Integrating Timetabling and Crew Scheduling at a Freight Railway Operator

Lukas Bach[†], Twan Dollevoet[‡*], and Dennis Huisman[‡⋆]

[†]Cluster for Operations Research And Logistics,
Department of Economics and Business,
Aarhus University, Aarhus, Denmark
luba@asb.dk

[‡]Erasmus Center for Optimization in Public Transport (ECOPT)
and Econometric Institute, Erasmus School of Economics,
Erasmus University Rotterdam, Rotterdam, the Netherlands
dollevoet@ese.eur.nl, huisman@ese.eur.nl

[⋆]Process quality & Innovation, Netherlands Railways,
Utrecht, the Netherlands

Econometric Institure Report EI2014-03

Abstract

We investigate to what degree we can integrate a Train Timetabling / Engine Scheduling Problem with a Crew Scheduling Problem. In the Timetabling Problem we design a timetable for the desired lines by fixing the departure and arrival times. Also, we allocate time-slots in the network to secure a feasible timetable. Next, we assign engines in the Engine Scheduling Problem to the lines in accordance with the timetable. The overall integration is achieved by obtaining an optimal solution for the Timetabling / Engine Scheduling Problem. We exploit the fact that numerous optimal, and near optimal solutions exists. We consider all solutions that can be obtained from the optimal engine schedule by altering the timetable, while keeping the order of demands in the schedules intact. The Crew Scheduling model is allowed to re-time the service of demands if the additional cost is outweighed by the crew savings. This information is implemented in a mathematical model for the Crew Scheduling Problem. The model is solved using a column generation scheme. Hereby it is possible for the Crew Scheduling algorithm to adjust the timetable and achieve a better overall solution. We perform computational experiments based on a case at a freight railway operator, DB Schenker Rail Scandinavia, and show that significant cost savings can be achieved.

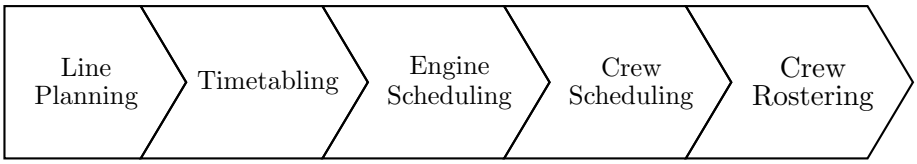
Keywords: Railway Crew Planning, Vehicle and Crew Scheduling, Partial Integration, Time Windows, Branch-and-Price

[*]Corresponding author

1 Introduction

In this paper we investigate to what extent we can integrate a Train Timetabling / Engine Scheduling Problem and a Crew Scheduling Problem (CSP). Our approach is inspired by a case at a freight railway operator, DB Schenker Rail Scandinavia (DBSRS).

The planning process at railway operators has been described at different levels of detail by, among others, Huisman et al. (2005), Caprara et al. (2007) and Lusby et al. (2011). In general, the planning process can be described in five phases as follows:



In the Timetabling Problem we design a timetable for the desired lines - from the Line Planning Problem - and fix the departure and arrival times. Also, we allocate time-slots in the network to secure a feasible timetable. In the Engine Scheduling Problem we assign engines to the lines in accordance with the timetable. We ensure that this allocation produces a feasible plan. Bach et al. (2014) solved an integrated version of these two problems at a freight railway operator.

In this paper we seek to integrate the Timetabling and Engine Scheduling phases with Crew Scheduling. The overall goal is to investigate whether it is favorable to leave these earlier stages open to adjustments in the Crew Scheduling phase. To this end, we first solve a CSP based on integration with the Timetabling and Engine Scheduling Problem (TESP). We seek to integrate by allowing changes to the timetable in the CSP phase. We do this while keeping the solution to the TESP feasible. Hence, we explore alternative feasible solutions. We then compare this to a completely sequential approach, where we solve the CSP in a classic approach without any link to the timetabling or engine scheduling phases.

The remainder of this paper is structured as follows. In Section 2, we describe the problem and introduce case specific details, and in Section 3, we review the relevant literature. In Section 4, we explain the integration with the TESP and present our integrated model, we present the solution method hereto in Section 5. In Section 6, we discuss the results of our computational experiments. Finally, we present our conclusions and future research directions in Section 7.

2 Problem Description

The demand in this model is strictly unit train demand: Any demand is for a full train driving from an origin to a destination station. It is not possible to aggregate demand. For each of these demands, we have a fixed time window wherein the demand should be fulfilled. For the crew, a single demand can be split into one or more tasks at relief points along the route. In the case considered in this paper there is a maximum of 228 demands per week.

We consider a planning horizon of one year during which the timetable has to be repeated each week. The timetable design is typically done more than half a year in advance. The contracts between DBSRS and their customers in general cover the timetabling period of a year.

The transit times and time-slots in the operational area of DBSRS are allocated by the infrastructure manager, who also assigns access to the network. We use these time-slots to model what times we can access the infrastructure. The time-slots are published with a time resolution of 1 minute and are available about every half hour in the main corridor. It is thus only possible to start usage of the tracks at discrete points in time.

We have a number of engines available and generate the same number of engine schedules, each containing a set of demands to perform during a week. A specific engine can perform different engine schedules from week to week. This can lead an engine to start and end at different stations in the beginning and end of the week. However, this is balanced over all engine schedules. For an engine schedule to be feasible, it must satisfy a set of constraints (mainly it can only use tracks when allowed to do so by a time-slot) and include buffer times at the beginning and end of demands, forcing the plan to be more robust.

The crew at DBSRS are divided into three major groups separated by the country in which they are employed. This means that they also have a different set of working regulations. Within these groups, the crew are furthermore assigned to crew bases within their country. The crew should always end their duty at their home base, but can travel anywhere otherwise allowed by the working regulations.

For a duty to be feasible it must respect the rules in Table 1. For German drivers, a duty becomes a night duty if 3 or more hours of the duty are within night time. In contrast, for Danish drivers, it is a night duty if just 1 minute is within night time. In Sweden there are only work time regulations related to a night duty and no fixed cost for a night duty. Instead, the cost of the actual time during night time is compensated.

If a driver crosses a border, EU rules are enforced. The rules for offsetting a night duty are only affecting Danish drivers, who with respect to this, now have the duty defined as a night duty similar to the German rules. For Danish and Swedish night duties, the maximum drive time is reduced to 8

Table 1: Duty related labor rules

	Germany	Denmark	Sweden
Min work hours	5	6	3
Max work hours	14	9	10
Max drive time	8	9	10[9]
Max drive time ND*	8	9[8]	10[8]
Max time without break	5.5	4.75	5
Length of break	(duty $<$ 8): 0.5	0.5	0.5
...	(duty $<$ 12): 0.75		
...	(duty $<$ 14): 2		
Night time	23:00 - 6:00	1:30 - 4:30	22:00 - 6:00

Note: *ND(Night Duty). Values in [] are for cross-border traffic, where it is more restricting than native rules.

hours. Further, the maximum drive time during the day becomes 9 hours.

As part of any duty there is a set of mandatory tasks. These are prepare and finalize duty tasks. To have a break, start, or end a duty, it is necessary to walk to/from a break room. This walking time is dependent on the station used, and it is added to the time of the task. There is also a maximum work time without a break. This ensures that a break is placed at a reasonable time during a duty. For all drivers the minimum break time to reset the time without break is 30 minutes. We refer to this as the minimum break. As it can be seen from Table 1, it is sometimes necessary to have at least one longer break for the duty to be feasible. We call this the required break.

The cost of a duty is modeled as a fixed part and variable part, this is roughly $\frac{1}{3}$ fixed and $\frac{2}{3}$ flexible. Having some of the cost flexible allows for a more balanced solution, not maximizing duty length. How the cost should be distributed between fixed and flexible is a managerial problem, the model can be used with any combination of the two. The variable part consists of a time-dependent cost, costs for night duties, and costs for cross-border activities.

The problem then becomes to cover all crew tasks during a week at the minimum cost using the crew from the three different countries.

3 Literature

Many successful applications of Operations Research in the railway context have been discussed in the literature over the last decades (see Kroon et al. (2009) for an example). Traditionally, the planning process at a railway operator is decomposed into five levels. The Crew Scheduling Problem (CSP) is the fourth level and is usually solved when the timetable and the engine

schedules are determined. In what follows, we review the recent literature on the CSP and on the integration of earlier levels with the CSP. For more information on the other scheduling levels, we refer to Caprara et al. (2007), Huisman et al. (2005), or Lusby et al. (2011) and references therein. We also refer to Bach et al. (2014) for more on the Timetabling and Engine Scheduling Problem.

In the seminal study by Caprara et al. (1999), the CSP is modeled as a set covering problem and solved by column generation. In this approach, the columns represent feasible duties, and the rows correspond to the tasks that have to be performed. The master problem is solved by Lagrangean relaxation, while feasible duties are generated in the pricing problem by solving a resource-constrained shortest path problem.

Recent studies on the CSP mainly focus on developing acceleration techniques in order to solve large-scale problems. Elhallaoui et al. (2005) aggregate multiple tasks into one and thereby reduce the size of both the master and the pricing problems. By updating the aggregation dynamically, the problem can still be solved to optimality.

Jütte and Thonemann (2012) divide the CSP into multiple regions and price the assignment of trips to these regions. This procedure is implemented by Jütte et al. (2011) for a real-world instance from DB Schenker. The authors show that large-scale instances can be solved in a reasonable computation time.

Several algorithms for the CSP have found their way to commercial decision support systems. PowerSolver is developed by Kwan and Kwan (2007) and applied to several instances from the UK. Abbink et al. (2011) describe LUCIA, a crew scheduling algorithm used by Netherlands Railways that can solve instances with tens of thousands of tasks. LUCIA is based on an algorithm for the crew rescheduling problem that was developed by Huisman (2007).

All the algorithms described so far deal with crew *planning*, where computation times of hours or days are acceptable. When disruptions occur in the daily operations, crew duties should be rescheduled in real-time. Potthoff et al. (2010) describe a method to reschedule duties within minutes for a case of Netherlands Railways. Similarly, Rezanova and Ryan (2010) develop a real-time crew rescheduling approach based on set partitioning and test it on real-life instances from Denmark.

In this paper we apply the CSP to the case from DBSRS. A particular difference is that we consider crew employed in different countries who thus work under different working regulations. The crew considered work in Sweden, Denmark, or Germany. The main working regulations are similar. Yet some differences have to be addressed. Papers considering train working regulations related to these countries include; Rezanova and Ryan (2010) for a Danish, and Jütte et al. (2011) for a German setting.

The integration of single-depot Vehicle and Crew Scheduling is studied

in Freling et al. (2003). Huisman et al. (2005) extend their model to multiple depots and focus on a sub/extraurban transit system. They show that there are significant cost savings to be achieved. This conclusion is supported by Groot and Huisman (2008), Mesquita et al. (2009), and Steinzen et al. (2010), among others. All these papers consider a full integration of the Vehicle and Crew Scheduling Problem.

Kliewer et al. (2012) extend the Vehicle and Crew Scheduling Problem with the addition of time windows for trips. By allowing to shift trips with up to 4 minutes in time, they achieve a further cost reduction. The approach is based on an integrated model presented by Steinzen et al. (2010) and tested on real-life data from public bus transportation. A similar approach to re-timing can be seen in Veelenturf et al. (2012), where a Crew Rescheduling Problem at Netherlands Railways is considered. Here, minor changes to the timetable result in improved solutions.

Gintner et al. (2008) present a partial integration of Vehicle and Crew Scheduling. The optimal vehicle flow can be decomposed into an optimal vehicle schedule in a number of different ways. All these alternative solutions to the Vehicle Scheduling Problem are implicitly explored in the crew scheduling phase.

This paper also partially integrates timetabling and vehicle scheduling with crew scheduling. Our approach allows to change the timetable, but leaves the order of the trips in the vehicle/engine schedule unchanged. In contrast to Kliewer et al. (2012), who also allow some degree of re-timing, our model is for a railway system whereas their model is for a public bus transit system. The most significant difference is that the access to the railway infrastructure has to be taken into account.

4 Methodology

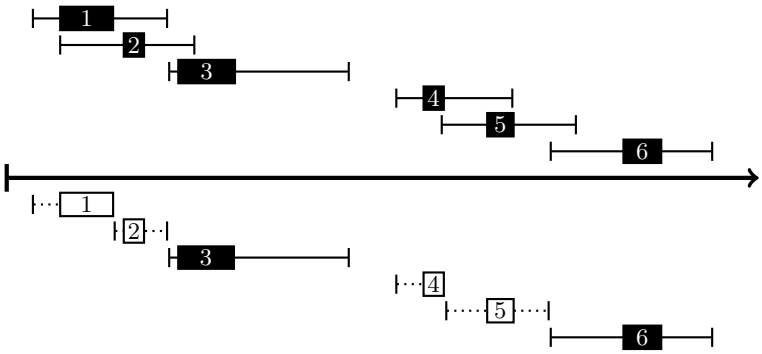
Both the combined Timetabling and Engine Scheduling Problem (TESP) as well as the Crew Scheduling Problem are solved using a column generation approach. We compare the integrated approach to the sequential approach, where we solve the CSP using a timetable set by the TESP. By comparing this to the integrated approach, we can evaluate the quality and the possible improvements by integration.

4.1 Integration

When solving the TESP as in Bach et al. (2014), there are a number of alternative feasible solutions. In the integrated approach, we seek to explore these by allowing re-timing of the individual demands. An engine schedule will remain feasible as long as the order of the demands is unchanged. In an optimal solution, a demand is serviced at a point in time within its time window, and by a specific engine schedule. An engine schedule services one

or more demands. The demands in a schedule can be far apart time-wise, such that service at any point in time within their time window is allowed. If this is the case, we say that a demand can move freely within its time window. This holds for many demands. The others are either fixed or can use a restricted part of the time window. We will explain this in the following section. We always keep the TESP feasible. To maintain feasibility, we need to make sure that no more than one demand use the same time-slot. This approach explores many alternative solutions for the crew.

Figure 1: Time window reduction



To keep the schedules feasible, we adjust the time windows of the demands, maintaining the order of demands and thus keeping the schedules feasible. The algorithm we use to adjust the time windows keeps a time window unchanged if there is no overlap with the next or the previous time window. Otherwise we start from the time-slot used by the optimal solution and keep the next time-slot intact while there is no overlap. When it reaches an overlap to the next time window, the current time window is cut. In Figure 1, the top part represents an engine schedule and the time windows of the demands. On the bottom part the adjusted time windows are shown.

4.2 Timetabling and Engine Scheduling Problem

The main objective in the Timetabling and Engine Scheduling Problem is to fix the timetable. The TESP is formulated and solved as in Bach et al. (2014), and the solutions hereof are used as input for the integrated approach. No variables explicitly define the timetable. Instead, the timetable is extracted from the time-slots chosen in the model.

The TESP is formulated as a Set Partitioning Problem, with Ω being the set of engine schedules, r, under consideration. D is a set of all demands, d. N is the set of all stations, indexed by n. S is a set of all time-slots, s. E is the set of engine types, e. w_e is the engine availability for engine type e, given as a parameter.

The following parameters are determined in the pricing problem: c_r is the cost of schedule r. The parameter α_r^d indicates if demand d is covered on schedule r. β_r^n is equal to 1 if a station n is used on schedule r as origin station, if it is used as destination station it is equal to -1. If a station is used as both origin and destination or neither, β_r^n is equal to 0, i.e., the schedule is balanced with respect to this station. If time-slot s is used by schedule r, then γ_r^s is equal to 1. We set δ_r^e equal to 1 if schedule r is driven by engine type e. The integer program looks as follows:

$$\min \sum_{r \in \Omega} c_r x_r \tag{1}$$

$$\text{s.t.,} \sum_{r \in \Omega} \alpha_r^d x_r = 1, \quad \forall d \in D \tag{2}$$

$$\sum_{r \in \Omega} \beta_r^n x_r = 0, \quad \forall n \in N \tag{3}$$

$$\sum_{r \in \Omega} \gamma_r^s x_r \leq 1, \quad \forall s \in S \tag{4}$$

$$\sum_{r \in \Omega} \delta_r^e x_r \leq w_e, \quad \forall e \in E \tag{5}$$

$$x_r \in \{0, 1\}, \quad \forall r \in \Omega \tag{6}$$

The objective function (1) minimizes the cost of the selected engine schedules. In constraints (2) we make sure that all demands are serviced by the engines. To ensure a necessary balance between the starting and ending stations of the engines, we have the balance constraint, constraints (3). In constraints (4) we ensure that each time-slot is used at most once. Engine availability is ensured by constraints (5).

In the pricing problem we handle one weekly schedule of an engine in the network. To get the engine schedule, we solve a Shortest Path Problem. The start of service for the demand must be within its time window, which is defined by $[a_d, b_d]$ where a_d is the earliest start and b_d is the latest start of service. Within the time window demands can only be serviced when a path is available. Hence, service can be initiated at a set of discrete times t stored in the set U_d. The selected time of service is induced from the set of time-slots used by the schedule and is in this way connected to the master-problem. Furthermore, constraints considering shunting time, wait time after demands, deadheading, and flow conservation for intermediate stations are treated in the pricing problem.

4.3 Crew Scheduling Problem

The CSP is formulated as a Set Covering Problem, and solved by a column generation approach in which we generate duties ad-hoc in a pricing problem.

From the TESP we have demands $d \in D$. Each d is a demand going from one station to another. On this route there can be one or more relief points for the duties. Thus d is split into one or more crew tasks. P is the set of all tasks, indexed by p. Ω is the set of all duties (columns) in the restricted master problem, indexed by r. The cost of a duty, r, is given by c_r, α_r^p indicates if a crew task p is covered by duty r.

$$\min \sum_{r \in \Omega} c_r y_r \tag{7}$$

$$\text{s.t.}, \sum_{r \in \Omega} \alpha_r^p y_r \geq 1, \quad \forall p \in P \tag{8}$$

$$y_r \in \{0, 1\}, \forall r \in \Omega \tag{9}$$

The objective function (7) minimizes the cost of the selected duties. Constraints (8) ensure that all tasks are covered by at least one duty. The duties are generated in multiple pricing problems by solving a Shortest Path Problem on a graph representing the tasks and adhering to the working regulations, see Section 4.5.

4.4 Integrated Model

Constraints (1)-(6) form the master problem for the TESP, and constraints (7)-(9) form the master problem for the CSP. To integrate the two problems, we must connect the timing of the engines with the crew duties. To include the engines, we take the set of demands D. For each d we take the corresponding set U_d and make a copy of d for each time t at which the demand can be performed. We assign these copies d' to the set D'_d. The set D'_d now contains a node for each time it is possible to initiate service of d. For notational convenience, we define D' as the union of the sets D'_d over $d \in D$.

We introduce the decision variable $x_{d'}$ which is equal to 1 if the demand is performed at the given time, 0 otherwise. Thus we also add the constraints in equations (10) to an integrated master problem. We hereby ensure that we service each demand exactly once.

$$\sum_{d' \in D'_d} x_{d'} = 1, \quad \forall d \in D \tag{10}$$

Each demand d corresponds to one or more tasks p. If we move service of a demand in time, we also have to cover the crew tasks at different times. For each d' we also make a set of copies of the corresponding crew tasks p, so we now have a p' for each $d' \in D'_d$ contained in a set P'_p for all tasks $p \in P$. For each of these copies we add a decision variable $z_{p'}$, which is equal to 1 if the task is performed at the time the copy represents, or 0 otherwise.

We also have to introduce constraints such that only one of these are used. Veelenturf et al. (2012) introduce successor sets to link tasks together. We do not need sets as we only have one feasible successor task, because we can only 'delay' the train initially, whereas they can delay the trains at intermediate stations. Therefore, we introduce $succ(p')$ which points to the next task of p', $succ(p') = \emptyset$ if there is no successor, i.e., it is the last task of the demand. The relationship is kept by constraints (11).

$$z_{p'} - z_{succ(p')} = 0, \forall p' \in \{P' : succ(p') \neq \emptyset\} \tag{11}$$

We link the demand copies $d' \in D'$ in the same way to the first task, so the time chosen for service with the engine always corresponds to the time chosen for the crew tasks. Hence, we define $task(d')$ as the first task p' corresponding to d'. Then we form constraints (12) which are similar to constraints (11).

$$x_{d'} - z_{task(d')} = 0, \quad \forall d' \in D' \tag{12}$$

We also let the sum of copies of the same task be equal to 1, so only one time for a task can be chosen. The constraint is given in equation (13).

$$\sum_{p' \in P'_p} z_{p'} = 1, \quad \forall p \in P \tag{13}$$

We let the parameter $\gamma_r^{p'}$ be equal to 1 if task copy p' is used on schedule r, 0 otherwise. Let the parameter EDC be the engines' driver capacity such that $EDC = 2$ is interpreted as there can be 1 engine driver and 1 deadheading engine driver in the engine. Then we let constraints (14) ensure that $z_{p'}$ attains a positive value if the corresponding task copy is chosen.

$$EDC z_{p'} - \sum_{r \in \Omega} \gamma_r^{p'} y_r \geq 0, \quad \forall p' \in P' \tag{14}$$

After introducing constraints (10)-(14) and variables $z_{p'}$ and $x_{d'}$, it can be seen from equation (11) and (12) that $x_{d'}$ and all $z_{p'}$ corresponding to this demand copy will always be equal. This is because there is always one and only one successor for each p'. Because of this, we can remove the variables $z_{p'}$ from the model if we replace them with $x_{d'}$ in constraints (14). The addition of the parameter $dem(p')$, which points to the d' that a crew task is derived from, makes it possible to replace $z_{p'}$ with $x_{dem(p')}$ in equations (14).

From the train scheduling problem we know that some demands at a certain time use the same time-slots. When allowing changes to the optimal solution, we need to make sure that no more than one demand is using a time-slot. For all $s \in S$, we let M_s be the set of demand copies d' using the

same time-slot s. To ensure that no more than one of the demand copies use the same time-slot, we add constraints (15).

$$\sum_{d' \in M_s} x_{d'} \leq 1, \quad \forall s \in \{S : |M_s| > 1\} \tag{15}$$

The objective function (16) together with constraints (17)-(22) form the integrated master problem. We explore sub-optimal solutions from the TESP. The costs of the underlying time-slots are time-dependent in cost. Hence, it is necessary to introduce the cost of choosing a sub-optimal solution, this is captured by $c_{d'}$.

$$\min \sum_{r \in \Omega} c_r y_r + \sum_{d' \in D'} c_{d'} x_{d'} \tag{16}$$

$$\text{s.t.,} \sum_{d' \in D'_d} x_{d'} = 1, \quad \forall d \in D \tag{17}$$

$$\sum_{d' \in M_s} x_{d'} \leq 1, \quad \forall s \in \{S : |M_s| > 1\} \tag{18}$$

$$\sum_{r \in \Omega} \alpha_r^p y_r \geq 1, \quad \forall p \in P \tag{19}$$

$$EDC x_{dem(p')} - \sum_{r \in \Omega} \gamma_r^{p'} y_r \geq 0, \quad \forall p' \in P' \tag{20}$$

$$y_r \in \{0, 1\}, \quad \forall r \in \Omega \tag{21}$$

$$x_{d'} \in \{0, 1\}, \quad \forall d' \in D' \tag{22}$$

A stronger bound can be achieved by adding the constraints in (23). The parameter EDC in (21) is of a big M type. In the relaxation this allows for fractional x variables. This indicates that tasks corresponding to the same demand are serviced at different times.

$$\sum_{r \in \Omega} \gamma_r^{p'} y_r - x_{dem(p')} \geq 0, \forall p' \in P' \tag{23}$$

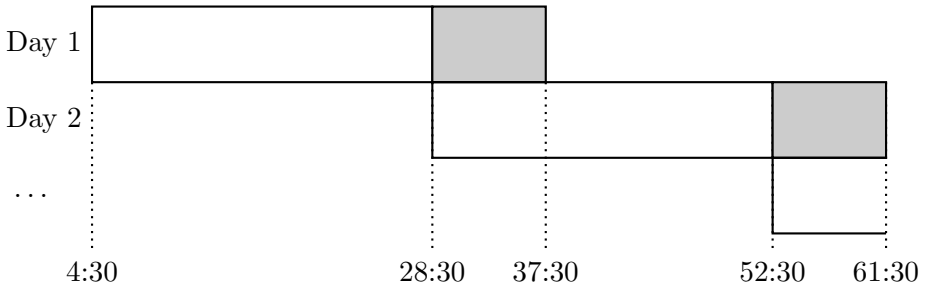
Equation (23) ensures that all tasks in a demand are serviced at least with the same fraction as the corresponding demand. However, this is not part of our model as it also slows down the solution process significantly.

4.5 Pricing Problem

The pricing problem for duty generation is modeled as a Shortest Path Problem with Resource Constraints (SPPRC) on a graph. Using a shortest path algorithm on a cyclic network is a problem. Abbink et al. (2011) deal with this by splitting the pricing problem into days of the week. This gives

one pricing problem per day, beginning with the first time-period of the day and ending at the latest plus the maximum work time allowed in a duty. In this way all possible duties are captured, and the individual pricing problems are now acyclic.

Figure 2: Splitting the week into days



Note: Example for the Danish labor rules where the maximum working time is 9 hours.

Splitting the pricing problem into individual days could be done at a number of arbitrary points during the week with 24 hours in between. Intuitively this would be at midnight which would separate the days. Figure 2 shows the days split at 04:30. Having multiple pricing problems also allows us to solve these in a parallel manner, not necessarily solving all the pricing problems to optimality in each iteration. In Section 5 the splitting of the pricing problem will be explained in greater detail.

5 Implementation

To establish a benchmark to see how much the solution to the CSP can be improved by our integration approach, we use our model in two ways. First, for the benchmark, we fix the x variables - representing the chosen timetable - to the solution provided by TESP. We then try to increase the lower bound by branching in a breadth first manner as we are in this case not interested in an integer solution, but in increasing the lower bound as much as possible.

The objective is to compare the cost of this benchmark solution to the cost of the integrated approach. For the comparison we use the total crew costs and the additional costs of choosing a sub-optimal engine schedule in the integrated approach.

In the integrated approach we let the timetable variables x flow freely within the adjusted time windows. To reach an integer solution, we first branch on the timetable variables, then we try to find an integer solution for the crew by performing a depth first search while branching on the tasks in

the duties. In this section, we will cover the graphs for the pricing problems, the algorithm for finding the shortest path, and the branching approaches.

5.1 Pricing Problem Splitting

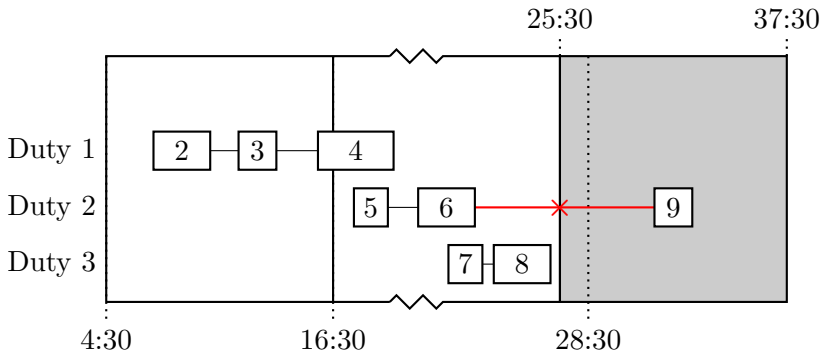
There are both different costs and additional working regulations depending on whether or not a duty is a night duty. To deal with this, we have to record whether a duty (for German and Danish drivers) is a night duty and apply the appropriate regulations. A resource could be added to the labels, recording whether we spend sufficient night time, and this resource could offset the night cost and the regulations. We can avoid this resource if we can a priori determine whether we are generating a night duty. We can do this if we, apart from splitting the pricing problem into days, also split it into day / night duties. By doing so, we avoid adding an extra resource.

Consider first the Danish rules for night duties. A Danish duty is a night duty if 1 or more minutes are spent between 1:30 and 4:30. If we split the day such that it begins at 4:30, then any duty should start between 4:30 and 28:30. As the maximal working time is 9 hours, duties can end at 37:30 at the latest. We now know that any task or arc crossing time 25:30 (24:00 + 1:30) will offset a night duty. As the graph is acyclic, this can indeed only happen once. In Figure 3, we depict the period between 4:30 and 37:30 for a specific day. We create two pricing problems for this instance. The first one contains only the nodes between 4:30 and 25:30. By construction, only day duties can be generated in this pricing problem. Such a day duty can start and end anytime within this period. The second pricing problem will generate night duties only. It considers the period between 16:30 and 37:30. A duty can start in the period from 16:30 and 28:30 and should end after 25:30. Doing this we can be sure that the duty is a night duty.

This can be extended to the German rules that define a night duty as a duty lasting 3 hours between 23:00 and 6:00. This is equivalent to spending one minute between 2:00 (23:00 + 3:00 - 24:00) and 3:00 (6:00 - 3:00). This holds because the minimum duty length is in any case more than 3 hours. Thus if we split the day at 3:00, we can apply a similar procedure as for the Danish rules, only adjusting the specific times at which duties can start and end.

Furthermore, the pricing problems are split by nationality and depot, such that any problem has a fixed start/end point and is governed by a given set of labor rules. For the German drivers where the length of the required break depends on the duty length, a separate pricing problem will be solved for the different possible lengths of the duty, i.e., less than 8, 12, and 14 hours. When drivers cross borders they must adhere to a set of EU working rules. By solving a separate pricing problem when we allow the drivers to cross borders, we still have a distinct set of working rules that does not change as a result of future decisions in the pricing problem.

Figure 3: Defining a night duty



Note: Duties 1-3 contain a number of tasks. Duty 1 is a regular day duty, duty 2 is a night duty as it crosses 25:30. Duty 3 does not cross 25:30, so it is a day duty. As it ends before 25:30, it can only be generated in the pricing problem for day duties.

In Figure 4 it can be seen how the final number of individual pricing problems increases by this method. In total with the rules applied, we get about 350 pricing problems.

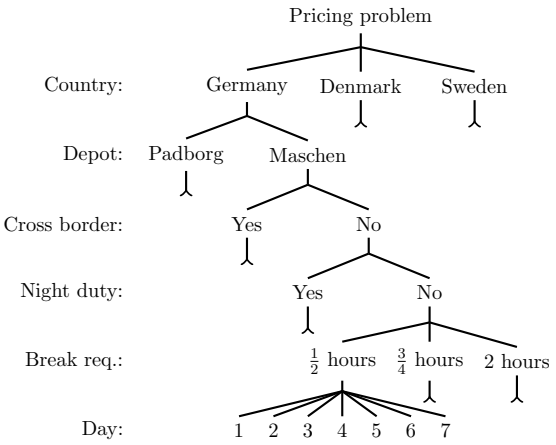
5.2 Pricing Problem Graphs

In the pricing problem we solve the SPPRC on multiple graphs, which are generated as follows. The graphs consist of nodes representing tasks, each node has a specific work time (wt_i) and drive time (dt_i) given. The arcs connecting the nodes have work time (wt_{ij}) and drive time (dt_{ij}). They also have a break time (bt_{ij}) assigned, representing how long a break is possible between two tasks. A parameter, break at end (be_{ij}), indicates whether a break is at the end of an arc or in the beginning of it. If it is possible to place the break in either end, two parallel arcs are introduced. These parameters are important when calculating the time since break requirement.

For each individual pricing problem a graph is created with the nodes that fall within the time frame, regulations, and the geography of the problem. For example, a task is represented by a node if it is reachable from the depot within the maximum working time of the duty. If the problem contains cross border activities, only tasks where it is possible to connect to a cross border task are included. The same applies to night duties. Only tasks that can be part of a night duty are included. For non-night duties the opposite applies.

To reduce the probability of cycles in the duties, the following rules are enforced prescribing which nodes can be connected. First, we define a cycle as a duty performing the same task at two different times. This

Figure 4: Pricing problems



is an infeasible duty, but it can be generated because we do not enforce elementary duties. If we do not allow any node to connect to any other node that represents the same task at a different time, we reduce the number of infeasible connections. This can be expanded to disallow any connection from a node to any other nodes coming from the same engine demand but at a different time. Finally, we can remove connections that will not be feasible with respect to the time-slots in the master problem. Because some copies of different demands might use the same infrastructure, only one of these can be chosen. Equivalently, servicing tasks from different demands that will use the same infrastructure simultaneously, cannot be part of an integer solution.

5.3 Resource Extension Functions

The Shortest Path Problem with Resource Constraints (SPPRC) is solved on the graphs described in Section 5.2. A labeling algorithm is used to solve the problem. We introduce 4 resources that are necessary in this problem: work time (WT), drive time (DT), time since last break (TSB), and break (B).

Resource extension functions from node i to j are then formulated as follows:

$$WT_j = WT_i + wt_{ij} + wt_j \tag{24}$$

$$DT_j = DT_i + dt_{ij} + dt_j \tag{25}$$

$$TSB_j = \begin{cases} wt_j & \text{if } be_{ij} = 1, bt_{ij} \geq minBreak \\ dt_{ij} + wt_j & \text{if } be_{ij} = 0, bt_{ij} \geq minBreak \\ TSB_i + wt_{ij} + wt_j & \text{otherwise} \end{cases} \tag{26}$$

$$B_j = \begin{cases} 1 & \text{if } bt_{ij} \geq reqBreak \\ B_i & \text{otherwise} \end{cases} \tag{27}$$

We do not extend a label if it is not resource feasible, i.e., if the resources exceed the given maximums. For WT, DT, and TSB the maximums are dependent on the regulations and are thus different among the graphs. Any of these three resources can dominate another resource of equal or larger size. Let us define the capacities for the resources as $maxWorkingTime$, $maxDrivingTime$, and $WithoutBreak$. The break resource (B) is a binary resource equal to 1 if it is fulfilled. The break resource can only dominate if it is greater than or equal to the resource of another label. Furthermore, if WT + "drive time to the depot" exceeds the maximum work time, then we do not extend the label. If the required break has not yet been held, we do not extend the label if WT + "drive time to the depot" + "time of the required break" exceeds the maximum work time.

When solving the pricing problems, a large number of labels are created and extended at each node. A label with a higher resource consumption in either of the resources WT, DT, or TSB cannot dominate a label that has a lower consumption of just one of the resources. As a consequence, many of the labels with high resource consumption cannot be dominated, even if they appear to be 'bad' labels. If we can relax the dominance criteria so more labels can be dominated at each node, we can speed up the algorithm.

A label $\boldsymbol{a} \succ \boldsymbol{b}$ if all resources in $\boldsymbol{a}$ are dominating the resources in $\boldsymbol{b}$. In some cases we can improve this dominance rule by expiring some resources. The resource drive time (DT) can be expired - or set equal to 0 - if $(maxWorkingTime - WT) \leq (maxDrivingTime - DT)$. The label then dominates any other label with respect to DT. We can ignore the drive time and expire the resource because the duty will still be feasible if we drive for the remainder of the allowed working time. A similar argument holds for the time since break (TSB) resource: If $(maxWorkingTime - WT) \leq (maxTimeWithoutBreak - TSB)$, then we can expire the TSB resource in a similar fashion. These additional rules aid the dominance of more labels and thus speed up the algorithm.

5.4 Solving the Shortest Path Problem

Splitting the pricing problem into multiple smaller pricing problems creates a number of smaller problems to solve. In each iteration of the Branch-and-Price algorithm we have to prove that no duties with negative reduced costs

exist. Hence, it is necessary to solve all the individual pricing problems. In each intermediate iteration in the column generation it is only necessary to generate some columns with negative reduced costs. We do not need all possible columns nor the best. Thus, only a subset of the pricing problems need to be solved in each iteration. This subset is solved in parallel as the individual pricing problems are independent.

5.5 Branching

We branch in two stages, first for the timetable variables x, and second for the duties y. To reach an integer timetable we fix one $x_{d'}, \forall d' \in D'_d$ to 1 for all $d \in D$. Our branching approach has two ways of fixing this. First, we fix exactly one of the variables to 1 on the left branch and 0 on the right branch. Second, we split the set D'_d in two by the fractional values of the $x_{d'}$ variables, such that the fraction is equally divided on both branches.

The branching procedure first checks whether any $x_{d'}$ variables satisfy $0.8 \leq x_{d'} < 1, \forall d' \in D'$. If this is the case, we select the one closest to 1 for branching. If none of the $x_{d'}$ variables satisfy the criterion, the second rule is applied. In the second rule the $d \in D$ with the largest number of non-zero variables $x_{d'}, \forall d' \in D'_d$ is branched on.

To find integer solutions for the duties, we employ a classic follow-on branching scheme based on Ryan-Foster branching (Ryan and Foster, 1981) on the crew task. The implementation of this is similar to the implementation used in solving the TESP in Bach et al. (2014). The follow-on scheme ensures that a task a is always performed immediately before a task b on the left branch, and task b is never performed immediately after task a on the right branch.

6 Computational Experiments and Results

Experiments have been carried out on an Intel Core i7 2.8 GHz and implemented with C++ using ILOG CPLEX 12.4 as the linear programming solver, as a parallel algorithm using up to 8 threads.

In this section we will first discuss implementation details. Then, we describe the instances used, before we present results for the pure Crew Scheduling Problem (CSP), and then finally, we will compare these results to those of the Integrated Crew Scheduling Problem (I-CSP).

6.1 Implementation Details

Implementing and solving the model gives some challenges: Finding an optimal solutions to the I-CSP model in reasonable time is not possible. Therefore it is solved in two stages. First the timetable related variables are fixed, and then we move to the crew phase. As we will address later, the

LP-relaxation is very tight when the timetable is fixed. But letting the timetable be adjusted gives a quite weak LP-relaxation. Thus branching to get the fixed/integer timetable takes a long time. This is the main reason as to why we do not solve the model to optimality. Instead we branch on the timetable as described but add an acceptance criterion for accepting a node. We either pick the 1-branch if the new objective value is less than 0.01% worse than the predecessor. If this criterion is not met, we choose the branch with the lowest objective value. In this way we can reach a fixed/integer timetable in reasonable time and then move on to solving the crew part of the problem. This part is solved as for the pure CSP and is solvable in reasonable time as will be shown in Section 6.3.

6.2 Instances

Observing Table 2 we have the instances used to test the algorithms. These are based on the instances and the results from Bach et al. (2014). We

Table 2: Data instances

Instance	**TW**	**Engine:**			**Crew:**		**Excl.**
	width	**Avail.**	$\|D\|$	$\|D'\|$	$\|P\|$	$\|P'\|$	**sets**
CSP-90-24-s	6	24	90	650	462	3,424	637
CSP-90-30-s	6	30	90	772	462	3,947	713
CSP-90-30-ex3	9	30	90	1,032	462	5,287	779
CSP-180-24-s	6	24	180	1,131	956	6,059	1,118
CSP-180-30-s	6	30	180	1,312	956	6,903	1,265
CSP-180-30-ex3	9	30	180	1,779	956	9,712	1,586
CSP-228-24-s	6	24	228	1,511	1,214	7,905	1,555
CSP-228-30-s	6	30	228	1,745	1,214	9,104	1,711
CSP-228-30-ex3	9	30	228	2,295	1,214	12,355	2,024

have tested on three main groups of instances. The main parameter is the number of demands, which is the size of the set D. The second parameter is the number of engines available. We have two different test sets: one with 24 and one with 30 engines. If two test instances are equal in all parameters except the engines available, we will expect more dense schedules or higher utilization. This leads to less possible changes to the timetable when we modify the time windows. The set D' holds the different time-wise possibilities for serving the demands. The third parameter is the average width of a time window. In the standard "s" instances it is 6 hours, and in the extended time window instances "ex3", the time windows are extended by 3 hours, to 9 hours in total. For each demand in D, as described earlier, we split the demand into one or more tasks at relief points. The set P consists of all these tasks. The set P' holds all the crew tasks at different

times of service. Finally, in the table we have the Exclusion sets, which are sets containing more than one demand d' that cannot be part of the same solution.

6.3 Computational Results

In Tables 3 and 4 we report results for the CSP and I-CSP. In the tables the column headings are interpreted as follows: The gap is the gap from the integer solution to the lower bound for the crew phase (LB-C), to the root relaxation for the crew phase (Root-C), and to the root relaxation for the timetabling phase (Root-TT). An * means that an optimal solution was found. In Table 4 the gaps are given as the average gaps over all repetitions. Hence an * means that an optimal solution was found for each of the repetitions for that instance. We also report run times. Here Root-TT is the time to solve the timetabling root node. Int-TT / Root-C is the time to reach an integer timetable, which is the same as solving the root node for the crew phase (Root-C). Int-C is the additional time it takes to reach an integer solution for the crew. In Table 3 the column (Columns) states the total number of columns generated at the root node and during branching. The column (Branch nodes) gives the number of nodes explored in the branching tree.

In Table 3, we present test results for the CSP. From the table we can see

Table 3: Crew Scheduling Problem

Instance	**% Gap:**		**Columns:**		**Branch**	**Time (seconds):**		
	LB-C	**Root-C**	**Root**	**Branch**	**nodes**	**Root-C**	**Int-C**	**Total**
CSP-90-24-s	*	0.0174	5,552	807	23	15	20	26
CSP-90-30-s	*	0.0153	5,304	378	7	17	23	23
CSP-90-30-ex3	*	*	5,591	0	1	26	27	27
CSP-180-24-s	0.0538	0.0841	10,840	398,644	4,093	29	117	†
CSP-180-30-s	*	0.0115	10,202	35,646	388	35	88	484
CSP-180-30-ex3	0.0661	0.0903	10,918	406,115	3,812	59	175	†
CSP-228-24-s	0.0047	0.0240	12,840	394,961	2,041	51	239	‡
CSP-228-30-s	0.0436	0.0736	12,471	677,403	5,465	56	187	‡
CSP-228-30-ex3	0.0644	0.0821	12,876	665,440	4,916	88	297	‡

* indicate a zero gap, i.e., an optimal solution is found.
† is maximum runtime (3 hours) for medium sized instances (demand = 180).
‡ is maximum runtime (9 hours) for large sized instances (demand = 228).

that the model can be solved to proven optimality in reasonable time for the small instances that can be solved in less than 30 seconds. For the medium sized instances we can prove optimality for one instance. In general we get integer solutions very fast. In the worst case this is 3 minutes. Considering the real-life sized instances, we obtain feasible solutions for all of them within 5 minutes. These results indicate that the model scales rather well. This is primarily due to the structure of the pricing problems that are split into

days. Thus, adding extra demands can at worst affect two consecutive days. Hence, not all pricing problems increase in size when adding a demand.

Looking at the gaps, it can be seen that the root relaxation is a very strong lower bound on the integer solution. The worst root gap is less than 0.1%, a gap that is further narrowed to less than 0.07% when branching. In all cases this gap is so small that the root can be used as a very precise proxy for the objective value of the integer CSP solution. We exploit this in the two-phase approach described in the previous section. The results presented in Table 3 are used as benchmark solutions for evaluating the quality of the I-CSP solutions.

Table 4: Integrated Crew Scheduling Problem

Instance	**% Improvement:**			**% Gap:**			**Time (seconds):**		
	Avg	**Min**	**Max**	**LB-C**	**Root-C**	**Root-TT**	**Root**	**Int-TT**	**Int-C**
CSP-90-24-s	10.10	9.16	11.09	0.011	0.046	14.444	344	1512	24
CSP-90-30-s	12.63	11.87	13.23	0.013	0.059	14.929	494	2106	48
CSP-90-30-ex3	12.89	12.26	13.84	0.047	0.079	15.388	1165	3218	152
CSP-180-24-s	7.59	7.48	7.79	0.149	0.168	14.759	1103	5453	580
CSP-180-30-s	8.38	8.02	8.64	0.146	0.157	15.526	1636	7430	492
CSP-180-30-ex3	9.88	9.59	10.24	0.288	0.289	16.675	4810	14857	1865
CSP-228-24-s	7.11	6.28	7.62	0.188	0.195	15.544	2216	10442	779
CSP-228-30-s	7.42	6.88	8.08	0.145	0.152	16.377	3655	14922	1742
CSP-228-30-ex3	9.12	8.58	9.74	0.212	0.219	17.633	9856	26140	645

In Table 4, we present test results for the I-CSP. The solution method in itself is deterministic. However, as the pricing problems are solved in parallel, columns are inserted in different orders. This leads to different dual values and as a consequence, to different columns being generated in the following iteration. When branching, different branching decisions can be made due to having an alternative solution to the LP-relaxation. Because the algorithm is terminated before we explore the entire search tree, we might obtain a different solution in each repetition. Therefore, the table is based on multiple runs for each of the test instances. The table shows 6 repetitions for each instance.

For all instances a significant cost reduction is achieved in both worst and best case. Compared to the CSP, this cost reduction clearly shows that adjusting the timetable at the CSP phase is an advantage. Observing Table 4, it is clear that the cost reductions fluctuate for each instance, which is expected due to the heuristic nature of the solution method. We believe that this lies within an acceptable range. A longer runtime will lead to greater stability as more alternative timetables can be explored. When comparing improvements among different test instances, it should be considered that these are not necessarily comparable. Consider a given timetable with engine schedules and its CSP solution, in theory the timetable can be the timetable that is also optimal for the CSP. In this case we would have a 0% improvement for the I-CSP. This would not tell us anything about

the quality of the I-CSP method but rather that we were lucky to have the optimal timetable. Thus, when evaluating the consistency of the method, we compare among single instances. When evaluating the quality of the method, we note that for all test instances we get significant improvements, which shows that the method is useful.

From Table 4 it can be seen that improvements for the I-CSP are between 6.28% and 13.84%. The smallest and largest average improvement are 7.11% and 12.89%, respectively. From the results it is clear that the I-CSP delivers significant improvements compared to the benchmark cases. Observing the data sets with 90 demands, it can be seen that the least flexible *CSP-90-24-s* is also the one with the lowest average improvement, compared to *CSP-90-30-s* and *CSP-90-30-ex3*. The tendency that the improvement is linked to the degree of flexibility is also present for the instances with 180 and 228 demands. It is a tendency that strongly suggests that the improvements found stem from the degree of flexibility.

Another tendency suggests a correlation between the size of the demand and the improvement. This can be explained by the complexity of the underlying Timetable and Engine Scheduling Problem. Here the schedules are less compact in the smaller instances, and thus there is more flexibility left for the I-CSP. It could also suggest that the method is able to exploit the flexibility better on smaller instances. From Table 2 it can be seen that there is more flexibility for each crew task and demand for the smaller than the larger instances.

To ensure that the stability of the method is consistent, we have tested the I-CSP approach with further repetitions on selected instances *CSP-228-30-s* and *CSP-228-24-s*. Here 12 repetitions has been performed without changing the conclusions of Table 4.

7 Conclusions and Future Research

In this paper we set out to investigate to what extent the Crew Scheduling Problem (CSP) can be integrated into the earlier planning stages at a freight railway operator. We have presented a method for integration and a model for the Integrated Crew Scheduling Problem (I-CSP). We have suggested a heuristic Branch-and-Price approach to solve the I-CSP. The suggested algorithm has been tested on a set of test instances derived from a real-life case at a freight railway operator. The results from the I-CSP have been compared to a set of benchmark results obtained by solving a standard CSP model. The comparisons show that the proposed I-CSP method gives improvements from 6.28% to 13.84% for the different instances.

There are several interesting paths for future research. First, it has been shown that the lower bound obtained from LP-relaxation to the I-CSP is weak and finding it is time consuming. This affects the branching process

and makes it impossible to close the optimality gap for real-life instances. By identifying key demands where flexibility is important and demands where flexibility is less or not important, it could be possible to search among a larger set of the good solutions. This could be implemented by examining the quality of the duties and only allowing changes to demands covered by low quality duties.

Second, we have shown that added flexibility greatly benefits the overall solution. Additional flexibility can be introduced by the way the time windows are split. The current algorithm ensures that the decision on re-timing of one demand is completely independent of others. If instead we allow the full, original, time window to be used, while adding precedence constraints on the demands in the same engine schedule, we would achieve additional flexibility.

References

Abbink, E., L. Albino, T. Dollevoet, D. Huisman, J. Roussado, and R. Saldanha (2011). Solving large scale crew scheduling problems in practice. *Public Transport 3*(2), 149–164.

Bach, L., M. Gendreau, and S. Wøhlk (2014). Freight railway operator timetabling and engine routing. In *Lukas Bach, Routing and Scheduling Problems - Optimization using Exact and Heuristic Methods*, Ph.D. Thesis, Chapter 3, pp. 73–107. Aarhus University.

Caprara, A., M. Fischetti, and P. Toth (1999). A heuristic method for the set covering problem. *Operations Research 47*(5), 730–743.

Caprara, A., L. Kroon, M. Monaci, M. Peeters, and P. Toth (2007). Passenger railway optimization. In C. Barnhart and G. Laporte (Eds.), *Transportation*, Volume 14 of *Handbooks in Operations Research and Management Science*, pp. 129–187. Amsterdam: Elsevier.

Elhallaoui, I., D. Villeneuve, F. Soumis, and G. Desaulniers (2005). Dynamic aggregation of set-partitioning constraints in column generation. *Operations Research 53*(4), 632–645.

Freling, R., D. Huisman, and A. P. M. Wagelmans (2003). Models and algorithms for integration of vehicle and crew scheduling. *Journal of Scheduling 6*(1), 63–85.

Gintner, V., N. Kliewer, and L. Suhl (2008). A crew scheduling approach for public transit enhanced with aspects from vehicle scheduling. In M. Hickman, P. Mirchandani, and S. Voss (Eds.), *Computer-aided Systems in Public Transport*, Volume 600 of *Lecture Notes in Economics and Mathematical Systems*, pp. 25–42. Springer Berlin Heidelberg.

Groot, S. and D. Huisman (2008). Vehicle and crew scheduling: Solving large real-world instances with an integrated approach. In M. Hickman, P. Mirchandani, and S. Voss (Eds.), *Computer-aided Systems in Public Transport*, Volume 600 of *Lecture Notes in Economics and Mathematical Systems*, pp. 43–56. Springer Berlin Heidelberg.

Huisman, D. (2007). A column generation approach for the rail crew re-scheduling problem. *European Journal of Operational Research 180*(1), 163–173.

Huisman, D., R. Freling, and A. P. M. Wagelmans (2005). Multiple-depot integrated vehicle and crew scheduling. *Transportation Science 39*(4), 491–502.

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

Jütte, S., M. Albers, U. W. Thonemann, and K. Haase (2011). Optimizing railway crew scheduling at DB Schenker. *Interfaces 41*(2), 109–122.

Jütte, S. and U. W. Thonemann (2012). Divide-and-price: A decomposition algorithm for solving large railway crew scheduling problems. *European Journal of Operational Research 219*(2), 214–223.

Kliewer, N., B. Amberg, and B. Amberg (2012). Multiple depot vehicle and crew scheduling with time windows for scheduled trips. *Public Transport 3*(3), 213–244.

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

Kwan, R. S. and A. Kwan (2007). Effective search space control for large and/or complex driver scheduling problems. *Annals of Operations Research 155*(1), 417–435.

Lusby, R., J. Larsen, M. Ehrgott, and D. Ryan (2011). Railway track allocation: models and methods. *OR Spectrum 33*(4), 843–883.

Mesquita, M., A. Paias, and A. Respício (2009). Branching approaches for integrated vehicle and crew scheduling. *Public Transport 1*(1), 21–37.

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

Rezanova, N. J. and D. M. Ryan (2010). The train driver recovery problem - a set partitioning based model and solution method. *Computers & Operations Research 37*(5), 845–856.

Ryan, D. M. and B. A. Foster (1981). An integer programming approach to scheduling. In A. Wren (Ed.), *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling*, pp. 269–280. North-Holland.

Steinzen, I., V. Gintner, L. Suhl, and N. Kliewer (2010). A time-space network approach for the integrated vehicle- and crew-scheduling problem with multiple depots. *Transportation Science 44*(3), 367–382.

Veelenturf, L. P., D. Potthoff, D. Huisman, and L. G. Kroon (2012). Railway crew rescheduling with retiming. *Transportation Research Part C 20*, 95–110.