heuristic based on column generation and Lagrangian relaxation to solve the OCRSP for a fixed subset of duties. In Section 6, we discuss how we can dynamically adjust the subsets of duties. Computational results are reported in Section 7. We finish with some concluding remarks and directions for further research in Section 8.

2 Problem description

We will now introduce some terminology and give a description of the OCRSP. From this point on, we will focus on train drivers. The problem of rescheduling conductors is, however, quite similar.

The operator's timetable contains all train movements necessary to operate the published service trips. A service trip (commonly known as train) is operated on a line, where a line is specified by a start and an end station and a number of intermediate stops. Furthermore, there are a number of empty train movements and shunting activities that need to be performed in order to efficiently use the rolling stock. An example (see Figure 1) of a line operated by NS is the 700-line from Groningen (Gn) to Amsterdam Airport Schiphol (Shl) with 10 intermediate stops (not all shown in the figure). Because Schiphol is an underground station with limited space, all trains ending in Schiphol are driven empty to the nearby shunt yard Hoofddorp (Hfdo), where they are turned. All operations that need to be performed by a driver are represented by a task, where a task is an elementary sequence of activities starting and ending at a relief point. Relief points are the subset of all stations where drivers can transfer from one rolling stock unit to another one. The trains of the 700-line, for example, are split up into the following tasks: Groningen–Zwolle (Zl), Zwolle–Amersfoort (Amf), Amersfoort–Hoofddorp.

On the day of operations the crew schedule is given by the original duties, each assigned to a driver. These original duties are either active duties, in the sense that they are a sequence of tasks, or reserve duties, meaning that a driver is on standby at a major station for a given time period. All duties start and end at the same crew base, where crew bases are a subset of the relief points. For repositioning from one relief point to another, duties can also contain taxi trips or deadhead tasks. The latter means that a driver is traveling as a passenger on a task.

Due to a disruption on the day of operations the timetable is modified according to the estimated duration of the disruption. This could mean that the new timetable and the driver duties have become incompatible. In this case it is necessary to reschedule the drivers.

Because NS operates very few night trains, for rescheduling it is generally sufficient to consider a crew schedule of a single day.

Given the point in time of rescheduling, for every unfinished original duty we need to find a replacement duty. A replacement duty is composed of all tasks of the associated original duty that started before the time of rescheduling, and a feasible completion. Feasible completions are (possibly empty) sequences of tasks such that the replacement duty satisfies the following rules.

- The replacement duty needs to start and end at the same crew base associated with the original duty. Furthermore, a replacement duty may end earlier or at the planned time. In addition, it is allowed to end up to 60 minutes later than the planned end time of the original duty.
- If in a replacement duty two tasks on different rolling stock units are performed after each other, then a minimum connection time between the two tasks needs to be respected. This connection time is less during rescheduling than in the planning phase.
- Every replacement duty longer than 5 1/2 hours must contain a meal break of at least 30 minutes. Meal breaks are possible only at relief points that have a canteen. Moreover, the working time before and after the break is not allowed to exceed 5 1/2 hours.
- Every driver is licensed to drive on certain parts of the railway network. Moreover, he is licensed to drive certain rolling stock types. These two attributes determine the set of tasks that can be performed by a replacement duty.

If an original duty is not affected by a disruption, one feasible completion is to follow the sequence of tasks as in the original duty.

We will consider a small example where the route between Hoogeveen (Hgv) and Beilen (Bl) (see Figure 1) is blocked from 7:10 to 10:10. Three train lines use this route each with a frequency of once per hour: The 500-line (intercity) between The Hague (Gvc) and Groningen, the earlier mentioned 700-line (intercity) between Schiphol and Groningen, and the 9100-line (regional) from Zwolle to Groningen.

According to the emergency scenarios, the trains of the 500-line coming from Groningen are turned in Assen (Asn) and the trains from The Hague are turned in Hoogeveen, respectively. The same pattern is applied to the 700-line. The regional trains of the 9100-line from Zwolle are turned in Meppel (Mp) and the trains from Groningen are turned in Beilen, respectively.

Figure 2.a shows how original duty D_1 from crew base Groningen was planned. At 7:10, when the rescheduling takes place, the duty is performing task t_1 belonging to the 700-line. Since the route is blocked south of Beilen, the train is turned in Assen, which results in a modified task t_1^r. This means that after performing his first task of the day the driver will be

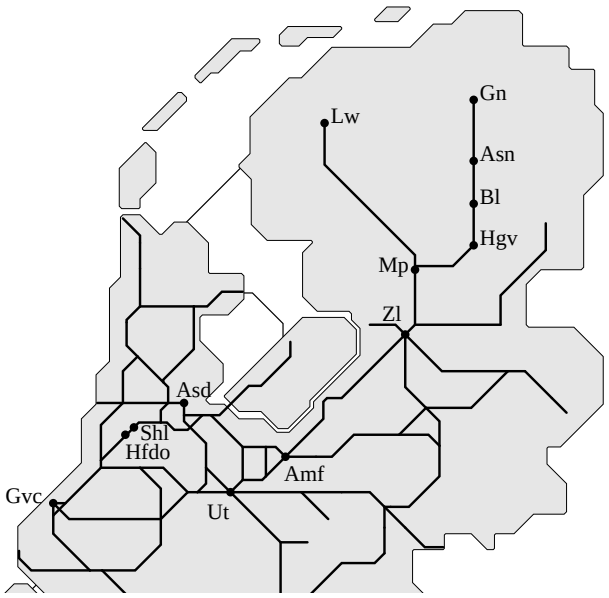

Figure 1: The central and northern part of the railway network operated by NS.

back in Groningen. It is not possible to get to Zwolle in time to perform task t_3 which was supposed to be the next task. Hence duty D_1 is not feasible anymore. Now we will show two examples of feasible completions of D_1. Given 7:10 as the time of rescheduling, D_1 must first complete task t_1^r. Then it is possible to deviate from the plan. One possibility (Figure 2.b) would be to go by taxi from Groningen to Zwolle and drive task t_6 to Amersfoort. After a meal break (MB) in Amersfoort, task t_9 to Hoofddorp on the 700-line could be performed, followed by t_{11} and t_{13} also on the 700-line. Finally, the duty could finish with driving the regional train (9100-line) from Zwolle to Groningen (t_{15}). Note, that in this feasible completion the last two tasks are the same as in the original duty, except that in the original duty the driver was deadheading as a passenger on task t_{13}. Another possibility shown in Figure 2.c would be to continue driving rerouted tasks of the 700-line (t_2^r, t_4^r) before going from Groningen to Zwolle (t_7), also on the 700-line. After a meal break in Zwolle the driver could perform task t_{10} back to Groningen. Since it is allowed to end up to 60 minutes later, the duty could finish with driving two trains (t_{14}, t_{16}) of the 9100-line to Zwolle and back.

Now we can formulate the OCRSP as follows. Given the modified timetable and the planned crew schedule, find a new crew schedule that covers as many tasks as possible such that every original duty is assigned to one feasible completion. The objective of the OCRSP is a trade-off between different aspects, namely feasibility, operational costs, and robustness. We will now briefly discuss these aspects.

First of all, feasibility is very important. In contrast to crew scheduling in the planning phase, it is not at all obvious that all tasks can be covered by a rescheduling solution. If a task cannot be covered, canceling it would lead to a feasible crew rescheduling solution, but

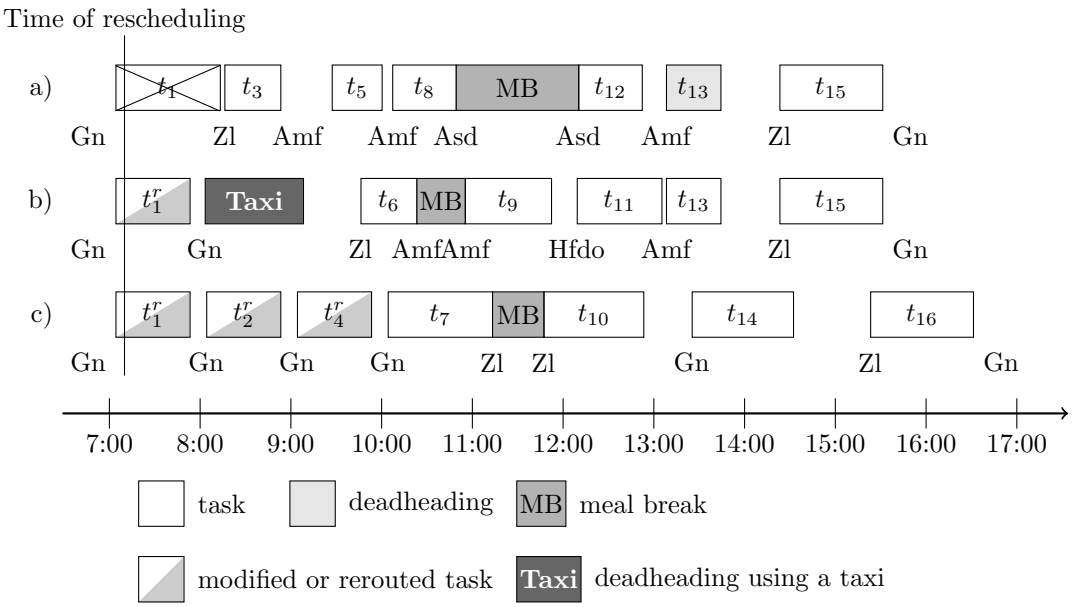

Figure 2: Examples of feasible completions for an affected original duty from crew base Groningen (Gn).

its cancellation could destroy the underlying rolling stock circulations. This conflict needs to be resolved and might lead to a different modified timetable, probably containing less tasks, and different rolling stock circulations.

Operational costs are the second aspect in the objective. Compared to the airline industry, where, especially in North America, operational costs play a major role in rescheduling (see Yu et al. (2003) and Lettovský et al. (2000)), for European railway operators, actual operational costs are less important, because the crew payments are mainly based on fixed salaries. Nevertheless, some parts of a rescheduling solution influence the operational costs. Operator specific compensations for extra work due to changed duties is a typical example. Also taxis for repositioning, or to take home stranded drivers, create additional costs. In contrast to the airline industry, deadheading on trains can be considered to have no additional cost.

The third criterion in the objective is robustness. Currently, humans are involved in the implementation of every rescheduling solution and can cause a failure. A crew dispatcher could, for example, because of the high stress level he is confronted with, forget to call a driver and inform him about the changes. Therefore, a solution is considered to be more robust if less original duties are changed. During rescheduling, connection times between train tasks may be shorter compared to the planning phase. In order to minimize the possibility of delay propagation, short connection times should be avoided if possible.

3 Literature review

During the last decade crew rescheduling, also known as crew recovery, has received a lot of attention in the airline literature. The application of a crew rescheduling decision support system at Continental Airlines (Yu et al. (2003)) won the Franz Edelman award. Stojković et al. (1998) published the first results for a rescheduling model dealing with crew pairing and rostering simultaneously. They apply a column generation approach for a preselected subset of crews. Column generation in combination with core problems defined by a selection of crews and/or time windows has also been used by Lettovský et al. (2000); Nissen and Haase (2006); Medard and Sawhney (2007). Abdelghany et al. (2004) propose a rolling horizon approach tailored for airlines operating a hub-and-spoke network.

To the best of our knowledge, the first attempt to come up with an approach for integral railway disruption management was done by Walker et al. (2005). They present a model that manipulates the timetable and the crew schedule at the same time in order to deal with disruptions. The objective is to minimize simultaneously the deviation of the new timetable from the original one and the actual cost of the crew schedule. One part of the model represents the timetable adjustment, the other part corresponds to a set partitioning model for the crew schedules. Both parts are linked in order to get a compatible solution. However, they considered only a single line and not a complete railway network. Therefore, their approach is not applicable to the Dutch situation.

Rezanova and Ryan (2006) present a column generation approach to railway crew rescheduling. Compared to the approaches in the airline literature their method can be seen as a refined method. Initially, they include only those original duties in the problem, which are directly affected by the disruption. The set of tasks considered in the initial problem is the set of tasks that have been assigned to the affected duties and end within the selected recovery horizon. In the cases where tasks remain uncovered in the solution of the initial problem, they propose to extend the problem by adding reserve duties and original duties that are close to the uncovered tasks in a geographical sense. The approach was tested on instances from the suburban railway of Copenhagen. In these instances no tasks needed to be canceled in the solution of the initial core problem and therefore the extension step was never used.

The paper of Huisman (2007) deals with crew rescheduling, due to track maintenance, in short-term planning. Almost all duties and tasks of the daily crew schedule of NS are included in the optimization problem. In contrast to the OCRSP, the number of available crews is not fixed in short-term planning. Moreover, computation times of several hours, as reported in the paper, are acceptable in this phase but not in the operations.

4 Mathematical model and solution approach overview

In the remainder of the paper we use the following notation.

- S: Set of stations (in our case limited to relief points).
- D: Set of crew bases.
- N: Set of tasks which have not started at the time of rescheduling, where for every $i \in N$ we have:
  - ds_i: Departure station.
  - dt_i: Departure time.
  - as_i: Arrival station.
  - at_i: Arrival time.
- $\Delta = \Delta_A \cup \Delta_R$: Set of unfinished original duties, where Δ_A are active and Δ_R are reserve duties, respectively. Moreover, for every $\delta \in \Delta$ we have:
  - cs_δ: The station where the original duty is at the time of rescheduling or the arrival station of the task performed by the driver at the time of rescheduling.
  - b_δ: The crew base where the original duty starts and ends.
- K^δ: Set of all feasible completions for original duty $\delta \in \Delta$. For every feasible completion $k \in K^\delta$ we have:
  - c_k^δ: Cost of feasible completion k for original duty δ. The cost of a feasible completion is zero if the duty is not modified. Otherwise, the cost is the sum of the cost for changing a duty, the cost for taxis, and the penalties for short connection times and overtime.
  - a_{ik}^δ: Binary parameter indicating if task i is covered by feasible completion k or not.
- f_i: Cost for canceling task i.

We can formulate the OCRSP using binary variables x_k^δ corresponding to the feasible completions of duty δ and binary variables y_i indicating if task i is canceled (1) or not (0).

$$\min \sum_{\delta \in \Delta} \sum_{k \in K^\delta} c_k^\delta x_k^\delta + \sum_{i \in N} f_i y_i \tag{1}$$

$$\text{s.t.} \quad \sum_{\delta \in \Delta} \sum_{k \in K^\delta} a_{ik}^\delta x_k^\delta + y_i \geq 1 \quad \forall i \in N \tag{2}$$

$$\sum_{k \in K^\delta} x_k^\delta = 1 \quad \forall \delta \in \Delta \tag{3}$$

$$x_k^\delta, y_i \in \{0,1\} \quad \forall \delta \in \Delta, \forall k \in K^\delta, \forall i \in N \tag{4}$$

In the above model, constraints (2) make sure that every task is either covered by a feasible completion or canceled. Furthermore, constraints (3) ensure that every original duty is assigned to exactly one feasible completion.

Note that, in the above model, deadheading can occur in two ways. Firstly, a feasible completion can explicitly use deadheading on tasks, e.g., if the driver of the original duty does not have the required route knowledge. In this case, the corresponding a^{δ}_{ik} coefficient is equal to 0. Secondly, a task can be overcovered in the solution to the model, then one of the drivers has to perform the driving, the other(s) deadhead on this task, but all coefficients a^{δ}_{ik} are equal to 1.

Recall from Section 1 that we can have about 1,000 original duties, of which about 90 are reserve duties, in (3). Moreover, the number of set covering constraints in (2) can be up to 10,000. The number of feasible completions for an original duty can range from only a few if the duty is almost finished when we reschedule, to millions if the duty has not started or has just started. If rescheduling is done on the day of operations, the emphasis is on obtaining the best possible solution within a couple of minutes of computation time rather than solving (1)–(4) to optimality.

Moreover, a local disruption like the one described in Section 2 affects only a limited number of original duties. Because we want to stay close to the planned schedule, it seems highly unlikely that an original duty covering tasks only in another part of the country will be modified in an optimal solution of (1)–(4) in this case. Therefore, it seems reasonable to consider a core problem containing only a subset of the original duties and tasks.

The advantage of a core problem is its reduced size, which will lead to shorter computation times. A drawback is that the solution quality might depend on the choice of the core problem. In particular, one might be able to reduce the number of canceled tasks by increasing the size of the core problem.

For the case of airline crew rescheduling this has been observed by Lettovský et al. (2000) and Nissen and Haase (2006). In both papers the core problems are generated using a set of parameters, which makes it possible for the dispatcher to solve the problem again with a larger core problem, if he is unsatisfied with the quality of the solution obtained so far. The drawback of this scheme is that computation times increase rapidly with the size of the core problems.

In order to overcome this drawback, we propose a different way, illustrated in Figure 3, of exploring promising parts of the solution space of (1)–(4). As in the other approaches, we start with an initial core problem. This initial core problem is defined such that it has a high probability of containing a good solution and is of a size that allows us to explore it within a small amount of time. If tasks need to be canceled in the solution obtained for the initial core problem, we try to cover them by exploring a neighborhood for each uncovered task in turn. We use a heuristic based on column generation and Lagrangian relaxation to explore

the core problems. This heuristic is described in Section 5. In Section 6, we discuss how we define the core problems.

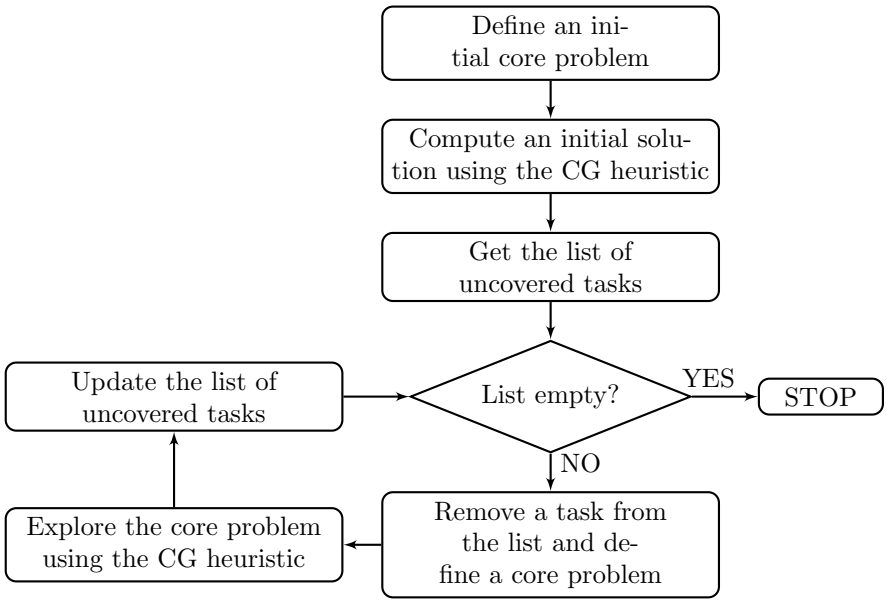

Figure 3: Overview of the algorithm

Starting with an initial feasible solution and trying to improve it iteratively by fixing a part of the solution and reoptimizing the remaining part has been proposed for several combinatorial problems. Examples are the heuristic of Caprara et al. (1999) for the set covering problem, and the large neighborhood search (LNS) heuristics of Ropke and Pisinger (2006) and Prescott-Gagnon et al. (2007) for the vehicle routing problem with time windows and of Pepin et al. (2008) for the multiple depot vehicle scheduling problem. The latter two papers have in common with this paper that they use heuristic column generation for neighborhood exploration.

5 Exploring the core problems

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. More formally a core problem reads:

$$\min \sum_{\delta\in\bar{\Delta}} \sum_{k\in\bar{K}^\delta} c_k^\delta x_k^\delta + \sum_{i\in\bar{N}} f_i y_i \tag{5}$$

$$\text{s.t.} \quad \sum_{\delta\in\bar{\Delta}} \sum_{k\in\bar{K}^\delta} a_{ik}^\delta x_k^\delta + y_i \geq 1 \quad \forall i \in \bar{N} \tag{6}$$

$$\sum_{k\in\bar{K}^\delta} x_k^\delta = 1 \quad \forall \delta \in \bar{\Delta} \tag{7}$$

$$x_k^\delta, y_i \in \{0,1\} \quad \forall \delta \in \bar{\Delta}, \forall k \in \bar{K}^\delta, \forall i \in \bar{N} \tag{8}$$

where $\bar{K}^\delta \subseteq K^\delta$. This subset contains all feasible completions that are represented by a path in graph $\bar{G}^\delta$, discussed in Section 5.2.

To find good feasible solutions to (5) subject to (6)–(8) fast, we use a Lagrangian heuristic similar to the one proposed by Huisman (2007). Therefore, we relax the covering constraints (6) in a Lagrangian way introducing nonnegative Lagrangian multipliers $u_i, i \in \bar{N}$. The Lagrangian subproblem then becomes:

$$\begin{aligned} \Theta(u) &= \min \sum_{\delta\in\bar{\Delta}} \sum_{k\in\bar{K}^\delta} c_k^\delta x_k^\delta + \sum_{i\in\bar{N}} f_i y_i + \sum_{i\in\bar{N}} u_i (1 - \sum_{\delta\in\bar{\Delta}} \sum_{k\in\bar{K}^\delta} a_{ik}^\delta x_k^\delta - y_i) \\ \text{s.t.} &\quad (7) \text{ and } (8) \end{aligned}$$

which can be rewritten as

$$\begin{aligned} \Theta(u) &= \min \sum_{i\in\bar{N}} u_i + \sum_{\delta\in\bar{\Delta}} \sum_{k\in\bar{K}^\delta} (c_k^\delta - \sum_{i\in\bar{N}} u_i a_{ik}^\delta) x_k^\delta + \sum_{i\in\bar{N}} (f_i - u_i) y_i \\ \text{s.t.} &\quad (7) \text{ and } (8) \end{aligned} \tag{9}$$

The Lagrangian subproblem is separable and therefore the optimal solution to it can be found with the following procedure. In order to not violate constraints (7) we set $x_k^\delta = 1$ for one $k \in \arg\min\{\bar{c}_k^\delta(u) : k \in \bar{K}^\delta\}$ for each $\delta \in \bar{\Delta}$, where $\bar{c}_k^\delta(u) := c_k^\delta - \sum_{i\in\bar{N}} u_i a_{ik}^\delta$ is the reduced cost of feasible completion k. All other x_k^δ variables are set to 0. Furthermore, for each $i \in \bar{N}$, we set $y_i = 1$ if $f_i - u_i < 0$ and $y_i = 0$ otherwise.

Now the Lagrangian dual problem is to find

$$\Theta^* = \max \Theta(u), \quad u \geq 0$$

Because the number of feasible completions for every driver can still be huge we combine Lagrangian relaxation with column generation. We assume that the reader is familiar with the general ideas of column generation (for references see e.g. Desrosiers and Lübbecke (2005)). We will thus consider a *restricted master problem* (*RMP*) of (7)–(9) containing only a subset of the x_k^δ variables. In the nth column generation iteration the x_k^δ variables in the RMP are given by $\cup_{\delta\in\bar{\Delta}}\{x_k^\delta : k \in \bar{K}_n^\delta\}$, where $\bar{K}_n^\delta \subseteq \bar{K}^\delta$ is a subset of feasible completions. Let Θ_n^* be the optimal value of the associated Lagrangian dual problem. For every RMP we

use subgradient optimization (see e.g. Fisher (1981)) to approximate Θ_n^*. Let u_n^* be the corresponding multiplier vector. To check if Θ_n^* is a good approximation of Θ^*, or if we need to add feasible completions to the RMP in order to potentially improve on Θ_n^*, we solve a pricing problem for every original duty $\delta \in \bar{\Delta}$. The pricing problems are modeled as resource constrained shortest path problems in dedicated graphs as described later in Section 5.2. Let $r_n^\delta := \min\{\bar{c}_k^\delta(u_n^*) : k \in \bar{K}_n^\delta\}$ be the smallest reduced cost of the already generated feasible completions for original duty δ and $z_n^\delta := \min\{\bar{c}_k^\delta(u_n^*) : k \in \bar{K}^\delta\}$ the optimal value of the pricing problem for δ. Then the feasible completion k corresponding to z_n^δ should be added to the RMP if $z_n^\delta - r_n^\delta < 0$. Moreover, $LB_n := \Theta_n^* + \sum_{\delta \in \bar{\Delta}} (z_n^\delta - r_n^\delta)$ is a lower bound on Θ^*.

Furthermore, when the subgradient method terminates, we invoke a greedy procedure to find feasible solutions to the core problem. This procedure, which takes as input a multiplier vector, is repeated up to *maxMulti* times using the multiplier vectors obtained in the last *maxMulti* iterations of the subgradient algorithm. In our experiments, *maxMulti* was set to 100 or 200. The greedy procedure is presented in Figure 4. First, we order the original duties by increasing reduced cost of the x_k^δ variables that were set to one in the Lagrangian subproblem solution. Moreover, we set all $y_i = 1$ (Line 1). We initialize $\hat{u}$ with the current vector of multipliers u (Line 2). Then, we choose for every original duty the best feasible completion with respect to quasi reduced cost depending on $\hat{u}$ (Lines 3 – 6). If there are uncovered tasks, we try to cover them by reserve duties that are idle (Lines 9 – 13), where idle means that we have chosen the feasible completion of their duties that covers no tasks.

```
1  Order the original duties δ ∈ Δ̄. Set y_i = 1 for all i ∈ N̄;
2  Set û = u;
3  forall δ ∈ Δ̄ do
4      Choose k*(δ) ∈ arg min{c̄_k^δ(û) : k ∈ K̄_n^δ} and set the corresponding x^δ_{k*(δ)} = 1;
5      Set û_i = 0 and y_i = 0 for all i ∈ N̄ with a^δ_{ik*(δ)} = 1;
6  end
7  Set û_i = f_i if y_i = 1 for all i ∈ N̄;
8  Construct the set of idle reserve duties Δ̄_I := {δ ∈ Δ̄_R : a^δ_{ik*(δ)} = 0 for all i ∈ N̄};
9  forall δ ∈ Δ̄_I do
10     Set x^δ_{k*(δ)} = 0;
11     Choose q*(δ) ∈ arg min{c̄_k^δ(û) : k ∈ K̄_n^δ} and set the corresponding x^δ_{q*(δ)} = 1;
12     Set û_i = 0 and y_i = 0 for all i ∈ N̄ with a^δ_{iq*(δ)} = 1;
13 end
```

Figure 4: Greedy procedure to construct feasible solutions

When we explore a new core problem, we warm start the RMP with columns generated earlier if possible. In order to do so, we store all generated columns in a column pool. If a

new core problem contains original duties that have been considered in other core problems, we scan the column pool and add columns to the RMP if all tasks covered by the column are included in the new core problem.

5.1 Acceleration strategies and column fixing

We apply two acceleration strategies, partial pricing and early termination, in our column generation procedure. We stop the pricing as soon as we have found attractive columns for 30 % of the pricing problems. Note that we can compute LB_n only if we solve all pricing problems. Moreover, we terminate the column generation process when the relative gap between LB_n and Θ_n^* is small, in our case below 0.1 %.

When the relative gap between the cost UB^* of the best feasible solution found so far and the lower bound LB_n is larger than a threshold, we switch to a column fixing phase, where we perform a depth first search in a branch and bound tree without backtracking. We may fix the columns of several original duties in every node and we apply column generation in every node.

For selecting the columns that we fix in a given node, we use information about how often a column appeared in a Lagrangian subproblem solution while solving the last RMP. For every feasible completion k for every original duty δ, we compute the ratio $R_k^\delta = \frac{s_k^\delta}{U}$, where s_k^δ is the number of times x_k^δ was set to 1 in a Lagrangian subproblem solution during subgradient optimization of the last RMP and U is the number of iterations performed by the subgradient algorithm. We order the feasible completions by decreasing values of R_k^δ. We start with setting the feasible completion with the largest value of R_k^δ to 1 and all other feasible completions from the same original duty to 0. For the same node, we then continue with the feasible completion with the next largest value of R_k^δ as long as $R_k^\delta \geq 0.7$ and the number of original duties for which we fixed the feasible completions in this node is less than μ percent of the original duties (μ was set to 10 in our experiments). This scheme is closely related to the α-fixing procedure proposed by Holmberg and Yuan (2000).

5.2 Pricing problems

For every original duty $\delta \in \bar{\Delta}$ we build a graph $\bar{G}^\delta$ in which every feasible completion k that satisfies the following criterion is represented by a path in $\bar{G}^\delta$: Every task i covered by k as well as every task that is used for deadheading in k belong to $\bar{N}$. Moreover, given a vector of Lagrangian multipliers u, the cost of every path corresponds to the reduced cost of the feasible completion.

In these graphs we use several types of nodes and arcs in order to model the feasible completions. The source of graph $\bar{G}^\delta$ captures the position of original duty δ at the time of rescheduling. There are three possibilities: The duty might not have started (i). If the duty

has started, the driver is either performing a task (ii), or he transfers at a station (iii). The sink node of $\bar{G}^{\delta}$ corresponds to the end of an original duty.

Besides the source and sink, we introduce a pair of nodes for the departure and the arrival of each task i. These nodes are connected by an arc representing driving task i. A copy of the arc is used to model deadheading of a driver on task i, if the driver is not allowed to drive task i due to his route and/or rolling stock licenses.

A transfer arc from the arrival node of a task i to the departure node of a task j exists, if a driver can perform task j immediately after task i. In general, this is possible if task j starts at the end station of task i and if either the time between the arrival and the departure is larger than the minimum connection time, or the two tasks are operated with the same rolling stock.

Transfer arcs have a property indicating if this transfer can be used as a meal break. This is the case if the transfer takes place at a station that has a canteen and the transfer time is long enough.

From some stations there are taxi connections to other stations for given periods of the day. This occurs for example if the shunting area is located far from a station or crew base. In this case drivers travel by taxi between the stations and the shunting areas to perform pull-out and pull-in tasks. Moreover, alternative ways of transportation might be used during rescheduling to reposition drivers. These deadhead transfers are also modeled by taxi arcs although they could be bus trips or trips on trains of other operators in reality.

Constructing the graphs in this way, not every path corresponds to a feasible completion because it might violate the meal break rule. Therefore, we solve the subproblems as resource constrained shortest path problems (see Irnich and Desaulniers (2005)). As resources we use the working time before and after the meal break.

6 Defining the core problems

6.1 Initial core problem

After initial experiments we came up with the following selection of the subset of original duties $\bar{\Delta}$ for the initial core problem. This selection is a good compromise between computation time and solution quality.

We build $\bar{\Delta}$ in four steps. In the first step, we add all tasks which are canceled or modified (rerouted) to N_1. Secondly, we build a set N_2 where we add an unmodified task j if it has the same pair of start and end stations as one of the tasks in N_1 and if its departure time dt_j lies in the interval $[t_0, t_1 + 60 \text{ minutes}]$, where t_0 is the earliest departure time of a task $i \in N_1$ with $ds_i = ds_j$ and $as_i = as_j$, t_1 is the latest arrival time of a task $i' \in N_1$ with $ds_{i'} = ds_j$ and $as_{i'} = as_j$. In the third step we add a task j to N_3 if it is part of the same train as one of the tasks in $N_1 \cup N_2$. Finally, we define the subset of original duties

$\bar{\Delta} := \Delta_R \cup \{\delta \in \Delta_A : \delta \text{ covers at least one task in } N_1 \cup N_2 \cup N_3\}$. Note that we include all reserve duties in the initial core problem.

6.2 Neighborhoods for uncovered tasks

Given our crew rescheduling problem, the largest improvement and the one we are mainly interested in is covering tasks that have not been covered in the solution of the initial core problem. Therefore, we are interested in neighborhood operators which, given an uncovered task, define a neighborhood such that exploring the neighborhood could lead to a crew schedule that covers more tasks.

In the first step we select a number of candidates. These duties can possibly cover the uncovered task. Usually this would leave other tasks uncovered and in order to assign them to other duties we select in step two for each candidate duty a number of similar duties that offer possibilities to swap parts of the duties.

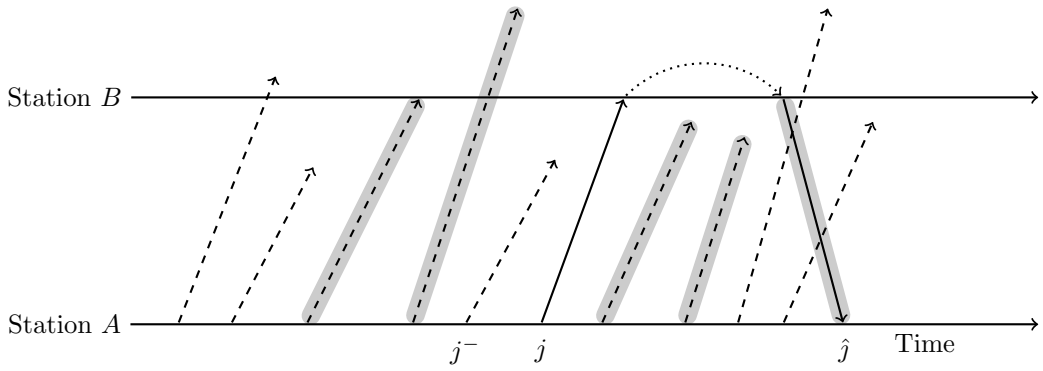

Figure 5: Selecting replacement duties that cover tasks leaving from station A just before and after task j.

The candidates in the first step are selected as follows (see Figure 5). Given the departure time and station (A in the example) of the uncovered task j we look at task j^- that departs from the same station the closest before task j. Then we consider the replacement duty σ that covers j^- in the current solution and check heuristically, considering rolling stock and route knowledge, if σ could cover j. If yes, then we select σ as a candidate and continue the next task that departs from station A before j^- until we have selected r candidates. We repeat the procedure considering tasks that depart from station A after task j.

Furthermore, we select the replacement duty which covers task $\hat{\jmath}$, the first task that leaves station B and goes back to station A such that a driver can transfer from j to $\hat{\jmath}$. Including this original duty ensures that it is possible to perform task j and then deadhead back to station A.

In Figure 5, we have marked in gray the tasks covered by replacement duties that have been selected. Note that, because of missing route knowledge, task j^- is not marked.

In the second step we select for every candidate the s most similar duties that have not been selected yet. We define similarity between duties as the number of stations that are visited around the same time. We also add a bonus if they share the same current station and crew base. The idea behind this measure is that two duties can possibly swap parts if they have a departure from the same station around the same time and both reach another station later in their duty again around the same time. Given a candidate σ, another duty τ and the set of tasks N_σ and N_τ covered by the duties, we compute the similarity as

$$S(\sigma, \tau) := B(\sigma, \tau) + \sum_{i \in N_\sigma} \sum_{j \in N_\tau} \gamma_{ij}$$

where

$$\gamma_{ij} := \begin{cases} 1 \text{ if } ds_i = ds_j \text{ and } |dt_i - dt_j| \leq \omega \\ 0 \text{ otherwise} \end{cases}$$

indicates that tasks i and j depart from the same station within at most ω minutes from each other. The bonus function $B(\sigma, \tau) := B_b(\sigma, \tau) + B_{cs}(\sigma, \tau)$ sums up the bonus for the same crew base B_b and same current station B_{cs}, respectively. For our experiments we use

$$B_b(\sigma, \tau) := \begin{cases} 0.6 \text{ if } b_\sigma = b_\tau \\ 0 \text{ otherwise.} \end{cases}$$

$B_{cs}(\sigma, \tau)$ is defined accordingly.

7 Computational experiments

We implemented our solution approach in C++ and compiled it with the Visual C++ 8.0 compiler. We ran our experiments on an *Intel Pentium D* processor with 2 GB RAM clocked at 3.4 GHz.

For the objective function we specified the following cost coefficients. The value of f_i depends on the type of the task. Canceling a task from a station A to another station B would make the underlying rolling stock schedule infeasible, therefore we set $f_i = 20,000$ for these tasks. Under the mild assumption that the rolling stock assigned to a task from A to A can be moved to the shunting area and pulled out again, these tasks leave the rolling stock schedule intact. Since this situation is preferred from the point of view of the overall disruption management process, we set the corresponding f_i to $3,000$. The cost of each feasible completion of a duty is zero if the duty is unchanged or the sum of penalties depending on the way the duty is changed. We used the following values for penalties: 400 if a duty is changed, 50 for every task that is not assigned to its original duty, 1 for every transfer between two tasks that was not used in the original plan by some duty and $1,000$ if the driver has to be

Location	ID	Time	Type	Affected duties
Abcoude	Ac A	11:00-14:00	two sided blockage	59
Abcoude	Ac B	16:30-19:30	two sided blockage	53
Beilen	Bl A	07:00-10:00	two sided blockage	15
Beilen	Bl B	16:00-19:00	two sided blockage	15
's-Hertogenbosch	Ht A	08:00-11:00	two sided blockage	55
's-Hertogenbosch	Ht B	15:30-18:30	two sided blockage	51
Lelystad	Lls A	04:00-07:00	two sided blockage	25
Lelystad	Lls B	13:00-16:00	two sided blockage	22
Zoetermeer	Ztm A	08:00-11:00	reduced number of trains	21
Zoetermeer	Ztm B	11:30-14:30	reduced number of trains	25

Table 1: Summary of the different scenarios

repositioned using a taxi. In the experiments we had no penalties for short connection times and overtime.

7.1 Instances

As a starting point for our instances we remodeled five scenarios, spread over the country, that happened in the past. All scenarios lasted about three hours. Therefore, we chose an estimated duration of 3 hours for our remodeled instances. For every historic scenario, we generated a second disruption with the same estimated duration but at a different time of the day. We modified the timetable following the main ideas behind the emergency scenarios. Because rescheduling of rolling stock is in itself a difficult optimization problem, we considered a simplified rolling stock schedule, which can easily be adapted to the new timetable. For the original duties we used a crew schedule that was operated by NS on a workday somewhere in September 2007.

A general description of the 10 cases is given in Table 1. The disruptions around Abcoude, which is located between Utrecht and Amsterdam, and around 's-Hertogenbosch, which is located south of Utrecht, involve heavily used routes. More than 50 original duties are affected by the disruptions in these instances. The involved routes in the instances at Beilen and Lelystad are not so heavily used, but the route blockages disconnect some ends of the railway network from the remaining part. The instances at Zoetermeer also involves a heavily used route, but in these instances a reduced number of trains can be operated on the involved route, because it is not completely blocked.

7.2 Results for initial core problems

In a first series of experiments, we applied our algorithm to the described scenarios but we solved only the initial core problems. These core problems were constructed as described in Section 6.1. In addition, we included a number of reserve duties. Recall that the crew schedule of NS contains about 90 reserve duties on an average workday. At the time a large disruption occurs and rescheduling takes place, not all reserve duties might be available for rescheduling. One reason is that reserve duties are also used when a train driver missed a connection because of a delay. Moreover, another disruption could have happened earlier and reserve duties might have been used in order to recover from this disruption. In order to take this into account somehow during our experiments, we derived three sets of reserve duties R_1–R_3 from a given initial plan R_0 in the following way. In the set R_1 (R_2), every reserve duty from R_0 had a probability of 50% (25%) to be included in R_1 (R_2) as well. Based on drawing a single random number between 0 and 1 for every reserve duty in R_0 we obtained the sets R_1 and R_2. Note here that the drawing for R_2 was done independently of the drawing for R_1. This procedure resulted in sets R_1 with 46 reserve duties and R_2 with 20 reserve duties. Finally, set R_3 does not contain any reserve duty at all.

In Tables 2–4 we report the results for the three sets of reserve duties R_1–R_3, respectively. Here the columns have the following meanings. The first column is the *ID* of the instance. $|\bar{\Delta}|$ is the number of original duties in the initial core problem (finished reserve duties are excluded) and $|\bar{N}|$ is the number of set covering constraints in (6). Column *LB* reports the value of LB_n when terminating the column generation. *UB* is the value of the best feasible solution. *GAP* is the percentage gap between *LB* and *UB*. *Time* is the computation time in seconds. In the last five columns, we give some insight into the best feasible solution. *A-A* and *A-B* denote the number of tasks of the types A-A and A-B that need to be canceled. *Taxi* is the number of additional taxi trips used. *MD* is the number of active original duties that are feasible but modified in the solution. *UR* is the number of reserve duties that cover tasks in the solution. Note that the total number of original duties that are modified is the sum of *MD*, *UR* and the number of affected duties (see Table 1).

First of all, we observe that all computation times are less than 4 minutes. The computation time mainly depends on the size of the core problems in terms of set covering constraints. For example, for set R_1, computation times range from 8 seconds for Bl B to 170 seconds for Ht A.

Moreover, the number of canceled tasks is at most 1 for the experiments with reserve duties and at most 2 for the experiments without reserve duties. The number of instances where all tasks are covered in the solution, is equal to 7, 8, and 6 for sets R_1, R_2 and R_3, respectively. It is not surprising that without reserve duties (R_3) the number of instances where all tasks can be covered is less compared to the experiments with reserve duties (R_1 and R_2). Interestingly, with R_2 we can cover all tasks in more instances compared to R_1, while

ID	$\lvert\bar{\Delta}\rvert$	$\lvert\bar{N}\rvert$	LB	UB	GAP (%)	Time (s)	A-B	A-A	Taxi	MD	UR
Ac A	163	602	63295	63588	0.5	144	1	0	1	5	6
Ac B	143	736	34351	34554	0.6	155	0	0	0	0	4
Bl A	80	234	10586	10586	0.0	11	0	0	1	0	2
Bl B	63	186	9541	9541	0.0	8	0	0	0	0	3
Ht A	134	617	38755	39454	1.8	170	0	0	5	0	4
Ht B	118	658	57731	58148	0.7	135	1	0	3	0	3
Lls A	81	239	17671	17775	0.6	16	0	0	5	0	5
Lls B	149	402	20003	20003	0.0	68	0	0	2	0	2
Ztm A	112	508	11976	12043	0.6	68	0	0	1	0	1
Ztm B	110	423	35192	35196	< 0.1	39	1	0	1	0	1

Table 2: Results for initial core problems with reserve R_1

ID	$\lvert\bar{\Delta}\rvert$	$\lvert\bar{N}\rvert$	LB	UB	GAP (%)	Time (s)	A-B	A-A	Taxi	MD	UR
Ac A	137	602	45544	46303	1.7	129	0	0	1	7	5
Ac B	124	736	35335	35871	1.5	165	0	0	0	1	4
Bl A	54	234	10588	10588	0.0	8	0	0	1	0	2
Bl B	44	186	10414	10604	1.8	8	0	0	0	3	1
Ht A	108	617	39212	39699	1.2	129	0	0	6	0	2
Ht B	96	658	58951	59099	0.3	129	1	0	3	0	2
Lls A	55	239	19272	19272	0.0	10	0	0	5	0	2
Lls B	121	402	20407	20407	0.0	42	0	0	6	2	1
Ztm A	86	508	12196	12343	1.2	42	0	0	0	0	1
Ztm B	84	423	35394	35394	0.0	31	1	0	0	0	0

Table 3: Results for initial core problems with reserve R_2

ID	$\lvert\bar{\Delta}\rvert$	$\lvert\bar{N}\rvert$	LB	UB	GAP (%)	Time (s)	A-B	A-A	Taxi	MD	UR
Ac A	117	602	87125	87969	1.0	115	2	0	1	16	-
Ac B	111	736	35924	37687	4.9	181	0	0	0	5	-
Bl A	34	234	11517	11642	1.1	9	0	0	3	0	-
Bl B	31	186	13115	14457	10.2	8	0	1	1	5	-
Ht A	88	617	39998	40208	0.5	150	0	0	7	0	-
Ht B	82	658	61914	62609	1.1	132	1	1	3	0	-
Lls A	35	239	22775	22775	0.0	7	0	0	8	1	-
Lls B	103	402	20805	21253	2.2	58	0	0	7	2	-
Ztm A	66	508	12351	12351	0.0	29	0	0	0	1	-
Ztm B	64	423	35391	35394	< 0.1	24	1	0	0	0	-

Table 4: Results for initial core problems with reserve R_3

the absolute number of reserve duties is only 20 compared to 46 in R_1. This indicates that it is important where and when reserve duties are available for rescheduling. Furthermore, we observe that at most 6 reserve duties are used.

Furthermore, we see that the impact of crew rescheduling on the whole crew schedule differs significantly. The impact is limited considering the experiments with reserve duties. Without reserve duties more duties are modified for most of the instances. In some cases, Ac B, Lls A and Ztm A, this can compensate for the absence of reserve duties.

7.3 Results with neighborhood exploration

We have seen that the solutions to the core problems are good in terms of number of canceled tasks especially when reserve duties are available. However, we would like to see if some of the uncovered tasks can be covered when we explore a neighborhood as defined in Section 6.2. In the following, we only consider the instances where cancelation of tasks occur in the solution of the initial core problem.

We present our results in Tables 5 – 8. In these tables the first column is the *ID* of the instances. Column *It* is the number of the core problem exploration in the overall algorithm (see Figure 3), where a 1 corresponds to the initial core problem and a number greater than 1 corresponds to a neighborhood exploration. *Fixed* provides the total cost of the fixed duties when the current core problem is solved. $|\bar{\Delta}|$ and $|\bar{N}|$ are the number of original duties and set covering constraints in the core problem. *LB*, *UB*, and *GAP* give the values for LB_n, the best feasible solution cost and the percentage gap. *TI* is the computation time needed for exploring the core problem. The next four columns show the status of the overall algorithm. *Sol*, which is equal to *Fixed* + *UB*, is the objective value of the new crew schedule. *TT* is the total computation time of the algorithm. *A-B* and *A-A* are the number of canceled tasks of the corresponding types.

We first tried a relatively small neighborhood, where r and s were set to 3. When considering R_1 as the set of reserve duties we can improve the solution of the initial core problems in 2 out of 3 cases and find solutions that cover all tasks (see Table 5). Exploring the neighborhoods of the uncovered tasks took between 8 and 19 seconds. With R_2 as the set of reserve duties we can improve the solution in 1 of the 2 cases (Table 6).

For the case Ht B with R_1, we also tried to obtain a better solution by exploring a larger neighborhood of the uncovered task. We tried the settings $r = s = 5$ and $r = s = 8$. This increased the size of the core problems to $|\bar{\Delta}| = 99$ and $|\bar{N}| = 622$ for the first and $|\bar{\Delta}| = 190$ and $|\bar{N}| = 1383$ for the second. As a result the time needed to explore the neighborhoods went up to 44 and 373 seconds, respectively. However, we could not find better solutions in terms of canceled tasks. We obtained similar results for the same experiments with Ht B and R_2.

ID	It	Fixed	$\lvert\bar{\Delta}\rvert$	$\lvert\bar{N}\rvert$	LB	UB	GAP (%)	TI (s)	Sol	TT (s)	A-B	A-A
Ac A	1	0	163	602	63295	63588	0.5	144	63588	144	1	0
Ac A	2	41831	72	225	3363	3465	3.0	19	45296	163	0	0
Ht B	1	0	118	658	57731	58148	0.7	135	58148	135	1	0
Ht B	2	31415	60	263	26733	26733	0.0	11	58148	146	1	0
Ztm B	1	0	110	423	35192	35196	<0.1	39	35196	39	1	0
Ztm B	2	14693	72	167	1005	1005	0.0	8	15698	47	0	0

Table 5: Results for neighborhood exploration with $r = 3, s = 3$ using R_1

ID	It	Fixed	$\lvert\bar{\Delta}\rvert$	$\lvert\bar{N}\rvert$	LB	UB	GAP (%)	TI (s)	Sol	TT (s)	A-B	A-A
Ht B	1	0	96	658	58951	59099	0.2	129	59099	129	1	0
Ht B	2	32314	40	298	27431	27431	0.0	10	59099	139	1	0
Ztm B	1	0	84	423	35394	35394	0.0	31	35394	31	1	0
Ztm B	2	14841	47	166	1056	1056	0.0	6	15897	37	0	0

Table 6: Results for neighborhood exploration with $r = 3, s = 3$ using R_2

In Table 7, we present the results of the neighborhood exploration with $r = 4$ and $s = 4$ when no reserve duties are present (R_3). For 2 out of the 4 considered instances we can significantly improve on the solutions of the initial core problems. Moreover, for Ztm B we were able to find a solution that covers all tasks.

We run our algorithm again after increasing r and s to 6. With this setting, which generates larger neighborhoods, we found better solutions for 3 of the 4 instances (see Table 8), but only for Ht B we found a solution covering more tasks compared to the smaller neighborhood.

Comparing the results with the two different choices of r and s we can see that we can obtain slightly better results by spending more time in exploring larger neighborhoods.

8 Summary and conclusion

We have proposed an algorithm to solve the OCRSP. Given a disruption and a real-life crew schedule from NS, we have shown how to select a subset of the original duties in the crew schedule such that we can find solutions of good quality within a short amount of time. This was achieved by combining column generation and Lagrangian relaxation into a heuristic algorithm. The proposed column fixing also enables us to obtain good solutions for the larger instances.

Furthermore, we developed an extension, exploring neighborhoods of tasks which could not be covered with the initial selection of duties. We have shown that with this extension, it is possible to reduce the number of canceled tasks in many cases. This is an important

ID	It	Fixed	$\lvert\bar{\Delta}\rvert$	$\lvert\bar{N}\rvert$	LB	UB	GAP (%)	TI (s)	Sol	TT (s)	A-B	A-A
Ac A	1	0	117	602	87125	87969	1.0	115	87969	115	2	0
Ac A	2	37973	48	272	49996	49996	0.0	9	87969	124	2	0
Ac A	3	41586	53	341	28591	28591	0.0	14	70177	138	1	0
Bl B	1	0	31	186	13115	14457	10.2	8	14457	8	0	1
Bl B	2	7530	36	307	6927	6927	0.0	7	14457	15	0	1
Ht B	1	0	82	658	61914	62609	1.1	132	62609	132	1	1
Ht B	2	30264	28	274	32345	32345	0.0	33	62609	165	1	1
Ht B	3	37900	25	229	24707	24709	<0.1	12	62609	177	1	1
Ztm B	1	0	64	423	35391	35394	<0.1	24	35394	24	1	0
Ztm B	2	13887	27	167	2010	2010	0.0	10	15897	34	0	0

Table 7: Results for neighborhood exploration with $r = 4, s = 4$ using R_3

ID	It	Fixed	$\lvert\bar{\Delta}\rvert$	$\lvert\bar{N}\rvert$	LB	UB	GAP (%)	TI (s)	Sol	TT (s)	A-B	A-A
Ac A	1	0	117	602	87125	87969	1.0	117	87969	117	2	0
Ac A	2	33200	95	556	36380	36383	<0.1	61	69583	178	1	0
Bl B	1	0	31	186	13115	14457	10.2	8	14457	8	0	1
Bl B	2	5571	67	511	8485	8485	0.0	25	14056	33	0	1
Ht B	1	0	82	658	61914	62609	1.1	131	62609	131	1	1
Ht B	2	25196	89	836	37347	37413	0.2	205	62609	336	1	1
Ht B	3	32423	86	789	28994	29199	0.7	392	61622	728	1	0
Ztm B	1	0	64	423	35391	35394	<0.1	24	35394	24	1	0
Ztm B	2	13334	84	613	2563	2563	0.0	52	15897	76	0	0

Table 8: Results for neighborhood exploration with $r = 6, s = 6$ using R_3

improvement compared to algorithms which rely on an a priori defined core problem. In our experiments, considering two sets of reserve duties, we can cover all tasks in 9 out of 10 instances. In the case where we do not consider any reserve duties, we can increase the number of instances where no tasks need to be canceled from 6 after solving the initial core problem to 7 by our neighborhood exploration scheme, while for 2 others we reduce the number of canceled tasks.

We believe that the idea of neighborhood exploration can be used in other areas of rescheduling as well. Moreover, our algorithm can easily be extended by new neighborhood definitions. This could further improve the performance of the algorithm.

We have shown that OR methods can deal with the size, complexity and time restrictions of crew rescheduling on the day of operations. However, railway operations are very dynamic and major disruptions as discussed in this paper happen along with disturbances that cause small delays. Future research challenges, probably involving simulation/optimization techniques, lie in evaluating the cross-effects between delays and disruptions, the consequences of several disruptions which are overlapping in time, and the influence of a duration that differs from the expected duration. These should be the next steps on the way to introduce OR-based decision support systems into control centers of passenger railway operators.

Acknowledgments

This research is made possible with support of Transumo. Transumo (TRANsition SUstainable MObility) is a Dutch platform for companies, governments and knowledge institutes that cooperate in the development of knowledge with regard to sustainable mobility. Furthermore, this research was partially sponsored by the Future and Emerging Technologies Unit of EC (IST priority 6th FP), under contract no. FP6-021235-2 (ARRIVAL).

References

A. Abdelghany, G. Ekollu, R. Narasimhan, and K. Abdelghany. A Procative Crew Recovery Decision Support Tool for Commercial Airlines during Irregular Operations. *Annals of Operations Research*, 127:309–331, 2004.

A. Caprara, M. Fischetti, and P. Toth. A Heuristic Method for the Set Covering Problem. *Operations Research*, 47:730–743, 1999.

A. Caprara, L. G. Kroon, M. Monaci, M. Peeters, and P. Toth. Passenger Railway Optimization. In C. Barnhart and G. Laporte, editors, *Transportation*, volume 14, pages 129–187. Elsevier, 2007.

J. Desrosiers and M. E. Lübbecke. A Primer in Column Generation. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*. Springer, New York, 2005.

M. L. Fisher. The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science*, 27:1–18, 1981.

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

D. Huisman. A column generation approach to solve the crew re-scheduling problem. *European Journal of Operational Research*, 180:163–173, 2007.

D. Huisman, L. G. Kroon, R. M. Lentink, and M. J. C. M. Vromans. Operations Research in passenger railway transportation. *Statistica Neerlandica*, 59:467–497, 2005.

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

J. Jespersen-Groth, D. Potthoff, J. Clausen, D. Huisman, L. G. Kroon, G. Maróti, and M. Nyhave Nielsen. Disruption Management in Passenger Railway Transportation. Technical Report EI 2007-05, Erasmus University Rotterdam, 2007.

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

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

C. P. Medard and N. Sawhney. Airline crew scheduling from planning to operations. *European Journal of Operational Research*, 183:1013–1027, 2007.

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

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

A.-S. Pepin, G. Desaulniers, A. Hertz, and D. Huisman. Comparison of heuristic approaches for the multiple depot vehicle scheduling problem. *Journal of Scheduling*, (DOI 10.1007/s10951-008-0072-x), 2008.

E. Prescott-Gagnon, G. Desaulniers, and L.-M. Rousseau. A Branch-and-Price-Based Large Neighborhood Search Algorithm for the Vehicle Routing Problem with Time Windows. Technical Report G-2007-67, Les Cahiers du GERAD, 2007.

N. J. Rezanova and D. M. Ryan. The Train Driver Recovery Problem - a Set Partitioning Based Model and Solution Method. Technical Report IMM-2006-24, Technical University of Denmark, 2006.

S. Ropke and D. Pisinger. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, 40:455–472, 2006.

M. Stojković, F. Soumis, and J. Desrosiers. The Operational Airline Crew Scheduling Problem. *Transportation Science*, 32:232–245, 1998.

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

G. Yu, M. Argüello, S. Gao, S. M. McCowan, and A. White. A New Era for Crew Recovery at Continental Airlines. *Interfaces*, 33:5–22, 2003.