A Column Generation Approach to solve the Crew Re-Scheduling Problem

Dennis Huisman[1,2]

[1] Erasmus Center for Optimization in Public Transport (ECOPT) &,
Econometric Institute, Erasmus University Rotterdam, P.O. Box 1738,
NL-3000 DR Rotterdam, The Netherlands

[2] Department of Logistics, NS Reizigers,
P.O. Box 2025, NL-3500 HA Utrecht, The Netherlands

E-mail: huisman@few.eur.nl

Econometric Institute Report EI2005-54

Abstract

When tracks are out of service for maintenance during a certain period, trains cannot be operated on those tracks. This leads to a modified timetable, and results in infeasible rolling stock and crew schedules. Therefore, these schedules need to be repaired. The topic of this paper is the rescheduling of crew.

In this paper, we define the Crew Re-Scheduling Problem (CRSP). Furthermore, we show that it can be formulated as a large-scale set covering problem. The problem is solved with a column generation based algorithm. The performance of the algorithm is tested on real-world instances of NS, the largest passenger railway operator in the Netherlands. Finally, we discuss some benefits of the proposed methodology for the company.

Keywords: transportation, railways, crew re-scheduling, large-scale optimization, column generation

1 Introduction

Every day, there are a lot of minor modifications in the timetable of a railway company due to some extra trains (charter trains, extra trains for sport events etc.) or speed limitations on some parts of the infrastructure due to track

maintenance. In general, these minor modifications have no influence on the rolling stock and crew schedules. However, in the Netherlands (as well as in some other intensively used European railway networks), large parts of the network are temporarily out of service due to large maintenance projects or construction works on the extension of the network. This mainly happens during weekends, and sometimes on weekdays as well. Figure 1 depicts the railway network in the Netherlands. For each line, it shows the number of weekends in the year 2005 that the line is out of service.

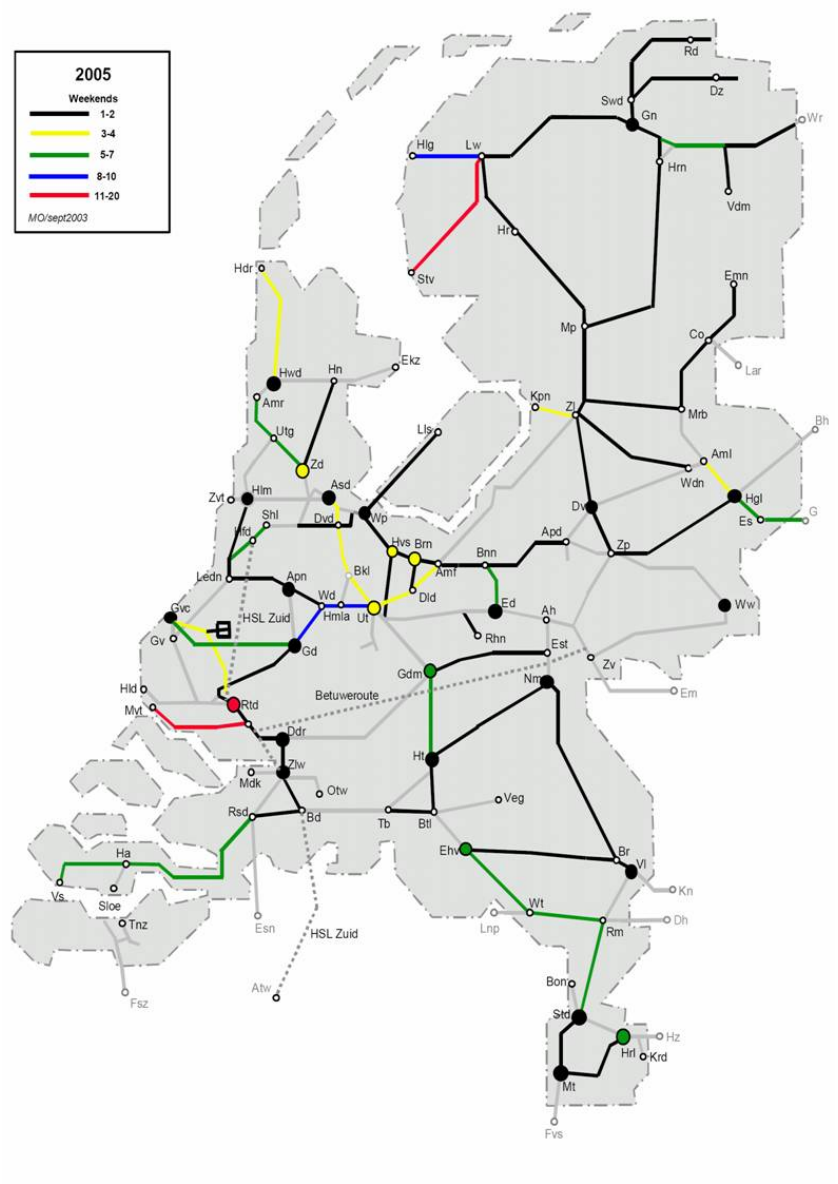

Figure 1: The number of weekends in 2005 when lines are out of service

For instance, if we look at the line Amsterdam (Asd) - Utrecht (Ut), 4

weekends a year the line is out of service. This is due to the quadrupling of this heavily used line. In those weekends, there are no trains possible on this line, which leads to modifications in the timetable. Some north-south Intercity services are rerouted via Hilversum (Hvs) and get different arrival and departure times. Other trains are canceled and do not drive at all or on only a part of their route. In this example, the Intercity service between Nijmegen (Nm) and Den Helder (Hdr) is split into two services, Nm - Ut and Asd - Hdr. The rolling stock and the crew need to be rescheduled then as well. For instance, a duty from the crew base Nm, which normally consists of a driving task Nm - Hdr and back, is not possible anymore and should be changed. For this example, in total, about 14% of the original duties becomes infeasible. It can be even worse if several lines in the network are out of service on the same day. This happens regularly in weekends in Spring and Autumn, since construction works are usually scheduled during these periods. Due to weather conditions in Winter and holidays in Summer, construction works are usually not preferred in these periods.

As can be seen from the example above, rolling stock and crew schedules can be completely destroyed, even if the modifications in the timetable are only local. So the rolling stock circulation and the crew schedules need to be repaired. In this paper, we look at the rescheduling of crew. In fact, we even limit ourselves to the rescheduling of train drivers. However, the problem is more or less similar for conductors.

Although crew scheduling is one of the most successful OR applications in the railway industry (see e.g. Abbink *et al.* (2005); Fores *et al.* (2001); Kohl (2003); Kroon & Fischetti (2001) for applications at several railway companies), to our knowledge, no research has been done yet on the rescheduling of crew. Until now, most focus has been on constructing crew schedules from scratch. The Crew Re-Scheduling Problem (CRSP) is different in the sense that an initial crew schedule has already been given, but should be modified since it is not feasible anymore due to changes in the underlying timetable and/or rolling stock schedule. Since the crews have already been hired, the usual objectives like minimizing crew costs do not make much sense anymore. The problem is more a matter of finding a feasible schedule given the amount of crew available with as few modifications as possible.

Before we give a detailed description of the CRSP in Section 3, we first describe the crew scheduling process at NS (Section 2). NS is the largest passenger railway operator in the Netherlands and provides services on 90% of the railway network. In Section 4, a mathematical model is presented. Afterwards, in Sections 5 and 6, we discuss an algorithm based on column generation techniques to solve the CRSP. In Section 7, we discuss some computational experiments on real-world cases of NS. The paper is concluded with some final remarks on the advantages of the suggested approach for NS and some possible directions for future research (Section 8).

2 Crew scheduling at NS

For an overview on all planning problems that play a role at a passenger railway operator, we refer to Huisman *et al.* (2005b). In this section, we limit ourselves to the crew scheduling process at NS.

At NS, the crew scheduling process has been split in two stages. First, the crew schedules for the annual plan are constructed. This plan deals with a general Monday, Tuesday and so on. There are about 6 times a year modifications in this annual plan. The schedules for the individual days, e.g. Friday December 2, 2005, are based on the schedules for the corresponding general day (in the example the Friday plan). The specific changes in the timetable and rolling stock schedule for a particular day leads to a modified crew schedule for this day. As mentioned before, this is mainly due to lines that are (partly) out of service on a particular day for maintenance or renewal.

The crew scheduling package Turni is used to construct the standard crew schedules for the annual plan (Abbink *et al.* , 2005). Until now, the modified schedules for the so-called daily plans are still constructed manually. That means that a planner modifies the current schedule with pencil and paper. In general, the planner has about two weeks to construct a new schedule.

To get a feeling of the size of the crew scheduling problems at NS, we report some numbers. On a weekday, about 5,000 timetabled trips are scheduled (on Saturdays and Sundays slightly less). To operate these trips about 2,800 rolling stock units are used, while there are about 3,000 drivers and 3,500 conductors employed. These crew members operate from 29 different crew bases spread over the country.

The complexity of the crew scheduling problems at NS is not only influenced by its size, but also in the rules that the crew schedules have to satisfy. Therefore, we briefly describe some rules in the standard (annual) crew scheduling problem. The first set of rules is defined at the level of the individual duties, for instance the length of every duty should not exceed the maximum spread time, there is a meal break in a duty with a certain minimum length, and so on. Moreover, there are requirements on the complete set of duties of each crew base. Hereby, one can think of a maximum average working time for all drivers and a fair division of the work over the bases. The latter constraints are very typical for the Dutch situation and are known as "Sharing Sweet & Sour" rules. They aim at allocating the popular and the unpopular work as fairly as possible among the different crew bases. For instance, some routes are more popular than others and Intercity trains are preferred over regional trains. For a detailed description of these rules, we refer to Abbink *et al.* (2005).

Due to these complex "Sharing Sweet & Sour" rules, a duty is a kind of tour through the railway network. As a result, a modification in the timetable in a part of the network leads to a complex rescheduling problem.

For instance, if there are no trains between two stations A and B, and in a small subset of the duties there are only tasks from A to B and back, while in the other duties there no tasks between A and B at all, the rescheduling problem is completely trivial. The first set of duties can be dropped, while the second set remains the same. However, at NS a similar situation is very complex since in many duties there are tasks from A to B or vice versa, while these tasks are possibly in a chain like e.g. A-C, C-B, B-A, A-C, C-A, A-B, B-C, C-A. That means that simply removing the tasks B-A and A-B leads to an infeasible duty. So much more tasks and duties than only the directly affected ones should be rescheduled as well.

3 Problem description

Before we give a definition of the CRSP, we first introduce some terms. A *task* is the smallest amount of work that has to be assigned to one driver. A task starts and ends at a *relief location*, which is either a crew base or another location where a change of driver is allowed. At most relief locations, a canteen is available, such that it is possible for the crew to have a meal break. A sequence of tasks on the same rolling stock unit is called a *piece (of work)*. A *duty* consists of one or more pieces of work with a minimum *connection time* between them. It is allowed that a driver travels as passenger on a certain train, such a task is called a *passenger task*. The minimal required connection time before a passenger task is, in general, shorter than for a regular task. Finally, there are several rules which each duty has to fulfill:

- A duty starts with a sign-on time, which depends on the type of the first task (i.e. regular task, passenger task, etc.), at the base of the crew member.
- A duty ends with a sign-off time at the base of the crew member.
- Each piece in a duty should not exceed a maximum length.
- A duty should not exceed a maximum length according to the collective labor agreement. This maximum length depends on the start and/or the end time of the duty.
- In each duty longer than a certain minimum length, there should be a meal break with a certain minimum length at one of the relief locations with a canteen.

Furthermore, for drivers there are extra requirements with respect to their knowledge about specific rolling stock types and routes. However, it is assumed that all drivers from a certain crew base have the same knowledge.

Finally, there are some rules, which deal with the deviations from the annual plan. We call the duties in the annual plan *original duties*.

The CRSP can now be defined as follows. Given the number of crews available and their original duties which can be slightly modified, find a schedule such that all tasks are covered. With a slightly modification of the original duties, we mean that a duty cannot start much earlier (or finish much later) than the original duty. In principle, deviations of 30 minutes are allowed, however, sometimes a little bit larger deviations are accepted as well.

However, it is often not possible to find a feasible solution with only modifying the original duties, therefore some rules can sometimes be relaxed and extra duties can be generated. For these extra duties, there is extra crew available at certain crew bases, the so-called *amplitude duties*. These duties have a long spread time in which a "real" duty could be placed. Notice that it is sometimes even not possible to find a feasible solution with all available crew (regular and amplitude duties). Then an extra duty is necessary and a crew member is asked to work an extra day (against a higher wage). This option should be avoided as much as possible.

Moreover, the crew can use buses or taxis to go from one location to another one when there is no train traffic possible. However, the latter option is not preferred, because the driving time of buses and taxis is very sensitive to traffic conditions and can therefore easily lead to delays. To avoid this, bus and taxi trips are mainly used at the start or end of a duty.

4 Mathematical formulation

In this section, we propose a mathematical formulation for the CRSP, which is equivalent to a set covering formulation.

Let N be the set of tasks, where $N^p \subset N$ is the set of passenger tasks. Furthermore, let B and Δ be the set of crew bases and original duties, respectively. Moreover, define K^δ as the set of all feasible duties which could replace original duty $\delta \in \Delta$. The costs of a duty k corresponding to original duty δ is denoted by c_k^δ. The parameter a_{ik}^δ is 1 if task i is part of this duty, and 0 otherwise. Finally, decision variables x_k^δ indicate whether a duty k corresponding to original duty δ is selected in the solution or not. The CRSP can be formulated as follows.

(CRSP):

$$\min \sum_{\delta \in \Delta} \sum_{k \in K^\delta} c_k^\delta x_k^\delta \tag{1}$$

$$\sum_{\delta \in \Delta} \sum_{k \in K^\delta} a_{ik}^\delta x_k^\delta \geq 1 \qquad \forall i \in N \backslash N^p, \tag{2}$$

$$\sum_{k \in K^\delta} x_k^\delta \geq 1 \qquad \forall \delta \in \Delta, \tag{3}$$

$$x_k^\delta \in \{0,1\} \qquad \forall \delta \in \Delta, \forall k \in K^\delta. \tag{4}$$

The objective (1) is to minimize the total costs of the duties. The first set of constraints (2) guarantee that every task not being a passenger task is covered by at least one duty. If a certain task is assigned to two or more duties, it will only be a 'real working' task in one of the duties. In the other duties, the task will actually be a passenger task. The constraints (3) require that each original duty is replaced by at least one new one. We have "$\geq$" instead of an equality sign, since it is sometimes possible that, due to the changes in the timetable, more duties are necessary than in the original schedule. By introducing high fixed costs for each duty, the number of extra duties will be minimized. In particular, there will be no extra duties if they are unnecessary. Moreover, we assume here that an original duty can always be replaced by a new one. We can guarantee this by allowing that an original duty is replaced by an empty one, which practically means that the driver assigned to this duty will get an extra day off.

5 Algorithm

In this section we describe a column generation based algorithm to solve the CRSP. We assume that the reader is familiar with the basic ideas of column generation (recent surveys on this topic are Barnhart *et al.* (1998); Lübbecke & Desrosiers (2002); Desaulniers *et al.* (2005)).

An outline of the algorithm is given in Figure 2.

In some steps of the algorithm, we work with the concept of pieces of work. Recall from Section 3 that a piece is a sequence of tasks on the same train. All pieces can be easily found by enumeration. We use the notation $p \subset p'$ if piece p is a part of piece p' starting with the same task, i.e. piece p' is completely similar to piece p but has at least one task more at the end. In the remainder of this section, we discuss the steps 1 until 5 in detail.

5.1 Generation of "look-alike" duties (Step 1)

For each original duty where at least one of the original tasks does not exist anymore, we generate some alternative duties. These duties should look like the original one. Moreover, there are some hard constraints: the duty should still start and end in the same crew base and the start (and end) time of the duty should not be much earlier (later) than those of the original duty.

The idea of the "look-alike" duties is as follows. It is preferred that pieces of work from the original duty, which have not been changed, will be present in the new duty as well. This is done by replacing the pieces, which do not

Step 0: Initialization
Construct all pieces of work.
Step 1: Generation of "look-alike" duties
Generate for each original duty δ a (large) set of "look-alike" duties.
These duties form the column pool $\tilde{K}$.
Take a small subset of the columns $K \subset \tilde{K}$ as initial set of columns.
Step 2: Computation of dual multipliers
Solve a Lagrangian dual problem with the set of columns K.
Step 3: Selection of new duties
Select duties with negative reduced cost in $\tilde{K}$ and add them to K.
If they are found, return to Step 2;
otherwise, go to Step 4.
Step 4: Generation of new duties
Order the set of original duties $\delta \in \Delta$ randomly.
Generate for each regular duty $\delta \in \Delta$ a set of new duties with negative reduced cost, and add them to K until a maximum number of duties has been added.
Generate for each amplitude duty $\delta \in \Delta$ a set of new duties with negative reduced cost, and add them to K.
Compute an estimate of a lower bound for the overall problem.
If the gap between this estimate and the lower bound found in Step 2 is small enough (or another termination criterion is satisfied), go to Step 5;
otherwise, return to Step 2.
Step 5: Construction of feasible solution
Solve the remaining set covering problem with the duties in the set K heuristically.

Figure 2: Algorithm to solve the CRSP

exist anymore, by alternative ones such that the remainder of the new duty is the same as the original one. We generate them by complete enumeration.

For instance, consider an original duty which consists of three pieces p, q and r (see Figure 3). Only tasks in piece q have been changed, and therefore this piece does not exist anymore. All duties starting with piece p and ending with piece r are defined as "look-alike" duties and are generated. Notice that we do not require that there is again exactly one piece in between the pieces p and r. This could be more or less than 1. In general, not many duties will be generated in this way. Therefore, we also generate duties starting with all pieces p', where $p \subset p'$. In a similar way, we replace piece r by a set of alternatives r', and we also consider feasible duties starting with p' and ending with r'.

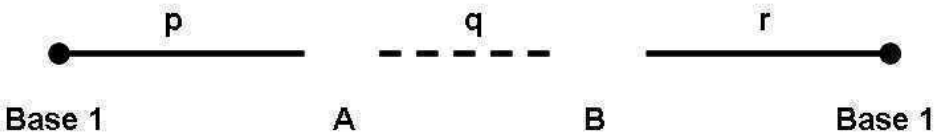

Figure 3: An original duty consisting of three pieces

5.2 Computation of dual multipliers (Step 2)

We use Lagrangian relaxation to obtain the dual multipliers (see Huisman *et al.* (2005a) for an overview on combining column generation and Lagrangian relaxation). Therefore, we relax the constraints (2) with Lagrangian multipliers λ_i. We denote the vector of multipliers with λ. Then, the reduced cost of a duty k is defined as:

$$\overline{c}_k^{\delta} = c_k^{\delta} - \sum_{i \in N \setminus N^p} \lambda_i a_{ik}^{\delta}. \tag{5}$$

The remaining Lagrangian subproblem, denoted by $\theta(\lambda)$, can be easily solved by pricing out the x-variables, i.e. all x-variables which have negative reduced costs are 1. Moreover, for each $\delta \in \Delta$ the x-variable with the smallest reduced cost will be 1 as well (even if the reduced cost is positive). We denote b^{δ} as the smallest reduced cost for each δ if this is a positive number, and 0 otherwise. The remainder of the variables will be equal to 0.

With subgradient optimization (see e.g. Beasley (1995)) we solve the Lagrangian dual problem $max_{\lambda}\theta(\lambda)$. The best value of $\theta(\lambda)$ in the subgradient algorithm is denoted by $\overline{\theta}$. Furthermore, we need an additional procedure to update the Lagrangian multipliers after solving the Lagrangian relaxation. This is necessary to assure that all duties in the current master problem have non-negative reduced costs so that these duties will not be generated again in the pricing problem. This procedure is similar to the one in Huisman *et al.* (2005a).

5.3 Selection of new duties (Step 3)

With the multipliers obtained in Step 2, we can easily calculate the reduced costs $\overline{c}_k^{\delta}$ of all duties in the set $\tilde{K} \setminus K$. We add the m duties with the smallest value $\overline{c}_k^{\delta} - b^{\delta}$.

5.4 Generation of new duties (Step 4)

It is very likely that an optimal (or nearly optimal) solution cannot be found with only the "look-alike" duties. Therefore, other duties with negative reduced costs should be generated. This is done in Step 4 by solving a pricing

problem for the original duties (see Section 6). This includes the regular duties and the amplitude duties. However, it is unnecessary to generate new duties for each original duty in each iteration. If we have found enough new duties, we return to Step 2. To prevent that we start every iteration with the same original duties, we order them randomly at the start of an iteration. Moreover, we always consider all amplitude duties, since they do not have "look-alike" duties and there are only a few of them.

5.5 Construction of feasible solution (Step 5)

To find a feasible integer solution, we use the very successful heuristic in the crew scheduling package Turni to solve the set covering problem. This heuristic is based on the ideas of Caprara *et al.* (1999) with some local improvement heuristics.

6 Pricing problem

For each original duty, the pricing problem corresponds to a simple shortest path problem in a dedicated network. To reduce memory usage, the same network is used to solve all pricing problems.

In this network $N = (V, A)$, the set of nodes V consists of all feasible pieces, and a source and sink corresponding to the crew base. The set of all arcs is denoted by A. There is an arc between two pieces p and q if the start location of piece q is the same as the end location of piece p, and the time between the start of piece q and the end of piece p is at least equal to the connection time, which depends on the first task in piece p. Moreover, piece p and q should not be on the same rolling stock, since then there is another piece r which can be used (or the time on the same rolling stock is too long). Sometimes taxi and bus trips can be used, in that case, the start and end location should not be equal, but the time between the start of piece q and the end of piece p should be enough to use a taxi or bus. In this way, an arc can consists of a taxi or bus trip, and the costs associated with the arc includes the costs of this taxi or bus trip. Finally, there are arcs from the source to all pieces p for which the start location of p is equal to the location of the crew base or could be connected by a taxi or bus trip. In a similar way, there are arcs to the sink.

The nodes in the network are ordered efficiently such that to solve a particular pricing problem, it can be easily checked if some nodes can be used or not. To be more precise, the nodes are ordered on the following criteria:

- List of crew bases that can do all tasks in the piece,

- start location, and
- start time.

In this way, it is immediately clear that, if the crew base is not in the list of crew bases, all those nodes do not have to be considered in the pricing problem. In other words, although there is only one network used, it can be used very efficiently by solving the pricing problem for all different, original duties.

We solve the pricing problem as a simple shortest path problem, since almost all restrictions can be dealt with in the network. Examples are the connection time between two pieces, the earliest start time and latest end time of a duty and the duty length (by checking all possible start nodes). The only constraint that cannot be handled in this network is the meal break. However, in the computational experiments, we noticed that there is almost always automatically place for a meal break in a duty. This is caused by the fact that the minimum length of a meal break is not so much longer than the minimum connection time between two pieces (30 and 15, minutes respectively) and the fact that on a lot of stations trains leave every 30 minutes in each direction. If the shortest path does not correspond to a feasible duty since the meal break is missing, we just skip that path and search for a new one. This is implemented by generating the k shortest paths and by taking the best one(s) of them which include(s) a meal break.

However, notice that the pricing problem could also be solved exactly in polynomial time. By constructing a slightly different network, where a shortest path problem is solved between each possible start node and each possible node where a meal break starts, and from each possible node where a meal break ends to each possible end node and all possible combinations are considered. Although, from a theoretical point of view this would be nicer, we decided to solve the pricing problem in the first way, since in this way the pricing problem could be solved fast and reasonably well.

7 Computational experience

The presented algorithm has been applied to solve several real-world instances. These instances vary in size and complexity. We will discuss the results of the algorithm for two different cases, one of medium-size and a large one. The following parameter settings have been used:

- The number of subgradient iterations in the master problem is 1000.
- At most 1000 columns are selected from the column pool in Step 3. We iterate between Step 2 and 3 until the potential improvement is less than half the costs of an unmodified duty.

number of duties	586
number of tasks	5,683
number of expired tasks	355

Table 1: Data Case 1

- In Step 4, we add at most 100 alternative duties for each original duty.
- We terminate the column generation algorithm if the potential improvement is less than 0.1% or if the number of main column generation iterations (Step 4) is equal to 30.
- The set covering heuristic in Step 4 has a maximum computation time of 4 hours.

All tests are executed on a Pentium IV pc (3 GHz, 512 MB RAM).

7.1 Case 1: November 26, 2005

We first look at a rather small case in the Eastern part of the Netherlands. On Saturday November 26, 2005, the tracks between Almelo (Aml), Hengelo (Hgl) and Enschede (Es) were renewed. Therefore, the Intercity services from the Western part of the country to Es did not go further than Aml and returned there. Furthermore, the local train services between Zwolle (Zl) and Es returned in Aml as well. For the passengers, bus services were introduced between Aml and Es (express and stop). Finally, there were some extra empty rolling stock movements since rolling stock could not stay overnight in Aml, but only in Zwolle and Deventer (Dv). The crew bases in this area are Zl, Zutphen (Zp), Hgl and Es. Moreover, crews from bases in the Western part of the country drive trains to Es as well. The crews from Hgl and Es could use the bus services to travel from their crew base to Aml and vice versa. Moreover, to be able to drive the early and late trains, taxi services could be used from the different crew bases to Dv.

In total 72 duties (including all duties from Hgl and Es) were affected by the changes in the timetable. Other relevant figures are mentioned in Table 1

In Figures 4 and 5, we illustrate the performance of the column generation algorithm. On the horizontal axis the main and sub iterations of the algorithm are shown, where a main iteration coincides with generating new duties (Step 4), while in a sub iteration duties are only selected from the column pool (Step 3).

From Figure 4, we can see that the main improvements in the value of $\overline{\theta}$ in the subgradient algorithm, are in the main iterations and in the sub iterations of the first main iteration. Moreover, we see that selecting from the column pool is only done more than once in the first few main iterations. The

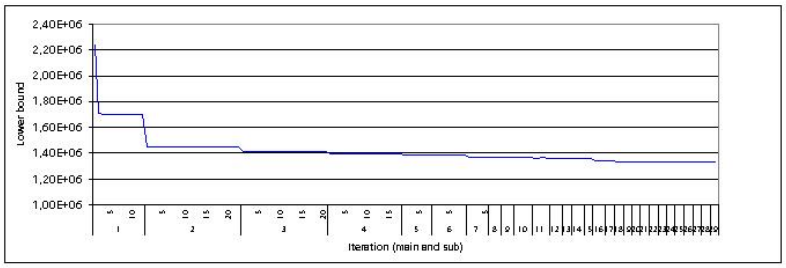

Figure 4: Value of $\overline{\theta}$

computation time of the first main iterations is very low, since the generation of new duties can be stopped quite soon. The reason is that we return to the master problem when for 20 original duties the maximum number of new duties is found. In the last iterations, only a few duties are generated per original duty. Therefore, all of them are considered. To prove that there are no duties with negative reduced costs anymore, we should solve the pricing problem for all original duties.

The results of this case are given in Table 2. First, we mention the total number of pieces, which is relevant for the size of the network in the pricing problem, since we have a node for each piece. Then, we mention the results: the number of unmodified duties (including duties which have the same tasks as before, but where some tasks have a different start or end time), the number of modified duties (including empty ones), the number of used amplitude duties and the number of extra duties.

It is remarkable that there are only a few look-alike duties, this is due to the fact that the duties from the crew bases Hgl and Es are all completely different. The computation time of Step 1 can be neglected compared to the other steps. Moreover, we can see that the number of modified duties is 74, which is only a few more than the 72 that were strictly necessary. The planners had a careful look at this solution and implemented it with a few minor changes.

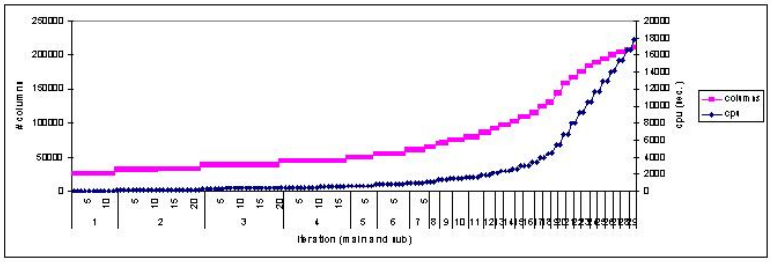

Figure 5: Number of columns and cpu time

7.2 Case 2: September 17, 2005

A lot of construction works were carried out on Saturday September 17, 2005. The main lines between Utrecht (Ut) and 's Hertogenbosch (Ht), and between Amersfoort (Amf) and Apeldoorn (Apd) could not be used at all. Moreover, several other construction works were carried out in other areas of the country.

In Table 3, we mention the number of duties in the original schedule, the number of tasks and the number of expired tasks. As mentioned before, on several crew bases in the country, amplitude duties could be used. In total, there were 26 of these duties available for this case. Moreover, taxi and bus

# pieces	23,226
# look-alike duties	8,767
# generated duties (in Step 4)	184,420
cpu time Step 1	0.2 hours
cpu time Step 2-4	4.9 hours
cpu time Step 5	4 hours
# unmodified duties	512 (498 + 14)
# modified duties	74 (60 + 14)
# used amplitude duties	8
# extra duties	4

Table 2: Results Case 1

number of duties	773
number of tasks	7740
number of expired tasks	827

Table 3: Data Case 2

# pieces	37,048
# look-alike duties	169,974
# generated duties (in Step 4)	203,961
cpu time Step 1	2.55 hours
cpu time Step 2	0.45 hours
cpu time Step 3	0.01 hours
cpu time Step 4	8.63 hours
cpu time Step 5	4 hours
# unmodified duties	548 (512 + 36)
# modified duties	228 (214 + 14)
# used amplitude duties	6
# extra duties	3

Table 4: Results Case 2

trips could not be used in this case.

The results of the algorithm are mentioned in Table 4. We denote again the number of pieces generated, the number of look-alike duties and the number of duties generated in the pricing problem (Step 4). Afterwards, the computation times of the different steps are denoted. Finally, we give the number of unmodified (including duties which have the same tasks as before, but where some tasks have a different start or end time), the number of modified duties (including empty ones), the used amplitude and extra duties in the solution.

We can see that 29.5 % of the original duties will be modified. 14 of the original duties will be empty, i.e. a driver will get a day off. On the other hand, 3 extra duties are necessary. The planners had a careful look at this solution and were very satisfied about its quality. Moreover, a total computation time of less than 16 hours is acceptable for such a large case.

It is more complicated to look at the quality of the solution from a theoretical point of view. Since the column generation algorithm is not stopped after convergence but terminated earlier, a lower bound could not be obtained. However, we did an extra run to obtain a good lower bound and compared it with the solution in Table 4. The gap between the value of the feasible solution above and this lower bound was 3.3%.

8 Conclusions and Future research

In this paper, we discussed the Crew Re-Scheduling Problem (CRSP), where crew duties need to be repaired because of changes in the underlying timetable and rolling stock schedule. This occurs mainly due to construction works on the infrastructure. We showed that the presented algorithm, which uses column generation techniques, performed very well on several practical cases. These cases varied in size and complexity. Computation times were reasonable and the solution mostly outperformed the solutions made by the planners.

The advantages for NS to use the presented algorithm are recognized by the management. A project has started to incorporate the algorithm in a decision support system. In this way, the total time to construct a schedule for one particular day could be reduced significantly. This can result in less planners or more time to consider alternative schedules instead of constructing only one schedule.

In the future, we would like to look at real-time crew rescheduling in case of disruptions of the timetable as well. The similarity with the CRSP is that in such cases a line is also out of service for some time and the crew needs to be rescheduled. However, a difference is that in case of a disruption during the operation the duration is unknown, while it is known in case of construction works. Moreover, the decisions should be taken rapidly. In spite of these differences, we believe that some ideas presented in this paper can be used for this problem as well.

Acknowledgments The author would like to thank Vincent Vijfvinkel for his careful implementation of the shortest path algorithm to solve the pricing problem. Moreover, he would like to thank Ad Krijgsman, Bram Hamel, Erwin Abbink and Wilmet van Dijk for their comments on the results of some initial runs, which lead to further improvements of the algorithm. Finally, Leo Kroon is thanked for his comments on an earlier draft of this paper.

References

Abbink, E., Fischetti, M., Kroon, L, Timmer, G., & Vromans, M. 2005. Reinventing crew scheduling at Netherlands Railways. *Interfaces*, **35**, 393–401.

Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., & Vance, P.H. 1998. Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Operations Research*, **46**, 316–329.

Beasley, J.E. 1995. Lagrangean Relaxation. *Pages 243–303 of:* Reeves, C.R. (ed), *Modern Heuristic Techniques for Combinatorial Problems*. McGraw-Hill, London.

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

Desaulniers, G., Desrosiers, J., & Solomon, M.M. (eds). 2005. *Column Generation*. Springer, New York.

Fores, S., Proll, L., & Wren, A. 2001. Experiences with a Flexible Driver Scheduler. *Pages 137–152 of:* Voß, S., & Daduna, J.R. (eds), *Computer-Aided Scheduling of Public Transport*. Springer, Berlin.

Huisman, D., Jans, R., Peeters, M., & Wagelmans, A.P.M. 2005a. Combining Column Generation and Lagrangian Relaxation. *Pages 247–270 of:* Desaulniers, G., Desrosiers, J., & Solomon, M.M. (eds), *Column Generation*. Springer, New York.

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

Kohl, N. 2003. Solving the World's Largest Crew Scheduling Problem. *ORbit*, 8–12.

Kroon, L.G., & Fischetti, M. 2001. Crew Scheduling for Netherlands Railways "Destination: Customer". *Pages 181–201 of:* Voß, S., & Daduna, J.R. (eds), *Computer-Aided Scheduling of Public Transport*. Springer, Berlin.

Lübbecke, M.E., & Desrosiers, J. 2002. *Selected Topics in Column Generation*. Tech. rept. G-2002-64. Les Cahiers du Gerad, Montréal.