Railway Crew Rescheduling with Retiming

Lucas P. Veelenturf[1,*], Daniel Potthoff[2], Dennis Huisman[2,3], Leo G. Kroon[1,3]

[1] Rotterdam School of Management and ECOPT,
Erasmus University Rotterdam, P.O. Box 1738, NL-3000 DR Rotterdam,
The Netherlands
[2] Econometric Institute and ECOPT,
Erasmus University Rotterdam, P.O. Box 1738, NL-3000 DR Rotterdam,
The Netherlands
[3] Department of Logistics, Netherlands Railways,
P.O. Box 2025, NL-3500 HA Utrecht,
The Netherlands

ECONOMETRIC INSTITUTE REPORT EI 2009-24

September 15, 2009

Abstract

Railway operations are disrupted frequently, e.g. the Dutch railway network experiences about three large disruptions per day on average. In such a disrupted situation railway operators need to quickly adjust their resource schedules. Nowadays, the timetable, the rolling stock and the crew schedule are recovered in a sequential way. In this paper, we model and solve the crew rescheduling problem with retiming. This problem extends the crew rescheduling problem by the possibility to delay the departure of some trains. In this way we partly integrate timetable adjustment and crew rescheduling. The algorithm is based on column generation techniques combined with Lagrangian heuristics. In order to prevent a large increase in computational time, retiming is allowed only for a limited number of trains where it seems very promising. Computational experiments with real-life disruption data show that, compared to the classical approach, it is possible to find better solutions by using crew rescheduling with retiming.

1 Introduction

Passenger railway operations face unforeseen events like infrastructure malfunctions, accidents or rolling stock breakdowns every day. As a consequence of these events, parts of the railway infras-

[*]Corresponding author. E-mail: lveelenturf@rsm.nl, Phone: +31 10 4081567

tructure may be unavailable temporarily. Therefore, it may not be possible to operate the timetable as planned. Jespersen-Groth et al. (2007) describe the disruption management process as the accomplishment of three interconnected tasks: Timetable adjustment and rolling stock and crew rescheduling. Because of their complexity and the limited time available for decision making, these steps are carried out sequentially. First, an adjusted timetable is constructed by canceling, delaying or rerouting trains. In the next two steps it is checked whether modified rolling stock and crew schedules compatible with the adjusted timetable can be found. However, if during rolling stock or crew rescheduling no rolling stock or crew for a task of the adjusted timetable can be found, another iteration through the three steps is necessary. Then a different timetable, where some trains run on different times or are canceled, is needed.

An infeasibility of the crew rescheduling step suggests a new adjusted timetable where additional trains are canceled. If this is compatible with the rolling stock schedule this would be a solution. In this paper we will show that sometimes other solutions exist where no additional trains need to be canceled if the departures of some trains are just delayed by a couple of minutes. Given the high time pressure, always present during disruptions, and the complexity of the problems, dispatchers may or may not find the latter solutions. It is quite clear however, that up to 1,000 passengers waiting for a train on a busy station during peak hours would appreciate such solutions.

Recently, Operations Research (OR) models have been developed for the different tasks in railway disruption management. The paper of Nielsen (2008) deals with rolling stock rescheduling. Rezanova (2009) and Potthoff et al. (2008) present models and solution approaches for crew rescheduling. Clearly, integration of the different steps is beneficial. Walker et al. (2005) presents a model that does timetabling and crew rescheduling at the same time. However, because of the limited computation time that is available and the high detail of the timetabling decisions, integration seems reaching too far for large scale problems at this point in time.

In this paper, we look at an extension of the crew rescheduling problem, where some timetabling decisions are integrated into crew rescheduling. More precisely, the departure of trains may be delayed. This gives more flexibility to the third step in the disruption management process and may avoid undesired iterating. Moreover, this new approach is able to provide high quality solutions from a service level point of view.

The first contribution of this paper is a new formulation for crew rescheduling with retiming, where retiming options are modeled as discrete choices. Moreover, we show how to adapt the solution approach of Potthoff et al. (2008) in order to keep the increase in computation time for the extended model moderate. We evaluate our approach using real life data from Netherlands Railways (NS), the largest passenger railway operator in the Netherlands. Finally, we show that retiming allows to find better solutions compared to crew rescheduling without retiming.

The remainder of this paper is organized as follows. A problem description is provided in Section 2. The existing literature is reviewed in Section 3. In Section 4 we present the mathematical formulation. Our solution approach is discussed in Section 5. Computational results are presented

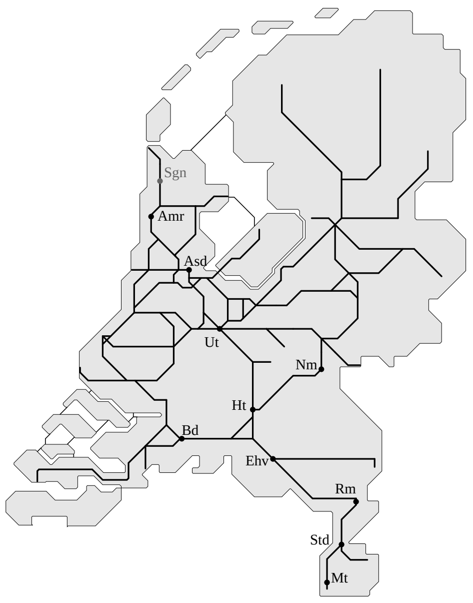

Figure 1: Part of the Dutch railway network used by NS

in Section 6. In Section 7 we draw some conclusions and give some recommendations for further research.

2 Problem description

We first introduce some railway terminology which is necessary to clearly describe the problem. Most of the services offered by passenger railway operators are regular service trips (commonly known as trains) on specified lines according to a published timetable. A line is determined by a start and an end station and a number of intermediate stops. NS operates all lines with a frequency of once or twice per hour. An example of such a line is the 800-line between Maastricht (Mt) and Alkmaar (Amr) with 13 intermediate stops. In the rush hours the 800-line is extended to a line between Maastricht and Schagen (Sgn) with 15 intermediate stops.

In order to operate the timetable, trains are split into trips between *relief points*. A relief point is a station where a driver can switch from one rolling stock unit to another. The work that must be performed by crew members is divided into *tasks*. Several tasks may correspond to the same trip. E.g. the task for driving the train, as well as one or more tasks for conductors. Note, that in this paper we will limit ourselves to train drivers.

The relief points on the 800-line are Maastricht, Sittard (Std), Roermond (Rm), Eindhoven (Ehv), ’s-Hertogenbosch (Ht), Utrecht (Ut), Amsterdam (Asd) and Alkmaar (see Figure 1). Note that the end station during peak hours, Schagen, is not a relief point, so the driver has to stay on the rolling stock and drive the next train of the 800-line from Schagen back to Alkmaar. This results in a task from relief point Alkmaar to relief point Alkmaar.

A *duty* is a sequence of tasks which is performed by one crew member on a single day. Duties start and end at the same crew base to ensure that a crew member who starts at a certain crew base will be back at the same crew base after his duty. The set of crew bases is a subset of the set of relief points. Sometimes a duty contains a so called *deadhead task*, which is used for repositioning. A deadhead task means that the driver is not driving the train, but he is a passenger on that train. Another possibility is that the duty contains a *repositioning task*. The repositioning task is comparable with the deadhead task, with the difference that it uses another way of transportation, for example a bus, a taxi or a train of another operator.

The operational crew rescheduling problem (OCRSP) takes an adjusted timetable and modified rolling stock schedule as input and tries to find a replacement duty for every original duty, such that as many tasks as possible of the adjusted timetable are covered. A replacement duty is consisting of an already performed (possibly empty) part of an original duty and a feasible completion. A feasible completion is a sequence of tasks following the already performed part of a duty, resulting in a replacement duty which has to satisfy a number of rules. For NS these rules are given below:

- A replacement duty needs to start and end at the same crew base as the original duty.
- A replacement duty may end up to 60 minutes later than the planned end time of the original duty.
- If, in a replacement duty, two subsequent tasks must be performed on different rolling stock units, a certain minimum transfer time has to be taken into account.
- A replacement duty which is longer than 5 1/2 hours must contain a meal break of at least 30 minutes at a relief point with a canteen. Moreover, the time before and after the meal break must be less than 5 1/2 hours.
- A replacement duty can only perform a task if the driver is qualified for the route and is licensed for the rolling stock type.

Not all original duties in the crew schedule have tasks assigned to them. There exist a number of *reserve* duties, where the driver is on *stand-by* for a specified amount of time at a major station. The purpose of these reserve duties is that they can be utilized during crew rescheduling.

If in a solution to the OCRSP a task cannot be covered by any crew member, it means that no compatible crew schedule for the adjusted timetable has been found. The railway operator has to come up with another adjusted timetable, for which it is possible to find a compatible crew schedule.

The idea of allowing retiming is to evaluate not just one fixed timetable but a number of similar timetables at once. By delaying the departure of some tasks more connections for drivers will become possible and hence more feasible completions may exist. Therefore, it may be possible to find a better crew schedule. Compared to classical crew rescheduling, the objective of the extension with retiming also aims for keeping the amount of delay as small as possible.

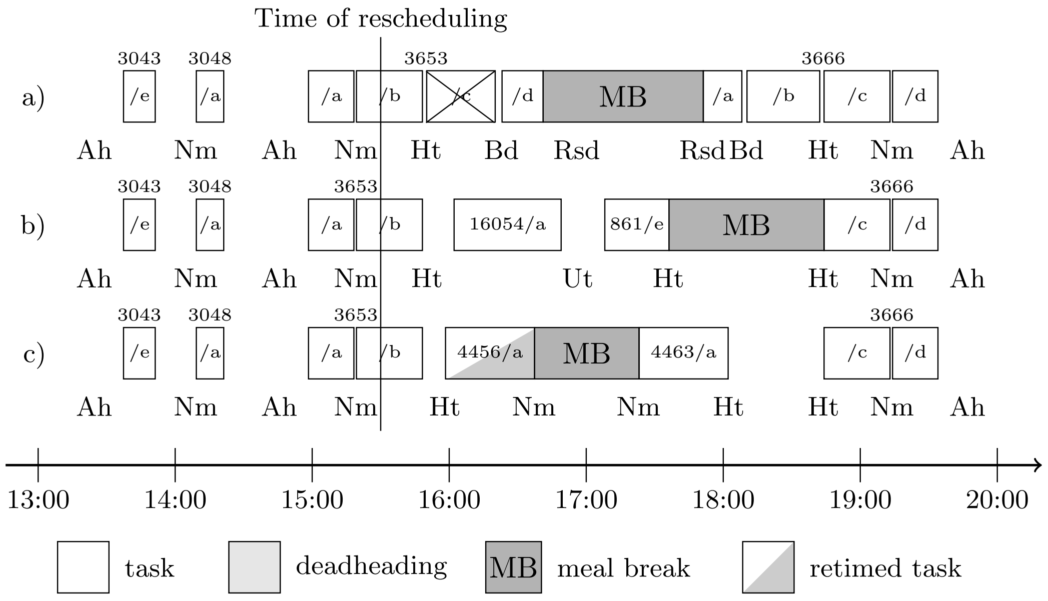

Figure 2: Replacement duties for duty "Ah 114"

In Figure 2.a we show the original duty Ah 114 from crew base Arnhem in case the two southbound routes from 's-Hertogenbosch to Breda and Eindhoven are blocked from 15:30 to 18:30. The duty has started with driving task $3043/e$ (the fifth task of train 3043) from Arnhem (Ah) to Nijmegen (Nm). At 15:30, when the disruption started, the driver has completed his next two tasks and is performing task $3653/c$. The meal break was planned in Roosendaal, thereafter the duty was supposed to end with driving train 3666 from Roosendaal to Arnhem, $3666/a$–$3666/d$. However, due to the route blockage, task $3653/c$ is canceled. Therefore, original duty Ah 114 has become infeasible. A replacement duty is shown in Figure 2.b. Note that because the rescheduling takes place at 15:30, the first four tasks of the duty cannot be changed. After those 4 tasks, the driver arrives in 's-Hertogenbosch at 15:48. If the next task has to be performed on different rolling stock, a minimal transfer time of 10 minutes must be respected. So the replacement duty is allowed to perform task $16054/a$ to Utrecht at 16:02, which is operated with a different rolling stock than task $3653/b$. From Utrecht the driver could go back to 's-Hertogenbosch by driving task $861/e$. After that the duty could end by performing tasks $3666/c$ and $3666/d$ just as in the original duty.

The motivation for allowing retiming is to make replacement duties possible that are not possible in a fixed timetable. For example, the planned departure time of task $4456/a$ is 15:56 and the task is operated by a different rolling stock than task $3653/b$, which means that due the minimum transfer time a transfer between task $3653/b$ and task $4456/a$ is only allowed if the latter task is delayed by more than 2 minutes. In Figure 2.c we show a replacement duty, not feasible without retiming, where tasks $4456/a$ is delayed by 2 minutes.

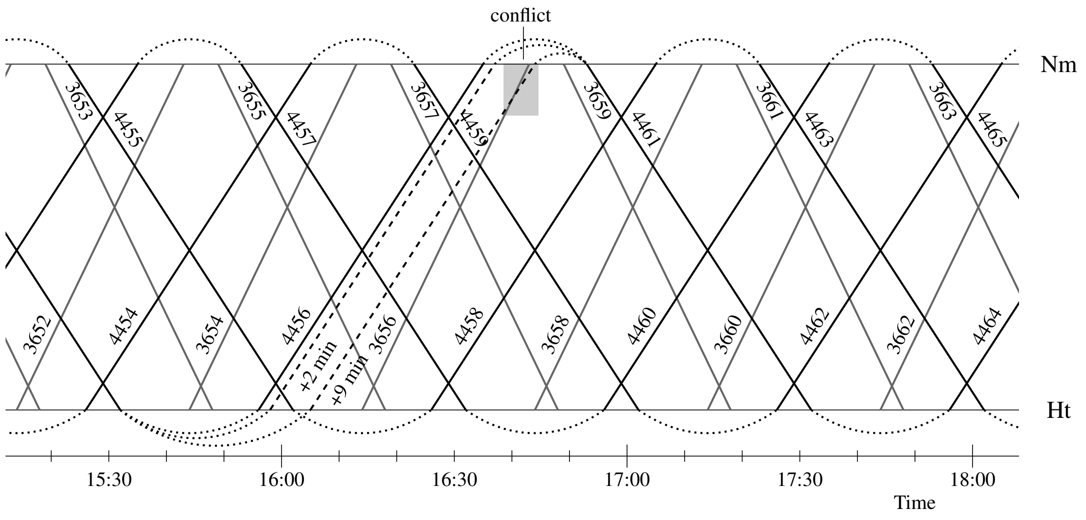

Figure 3: An example of a delayed task between ’s Hertogenbosch (Ht) and Nijmegen (Nm)

Modeling flexibility of departure times in a railway timetable is far from trivial due to a large number of interdependencies. Throughout the paper we will assume that: (i) A delayed departure of a task by χ minutes leads to a delayed arrival of the task by χ minutes. (ii) A delayed task does not affect other tasks using different rolling stock.

The first assumption is not always true in practice. On the one hand, the planned running time for a task may include some buffer time that could be utilized to (partly) absorb delays. On the other hand, a task that is running later than planned could experience an additional delay due to conflicts with other trains. For example, it might be possible that a delayed train has to wait at a signal in a station area. Conversely, a delayed task may affect other trains. A faster train may, for example get stuck behind a slower delayed train. Figure 3 shows part of the 2007 timetable for the route between ’s-Hertogenbosch and Nijmegen. Two lines use this route, the 3600 intercity line from Roosendaal to Arnhem and the 4400 regional line from ’s-Hertogenbosch to Nijmegen. If the departure of the regional train 4456 is delayed by e.g. 9 minutes, it still departs before the intercity train 3656. As indicated in the figure the faster intercity train 3656 catches up with the delayed regional train 4456. This causes a conflict in the timetable. If overtaking on the last part of the route is not possible, the intercity train will be stuck behind the regional train and experience a delay. This example shows that assumption (ii) does not always hold, however at this point in time it seems reasonable since the objective of this paper is to analyze the potential retiming in crew rescheduling might offer. However, if train 4456 would be delayed by 2 minutes, assumptions (i) and (ii) hold.

We would like to take into account the special structure of railway timetables and rolling stock schedules. Since the dwell times at intermediate stations are quite small, it seems likely that a delay of a task will propagate due to the rolling stock schedule. In Figure 3 the rolling stock schedule is indicated by dotted lines. The rolling stock assigned to task ’s-Hertogenbosch- Nijmegen or Nijmegen - ’s-Hertogenbosch of the 3600 line is also assigned to the subsequent task from Nijmegen to Arnhem and ’s-Hertogenbosch- Breda respectively. If the train 3658 would arrive 5 minutes late in ’s-Hertogenbosch, where its dwell time is 2 minutes, it is impossible that train 3658 can depart from ’s-Hertogenbosch towards Arnhem on time.

3 Literature review

While Walker et al. (2005) was the first paper looking at railway crew rescheduling, in the domain of airlines, crew rescheduling received the first attention much earlier in Johnson et al. (1994). Note that crew rescheduling is also know as crew recovery. For a recent review of literature on airline crew rescheduling we refer the interested reader to Clausen et al. (2009). Stojković and Soumis (2001) and Abdelghany et al. (2004) are the first papers that extend crew rescheduling by the possibility to retime flights.

In Stojković and Soumis (2001) some flights may be delayed within specified time windows while new duties for pilots are generated simultaneously. The problem is formulated as a multi-commodity network flow problem with time windows and *flight precedence constraints*. The purpose of the flight precedence constraints is to ensure that minimum turnaround times in the underlying aircraft rotations are not violated and to keep important passenger connections. The problem is separable per pilot and is solved with a branch-and-price algorithm.

The formulation of Stojković and Soumis (2001) is extended to the multi-crew case in Stojković and Soumis (2005). In the multi-crew case every flight has to be covered by exactly ν crew members. This is achieved by deriving ν tasks per flight which need to be covered exactly once. Again the departure time of some flights may be chosen within a time window. *Same departure time* constraints are added to the formulation to make sure that the same delay is chosen for all tasks derived form a flight. Two options are presented in order to deal with flights that cannot be covered ν times. In one option covering less than ν tasks is accepted while in the second option either all ν or non of the tasks derived form a flight may be covered. As in Stojković and Soumis (2001) the problem is solved with a branch-and-price algorithm using specialized branching decisions.

Abdelghany et al. (2004) present a rolling approach for multi-crew rescheduling with retiming of flights. The approach tries to resolve as many conflicts as possible in crew duties, that occur during irregular operations. In a preprocessing step flights from duties with conflicts and flights from selected candidate crews are divided into sets of resource independent flights, each leading to a recovery stage. Flights are resource independent if they cannot appear in a resource schedule together. In the rolling approach the recovery stages are tackled in increasing order of time. For each recovery stage an assignment problem with additional continuous variables for the departure times is solved with a Mixed Integer Programming solver. In the model every flight has three crew positions. Additional constraints enforce that neither duty limits nor connection times are violated. The model allows to assign less than three crew members to a flight, which means that the flight will be an open flight in the final solution.

Abdelghany et al. (2008) present an integrated approach to recover the flight schedule, aircraft and crew at the same time. The overall approach follows Abdelghany et al. (2004) but the Mixed Integer Program for the recovery stages is extended to deal with different resources, namely aircraft, pilots and flight attendants. Either the required number of resource units per type has to be assigned to a flight, or no resource units at all. The latter means that the flight is canceled. Moreover, qualification constraints are added. The pilot must, for example, be qualified for the assigned aircraft type.

Crew scheduling with flight retiming in the planning phase is discussed by Klabjan et al. (2002). Mercier and Soumis (2007) introduce an integrated aircraft routing, crew scheduling and flight retiming model.

Walker et al. (2005) present a model for simultaneous railway timetable adjustment and crew rescheduling. A timetabling part where the departure of tasks can be chosen within time windows is linked to a crew scheduling part where generic driver shifts are chosen. Here a generic driver shift is a sequence of tasks that is feasible with respect to the start and end locations of consecutive tasks. Shift length and task (piece-of-work) sequencing constraints ensure that the departure times are chosen such that only the break rule may be violated in the selected shifts. Breaks are added into the shifts during the branching process. A conflict free timetable could be achieved by adding an enormous number of train crossing and overtaking constraints. The authors propose to relax these constraints in the initial model and to resolve violations by branching on the waiting decisions between involved train pairs.

Recently, Rezanova and Ryan (2009) and Rezanova (2009) presented a solution approach for railway crew rescheduling under the assumption that the timetable is fixed. The problem is formulated as a set partitioning problem and possesses strong integer properties. The proposed solution approach is therefore a depth-first search in a branch-and-price tree. The problem is first initialized with a very small disruption neighborhood, which contains only duties that cover delayed, canceled or re-routed tasks and is limited by a recovery period. As long as constraints are uncovered, while solving the LP-relaxation, the disruption neighborhood is extended by either adding more duties to the problem or by extending the recovery period. In order to deal with new information becoming available the author(s) propose to use the crew rescheduling algorithm in a rolling time horizon approach similar to the one proposed by Nielsen (2008) for rolling stock rescheduling.

Also the paper of Potthoff et al. (2008) deals with railway crew rescheduling without retiming. The problem is modeled in a similar way as in Rezanova and Ryan (2009) and Rezanova (2009) with the difference that tasks may be covered by more than one duty. The presented solution approach uses a heuristic combining Lagrangian relaxation and column generation to explore core problems. First, an initial core problem containing the infeasible duties and some candidate duties is solved with the heuristic. If tasks cannot be covered, new core problems representing the neighborhood of an uncovered task are explored.

4 Mathematical formulation

In this section, we formulate the operational crew rescheduling problem with retiming as an integer linear program. Therefore, we will first introduce some notation. We will use copies of tasks to represent the retiming possibilities, as proposed by Mercier and Soumis (2007). The copies differ from each other in their departure and arrival times. Using copies of tasks limits the retiming possibilities since the departure time cannot be chosen continuously and the retiming possibilities of a task must be determined beforehand.

Denote by N, indexed by i, the set of tasks. Let s_i^{dep} (s_i^{arr}) denote the departure (arrival) station of task i. The planned departure and arrival time are given as t_i^{dep} and t_i^{arr}, respectively. The minimum required dwell time after task i is w_i. Moreover, for every task $i \in N$ a penalty f_i is defined for not covering it. Furthermore, we derive a number of copies $e \in E_i$ for every task $i \in N$. E_i contains at least the copy representing the planned departure time of task i. Denote by $N^c \subseteq N$ the tasks i for which $|E_i| \geq 2$. E is the union of all sets E_i. With $i(e)$ we refer to the source task copy e is derived from. With every copy $e \in E$ we associate the delay d_e compared to the planned departure time $t_{i(e)}^{\text{dep}}$ and a cost parameter g_e representing the penalty for the delay. Furthermore, let $B_e = \{e' \in E_{i(e)} \mid d_{e'} \leq d_e\}$.

If two tasks i and j are operated after each other in the same rolling stock duty, there is a minimum turnaround time u_{ij}. Note that the turnaround time is 0 if the same rolling stock composition is continuing in the same direction. Let $h_{ij} = \max(w_i, u_{ij})$ be the minimum time that is needed after the arrival of task i before the rolling stock is available for task j. Let $\hat{L}_e = \arg\min_{f \in E_j}\{d_f \mid (t_j^{\text{dep}} + d_f) - (t_i^{\text{arr}} + d_e) \geq h_{ij}\}$ be the copy representing task j with the smallest delay such that the rolling stock is available if copy e would be used for task i. Furthermore, let

$$L_e = \begin{cases} \emptyset & \text{if } \exists\, e' \in E_{i(e)} \text{ with } d_{e'} > d_e \text{ and } \hat{L}_e = \hat{L}_{e'}, \\ \hat{L}_e & \text{otherwise.} \end{cases} \tag{1}$$

$\Delta = \Delta_A \cup \Delta_R$ is the set of unfinished original duties, where Δ_A are active duties and Δ_R are stand-by duties. Let K^δ be the set of all feasible completions for duty $\delta \in \Delta$. With every feasible completion $k \in K^\delta$ we associate cost c_k^δ and binary parameters a_{ik}^δ and b_{ek}^δ. a_{ik}^δ is equal to 1 if feasible completion k for duty δ is qualified to drive task i and 0 otherwise. b_{ek}^δ is equal to 1 if feasible completion k for duty δ is making use of copy e and 0 otherwise. Note that b_{ek}^δ is 1 if feasible completion k is using copy e for deadheading.

Let x_k^δ and v_e be binary variables indicating if feasible completion k, or respectively copy e are chosen $x_k^\delta = 1$ ($v_e = 1$), or not $x_k^\delta = 0$ ($v_e = 0$). Furthermore, we introduce a binary variable z_i for every task $i \in N$. If task i is canceled, z_i will be set to 1, or otherwise z_i will be set to 0.

We can now formulate the operational crew rescheduling problem with retiming (OCRSPT) as

$$\begin{aligned}
\min \quad & \sum_{\delta\in\Delta}\sum_{k\in K^\delta} c_k^\delta x_k^\delta + \sum_{i\in N} f_i z_i + \sum_{e\in E} g_e v_e && && (2)\\
\text{s.t.} \quad & \sum_{\delta\in\Delta}\sum_{k\in K^\delta} a_{ik}^\delta x_k^\delta + z_i \geq 1 && \forall i \in N && (3)\\
& \sum_{k\in K^\delta} x_k^\delta = 1 && \forall \delta \in \Delta && (4)\\
& |\bar{\Delta}| v_e - \sum_{\delta\in\Delta}\sum_{k\in K} b_{ek}^\delta x_k^\delta \geq 0 && \forall e \in E && (5)\\
& \sum_{e\in E_i} v_e + z_i = 1 && \forall i \in N && (6)\\
& z_i + \sum_{e'\in B_e} v_{e'} - \sum_{f\in L_e} v_f \geq 0 && \forall i \in N^c, \forall e \in E_i && (7)\\
& x_k^\delta \in \{0,1\} && \forall \delta \in \Delta, \forall k \in K^\delta && (8)\\
& v_e \in \{0,1\} && \forall e \in E && (9)\\
& z_i \in \{0,1\} && \forall i \in N && (10)
\end{aligned}$$

We will refer to Model (2)–(10) as OCRSPRT$_1$. In the objective function (2) the deviation from the planned crew schedule, the penalties for canceling tasks, and the penalties for delays are minimized. Constraints (3) ensure that every task is either assigned to one or more qualified drivers, or is canceled. By (4) exactly one feasible completion must be selected for every original duty. Constraints (5) make sure that the binary variable v_e is set to 1 if copy e is used in any selected feasible completion. That only one copy per task may be used is modeled by Constraints (6). Moreover, these constraints guarantee that deadheading on tasks which are canceled is not possible. Constraints (7) model that the rolling stock must be idle for at least h_{ij} time units if task j is operated directly after task i by the same rolling stock. We assume that stand-by rolling stock may be used if necessary. If a task is canceled, the next task in the rolling stock duty will be served by stand-by rolling stock and may therefore depart at every possible departure time. Note that Constraints (7) are only required for tasks for which multiple copies are derived. Essential in Constraints (7) is that $|L_e| \leq 1$. If $|L_e| \geq 2$ we must use a Constraint (7) for every $f \in L_e$.

We will give an example to illustrate how the sets L_e and B_e interact in Constraint (7). Therefore we consider train 3552 from Eindhoven to Hoofdorp Shunt Yard (Hfdo) which consists of three tasks assigned to the same rolling stock schedule. Assume we derive two copies for the first two tasks with 0 and 3 minutes delay respectively. Detailed information about the copies are shown in Table 1. Let us assume that $h_{ij} = 2$ minutes between tasks $i(d)$ and $i(e)$ as well as between $i(d)$ and $i(f)$. Then according to (1): $L_d = \{e\}$, $L_{d'} = \{e'\}$, $L_e = \emptyset$ and $L_{e'} = \{f\}$. The last two result from the fact that the planned idle time in Utrecht is 6 minutes, so even if train 3552 arrives with a delay of 3 minutes in Utrecht, the next task can still depart at the planned time. This is an example where an introduced delay can be absorbed due to margins in the timetable.

Copy	Delay (min)	Origin	Detstination	Departure	Arrival
d	0	Ehv	Ht	14:47	15:06
d'	3	Ehv	Ht	14:50	15:09
e	0	Ht	Ut	15:08	15:37
e'	3	Ht	Ut	15:11	15:40
f	0	Ut	Hfdo	15:43	16:27

Table 1: Example of copies for train 3552 from Eindhoven (Ehv) to Hoofdorp Shunt Yard (Hfdo)

An alternative formulation OCRSPRT$_2$ can be obtained by replacing Constraints (5) in OCRSPRT$_1$ by

$$v_e - \sum_{k \in K^\delta} b^\delta_{ek} x^\delta_k \geq 0 \;\; \forall \delta \in \Delta, \forall e \in E \tag{11}$$

Proposition 1. (11) *implies* (5).

Proof. If $v_e - \sum_{k \in K^\delta} b^\delta_{ek} x^\delta_k \geq 0 \;\; \forall \delta \in \Delta$, then

$$\sum_{\delta \in \Delta} \big(v_e - \sum_{k \in K^\delta} b^\delta_{ek} x^\delta_k \big) \geq 0 \Rightarrow |\Delta| v_e - \sum_{\delta \in \Delta} \sum_{k \in K^\delta} b^\delta_{ek} x^\delta_k \geq 0$$

□

Proposition 2. *The reverse implication of Proposition 1 is not true.*

Proof. E.g. consider $\Delta = \{1, 2, 3\}, E = \{1, 2, 3\}$ and $\sum_{k \in K} b^\delta_{1k} x^\delta_k = 0.5$ for $\delta \in \{1, 2\}$ and $\sum_{k \in K} b^\delta_{1k} x^\delta_k = 0.0$ for $\delta = 3$. Then with $v_1 = 1/3$ Constraint (5) would hold, but Constraint (11) would be violated for $\delta = 1$ and $\delta = 2$. □

Denote by LP$_1$ the linear relaxation of model OCRSPRT$_1$ and by LP$_2$ the linear relaxation of model OCRSPRT$_2$.

Proposition 3. *LP*$_2 \geq$ *LP*$_1$

Proof. The proof follows directly from Proposition 1 and 2. □

Proposition 3 states that using Constraints (11) results in a tighter LP relaxation. However, $|E|$ constraints of type (5) are replaced by $|E||\Delta|$ constraints of type (11). Thus the number of constraints of type (11) is much larger than that of type (5).

After several experiments with the solution approach described in Section 5, we discovered that the approach of model OCRSPRT$_2$ resulted in less uncovered tasks and less retimed tasks than the approach of model OCRSPRT$_1$. In principle the models have the same integer solutions, but since we use an heuristical approach, we do not always find an optimal solution. We also noticed that the problem is solved slower if we use model OCRSPRT$_2$ instead of model OCRSPRT$_1$. However, we will accept the increase in computation time to receive better results. So, in the remainder of this paper we only consider model OCRSPRT$_2$.

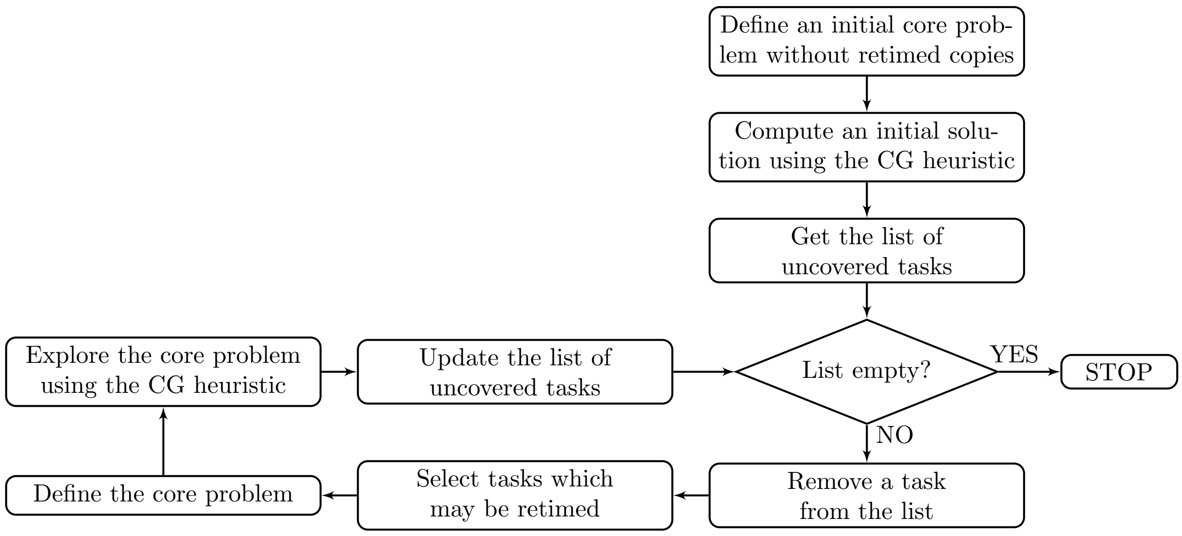
Define an initial core problem without retimed copies
Compute an initial solution using the CG heuristic
Get the list of uncovered tasks
List empty?
YES
STOP
NO
Remove a task from the list
Select tasks which may be retimed
Define the core problem
Explore the core problem using the CG heuristic
Update the list of uncovered tasks

Figure 4: Iterative neighborhood exploration with retiming (INER)

5 Solution approach

On an average workday a crew schedule of NS contains about 1,000 duties for drivers covering in total more than 10,000 tasks. Our aim is to provide solutions of good quality within a couple of minutes of computation time. Therefore, we will not consider all original duties and all tasks, but we will extract core problems containing only a subset of the duties and tasks. Moreover, we will use a Lagrangian heuristic embedded in a column generation scheme very similar to the one proposed by Potthoff et al. (2008). In this paper we will investigate two approaches, which use the same heuristic to explore the core problems, but differ in the way the core problems are defined.

Our first approach is outlined in Figure 4. We first define an initial core problem where retiming is not allowed. A solution for this core problem is computed using the column generation based heuristic. If the computed solution covers all tasks we stop, otherwise we iterate over the uncovered tasks and define one new core problem per uncovered task. We use a neighborhood definition to select the tasks for which we allow retiming and for constructing the core problems. The core problems are explored using the *column generation* (CG) *heuristic* and the list of uncovered tasks is updated. We will refer to this approach as *iterative neighborhood exploration with retiming* (INER). The difference to the approach presented by Potthoff et al. (2008) is that in INER retiming of some tasks is allowed in the neighborhood exploration phase.

Our second method, outlined in Figure 5, does not use an iterative neighborhood exploration. If the solution of the initial core problem contains some uncovered tasks, a second core problem is constructed and solved. This second core problem is an extension of the initial core problem, which is obtained by adding retiming possibilities. In the remainder of this paper we refer to this approach as *extended core problem with retiming* (ECPR).

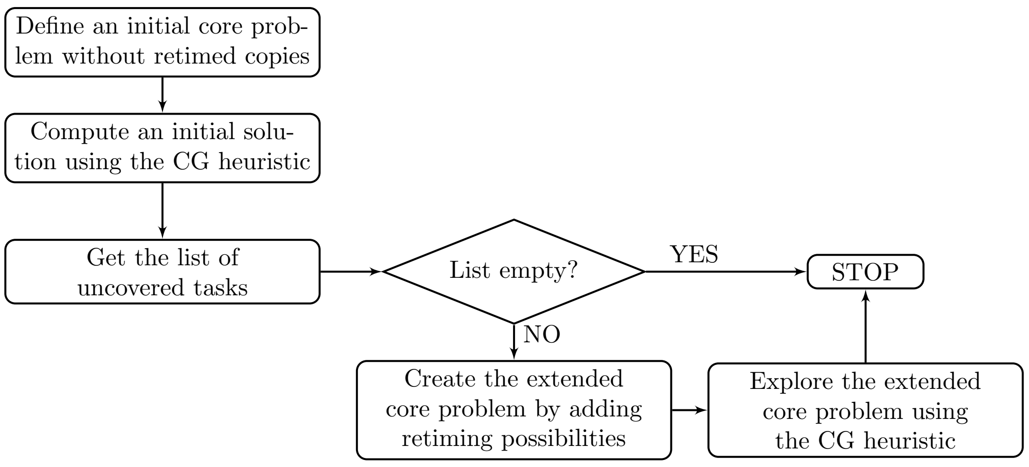
Define an initial core problem without retimed copies
Compute an initial solution using the CG heuristic
Get the list of uncovered tasks
List empty?
YES
STOP
NO
Create the extended core problem by adding retiming possibilities
Explore the extended core problem using the CG heuristic

Figure 5: Extended core problem with retiming (ECPR)

In both approaches INER and ECPR we relax the initial core problems by only using Constraints (3), (4), (8) and (10). Note that in this model it can happen that feasible completions are chosen that contain deadheading on tasks which are canceled. However, in the next core problem which is considered in both approaches, these deadheadings are not allowed anymore and a different solution will be computed. The reason we decided to use the relaxed model in the initial core problem is that the computation time for using OCRSPRT$_2$ is too long compared to the relaxed model.

5.1 Defining the initial core problems

The initial core problems in INER and ECPR are constructed in the same way as in Potthoff et al. (2008). The intention is to select the duties that are affected by the timetable adjustments and to add some duties which contain some tasks close in space and time to the modified tasks.

5.2 Neighborhoods for uncovered tasks in the INER approach

Given an uncovered task, we define a neighborhood which will be extended by retiming possibilities in a subsequent step. First we select a number of candidate duties. These duties can possibly cover the uncovered task. In order to offer some reassignment possibilities we also select a number of similar duties for each candidate duty.

The candidate duties are selected as follows. Given the departure time and station of the uncovered task j we look at the latest task j^- that departs from the same station before task j. Then we consider the replacement duty σ that covers j^- in the current solution and check heuristically, considering the qualification of the driver, if σ could cover j. If yes, then we select σ as a candidate and continue with the previous task that departs from station s_j^{dep} before j^- until we have selected r candidates. We repeat the procedure considering tasks that depart from station s_j^{dep} after task j.

Furthermore, we select the replacement duty which covers task $\hat{\jmath}$, the first task that leaves s_j^{arr} and goes back to station s_j^{dep} 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 s_j^{dep}.

In the next step we select for every candidate the s most similar duties that have not been selected yet. We define similarity between duties in terms of the numbers of stations that are visited around the same time. We refer to Potthoff et al. (2008) for the exact definition.

5.3 Core problems with retiming possibilities

The primary goal of retiming is to enable solutions where less tasks need to be canceled. In order to limit the computational effort we allow retiming only for a subset of the tasks. If we have an uncovered task which starts, for example, at 's Hertogenbosch, this indicates that there is at the start time of the task a shortage of crew in 's Hertogenbosch. By delaying some tasks starting at 's Hertogenbosch, we can probably prevent the shortage. Therefore, we propose the following procedure to determine this subset. Let N^u be the uncovered tasks after solving the initial core problem. Then, for an uncovered task $i \in N^u$ we construct a set N_i^c with tasks that may be retimed as $N_i^1 \cup N_i^2$, where $N_i^1 = \{j \in N \mid s_j^{\text{dep}} = s_i^{\text{dep}}$ and $t_j^{\text{dep}} \in [t_i^{\text{dep}} - p, t_i^{\text{dep}} + p]\}$ and N_i^2 is recursively defined as the set of all tasks which are linked to tasks in N_i^1 or N_i^2.

For INER $N^c = N_i^c$ for the uncovered task i currently under consideration. For the extended core problem in the ECPR approach the tasks that may be retimed are $N^c = \cup_{i \in N^u} N_i^c$.

Let $\ddot{N}$ contain all tasks covered by an original duty in the neighborhood of the uncovered task under consideration when using INER . For the ECPR approach $\ddot{N}$ is the set of tasks of the initial core problem. The core problems are then defined by a subset of the original duties $\bar{\Delta}$ and a subset of the tasks $\hat{N}$. Here $\bar{\Delta} = \{\delta \in \Delta \mid \delta$ is using a task $i \in \ddot{N} \cup N^c\}$ and $\hat{N}$ is the set of all tasks used by at least one original duty $\delta \in \bar{\Delta}$. Note that due to overcovering and deadheading it can happen that for a task $j \in \hat{N}$ not all duties δ using task j are in $\bar{\Delta}$. By definition of $\bar{\Delta}$ retiming is not allowed for these tasks. Denote by $\bar{N} = \{i \in \hat{N} \mid \delta \in \bar{\Delta}\ \forall \delta \in \Delta$ using task $i\}$.

Given $\bar{\Delta}$ and $\bar{N}$ we define $\bar{E} = \cup_{i \in \bar{N}} E_i$. Moreover, we denote by $\bar{K}^\delta$ the set of feasible completions for duty δ which only use tasks $i \in \hat{N}$. The mathematical model for a core problem is obtained by replacing N with $\bar{N}$, Δ with $\bar{\Delta}$, E with $\bar{E}$ and K with $\bar{K}$ in the Formulations (2)–(10), respectively.

5.4 Exploring a core problem

For computing near optimal solutions and lower bounds for the core problems we adapt the heuristic, based on a combination of column generation and Lagrangian relaxation, presented in Potthoff et al. (2008). For an introduction to column generation we refer to Desrosiers and Lübbecke (2005). Let us first describe the building blocks, before we present our column generation based heuristic in Section 5.4.4.

5.4.1 Combining column generation and Lagrangian relaxation

A lower bound for a given core problem can be obtained by Lagrangian relaxation. In this section we will present the details for model OCRSPRT$_2$. We relax Constraints (3), (11) and (7) of the core problems in a Lagrangian fashion using multiplier vectors λ, μ and η, respectively.

For simplicity we introduce $\gamma_e = \sum_{\{d \in \bar{E} \mid e \in \bar{L}_d\}} \eta_d - \sum_{\{d \in \bar{E} \mid e \in \bar{B}_d\}} \eta_d$. Then, the Lagrangian subproblem equals:

$$\begin{aligned} \Theta(\lambda, \mu, \eta) \quad = \quad & \min \sum_{i \in \bar{N}} \lambda_i + \sum_{\delta \in \Delta} \sum_{k \in \bar{K}^\delta} (c_k^\delta + \sum_{e \in \bar{E}} \mu_e^\delta b_{ek}^\delta - \sum_{i \in \bar{N}} \lambda_i a_{ik}^\delta) x_k^\delta + \sum_{i \in \bar{N}} (f_i - \lambda_i - \sum_{e \in \bar{E}_i} \eta_e) z_i \\ & + \sum_{i \in \bar{N}} \sum_{e \in \bar{E}_i} (g_e + \gamma_e - \sum_{\delta \in \bar{\Delta}} \mu_e^\delta) v_e \qquad (12) \\ & \text{s.t. } (4), (6), (8), (9) \text{ and } (10) \end{aligned}$$

For given vectors λ, η and μ, $\Theta(\lambda, \eta, \mu)$ can be calculated with a simple procedure. First, we determine the values for all x_k^δ variables. To ensure that Constraints (4) are not violated, for every duty $\delta \in \bar{\Delta}$ we set x_k^δ equal to 1 for exactly one $k \in \arg\min \{ \bar{c}_k^\delta(\lambda, \eta, \mu) | \, k \in \bar{K}^\delta \}$. Here $\bar{c}_k^\delta(\lambda, \eta, \mu) = (c_k^\delta + \sum_{e \in \bar{E}} \mu_e^\delta b_{ek}^\delta - \sum_{i \in \bar{N}} \lambda_i a_{ik}^\delta)$ is the Lagrangian reduced cost of feasible completion k. The values of the z_i and v_e variables can be determined independently from the x_k^δ variables. The algorithm in Figure 6 determines for every task $i \in \bar{N}$ the values for the variables z_i and v_e ($\forall e \in \bar{E}_i$) such that Constraints (6) are not violated.

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

Figure 6: Algorithm to determine z_i and v_e

The Lagrangian dual problem is to find the best Lagrangian lower bound Θ^*:

$$\Theta^* = \max \Theta(\lambda, \eta, \mu), \quad \lambda \geq 0, \eta \geq 0 \text{ and } \mu \geq 0 \qquad (13)$$

Since the number of feasible completions can be enormous for some original duties, we combine Lagrangian relaxation with column generation. Instead of considering all feasible completions we consider only a subset in a restricted master problem (RMP). Denote by $\bar{K}_n^\delta$ the feasible completions present in the nth RMP. A lower bound Θ_n^* for the nth RMP is obtained by subgradient optimization (see e.g. Fisher (1981); Beasley (1993)).

Let λ^n, η^n and μ^n be the vectors of the Lagrangian multipliers corresponding to Θ_n^*. In the pricing problems of our column generation algorithm we check, per original duty, if feasible completions exist that are not in the RMP, but have lower Lagrangian reduced cost than the feasible completions in the RMP. We will refer to them as *promising* feasible completions. The pricing problems are formulated as shortest path problem with resource constraints (see Section 5.4.3). If promising feasible completions exist we add them to the RMP. Let $p_n^\delta = \min\{\bar{c}_k^\delta(\lambda, \eta, \mu) \mid k \in \bar{K}^\delta\}$ be the solution value of the pricing problem for duty δ and let $r_n^\delta = \min\{\bar{c}_k^\delta(\lambda, \eta, \mu) \mid k \in \bar{K}_n^\delta\}$ be the smallest Lagrangian reduced cost of a feasible completion for duty δ in the nth RMP. After solving the pricing problems for all duties $\delta \in \bar{\Delta}$ we can compute a lower bound for the core problem as $LB_n = \Theta_n^* + \sum_{\delta \in \bar{\Delta}} (p_n^\delta - r_n^\delta)$.

5.4.2 Feasible solutions

Next to a good lower bound, we are especially interested in near optimal feasible solutions. Based on Lagrangian multiplier vectors λ, η and μ we try to generate feasible solutions with a Lagrangian heuristic called *GREEDY* shown in Figure 7.

In procedure *GREEDY*, we select for every duty the best feasible completion. If it is the first time that a certain task appears in a selected feasible completion, the copy which is used for that task, will be the only copy that is allowed to be used in all duties. So after a certain copy for a task is selected, all feasible completions which use another copy of the same task will be ignored. Moreover, we ignore feasible completions which use copies, which would violate the minimum idle time constraints (7). Since every set $\bar{K}_n^\delta$ contains the artificial completion without any copies, it is ensured that for every duty at least one feasible completion is left to select.

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

GREEDY does not always find a feasible solution, however in most cases it will. Only in the extraordinary case that a crew member is assigned to be a passenger on a train which is not covered by a driver, the solution is infeasible. This condition is checked in Line 22.

5.4.3 Solving the pricing problems

For every duty in a core problem, we construct a directed acyclic graph that contains all possible feasible completions. The nodes represent arrivals or departures of copies derived from the tasks. An arc goes from an arrival node to a departure node if it is possible to use the corresponding copies after each other in a feasible completion. Besides the cost, every arc has two additional parameters: A time consumption and a boolean value indicating if the arc can represent a meal break. The problem of finding the path corresponding to the feasible completion with the smallest Lagrangian reduced cost is solved as resourced constrained shortest path problem. For that purpose, we have adapted the generic dynamic programming algorithm presented in Irnich and Desaulniers (2005).

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

Figure 7: Procedure *GREEDY* to construct feasible solutions

5.4.4 The column generation based heuristic

Our column generation based heuristic using the building blocks as described in Section 5.4.1–5.4.3 is outlined in Figure 8. It can be seen as a depth first search in a branch-and-bound tree with column generation in every node. This is a common way of designing column generation based heuristics for crew scheduling problems (see Desaulniers et al. (2001)). In Line 5 a dual solution for the RMP is obtained by Lagrangian relaxation as explained above. Another specialty in our approach is that we generate solutions throughout the algorithm (see Line 6). We denote by UB^* the cost of the best found feasible solution. When solving the pricing problems for the original duties, we do pricing and stop if we have found promising columns for more than *maxPP* of the duties. In Line 9 we use three criteria to decide if we stop column generation in the current node. First, we stop if no columns have been added to the RMP. Second, we stop if Θ^*_n is close to LB_n. As a third criterion we use a maximum number of column generation iterations *maxItCG* to perform in the current node. In the root node, where no feasible completions have been fixed, *maxItGC* $= \infty$, in the other nodes we can use a relatively small number to speed up the algorithm.

After terminating column generation for a node we check in Line 13 if the best feasible solution of value UB^* is close to the lower bound LB_F which is the sum of the fixed part UB_F and the lower bound of the free variables Θ_n^*. If this is the case, we can terminate the algorithm since we know that it is unlikely to find a better feasible solution if we only fix more variables. Otherwise, we fix the feasible completions for more original duties. This is done based on the number of times a feasible completion was set to 1 in the solution of a Lagrangian subproblem during the last subgradient optimization.

```
1  $stopFix =$ false, $LB_F = -\infty, UB^* = \infty, UB_F = 0$;
2  while stopFix = false do
3      $stopColGen =$ false;
4      while stopColGen = false do
5          Compute the lower bound $\Theta_n^*$ for the RMP with subgradient optimization;
6          Call GREEDY with at most maxMV multiplier vectors and update $UB^*$;
7          Solve pricing problems and add promising feasible completions;
8          Compute $LN_n$ if all pricing problems have been solved;
9          if any stopping criteria for column generation is met then
10             $stopColGen =$ true, $LB_F = UB_F + LB_n$;
11         end
12     end
13     if any stopping criteria for fixing is met then
14         $stopFix =$ true;
15     else
16         Fix the feasible completions for at most maxFix original duties and update $UB_F$;
17     end
18 end
```

Figure 8: The algorithm to solve a core problem

6 Computational results

We will evaluate our two new approaches with retiming INER and ECPR on three disruption scenarios, *Ac:1*, *Ht:1*, and *Ztm:1*. These scenarios are based on past real life disruptions. Some information about the scenarios is presented in Table 2. Furthermore, we used a crew schedule from NS that was planned for some workday in September 2007. In order to evaluate the benefits of retiming, we compare our new methods with the method proposed in Potthoff et al. (2008). We will refer to the latter

Location	ID	Time	Type
Abcoude	*Ac:1*	11:00-14:00	two sided blockage, some trains are rerouted
's-Hertogenbosch	*Ht:1*	15:30-18:30	two sided blockage
Zoetermeer	*Ztm:1*	08:00-11:00	reduced number of trains

Table 2: Information about the disruption scenarios

as *iterative neighborhood exploration* (INE). Moreover, we will investigate the effect of considering stand-by duties. For that reason, we determine two cases. In the first case we do not use any stand-by duty and in the second case we use a set of 46 stand-by duties.

All approaches have been implemented in C++. The tests have been performed under Windows XP on a quad core 2.99 GHz CPU machine with 3.25 GB RAM memory. However, only a single core was used in the tests.

6.1 Parameter settings

First of all, we have some settings which are required to determine the core problems. In the definition of N_1 we set $p = 30$ minutes. For every task in $\bar{N}^c$ we derive four copies with delays d_e equal to 0,1, 3 and 5 minutes.

In the column generation based heuristic, we use the following settings. For partial pricing we set *maxPP* $= 0.3$. For calling *GREEDY* we set *maxMV* $= 100$. In the root node of our depth-first search *maxItCG* $= \infty$, in all other nodes we use *maxItCG* $= 10$. Furthermore, *maxFix* was set to 0.05.

6.2 Cost parameters for the objective function

We use the following settings to account for the different aspects in the objective function. First, the cost of changing a duty is set to 400. The cost for sending home a stranded driver by taxi is 3000. Using a task in a feasible completion costs 0 if the corresponding original duty was already covering that task, and 50 otherwise. Moreover, the cost of a transfer is 0 if the transfer was already in any original duty, and 1 otherwise. The usage of new repositioning tasks costs 1,000. The penalty for retiming a task is 200 per minute of delay.

The penalties f_i for canceling task i depend on the characteristic of the task. We say a task is of type A-B if $s_i^{\text{dep}} \neq s_i^{\text{arr}}$ and of type A-A if $s_i^{\text{dep}} = s_i^{\text{arr}}$. We set $f_i = 20,000$ if task i is of type A-B and $f_i = 3,000$ otherwise. This is motivated by the overall disruption management process. If only tasks of type A-A are canceled, the crew schedule is compatible with the underlying rolling stock schedule under the assumption that the rolling stock assigned to the canceled A-A tasks can remain idle at the platform or can be shunted to a nearby shunt yard and pulled out again for its next trip.

	Method	It	$\|\bar{\Delta}\|$	$\|\bar{N}\|$	$\|\bar{E}\|$	LB	UB	Gap	Sol	TT	A-B	A-A	DT	TD
								(%)		(s)				(min)
Ac:1	INE	1	176	629	0	58718	59211	0.8	59211	98	1	0	0	0
Ac:1	INE	2	98	259	0	25324	25324	0.0	59211	110	1	0	0	0
Ac:1	INER	2	115	317	106	31202	31202	0.0	59212	137	1	0	0	0
Ac:1	ECPR*	2	187	670	30*	58718	59116	0.7	59116	493	1	0	0	0
Ht:1	INE	1	126	660	0	61661	61744	0.1	61744	95	1	1	0	0
Ht:1	INE	2	77	391	0	30637	30637	0.0	61694	106	1	1	0	0
Ht:1	INE	3	72	372	0	30450	30450	0.0	61694	118	1	1	0	0
Ht:1	INER	2	87	455	37	17637	17809	1.0	45649	169	0	1	3	9
Ht:1	INER	3	79	413	56	14007	14007	0.0	45649	190	0	1	3	9
Ht:1	ECPR	2	147	835	119	43241	43751	1.2	43751	602	0	1	2	6
Ztm:1	INE	1	117	432	0	51940	51991	0.1	51991	25	2	0	0	0
Ztm:1	INE	2	99	247	0	43264	43264	0.0	51991	36	2	0	0	0
Ztm:1	INE	3	100	301	0	23563	23563	0.0	32339	50	1	0	0	0
Ztm:1	INER	2	133	398	175	5355	5667	5.8	13392	167	0	0	1	3
Ztm:1	ECPR†	2	186	768	185†	11982	12389	3.4	12389	572	0	0	1	3

Table 3: Results with stand-by drivers.

6.3 Numerical results

For the numerical results we use some abbreviations in Tables 3 and 4: "It" is the iteration number of the general solution approach as given in Figures 4 and 5. The costs in the columns "LB" and "UB" respectively are the lower bound on the optimal solution and the cost of the best found solution of the core problem. "Gap" represents the relative difference between the best solution and the lower bound of the core problem. The column "Sol" represents the cost of the total solution: the cost of the core problem ("UB") plus the rescheduling cost of the other duties that were selected in previous iterations, and that are needed to complete the solution. The total computation time in seconds including the current iteration is given in the column "TT". The columns "A-B" and "A-A" represent the number of uncovered tasks for the respective types. The last two columns give information about the used retimed copies. The column "DT" displays the number of delayed tasks and the column "TD" represents the total number of delayed minutes.

With INER and ECPR we solve formulation OCRSPRT$_2$. We compare the results with the INE method. Since the rescheduling model without retiming is used in the initial core problem of all three approaches, the results of the first iteration are the same. Therefore, we report this result only once for the method INE in Tables 3 and 4. A remark must be made that we were not able to use the ECPR approach with $p = 30$ in the definition of N_1 since it ran out of memory. For *Ac:1* (*) we had to set $p = 5$ and for *Ztm:1* (†) we had to set $p = 20$.

In Table 3 we show the results of the three approaches in case we use the 46 stand-by duties and in Table 4 we show the results without using stand-by duties. By using stand-by duties we notice that the ECPR method has in all cases the best solution. However, the computation times of this approach are

	Method	It	$\lvert\bar{\Delta}\rvert$	$\lvert\bar{N}\rvert$	$\lvert\bar{E}\rvert$	LB	UB	Gap	Sol	TT	A-B	A-A	DT	TD
								(%)		(s)				(min)
Ac:1	INE	1	130	629	0	61136	62187	1.7	62187	86	1	0	0	0
Ac:1	INE	2	59	287	0	27235	27235	0.0	62187	97	1	0	0	0
Ac:1	INER	2	79	351	106	34066	34066	0.0	62136	146	1	0	0	0
Ac:1	ECPR*	2	141	670	30*	60967	62390	2.3	62390	539	1	0	0	0
Ht:1	INE	1	90	660	0	65567	65803	0.4	65803	94	1	2	0	0
Ht:1	INE	2	44	407	0	34489	34489	0.0	65803	105	1	2	0	0
Ht:1	INE	3	39	407	0	31080	31080	0.0	65803	115	1	2	0	0
Ht:1	INE	4	40	405	0	29941	29941	0.0	63657	124	1	1	0	0
Ht:1	INER	2	52	454	37	23200	23208	< 0.1	52861	144	0	3	2	6
Ht:1	INER	3	45	439	56	16747	16747	0.0	50364	163	0	2	2	6
Ht:1	INER	4	54	429	82	17812	17812	0.0	46666	205	0	1	2	6
Ht:1	ECPR	2	114	871	157	44502	44660	0.4	44660	641	0	1	2	6
Ztm:1	INE	1	71	432	0	51991	51992	0.0	51992	16	2	0	0	0
Ztm:1	INE	2	54	249	0	42058	42058	0.0	51992	23	2	0	0	0
Ztm:1	INE	3	55	306	0	42159	42159	0.0	51992	36	2	0	0	0
Ztm:1	INER	2	86	390	175	5354	5818	8.7	14046	138	0	0	1	3
Ztm:1	ECPR†	2	140	768	185†	12058	12441	3.2	12441	477	0	0	1	3

Table 4: Results without stand-by drivers.

more than three times longer compared to the other two approaches. In terms of uncovered tasks the INER approach performs the same as ECPR, except for case *Ht:1* where in the solution of INER an additional task is delayed. By delaying at most 3 tasks, both retiming approaches have in cases *Ht:1* and *Ztm:1* less uncovered tasks than the INE approach. In case *Ac:1*, retiming did not result in better crew schedules. However, a remark must be made that the solution of the method INE for *Ac:1* is a crew schedule which is not compatible with the adjusted timetable since it has one driver deadheading on a canceled task.

The uncovered task in *Ac:1* is rerouted due to the disruption and takes half an hour longer. The crew member which was originally assigned to this task does not have the knowledge of the new route and is therefore not allowed to drive this train. This task has to be performed exactly at the moment of rescheduling. Therefore, it is not possible to cover the task without retiming it. Because of the minimum transfer time of 10 minutes, the task must be retimed with at least 10 minutes. The retiming approaches INER and ECPR only use a maximum retiming possibility of 5 minutes and were therefore not able to cover the task. In additional tests in which INER and ECPR also constructed retimed copies of 10 minutes delay, it was still not possible to cover all tasks. It seems to be that the costs of all these delays are not worth it.

If we do not use any stand-by duties (see Table 4), ECPR resulted twice in the best solution and INER found once the best solution. In terms of uncovered tasks and delayed minutes, the methods performed equally well. Except for case *Ac:1*, retiming of at most 2 tasks results in less uncovered tasks. Again the computation time of ECPR is by far the largest and INER has a computation time which is at most 2 minutes longer compared to INE.

We notice that the solutions in which stand-by duties are used have lower costs, but if we only consider the number of uncovered tasks and the number of delayed tasks, it was not necessary to use the stand-by duties. Moreover, for *Ht:1*, the use of stand-by duties has increased the number of delayed tasks.

7 Conclusions and future research

We presented two approaches to solve railway crew rescheduling with retiming. We have compared our new approaches with an approach that does not allow retiming. In 4 out of the 6 cases (*Ht:1* and *Ztm:1*, both with and without stand-by drivers), the new approaches found solutions with less canceled tasks. Moreover, the observed delay that was introduced into the timetable is very small, which makes it likely that those solutions could be implemented in practice. The computation times of the iterative neighborhood exploration with retiming (INER) approach are within a range that should make it applicable within a decision support system for disruption management.

In this paper we have limited ourselves to consider only train drivers. However, in a disrupted situation conductors need to be rescheduled as well. This could be done as in Stojković and Soumis (2005) and Abdelghany et al. (2008) by using multiple tasks per trip that represent roles in the optimization model.

In future work conflicts between trains due to retiming decisions should be taken into account as well. We believe that the presented model and solution approaches could be extended into that direction without sacrificing computation time too much.

Disruption management takes place in a very uncertain environment, e.g. it can only be estimated how long it will take before a broken switch is repaired. This means that, at the point in time when the first rescheduling decisions must be taken, it is not certain how the timetable will be adjusted during the rest of the day. Stochastic or robust optimization models could be used in order to deal with this kind of uncertainty.

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.

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

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

J. Clausen, A. Larsen, J. Larsen, and N. Rezanova. Disruption management in the airline industry - Concetps, models and methods. *Computers & Operations Research*, 2009. doi: 10.1016/j.cor.2009.03.027.

G. Desaulniers, J. Desrosiers, and M. M. Solomon. Accelerating Strategies in Column Generation Methods for Vehicel Routing and Crew Scheduling Problems. In C. C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 309–324. Kluwer, Boston, 2001.

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.

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

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.

V. Johnson, L. Lettovský, G. L. Nemhauser, R. Pandit, and S. Querido. Final report to Northwest Airlines on the crew recovery problem. Technical report, The Logistic Institute, Georgia Institute of Technology, 1994.

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

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

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

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

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

N. J. Rezanova and D. M. Ryan. The train driver recovery problem - A set partitioning based model and solution method. *Computers & Operations Research*, 2009. doi: 10.1016/j.cor.2009.03.023.

M. Stojković and F. Soumis. An Optimization Model for the Simultaneous Operational Flight and Pilot Scheduling Problem. *Management Science*, 47:1290–1350, 2001.

M. Stojković and F. Soumis. The operational flight and multi-crew scheduling problem. *Yugoslav Journal of Operations Research*, 15:25–48, 2005.

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.