Final version is available at InformsPubsOnline : http://dx.dou.org/10.1287/trsc.2015.0648

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

[†]Department of Applied Mathematics
SINTEF ICT, Oslo, Norway
Lukas.Bach@sintef.no

[‡]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

Abstract

We investigate to what degree we can integrate a Train Timetabling / Engine Scheduling Problem with a Crew Scheduling Problem. In the Timetabling / Engine Scheduling Problem we determine for each demand a specific time within its time window when the demand should be serviced. Furthermore, we generate engine duties for the demands. In our solution approach for the overall problem, we first obtain an optimal solution for the Timetabling / Engine Scheduling Problem. When solving the Crew Scheduling Problem, we then exploit the fact that numerous optimal, and near optimal solutions exist for the previous problem. We consider all these solutions that can be obtained from the optimal engine schedule by shifting the demands in time, while keeping the order of demands in the engine duties intact. In particular, in the crew scheduling stage it 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. 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.

[*]Corresponding author

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

1 Introduction

In most European countries, freight trains are operated in between passenger trains. Therefore, a joint timetable for all trains is constructed. In this timetable, the arrival and departure times of the trains at all stations in the network are specified. Furthermore, time-slots for freight trains are included. Such a time-slot represents the right for a railway operator to drive from one station to another at a specific time. Freight railway operators such as DB Schenker Rail Scandinavia (DBSRS) have to request and pay for these time-slots. The price for the time-slots may differ. Usually, time-slots during the peak hours for passenger traffic (e.g. between 7 and 9am, and 4 and 6pm) are more expensive. Since the contracts between the operator and its customers cover in general a period of one year, each week the same time-slots are requested. The whole set of requested time-slots can be seen as the timetable for the freight operator. This consists of a set of trips: movements from A to B with given departure and arrival times.

The planning process of a freight railway operator consists of several steps. The timetabling problem for the operator is to decide which time-slots to request. Furthermore, the operator has to decide for each demand, which time-slot is used to service it. By doing so, the departure and arrival times for the demands are determined, as well. In the Engine Scheduling Problem, engines are assigned to the different demands in accordance with the timetable. Bach et al. (2015) solved an integrated version of the Timetabling and Engine Scheduling Problem (TESP). That is, the time-slots are not optimized first, but are still open and optimized together with the engines.

In this paper, we seek to integrate the TESP with the Crew Scheduling Problem (CSP). The overall goal is to investigate whether it is favorable to leave some aspects of the earlier stages open to adjustments in the Crew Scheduling phase. We seek to integrate by allowing changes to the timetable when solving the CSP. 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 simple example below illustrates the possible advantages.

> **Example:** In this example, the following time-slots are available.

A-B	1-2pm	5-6pm	9-10pm
B-C	3-4pm	7-8pm	11-12pm
C-B	1-2pm	5-6pm	9-10pm
B-A	3-4pm	7-8pm	11-12pm

These time-slots are depicted in Figure **??**(a). The costs for the time-slots starting at 5pm is 100 higher than for the other time-slots. There are two demands from A to C. One of those has to be transported from A at 1pm. The other can be either transported at 5pm or at 9pm. Similarly, there are two demands from C to A. The first one has to be transported from C at 1pm, and the second one can be either transported at 5pm or 9pm.

The optimal solution of the TESP is depicted in Figure **??**(b). The time-slots at 1pm and 9pm are selected in both directions. The first engine starts in A at 1pm, arrives at and departs from B at 2pm and 3pm, respectively, and arrives in C at 4pm. Then it returns at 9pm to A, arriving there at 12pm. Similarly, the second engine starts in C at 1pm and returns in C at 12pm.

For the crew scheduling problem, this results in 8 tasks. Suppose that the only labor rules are a maximum working time of 8 hours and the requirement that at the end of the duty each crew member should be back at its base. There are only fixed cost, which are 1,000 per duty. Given the solution to the TESP, the optimal solution contains 4 duties. This solution is depicted in Figure **??**(c). The crew costs are 4,000.

However, the overall optimal solution contains only 2 duties. This solution is depicted in Figure **??**(d). It selects the more expensive time-slots at 5pm instead of those at 9 pm. The resulting cost increase is outweighed by the savings in crew costs. The total costs are reduced by 1,800 (2*1000 - 2*100) in comparison to the sequential solution. In this specific example, the crew and engine duties are equal.

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 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.

A set of time-slots for freight operators are scheduled in the timetable and represent the right to access the railway infrastructure between two stations at a certain time. 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. 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 duties, each containing a set of demands to perform during a week. A specific engine can perform different engine duties from week to week. This can lead to an engine starting and ending at different stations in the beginning and end of the week. However, this must be balanced over all engines. Furthermore, we distinguish between several types of engines. Some demands can only be serviced by specific engines. For example, if a certain track is not electrified, demands via that track cannot be serviced by electrical engines. Each engine duty must satisfy the following constraints: (1) It can only use tracks when allowed to do so by a time-slot. (2) There must be a certain minimum time between the beginning of a demand and the end time of the previous demand. This minimum time is input for the engine scheduling problem and can be seen as a buffer time to improve robustness of the schedule. (3) The engine type must be suitable for all demands in the duty.

The network that we consider is from DBSRS and is spread over three countries: Germany, Denmark and Sweden. The crew 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. For a detailed description of the working regulations, we refer to Bach (2014). Within each country, the crew are furthermore assigned to crew bases located in their country. The crew should always end their duty at their home base, but can travel anywhere otherwise allowed by the working regulations. If a driver crosses a border, EU rules are enforced.

As part of any duty there is a set of mandatory tasks. These are sign-on and sign-off tasks. In order to have a break, start, or end a duty, it is necessary for crew to walk to/from a break room. This walking time is dependent on the station used, and it is added to the duration of the task. Furthermore, a maximum duty length, a minimal break length and

a maximum time without a break are considered. The cost of a duty is modeled as a fixed part and variable part. The variable part consists of a time-dependent cost, costs for night duties, and costs for cross-border activities.

The crew scheduling problem then becomes to cover all crew tasks during a week at the minimum cost using the crew from the three different countries. In the integrated Timetabling, Engine and Crew Scheduling Problem, we have to find simultaneously a schedule for the engines and the crew and decide about the time-slots to request. The goal is to minimize overall costs while satisfying all constraints of the individual problems discussed before. In doing so, we make sure that the engine schedule and the crew schedule are compatible: Both an engine and a crew member must be available at the time when the demand is being served.

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). The Crew Scheduling Problem (CSP) is usually solved when the timetable and the engine schedule have been 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. (2015) for more details 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. 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.

Note that there are several papers (e.g. Freling et al. (2003)) where different types of crew duties are taken into account. However, to the best of our knowledge, no papers consider the dependency between the crew base, the performed work (cross-border or not) and the applicable rules.

In Freling et al. (2003) the integration of single-depot Vehicle and Crew Scheduling is studied. Huisman et al. (2005) extend their model to multiple depots and focus on a sub/extraurban transit system. They show that due to the integration significant cost savings can 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, but the application is focussed on transit scheduling.

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). In contrast to our model, the authors represent the vehicle schedule as a network flow. The solution methodology is tested on real-life data from public bus transportation. A similar approach to re-timing in the CSP 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. In particular, our approach allows to change the timetable when solving the CSP, but leaves the order of the demands in the engine duties unchanged. Besides the work by Kliewer et al. (2012), we are not aware of any other paper considering timetabling, vehicle, and crew scheduling simultaneously. Kliewer et al. (2012) also allow some degree of re-timing, but integrate the vehicle and crew scheduling problems fully. Furthermore, in their model, small changes of up to 4 minutes in the timetable are considered, where the changes we consider are much larger. As a consequence, our duties might change more considerably if a demand is re-timed. Furthermore, our model is for a railway system whereas their model is for a public bus transit system. The most significant difference with a bus transit system is that the access to the railway infrastructure has to be considered, which means that re-timing is limited to selecting other time-slots. Also, the costs of the time-slots are taken into account.

4 Mathematical Formulations

In order to assess the benefit of integrating the timetabling and the crew scheduling phase, we are going to compare the integrated approach to the sequential approach. In the sequential approach, we solve the CSP using a fixed timetable that is found by the TESP. By comparing this to our integrated approach, we can evaluate the possible quality improvements obtained by integration. In this section, we first describe the central aspect of our integrated approach. Then, we describe the TESP, the regular CSP and our integrated model. The regular CSP assumes a fixed timetable as input and is used in the sequential approach. Our integrated approach, on the contrary, only needs the engine schedule from the TESP as input.

The following observation is central to our integrated methodology. Recall from Section 2 that a time window is associated with each demand that has to be serviced. Furthermore, as described in the introduction, a train can only be operated when a time-slot is available. As a consequence, for each demand, a discrete set of departure options can be derived. To illustrate this using the example in Section 1, assume that the time window of the second demand from A to C prescribes that the demand should arrive at C between 6pm and 12pm. There are two trains from A arriving at C within this time window. It follows that the demand should either depart from A at 5pm, or at 9pm. A similar discrete set of departure options can be derived for every demand. Instead of selecting one departure option for each demand before optimizing the engine and crew schedules, we postpone the decision for a specific departure option. All departure options are taken into account when solving the TESP. Then, given the engine schedule that has

been found, the sets of departure options are reduced. These reduced sets of departure options are considered when optimizing the crew. We explain this in more detail in Section 4.3.

4.1 Timetabling and Engine Scheduling Problem

Both the TESP and the CSP are solved using a column generation approach. The TESP selects the time-slots to request. It does this while generating a set of weekly engine duties, that can be combined to cover all demands. The engine duties do not necessarily start and end at the same station but the complete schedule overall is balanced at each station. The TESP is formulated and solved as in Bach et al. (2015). There are no decision variables that explicitly define the selection of the time-slots. Instead, the selected time-slots are extracted from the engine duties chosen in the model.

The TESP is formulated as a Set Partitioning Problem, with Ω_E being the set of engine duties, indexed by 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. The parameter w_e denotes the engine availability for engine type e.

The following parameters characterize an engine duty: K_r is the cost of duty r. The parameter η_r^d indicates whether demand d is covered by duty r. In the engine duty, the demand is scheduled at a specific time. Therefore, the time-slots that must be requested can be deduced from the engine duties that are selected in the final solution. β_r^n is equal to 1 if a station n is used in duty 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 duty is balanced with respect to this station. If time-slot s is used by duty r, then γ_r^s is equal to 1. We set δ_r^e equal to 1 if duty r is driven by engine type e. Finally, we introduce the binary decision variable u_r, that indicates whether engine duty r is selected in the engine schedule ($u_r = 1$) or not ($u_r = 0$). The integer program is formulated as follows:

$$\min \sum_{r \in \Omega_E} K_r u_r \tag{1}$$

$$\text{s.t.,} \sum_{r \in \Omega_E} \eta_r^d u_r = 1, \quad \forall d \in D \tag{2}$$

$$\sum_{r \in \Omega_E} \delta_r^e \beta_r^n u_r = 0, \quad \forall n \in N, e \in E \tag{3}$$

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

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

$$u_r \in \{0, 1\}, \quad \forall r \in \Omega_E \tag{6}$$

The objective function (1) minimizes the cost of the selected engine duties. 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 engine duties, we have the balance constraints (3). These constraints ensure that for each station n and engine type e, the number of engines of type e departing from n equals the number of engines of type e arriving there. In constraints (4) we ensure that each time-slot is used for servicing at most one demand. Engine availability is ensured by constraints (5). For details on the solution method for the TESP, we refer to Bach et al. (2015).

4.2 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. The input for the CSP contains a set of demands $d \in D$. Each demand d represents a set of wagons that must be transported 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_C is the set of all crew duties (columns) in the restricted master problem, indexed by r. The cost of a duty r is given by c_r; the parameter α_r^p indicates if crew task p is covered by duty r. We define the parameter EDC as the engines' driver capacity. For example, if $EDC = 2$, there can be 1 engine driver and 1 deadheading driver in the engine. We introduce the binary decision variable y_r that indicates whether a crew duty $r \in \Omega_C$ is selected ($y_r = 1$) or not ($y_r = 0$). The integer program is formulated as follows:

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

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

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

The objective function (7) minimizes the cost of the selected duties. Constraints (8) ensure that all tasks are covered by at least one duty and that at most two drivers are scheduled on each specific task.

4.3 The Integrated Crew Scheduling Problem

When solving the TESP as in Bach et al. (2015), we obtain a fixed departure time for each demand and a set of engine duties. However, in this paragraph we explain that there are a number of alternative feasible solutions. In the integrated approach, we seek to explore these alternative solutions by allowing re-timing of the individual demands when solving the CSP. When re-timing the demands in the CSP, we make sure that the engine schedule remains feasible. In particular, each engine duty will contain the same demands in the same order. However, the precise departure times of the demands might change.

In an optimal solution to the TESP, a demand is serviced at a point in time within its time window, and by a specific engine duty. An engine duty services one or more demands. There can be large time intervals between two demands in a duty. As a consequence, it holds for some demands, that the engine duty remains feasible if the departure time of that demand is changed. We say that a demand can move freely within its time window if all of its original departure options are still available. If the engine duties contain much slack, this might hold for many demands. The departure times of the other demands are either fixed completely or can only vary among a restricted part of the original set of options. We will explain this in more detail in the following paragraph.

In Figure 1, we depict a typical engine duty above the horizontal line. Demands are indicated by numbered rectangles. The lines indicate the time windows of the demands. In the current solution, there is a large time interval between the end time of demand 2 and the start time of demand 4. Within this time interval, only demand 3 has to be serviced. Therefore, it is possible to re-time the service of demand 3 to another moment. As can be seen in Figure 1, demand 3 can be serviced anywhere within its time window, as long as demands 2 and 4 are fixed. Similarly, there is some time between the end time of demand 4 and the start time of demand 5. For demand 4, we cannot choose among all of the original departure options,

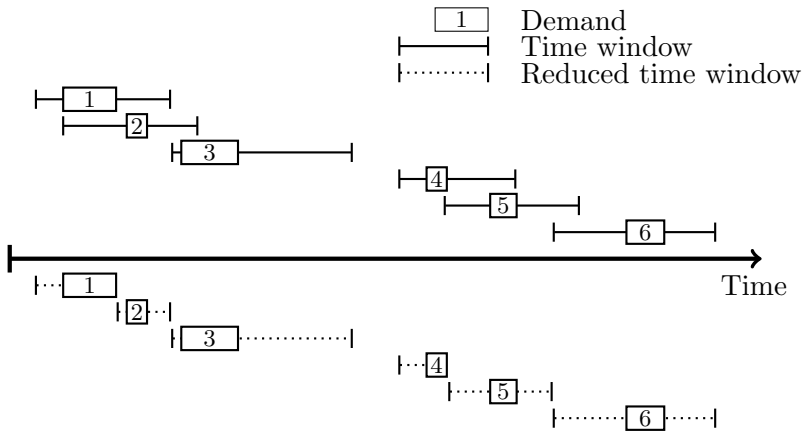

Figure 1: Time window reduction: On top, original time windows are shown for six demands in an engine duty. Below, for those demands, the reduced windows are depicted with dotted lines.

but it is possible to select any earlier time-slot. In our integrated approach, we allow changes to the exact timing of the demands when solving the CSP. We do this in such a way, that the engine duty remains feasible. We now explain how to do that.

To make sure that a feasible engine schedule still exists after solving the CSP, we do not consider all departure options in the crew scheduling phase. Instead, we define *reduced* time windows for all demands. We define these reduced time windows in such a way, that whatever combination of departure options we choose within these reduced time windows, the engine duty will always remain feasible. In the bottom of Figure 1, the reduced time windows are depicted by dotted lines. We now explain how we determine the reduced time windows. A time window remains unchanged if there is no overlap with the time windows of the previous and next demand in the engine duty. Otherwise we start from the departure option selected in the optimal solution. This departure option is always included in the reduced time window. We add the later departure options as long as they do not overlap with the time window of the next demand. If the departure option does overlap with the time window of the next demand, this departure option is not included in the reduced time window. Then, we consider earlier departure options. We add earlier departure options as long as they do not conflict with the departure option selected in the current TESP solution. To summarize the procedure, the time windows are extended to the right until the start of the next time window and to the left until the scheduled end time of the previous demand.

We now allow to select any departure option from the *reduced* time window when solving the CSP. By construction of the reduced time windows,

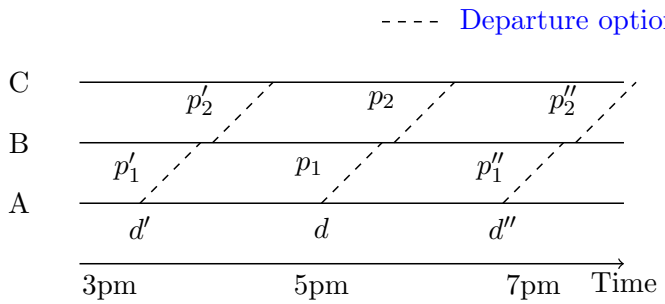

Figure 2: The alternative departure options d' and d'' and the corresponding crew tasks for a specific demand d.

the engine duty will remain feasible whatever departure times are selected in the CSP. A drawback of this approach is that less flexibility is available when solving the CSP. Because only a part of the departure options are available when solving the CSP, we say that the timetabling and crew scheduling phase are *partially* integrated. From now on, we refer to the partially integrated Crew Scheduling Problem as the I-CSP.

4.4 Formulation of I-CSP

As explained in the previous section, we allow to select any of the departure options in the reduced time windows when solving the CSP. In contrast to the TESP, in the CSP we do include explicit decision variables that indicate which departure option is selected. In order to integrate the timetabling aspect with the CSP, we must connect the timing of the demands with the selected crew duties. We do this by taking the set of demands D. For each d we take the discrete set of departure options in the *reduced* time window and make a copy of d for each departure option. We assign these copies d' to the set D'_d. The set D'_d now contains a copy for each departure option of d, that is, for each time it is possible to start service of demand d. An example is depicted in Figure 2. The departure of a demand d from A to C is scheduled at 5pm. However, two more departure options are available in the reduced time window: a departure at 3pm and a departure at 7pm. These departures are depicted as d' and d'', respectively. 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.

Each demand d corresponds to one or more tasks p. If we select a different departure option for a demand d, we also have to cover the crew tasks at different times. For each d' we 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 the example in Figure 2, the demand d is decomposed in two crew tasks p_1 and p_2. Similarly, the demand copies d' and d'' are decomposed in two crew tasks.

We link the demand copies $d' \in D'$ to the crew tasks, so that the time chosen for service with the engine always corresponds to the time chosen for the crew tasks. In order to do so, we define the parameter $\gamma_r^{p'}$ which is equal to 1 if task copy p' is used by crew duty r, and 0 otherwise. In contrast, the parameter α_r^p indicates whether duty r covers *any copy* of task p. Finally, recall from Section 4.2 that EDC represents the number of drivers that can be present on an engine. The integer program is formulated as follows.

$$\min \sum_{r \in \Omega_C} c_r y_r + \sum_{d' \in D'} c_{d'} x_{d'} \tag{10}$$

$$\text{s.t.,} \sum_{d' \in D'_d} x_{d'} = 1, \quad \forall d \in D \tag{11}$$

$$\sum_{d' \in M_s} x_{d'} \leq 1, \quad \forall s \in \{S : |M_s| > 1\} \tag{12}$$

$$\sum_{r \in \Omega_C} \alpha_r^p y_r \geq 1, \quad \forall p \in P \tag{13}$$

$$EDC x_{dem(p')} - \sum_{r \in \Omega_C} \gamma_r^{p'} y_r \geq 0, \quad \forall p' \in P' \tag{14}$$

$$y_r \in \{0,1\}, \quad \forall r \in \Omega_C \tag{15}$$

$$x_{d'} \in \{0,1\}, \quad \forall d' \in D' \tag{16}$$

In the objective function (10), we minimize the sum of the crew costs and the additional costs of the time-slots. Recall that the costs of the time-slots are time-dependent. For example, time-slots in the peak hours might be more expensive. Hence, it is necessary to introduce the additional costs of choosing a more expensive time-slot. These additional costs are captured by $c_{d'}$. Constraints (11) make sure that one departure option is selected for all demands. Some demands use the same time-slots if serviced on specific times. When allowing changes to the departure times of the demands, 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. Constraints (12) ensure that no more than one of the demand copies using the same time-slot is selected. Constraints (13) are the set covering constraints and were also present in the CSP. They ensure that each demand is covered by a crew duty. Finally, constraints (14) ensure that $x_{dem(p')}$ attains a positive value if a corresponding task copy is part of a selected duty. This ensures that the departure option that is selected is also the one that is covered by the crew duties.

A stronger bound can be achieved by adding the constraints in (17). The parameter EDC in (14) 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. The following constraints ensure that all tasks in a demand are serviced at least with the same fraction as the corresponding demand.

$$\sum_{r \in \Omega_C} \gamma_r^{p'} y_r - x_{dem(p')} \geq 0, \quad \forall p' \in P' \tag{17}$$

However, this is not part of our model as it also slows down the solution process significantly.

4.5 Extension: Re-timing at intermediate stations

We explained in Section 2 that every demand should be serviced as a direct train from its origin to its destination. This means that once a demand has departed, it must get to its destination as fast as possible. In cases where direct trains are not required, it might be possible to decrease costs even further by re-timing demands at intermediate stations along their route. In this way, demands can be scheduled more flexibly. Our model can handle this situation, if one set of constraints is changed and one set of constraints is added. First, the data must be adjusted. All demands must be cut at intermediate stations where it is possible to park a train. By doing so, a demand d is decomposed into a set of trips T_d that must be performed sequentially. We assume that all stations where trains can be parked are a relief point for the crew. In this way, the trips decompose into one or more crew tasks similarly as the demands did in Section 4.2. We define T as the union over $d \in D$ of the sets T_d. Then, for all trips $t \in T$, a discrete set of copies t' must be generated. The copies t' are similar to the copies d' of d. We define T'_t as the set of all copies t' of trip t. Finally, in the master problem, we must first select for each trip one of its copies. This means that $x_{d'}$ is replaced by $x_{t'}$ and that (11) is replaced by

$$\sum_{t' \in T'_t} x_{t'} = 1, \quad \forall t \in T. \tag{18}$$

Second, we must ensure that compatible copies are selected. We explain this with the example in Figure 2. Here, for a demand d from A to C, two alternative departure options are depicted. Assume now that it is allowed to park the train at station B for some time. Then the demand d can be decomposed similarly as the crew tasks. For each departure option d, there are now 2 trips: t_1 and t_2. For each trip, we must select one of the copies. It is now not required to select trips originating from the same demand. For example, we can combine trip t'_1 with trip t_2. On the contrary, it is not

possible to combine trip t_1 with trip t'_2. When scheduling the crew, we must ensure that the copies of t that are selected are compatible. We model this by introducing a set C of incompatible copies. For each trip t'_i, we define the set $C(t'_i)$ as those trips t'_{i+1}, that are incompatible with t'_i. In the example, we find $C(t''_1) = \{t'_2, t_2\}$ and $C(t_1) = \{t'_2\}$. We now introduce the following constraints in the master problem to make sure that compatible copies are selected.

$$x_{t'} + \sum_{t'' \in C(t')} x_{t''} \leq 1 \quad \forall t' \in T'. \tag{19}$$

These constraints are similar to the ones used by Veelenturf et al. (2012) in the Crew Rescheduling Problem to model the option of re-timing tasks. Note that in general, a trip can decompose in more than one crew tasks. In that case, Constraint (14) makes sure that the corresponding crew tasks are compatible, too.

We refer to the Integrated Crew Scheduling Problem with re-timing at Intermediate Stations as the I-CSP-IS.

5 Solution Approach

To establish a benchmark in order to see how much the solution to the CSP can be improved by our integrated 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 the TESP. This corresponds to the sequential approach where we first solve the TESP and then the CSP. In order to quantify the improvement obtained by integration, we compare the solution value of the I-CSP to a lower bound on the sequential solution value. In this way, we can be sure that the improvement stems from the integration, and not from our inability to solve the CSP to optimality. Thus, we try to increase the lower bound on the solution value of the sequential approach by branching in a breadth first manner. 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 sub-optimal time-slots in the integrated approach.

In the integrated approach we let the timetable variables x flow freely within the reduced 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. For more details on our implementation, we refer to Bach (2014).

5.1 Pricing Problem Splitting

The pricing problem for duty generation is modeled as a Shortest Path Problem with Resource Constraints. The nodes in the graph are the tasks that have to be performed. Two tasks a and b are connected by an arc if it is possible for a crew member to perform task b directly after performing task a. The network that we obtain in this way is cyclic. 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.

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. When drivers cross borders they must adhere to a set of EU working rules. By solving a separate pricing problem where we allow the drivers to cross borders, we still have a distinct set of working rules that is independent of the decisions in that pricing problem. In a similar fashion, we generate different pricing problems for generating night duties and general duties. In total with the rules applied, we get about 350 pricing problems.

5.2 Pricing Problem Graphs

In the pricing problem we solve a Shortest Path Problem with Resource Constraints (SPPRC, see Feillet et al. (2004)) 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.

We define a cycle as a duty performing copies of the same task at two different times. This is an infeasible duty, but it can be generated because

in the pricing problem we do not keep track of the copies that are selected in the duty. To reduce the probability of cycles in the duties, the following rules are enforced prescribing which nodes can be connected. If we do not allow any node to connect to any other node that represents a copy of the same node 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 deriving from the same engine demand but at an incompatible time. Finally, we can remove connections that will not be feasible with respect to the time-slots in the master problem. Constraints (12) indicate that at most one demand can be serviced in each time-slot. Some copies of different demands might use the same time-slot. In such cases, only one of these demand copies can be chosen. Then, we do not allow connections between the two demands. Equivalently, servicing tasks from different demands that will use the same time-slot simultaneously, cannot be part of an integer solution.

5.3 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 balanced over 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. (2015). 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.

5.4 Exploration of the search tree

Implementing and solving the model gives some challenges: Finding an optimal solution to the I-CSP model in reasonable time is not possible. Therefore, the problem is solved in two stages. First, the timetable related variables are fixed, and then we continue to variables from the crew phase. As we will address later, the LP-relaxation for the CSP is very tight when the

timetable is fixed. On the contrary, if the timetable can be adjusted, the LP relaxation is quite weak. Thus branching to get the fixed/integer timetable takes a long time. This is the main reason 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 continue 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 6.2.

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 for multiple repetitions with the same data. For the classical CSP the gap shows to be very small and hence this is not a problem. For the I-CSP variants the gap is larger and therefore we have performed multiple repetitions for each instance to provide a fair comparison.

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 mixed integer programming solver, as a parallel algorithm using up to 8 threads.

In this section we describe the instances used. Then, we present results for the pure Crew Scheduling Problem (CSP). Finally, we will compare these results to those for the Integrated Crew Scheduling Problem (I-CSP) and the Integrated Crew Scheduling Problem with re-timing at Intermediate Stations (I-CSP-IS).

6.1 Instances

In Table 1, we report the main characteristics of the instances used to test the algorithms. These are based on the instances and the results from Bach et al. (2015). We 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 used in the TESP solution. We have two different test sets: one with 24 and one with 30 engines. Please note that the engine duties are a fixed input to the I-CSP model and we do not re-optimize the number of engines used. We distinguish between 3

Table 1: The main characteristics of the instances

Instance	TW width	Engine: Used	$\|D\|$	$\|D'\|$	Crew: $\|P\|$	$\|P'\|$	Excl. 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

engine types. If two test instances are equal in all parameters except in the number of engines available, the same number of tasks is covered by a different number of engine duties. If the number of engine duties decreases, the engine duties contain more tasks on average. Thus, the engine duties are more dense and the engines have a higher utilization. This leads to less departure options in the reduced time windows. The set D' contains the different departure options for serving each demand.

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. The naming of the instances follows the convention CSP-"$|D|$"-"#Engines used"-"TW width". 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. The instances are ordered first by $|D|$ and second by $|D'|$. These parameters are assumed to reflect the difficulty of an instance.

6.2 Computational Results

In Tables 2, we report results for the CSP. In this table 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) or to the root relaxation for the crew phase (Root-C). An * means that an optimal solution was found. We also report computation times. Root-C is the time to solve the root node for the crew phase. Int-C is the time it takes to reach an integer solution for the crew. 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.

Table 2: 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 computation time (3 hours) for medium sized instances (demand = 180).
‡ is maximum computation time (9 hours) for large sized instances (demand = 228).

From the table we can see 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 within 3 minutes. Considering the instances with $|D| = 228$, which are 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 gaps are the total cost of the integer solution compared to the total cost of the root / lower bound solution. For the root of the CSP (Root-C) it is calculated as $\frac{(`Int-C' - `Root-C')}{`Root-C'}$. The worst (Root-C) gap is less than 0.1%, and the (LB-C) gap is in the worst case further narrowed to less than 0.07% when branching. In all cases these gaps are very small. We exploit this in the search tree exploration approach of the I-CSP as described in Section 5.4. The CSP has an objective value very close to the optimal value even with fractional crew schedules. Because of this we assume that when branching on the timetable variables in the I-CSP the objective value will be a good indication of the impact a branch decision will have on the crew cost.

The results presented in Table 2 are used as benchmark solutions for evaluating the quality of the I-CSP solutions.

In Table 3, we report results for the I-CSP. In this table, 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

Table 3: Integrated Crew Scheduling Problem

Instance	% Gap:			% Saving:			Time (seconds):		
	LB-C	Root-C	Root-TT	Avg	Min	Max	Root-TT	Int-TT	Int-C
CSP-90-24-s	0.011	0.046	14.444	10.1	9.16	11.09	344	1512	1536
CSP-90-30-s	0.013	0.059	14.929	12.63	11.87	13.23	494	2106	2154
CSP-90-30-ex3	0.047	0.079	15.388	12.89	12.26	13.84	1165	3218	3370
CSP-180-24-s	0.149	0.168	14.759	7.59	7.48	7.79	1103	5453	6033
CSP-180-30-s	0.146	0.157	15.526	8.38	8.02	8.64	1636	7430	7922
CSP-180-30-ex3	0.288	0.289	16.675	9.88	9.59	10.24	4810	14857	16722
CSP-228-24-s	0.188	0.195	15.544	7.11	6.28	7.62	2216	10442	11221
CSP-228-30-s	0.145	0.152	16.377	7.42	6.88	8.08	3655	14922	16664
CSP-228-30-ex3	0.212	0.219	17.633	9.12	8.58	9.74	9856	26140	26785

phase (Root-TT). For the computation times, Root-TT is the time to solve the timetabling root node. Int-TT is the time to reach an integer timetable, which is the same as solving the root node for the crew phase (denoted by Root-C in Table 2). Int-C is the time it takes to reach an integer solution for the crew. As described in Section 5.4, the table is based on multiple runs for each of the test instances. The table shows 6 repetitions for each instance.

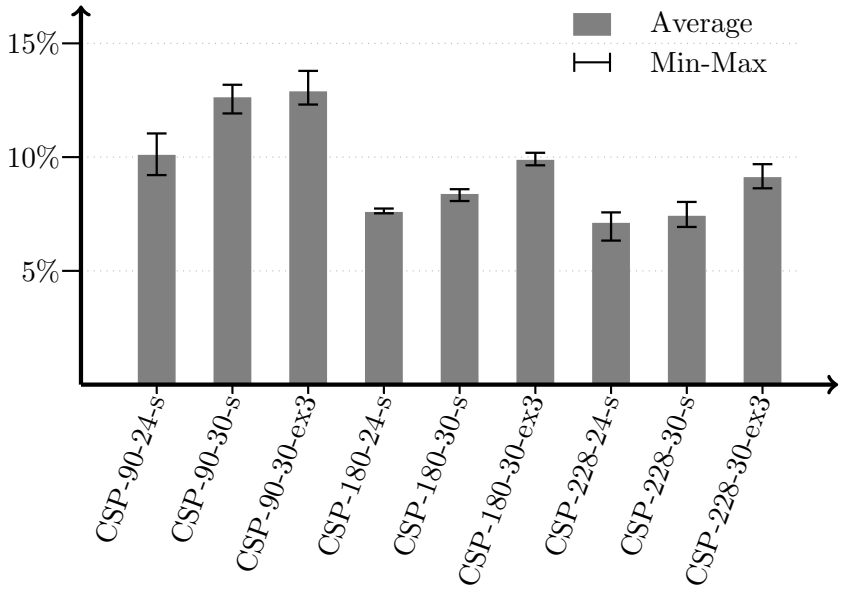

Figure 3: Results of I-CSP

Table 3 and Figure 3 present the total cost reductions, i.e., crew savings minus re-timing costs. The interval depicted for each instance in Figure 3 represents the minimum and maximum saving achieved. 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 3, 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 computation time will lead to greater stability as more alternative timetables can be explored. When comparing improvements among different test instances, it should be noted 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 3 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 number of demands 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 to exploit when solving the I-CSP. It could also suggest that the method is able to exploit the flexibility better on smaller instances. From Table 1 it can be seen that there is more flexibility for each crew task and demand for the smaller than the larger instances.

The I-CSP needs much more time than the CSP. The computation times increase by a factor in the range of 45 to 126. Among the computation times found in Tables 2 and Table 3, the longest computation time is about 7.5 hours. This should be seen in the context of this problem being solved once per year in a planning process that leaves enough time for a computation time of up to multiple days. The significant total cost reductions achieved, justify the additional computation time.

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 have been performed without changing the conclusions of Table 3.

Table 4: Integrated Crew Scheduling Problem with re-timing at Intermediate Stations

Instance	% Gap:			%	Time (seconds):		
	LB-C	Root-C	Root-TT	Saving:	Root-TT	Int-TT	Int-C
CSP-90-24-s	0.101	0.112	21.13	24.96	479	1655	1734
CSP-90-30-s	0.048	0.048	23.46	27.20	740	2310	2433
CSP-90-30-ex3	0.022	0.022	22.55	28.24	1790	3890	3929
CSP-180-24-s	0.028	0.032	14.06	31.30	715	11081	14521
CSP-180-30-s	0.191	0.193	19.05	29.59	1022	7255	14471
CSP-180-30-ex3	0.173	0.173	22.10	25.53	2779	39514	42257
CSP-228-24-s	0.198	0.198	19.28	24.40	1654	10345	14497
CSP-228-30-s	0.074	0.080	19.69	25.02	1830	26903	27406
CSP-228-30-ex3	0.288	0.288	22.95	24.74	4917	55797	57735

In Table 4 we represent results for the I-CSP with intermediate delays allowed. Here only one repetition has been performed as we wish only to show that this is indeed something the model is capable of solving. Also we wish to show that there is a potential if re-timing the engines at intermediate stations were allowed throughout the network. If we consider the performance, we can see that there is in general a larger gap to the root relaxation of the timetable when compared to Table 3. This can be explained by the increased number of timetable related variables. This number is reported as $|D'|$ in Table 1. The computation times are in general increased with respect to those for the I-CSP. However, this is to be expected as we increase the number of variables in the timetabling phase. The increase in computation time is within an acceptable factor of less than 3. In Figure 4 we compare the total cost savings, where the top bar represents the cost saving that can be achieved by allowing to re-time services at intermediate stations. It can be seen that the cost savings are significantly larger when allowing this re-timing at intermediate stations. Thus, there is a significant potential compared to the standard I-CSP. This conclusion is quite consistent over all instances and shows a potential of about a 15% additional saving.

In Table 5 we compare some characteristics of the solution process for both the I-CSP and the I-CSP-IS. First we have $|D'|$, the number of timetable variables, that increases significantly due to the intermediate delays. This change, has the biggest impact on the performance of the algorithm as it increases the size of the master-problem significantly. The increase is roughly a factor of 3. In both models a majority of the nodes in the branching tree are explored during the timetabling phase, the same is valid for the columns. The table also, in general, shows a significant increase in the number of nodes explored in the I-CSP-IS. The I-CSP-IS is a more complex model to solve, but our algorithm can solve it within reasonable time.

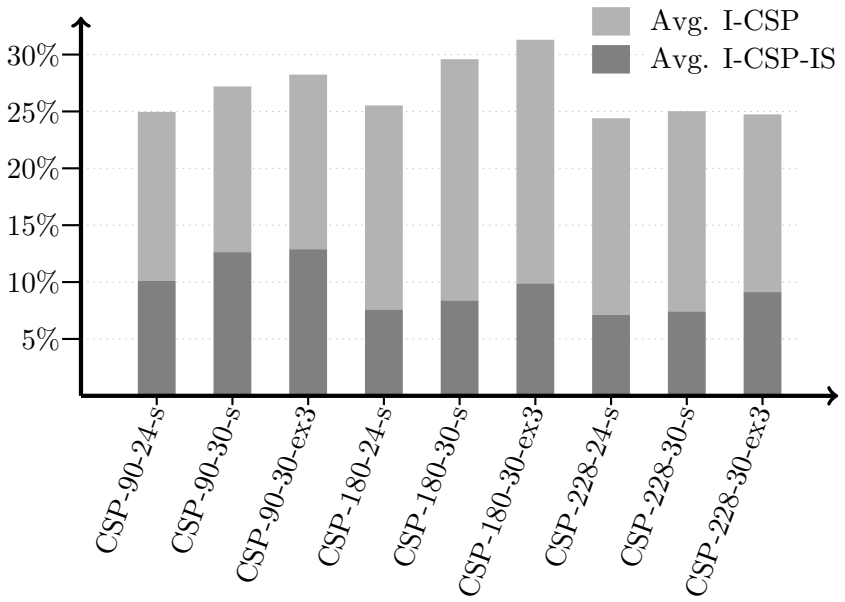

Figure 4: Comparison of results of I-CSP and I-CSP-IS

7 Conclusions and Future Research

In this paper we set out to investigate to what extent the Crew Scheduling Problem (CSP) can be integrated with 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). Furthermore, allowing for wider re-timing possibilities, we also introduced the I-CSP with re-timing at Intermediate Stations (I-CSP-IS). We have suggested a heuristic Branch-and-Price approach to solve the I-CSP and the I-CSP-IS. 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

Table 5: Problem size analysis

Instance	I-CSP					I-CSP-IS				
	$\|D'\|$	Nodes		Columns		$\|D'\|$	Nodes		Columns	
		TT	Crew	TT	Crew		TT	Crew	TT	Crew
CSP-90-24-s	650	217	17	27,054	1,094	1,991	389	40	44,780	2,550
CSP-90-30-s	772	265	88	31,176	2,694	2,279	435	27	50,802	1,470
CSP-90-30-ex3	1,032	282	85	37,309	4,880	3,070	441	18	56,262	707
CSP-180-24-s	1,131	842	74	151,479	12,362	3,537	2,929	68	275,075	7,931
CSP-180-30-s	1,312	743	86	201,787	10,282	4,018	1,853	508	194,544	44,627
CSP-180-30-ex3	1,779	356	71	152,527	10,316	5,728	1,559	169	265,165	20,180
CSP-228-24-s	1,511	879	247	247,555	37,477	4,649	1,097	484	144,160	67,223
CSP-228-30-s	1,745	640	42	263,893	9,541	5,317	3,085	30	372,363	4,660
CSP-228-30-ex3	2,295	379	112	194,998	21,014	7,254	4,753	198	599,077	26,902

compared to 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. For the I-CSP-IS, we obtained an additional improvement of the same order of magnitude. However, in the practical application we consider, the wider re-timing options cannot be implemented under the current infrastructure access regulations.

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 for 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 duty, 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. (2014). *Routing and Scheduling Problems - Optimization using Exact and Heuristic Methods*. Ph. D. thesis, Aarhus University.

Bach, L., M. Gendreau, and S. Wøhlk (2015). Freight railway operator timetabling and engine scheduling. *European Journal of Operational Research 241*, 309–319.

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.

Feillet, D., P. Dejax, M. Gendreau, and C. Gueguen (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks 44*(3), 216–229.

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 rescheduling 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.