A Column Generation Approach for the Integrated Crew Re-Planning Problem

Thomas Breugem[1], Twan Dollevoet[1], Dennis Huisman[1,2]

[1]Econometric Institute and Erasmus Center for Optimization in Public Transport
Erasmus University Rotterdam, The Netherlands

[2]Process quality and Innovation, Netherlands Railways
Utrecht, The Netherlands

breugem@ese.eur.nl, dollevoet@ese.eur.nl, huisman@ese.eur.nl

Econometric Institute Report Series - EI2019-31

Abstract

In this paper, we propose a column generation approach for crew re-planning, i.e., the construction of new duties and rosters for the employees, given changes in the timetable and rolling stock schedule. In the current practice, the feasibility of the new rosters is 'assured' by allowing the new duties to deviate only slightly from the original ones. In the Integrated Crew Re-Planning Problem (ICRPP), we loosen this requirement and allow more flexibility: The ICRPP considers the re-scheduling of crew for multiple days simultaneously, thereby explicitly taking the feasibility of the rosters into account, and hence allowing arbitrary deviations from the original duties. We propose a mathematical formulation for the ICRPP and develop a column generation approach to solve the problem. We apply our solution approach to practical instances from NS, and show the benefit of integrating the re-scheduling process.

Keywords: Crew Re-Scheduling, Crew Rostering, Railway Optimization

1 Introduction

Large maintenance and construction projects are crucial for heavily used railway networks to cope with the ever increasing demand. In 2019, for example, there were around 150 planned maintenance activities for the Dutch railway network that led to rerouting and/or the necessity of buses as alternative mode of transport[1]. These activities are inconvenient for the passengers and have a large impact on the crew schedules: Many crew members traverse large parts of the railway network, and hence their duties (i.e., days of work)

[1]source (in Dutch): `https://www.ns.nl/reisinformatie/werk-aan-het-spoor`.

become infeasible due to the maintenance activities. As a result, the crew schedule needs to be updated, preferably with as few modifications as possible.

The crew duties have to satisfy numerous rules, expressing, for example, a maximum on the length of the duty or the occurrence of a proper meal break. At NS, it is also required that each duty starts and ends at the same crew base. Figure 1 gives an example of a duty, passing the major stations The Hague (Gvc), Utrecht (Ut), and Zwolle (Zl). Note that the duty starts and ends at The Hague. Furthermore, a proper meal break of half an hour (indicated by a star) is specified, and the duty does not exceed 9.5 hours (which is the maximum length for duties starting after 6 in the morning).

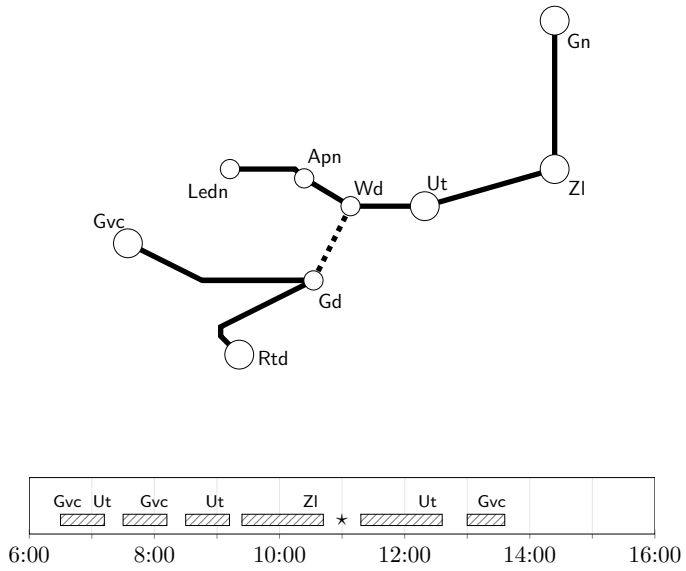


Figure 1: Schematic visualization of a part of the Dutch railway network and an example of a duty traversing this network for an employee based at The Hague (Gvc). Each block represents a trip. For each trip the departure station (top left) and/or arrival station (top right) are shown. The star indicates a meal break. The dashed line indicates scheduled maintenance between the two stations.

Besides rules on the duty level, the scheduled duties should also satisfy complex global rules related to the distribution of work among the crew bases. These rules, known as the 'Sharing-Sweet-and-Sour' rules (see Abbink et al. [2005]), assure that each crew base gets roughly the same type of work, i.e., the work allocation is fair. This implies, for example, that the duties for each crew base have roughly the same average length and the same percentage of work on high-quality rolling stock. These rules also include spatial variations: Trips marked as attractive are divided equally over all crew bases, to the extent to which this is possible. As a result, the 'Sharing-Sweet-and-Sour' rules 'enforce' that a crew member traverses different parts of the railway network within each duty. Hence, maintenance activities in a single part of the railway network can affect many different duties.

The impact of maintenance activities on the duties can be illustrated using the following example. Suppose that a maintenance activity is scheduled between Gouda (Gd) and Woerden (Wd), as indicated by the dashed line in Figure 1. This implies that the duty shown in Figure 1 is no longer valid, due to the trips between The Hague and Utrecht being canceled (as a result of the maintenance activities). As a consequence, the trip between Utrecht and Zwolle, originally covered by this duty, now has to be covered by a different (or even an additional) duty.

The problem of re-scheduling the crew has been formalized as the Crew Re-Scheduling Problem (CRSP) in Huisman [2007]. The CRSP differs from the traditional Crew Scheduling Problem (CSP) in the sense that the operational costs are of much less importance, a consequence of the fact that the original duties, and hence the amount of crew needed, are input to the CRSP. Instead, the CRSP aims at finding new duties for the already scheduled crew members such that all tasks are covered. If necessary, additional duties can be scheduled (i.e., an additional crew member will be asked to work) but this should be avoided whenever possible.

The newly constructed duties should assure that the roster of each employee remains feasible, implying that additional rules, such as a maximal workload and a minimum rest time between duties, have to be taken into account. In the current practice, the feasibility of the new rosters is 'assured' by allowing the new duties to deviate only slightly from the original ones: At NS, for example, the newly constructed duties are not allowed to start more than half an hour earlier than the original duties, or end more than half an hour later. We refer to this as *day-by-day* re-scheduling, as it allows the re-scheduling problem to be solved for each day separately.

It is clear that day-by-day re-scheduling limits the possible new duties that can be assigned, and hence a possible better solution, feasible with respect to the roster rules, might not be found. In the Integrated Crew Re-Planning Problem (ICRPP) we aim at capturing exactly this flexibility in start and end times: The ICRPP considers the re-scheduling of crew for multiple days simultaneously, thereby allowing more flexibility in the re-scheduling. The ICRPP further complicates the CRSP due to (i) the size of the problem, as multiple days need to be re-scheduled at once, and (ii) the possibility of larger shifts in start and end time, implying that the feasibility of the rosters should be taken into account explicitly.

The contribution of this paper is twofold. Firstly, we propose a mathematical formulation for the ICRPP and develop a column generation approach to solve the problem. Secondly, we apply our solution approach to practical instances from NS, and show the benefit of integrating the re-scheduling process. In particular, we show that the additional flexibility in the start and end times of the re-scheduled duties allows for a reduction in the number of additional duties compared to the day-by-day approach.

The remainder of this paper is organized as follows. In Section 2, we formalize the ICRPP and, in Section 3, we discuss related work. We then propose a mathematical formulation for the ICRPP in Section 4, and propose a column generation based solution approach in Section 5. In Section 6, we show the benefit of the newly developed solution approach using practical instances from NS, and we conclude the paper in Section 7.

2 Problem Description

The timetable and the rolling stock schedule together specify the work for a given planning day. The scheduled work is represented by *tasks* (i.e., indivisible blocks of work). Different types of tasks exist, such as driving and passenger tasks (i.e., riding a train as a passenger), but also operational tasks such as shunting and deadheading are specified in the plan. Certain tasks must always be covered (e.g., trip tasks, shunting tasks) to assure a correct execution of the plan, whereas other tasks are optional tasks (e.g., passenger tasks, taxi trips), and are solely added to the plan for additional flexibility.

In the crew planning phase, the planned tasks are assigned to the employees in the form of *duties*, i.e., sequences of tasks. Each duty has to satisfy certain rules, relating to the transfer times between tasks, the possibility of a meal break, the duration of the duty, among other things. These constraints follow from labor laws and the collective labor agreement. Furthermore, each duty should start and end at the same crew base. The duties are assigned to the crew members in the form of *rosters*: The roster of each employee specifies which duties to perform on which day. Furthermore, the roster specifies on which days the employee has a day-off. Similar to the duties, the rosters should satisfy numerous rules as agreed upon in the collective labor agreement. Typical roster rules include a minimum rest time between duties, a maximum workload over the week, and a sufficient number of days-off. We refer to Abbink et al. [2005] and Hartog et al. [2009] for an overview of the duty and roster rules at NS.

In crew re-planning the *original* duties and rosters are replaced by *alternative* duties and rosters, such that all tasks are covered in the disrupted situation. This implies that, after re-planning, each employee is assigned an alternative roster consisting of alternative duties. The alternative duties and rosters have to satisfy a number of rules, yet the rules in re-planning are generally less stringent than the rules for the initial planning phase. The 'Sharing-Sweet-and-Sour' rules, for example, are not explicitly taken into account in the re-planning process. The rules for the alternative duties are as follows.

- *Connection Times.* Between every two scheduled tasks there should be a sufficient connection time. This connection time depends on whether or not both tasks are on the same rolling stock unit. Furthermore, possible travel time (e.g., due to a passenger task) should be taken into account.

- *Duty Length.* A duty is not allowed to exceed a certain length. This maximum length depends on the start and end time of the duty. The duty length rules are specified by a start and/or end interval, and a maximum length for duties within these intervals.
- *Meal Break.* Each duty should allow for a proper meal break after a given amount of time. The meal break should always take place at one of the dedicated train stations, generally the larger stations containing a canteen.
- *Route and Rolling Stock Knowledge.* A crew member is only allowed to perform a certain trip if he or she has sufficient route knowledge. This knowledge is generally regional, i.e., it is assumed to be equal for all crew belonging to the same crew base.

Furthermore, the alternative rosters have to satisfy the following roster rules.

- *Rest Time.* After completing a duty it is required that an employee has a certain minimum time to rest. This implies that there should be a minimum amount of time between the end of a duty and the start of the duty on the next day.
- *Workload.* The total workload of a roster is not allowed to exceed a given maximum value. This maximum value depends on the work scheduled in the original roster: Ideally, the workload in the alternative roster should not deviate too much from the workload in the original roster.

Finally, each alternative duty has an associated cost. This cost can be decomposed into two parts: a fixed cost and a cost based on each connection (i.e., follow-up of two tasks). The fixed costs regulate the trade-off between different types of duties. Scheduling an additional alternative duty (i.e., an additional crew member has to perform it) is associated with a high fixed cost, as such a solution is costly and should be avoided. On the other hand, it is desirable to give some employees a day-off if possible, as the maintenance activities often mean that less tasks have to be covered. Hence, empty duties (i.e., duties without any tasks) are associated with a low fixed cost. Furthermore, there are costs associated with the connections, related to, for example, the actual cost of the connection (e.g., the cost of a taxi trip).

The alternative rosters are linked to the original rosters by means of *time windows*, specifying for each alternative duty an interval in which the start and end time of the duty must lie. The main purpose of the time windows is to assure that the newly constructed rosters remain feasible with the work scheduled outside of the re-planning period. At NS, for example, during re-planning a shift of at most half an hour in start and end time is always considered feasible with respect to the roster rules. As a result, the start and the end of the alternative roster can be at most half an hour earlier (respectively, later) than the original roster. Within each roster, however, the time windows can be loosened, by taking the roster rules into account explicitly. This is illustrated in Table 1, where we compare the time windows for the day-by-day approach with a fully integrated ap-

proach (i.e., allowing arbitrary shifts in start and end times). By varying the maximum allowed shift, one can analyze the trade-off between the number of necessary duties and the deviation from the original schedule.

	Original	Day-By-Day	Integrated
Day 1	12:30 - 21:00	12:00 - 21:30	12:00 - 08:30
Day 2	11:15 - 20:30	10:45 - 21:00	04:00 - 21:00

Table 1: Allowed start and end times for a re-scheduling period of two days. At NS, duties start after 04:00 and end the latest at 08:30 the next day. The column Original shows the originally planned start and end time of each duty, i.e., the interval in which the duty is scheduled. The column Day-by-Day shows the resulting interval according to the day-by-day approach, i.e., the interval is extended by half an hour on both sides. The column Integrated shows the possible additional flexibility: The first duty is allowed to end at any time before 08:30 the next day, and, similarly, the second duty can start already as early as 04:00. In this case, the rest time between the two duties should be explicitly enforced.

The Integrated Crew Re-Planning Problem (abbreviated ICRPP) can now be stated as follows: Given the original rosters, and the new set of tasks, determine an alternative roster (consisting of alternative duties) for each employee such that all tasks are covered. When necessary, additional duties can be scheduled, but at a high cost. The new rosters should be feasible with respect to the duty and roster constraints, and the alternative duties should respect the given time windows. The ICRPP aims at capturing the additional flexibility in the start and end times of the alternative duties, by having much looser requirements on the time windows compared to day-by-day rescheduling.

3 Literature Review

Crew planning is a well-established field of research in the Operations Research literature. In railway and mass transit optimization, the planning problem is generally decomposed into crew scheduling and crew rostering: Both crew scheduling (see, for example, Desrochers and Soumis [1989], Hoffman and Padberg [1993], Kroon and Fischetti [2001], Grötschel et al. [2003], and Abbink et al. [2005]) and crew rostering (see, e.g., Sodhi and Norris [2004], Hartog et al. [2009], Mesquita et al. [2013], and Borndörfer et al. [2015]) are well-studied problems. We refer to Kohl and Karisch [2004], Huisman et al. [2005], Caprara et al. [2007], Abbink et al. [2018], and Heil et al. [2019] for general overviews of crew planning in railway and airline optimization.

Only little research considers the integration of crew scheduling and rostering in public transport. Ernst et al. [2001] propose an integrated model able to construct both cyclic

and acyclic rosters. The solution approach relies on the complete enumeration of all possible duties for each day. As noted by the authors, this is tractable for the considered sparse railway network in Australia, but for larger railway networks (e.g., the Dutch railway network) approaches such as column generation should be considered to deal with the large number of possible duties. Mesquita et al. [2013] integrate the construction of the vehicle and duty schedules with crew rostering, and propose a Benders decomposition approach to solve the resulting problem. The benefit of the approach is shown using practical instances based on the urban bus systems of Lisbon and Porto. Finally, Borndörfer et al. [2017] consider the integration of crew scheduling and rostering. The proposed mathematical formulation links the duties and rosters through *duty templates*: coarse representations of duties expressing only the key characteristics (see Borndörfer et al. [2013]). By picking suitable templates, the number of linking constraints can be reduced drastically. The resulting model is solved using Benders decomposition, and it is shown that substantially improved rosters can be constructed without increasing the costs of the duty scheduling phase.

Railway re-scheduling and recovery has received considerable attention in recent years. We refer to Cacchiani et al. [2014] for a detailed overview. Huisman [2007] considers re-scheduling due to planned maintenance, similar to this work. Rezanova and Ryan [2010], Potthoff et al. [2010], and Sato and Fukumura [2012], on the other hand, focus on operational re-scheduling (or recovery), i.e., the re-scheduling of personnel during operations. Note that additional difficulties arise in this case, as crew members might already have started their duties at the moment of re-scheduling. Another key difference is the time available for re-scheduling: The re-scheduling due to planned maintenance is generally done numerous days in advance, and hence allows for multiple hours of computation time, whereas in operational re-scheduling the new duties should be obtained in a few minutes (or even seconds). Integrated approaches are proposed in Walker et al. [2005] and Veelenturf et al. [2012], where (part of) the timetable can be modified when re-scheduling. Finally, Veelenturf et al. [2014] propose a quasi-robust approach towards re-scheduling to cope with the uncertain length of the disruption period. The proposed formulation assures that a percentage of the re-scheduled duties should be recoverable. As a result, a subset of tasks is guaranteed to be covered for every disruption scenario.

Research regarding re-scheduling in the field of airline optimization precedes railway re-scheduling by numerous years (see, for example, Stojković et al. [1998], Lettovskỳ et al. [2000], Stojković and Soumis [2001], Petersen et al. [2012], among others). Clausen et al. [2010] gives a detailed overview of disruption management in airline optimization. The focus in airline recovery is generally on the operational planning phase, i.e., only short computation times are allowed. It is important to note that the railway and airline re-scheduling problem fundamentally differ: The railway re-scheduling problem deals with a single day in which many duties are to be re-scheduled, whereas the airline re-scheduling deals with pairings (i.e., a sequence of duties spanning multiple days), implying that

multiple days are to be taken into account. In airline optimization, the duties can generally be enumerated (see, e.g., Lavoie et al. [1988], Stojković et al. [1998]), implying that all rules related to the duties can be taken care of implicitly. The pairings should, however, satisfy numerous rules regarding, e.g., rest times, making the problem more closely related to the crew rostering problem.

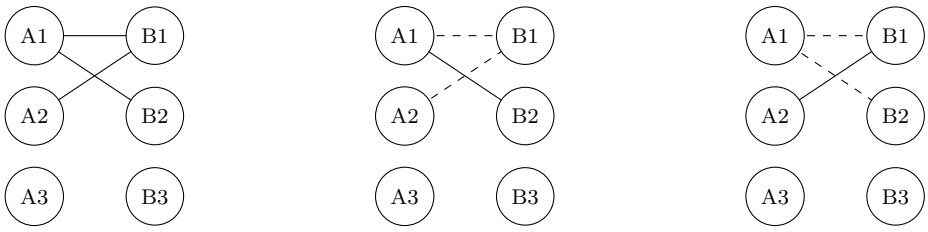
In this paper, we add to the literature by integrating railway crew scheduling and rostering in the re-planning phase, i.e., we simultaneously re-schedule the duties for multiple days, thereby taking the feasibility of the individual rosters into account. This problem extends the work on re-scheduling in railway optimization, where traditionally only one day is considered. Furthermore, it differs from current research on the integration of crew scheduling and rostering, as the original duties are considered input (similar to the way crew re-scheduling differs from crew scheduling). The resulting problem resembles the re-scheduling of crew pairings in airline optimization, yet differs fundamentally in the sense that the number of possible duties per day is huge, and hence a different solution approach is necessary.

4 Mathematical Formulation

We propose to formulate the ICRPP on the duty level, i.e., we propose a formulation in which each variable indicates whether a possible alternative duty is selected or not. This implies that the duty constraints are readily taken care of in the variable definitions (that is, only feasible duties are considered). The roster constraints, on the other hand, need to be modeled explicitly.

Let R denote the set of original rosters (one for each employee) and let T denote the set of days in the re-planning period. The set K_t, for all $t \in T$, denotes the set of tasks that need to be covered on day T. The set of alternative duties for day $t \in T$ is denoted by Δ_t, and the set of duties that can be assigned to roster r on day t is given by $\Delta_t^r \subseteq \Delta_t$. The set Δ_t^r directly incorporates the specified time window and restrictions due to route knowledge. The binary parameter a_{tk}^{δ} indicates whether the duty $\delta \in \Delta_t$ covers task $k \in K_t$. The cost of assigning duty $\delta \in \Delta_t^r$ to roster $r \in R$ is given by c_{rt}^{δ} and the cost of selecting $\delta \in \Delta_t$ as additional duty is given by f_t^{δ}.

The rest time constraints are modeled using clique constraints, similar to Ernst et al. [2001]. Let N denote the nights in the re-planning period, i.e., the transitions between two consecutive days in T. For each roster $r \in R$ and each night $n \in N$ we consider a bipartite graph where each edge represents a rest time violation between an end task at the beginning of n and a start task at the end of n. Figure 2 shows such a graph for three possible end tasks A1, A2, and A3, and three possible start tasks B1, B2, and B3 for a given roster r. In this case, ending with A1 is incompatible with starting with B1 or B2, and ending with A2 is incompatible with starting with B1.



Rest time graph | Maximal biclique 1 | Maximal biclique 2

Figure 2: Example of rest time constraints. The dashed arcs indicate the maximal bicliques: The first biclique involves tasks A1, A2, and B1, and the second biclique involves tasks A1, B1, and B2.

The rest time constraints for roster $r \in R$ and night $n \in N$ are obtained from all maximal bicliques (i.e., maximal complete bipartite subgraphs) in the bipartite graph, denoted by Q_n^r. We introduce the binary parameter $o_{rnt}^{\delta q}$, indicating whether the begin or end task of duty $\delta \in \Delta_t^r$ is part of the biclique $q \in Q_n^r$. In the first maximal biclique in Figure 2, for example, the parameter will equal 1 for all duties on the first day ending with either task A1 or A2 and all duties on the second day starting with B1. Note that, by definition, the parameter $o_{rnt}^{\delta q}$ is zero whenever t is not starting (or ending) before (or after) night n.

To model the workload constraints, let ℓ_{rt}^{δ} denote the length of duty $\delta \in \Delta_t^r$ and let w_r denote the maximum workload for roster R over the re-planning period T. We introduce the following decision variables.

- x_{rt}^{δ} for all $r \in R$, $t \in T$, and $\delta \in \Delta_t^r$. The binary decision variables x_{rt}^{δ} indicate whether duty $\delta \in \Delta_t^r$ is assigned to roster R.
- y_t^{δ} for all $t \in T$ and $\delta \in \Delta_t$. The binary decision variables y_t^{δ} indicate whether duty $\delta \in \Delta_t$ is scheduled as additional duty.

The ICRPP can now be formulated as follows.

$$\min \quad \sum_{r \in R} \sum_{t \in T} \sum_{\delta \in \Delta_t^r} c_{rt}^{\delta} x_{rt}^{\delta} + \sum_{t \in T} \sum_{\delta \in \Delta_t} f_t^{\delta} y_t^{\delta} \tag{1}$$

$$\text{s.t.} \quad \sum_{r \in R} \sum_{\delta \in \Delta_t^r} a_{tk}^{\delta} x_{rt}^{\delta} + \sum_{\delta \in \Delta_t} a_{tk}^{\delta} y_t^{\delta} \geq 1 \qquad \forall t \in T, k \in K_t \tag{2}$$

$$\sum_{\delta \in \Delta_t^r} x_{rt}^{\delta} = 1 \qquad \forall r \in R, t \in T \tag{3}$$

$$\sum_{t \in T} \sum_{\delta \in \Delta_t^r} o_{rnt}^{\delta q} x_{rt}^{\delta} \leq 1 \qquad \forall r \in R, n \in N, q \in Q_n^r \tag{4}$$

$$\sum_{t \in T} \sum_{\delta \in \Delta_t^r} \ell_{rt}^{\delta} x_{rt}^{\delta} \leq w_r \qquad \forall r \in R \tag{5}$$

$$x_{rt}^{\delta} \in \mathbb{B} \qquad \forall r \in R, t \in T, \delta \in \Delta_t^r \tag{6}$$

$$y_t^{\delta} \in \mathbb{B} \qquad \forall t \in T, \delta \in \Delta_t. \tag{7}$$

The Objective (1) expresses that we minimize the cost of the selected duties, consisting of the cost of the duties assigned to the rosters and the cost of the additional duties. Constraints (2) and (3) assure that each task is covered and that each roster is assigned exactly one duty for each day, respectively. Constraints (4) and (5) represent the roster constraints: (4) assure the minimum rest time is respected, and (5) enforce a maximum workload for each roster. Finally, the domains of the decision variables are specified in (6) and (7).

5 Solution Approach

We propose a column generation approach to solve the ICRPP. This type of approach has been succesfully applied to similar crew re-scheduling problems in railway optimization (see, for example, Huisman [2007] and Potthoff et al. [2010]), and is generally considered a state-of-the-art solution approach (for detailed surveys, see Barnhart et al. [1998], Lübbecke and Desrosiers [2005], Desaulniers et al. [2006] and Lübbecke [2011]). The main idea behind column generation is to solve a linear program with a huge amount of columns (i.e., variables) by only considering a subset of all possible columns. In each iteration, it is checked if possible profitable, i.e., negative reduced cost, columns are present among the columns not yet included. If this is the case, the procedure continues. Otherwise, the found solution is optimal, and the algorithm terminates.

We propose a solution method in which we iteratively select alternative rosters for the crew members. The algorithm continues until all employees are assigned an alternative roster, and, when necessary, selects additional duties to cover the remaining tasks. To select good alternative rosters, we base the selection of alternative rosters on the solution to the linear relaxation of (1)–(7). In particular, we select the alternative rosters that appear in the solution with a high (fractional) value. The linear relaxation is solved using column generation.

The remainder of this section is structured as follows. In Section 5.1 we give a global overview of the iterative selection procedure, and in Section 5.2, we discuss the selection procedure for alternative rosters in detail. Then, in Section 5.3, we discuss the pricing problem for alternative duties, which underlies the column generation algorithm. We conclude in Section 5.4 with an overview of the acceleration strategies used to speed-up the column generation procedure.

5.1 Iterative Selection Procedure

We obtain an integer solution for the ICRPP by iteratively selecting an alternative roster for one of the crew members. This type of approach is generally referred to as a diving heuristic, i.e., a depth-first search heuristic in the branching tree (see, for example, Joncour

et al. [2010], and references therein). The iterative procedure is schematically visualized in Figure 3.

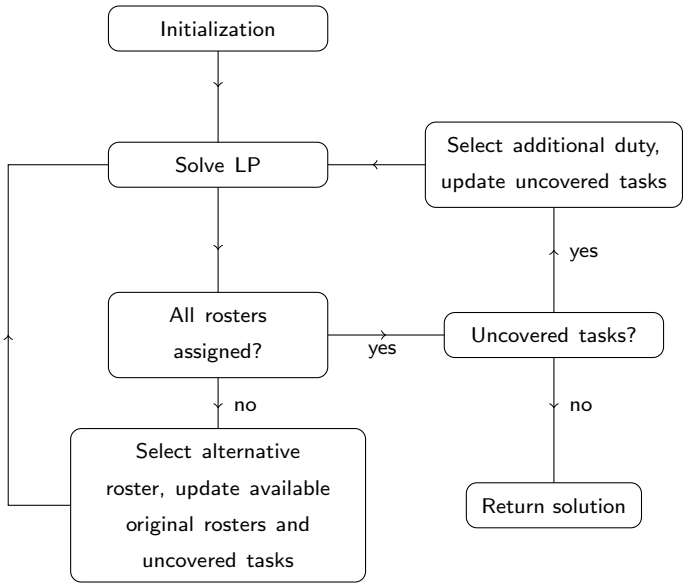


Figure 3: Iterative selection procedure for the ICRPP, consisting of two phases: Firstly, alternative rosters are selected for the crew members, and, secondly, additional duties are selected to cover possibly still uncovered tasks.

The selection procedure consists of two phases: Firstly, alternative rosters are selected for the crew members, and, secondly, additional duties are selected to cover possibly still uncovered tasks. The main motivation of selecting complete rosters, as opposed to separate duties, is to assure feasibility with respect to the roster rules. Note that the roster rules do not apply to the additional duties. Each time an alternative roster or additional duty is selected, the set of still available rosters (i.e., crew members) and uncovered tasks is updated. Once all tasks are covered, the algorithm terminates and the found solution is returned.

As is common for diving heuristics, both the alternative rosters and additional duties are selected based on the *highest value* rule, i.e., all rosters or duties with integer value are selected, together with one roster or duty assigned the highest (non-integer) value in the fractional solution (the latter assuring that the solution to the linear relaxation changes). Whereas defining the highest value is trivial for the additional duties (i.e., the value is directly obtained from the corresponding variable), this is not the case for the alternative rosters. In Section 5.2 we discuss the selection of alternative rosters in more detail.

5.2 Selecting Alternative Rosters

The roster rules imply that an arbitrary selection of alternative duties for an original roster can lead to infeasible solutions. Consider, for example, the fractional solution depicted in

Table 2. In this specific example, selecting one duty per day, based on the highest value rule, will lead to an infeasibility: Selecting duties 2 and 3 will lead to an alternative roster which violates the minimum rest time of 12 hours. Note that the fractional solution *does* satisfy (4).

	Day	Start	End	Value
1	1	12:00	20:00	0.4
2	1	12:30	21:00	0.6
3	2	08:00	16:00	0.4
4	2	09:00	17:00	0.3
5	2	09:00	17:30	0.3

Table 2: Example of a fractional solution for a given original roster. Each row indicates an alternative duty, and specifies the start and end time, and the (fractional) solution value.

To assure the feasibility of the constructed rosters, even when multiple alternative duties are selected in each iteration, we directly select *all* alternative duties for a single roster, thereby taking the roster constraints directly into account. Consider again the example of Table 2. The duties give rise to five alternative rosters: $\{1,3\}, \{1,4\}, \{1,5\}, \{2,4\}$, and $\{2,5\}$. Note that the roster $\{2,3\}$ is not considered here, as it violates the rest time constraint. The highest value rule is applied by taking the minimum over all assigned alternative duties. That is, the value corresponding to, for example, roster $\{1,3\}$ equals $\min\{0.4, 0.4\} = 0.4$, and the value corresponding to roster $\{2,4\}$ equals $\min\{0.6, 0.3\} = 0.3$. In this specific example, the highest value rule would select roster $\{1,3\}$. For each original roster, the alternative roster with highest value can be determined and the alternative with overall highest value can be added to the solution.

The above procedure can be formalized as follows. Let (x, y) be a solution to the linear relaxation of (1)–(7). Given x, we determine the feasible alternative rosters $P_r(x)$ for each $r \in R$, consisting only of alternative duties with a non-zero solution value. This set is determined by complete enumeration. For an alternative roster $\rho_r \in P_r(x)$, let $\rho_{rt} \in \Delta_t^r$ denote the alternative duty assigned to day $t \in T$. The value $\pi(x, \rho_r)$ for each $\rho_r \in P_r(x)$ is defined as

$$\pi(x, \rho_r) = \min_{t \in T} \{x_{rt}^{\rho_{rt}}\},$$

i.e., the value of the alternative roster is based on the minimum taken over all assigned alternative duties. In each iteration, we determine for each original roster $r \in R$ a candidate $\rho_r^\star$ maximizing $\pi(x, \rho_r)$ and add all alternative rosters $\rho_r^\star$ for which $\pi(x, \rho_r^\star) = 1$ to the solution, together with the alternative roster with highest non-integer value (i.e., the highest value below 1). The involved original rosters and tasks are removed from the pool of available original rosters and uncovered tasks, respectively, and the algorithm

continues.

5.3 Pricing Problem

The pricing of alternative and additional duties can be modeled as a series of Resource Constrained Shortest Path Problems (RCSPPs) on suitably defined graphs. Let B denote the set of crew bases. For every day $t \in T$ and crew base $b \in B$, we consider a dedicated digraph $G_t^b = (V_t^b, A_t^b)$, referred to as a pricing graph, where the nodes correspond to the tasks in K_t and the arcs represent feasible connections between tasks. The graph contains an additional source and sink node, representing the departure and arrival at the crew base.

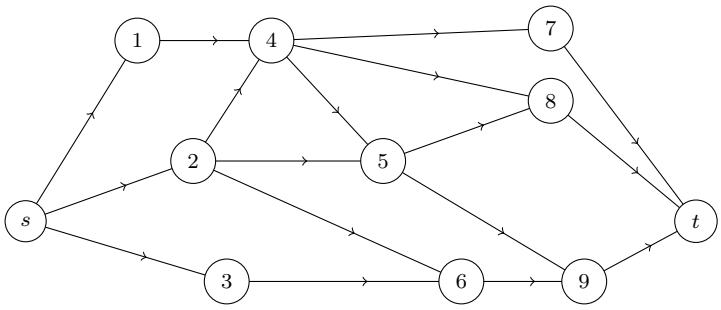


Figure 4: Pricing graph for alternative and additional duties. Each node corresponds to a task and each arc represents a feasible connection between two tasks. In this graph, a total of nine tasks are shown. Each path from the source s to the sink t corresponds to a possible duty.

Figure 4 gives a stylized example of a pricing graph. Note that any passenger tasks and taxi trips can be directly incorporated in the arc set. Furthermore, the route knowledge is readily incorporated by adding a node to the graph only for those tasks compatible with crew base b. Finally, the travel times from and to the crew base can be set accordingly using the source and sink arcs.

Let R_b denote the original rosters corresponding to base $b \in B$. For each base $b \in B$ and day $t \in T$, we solve a pricing problem for each original roster $r \in R_b$ to price out any negative reduced cost alternative duties for r on day t, and, furthermore, we solve one pricing problem to price out negative reduced cost additional duties for base b on day t. Note that the time window specified for original roster $r \in R_b$ on day $t \in T$ can be easily incorporated in the pricing graph G_t^b by discarding all tasks which cannot be covered within the specified time interval.

The reduced cost of the alternative and additional duties can be expressed as follows. Let λ_{tk} denote that dual variables corresponding to the coverage constraints (2), μ_{rt} to the assignment constraints (3), γ_{rn}^q to the rest time constraints (4), and, finally, ϕ_r to the

workload constraints (5). The reduced cost of x^δ_{rt} is given by

$$c^\delta_{rt} - \sum_{k \in K_t} \lambda_{tk} a^\delta_{tk} - \mu_{rt} - \sum_{n \in N} \sum_{q \in Q^r_n} \gamma^q_{rn} o^{\delta q}_{rnt} - \phi_r \ell^\delta_{rt}, \tag{8}$$

and the reduced cost of y^δ_t by

$$f^\delta_t - \sum_{k \in K_t} \lambda_{tk} a^\delta_{tk}. \tag{9}$$

As discussed in Section 2, the costs c^δ_{rt} and f^δ_t consist of a fixed cost and a cost per connection, and can therefore readily be modeled using the arc costs. As a result, the reduced cost can be modeled using appropriate arc costs, as well: The dual variables regarding the coverage constraints can be modeled as arcs costs using the incoming arcs of each task. Furthermore, the assignment constraint duals can be incorporated in the arc costs of arcs leaving the source. For the rest time constraints, we observe that membership of a biclique in the violation graph depends *only* on the start and end times of the alternative duty: The duals corresponding to the rest time constraints can therefore be readily modeled using arc costs on the arcs leaving and entering the source and sink, respectively. Finally, the length of the alternative duty decomposes over the length of the visited arcs and nodes, and hence can also be modeled using the arc costs.

The duty length and meal break rules are incorporated in the pricing problem in a similar way as proposed in Huisman [2007]. Recall that the duty length rules are expressed by a start and/or end interval, and a maximum length, e.g., a duty length rule could specify that a duty starting between 05:00 and 06:00 or ending between 02:30 and 08:30 can have a length of at most 8.5 hours, or could specify that a duty starting after 06:00 and ending before 02:30 the next day can have a length of 9.5 hours. All these rules together can be captured by a suitably picked maximum ending time for each possible start task (see Figure 5). Hence, the pricing problem, given a pricing graph (either for an alternative or an additional duty), can be modeled as a RCSPP for each start task, with a resource related to the meal break constraint.

Solving the RCSPP for each possible start node allows to incorporate the duty length constraint. This is done as follows: Given a start task s, we enforce the maximum duty length constraint by allowing only those end tasks e such that the duty starting with s and ending with e respects the maximum duty length constraint. This procedure is illustrated in Figure 6. For task s_1, starting at 05:15, we remove end tasks e_3, e_4, and e_5 from the set of possible end task, as the length of the resulting duty would be too long. For start task s_2 all the end tasks except e_5 are feasible.

We solve the RCSPP using a labeling algorithm incorporating *completion bounds*: lower bounds on the minimum cost of completing a path to the sink node. The underestimation follows from the fact that the lower bounds are based on the shortest path costs, i.e., the

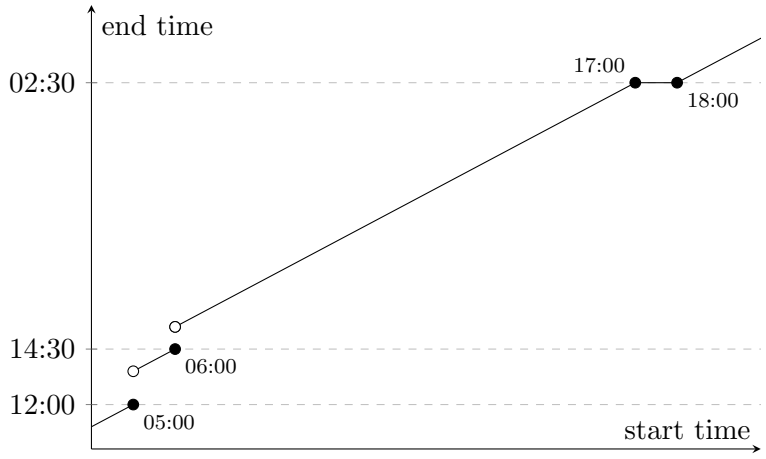


Figure 5: Duty length function, mapping each start time to the maximum end time. Duties starting between 04:00 and 05:00 are at most 7 hours, those starting between 05:00 and 06:00 at most 8.5 hours, and those starting after 06:00 at most 9.5 hours. Furthermore, duties ending after 02:30 in the night cannot exceed 8.5 hours.

meal break constraint is not taken into account. We solve the RCSPP heuristically by using Breadth First Search (BFS), thereby limiting the label set to at most k labels, based on the lowest estimated cost (i.e., the lowest lower bound). Note that this labeling strategy considers at least the k shortest paths in the graph, which, as argued in Huisman [2007], are often feasible for the meal break constraint and hence represent the set of most negative reduced columns. Whenever optimality is required, we switch to a Depth First Search (DFS) procedure (to limit the size of the label set), and terminate whenever either the k most negative reduced cost paths are found, or we prove that no more negative reduced cost paths are present.

5.4 Acceleration Strategies

The performance of column generation based algorithms depends heavily on the strategy for pricing negative reduced columns and the precise interaction between the master and pricing problems. This led to the development of acceleration strategies: strategies, often specified for classes of problems, to improve the column generation procedure (see e.g., Desaulniers et al. [2002] for a detailed overview). We consider three acceleration strategies for the ICRPP, generally referred to as *pre-processing* (i.e., reducing the number or size of the RCSPPs that need to be solved), *partial pricing* (i.e., solving only a subset of pricing problems) and the enforcement of *task-disjoint* columns.

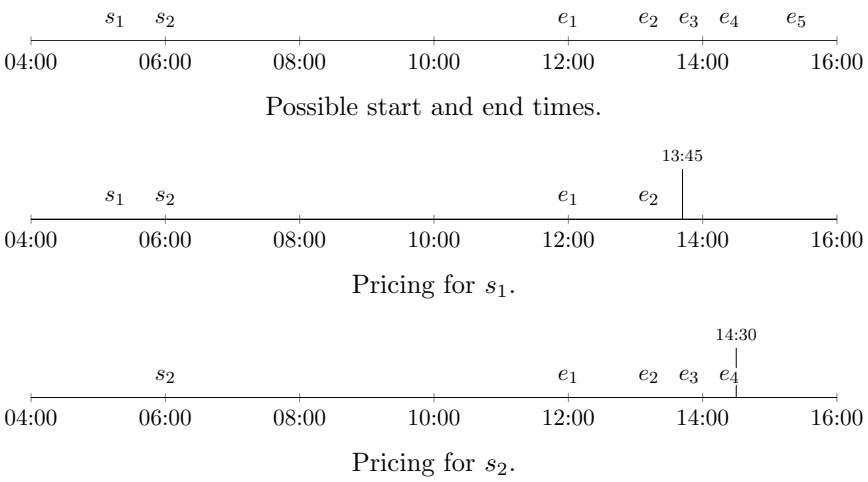


Figure 6: Two pricing problems resulting from the start tasks s_1 and s_2. For a given start task s, the duty length constraint is enforced by removing all start tasks before s and all end tasks e such that the time between s and e violates the duty length constraint.

Pre-Processing Start Tasks

Firstly, we limit the number of RCSPPs to be solved for a given pricing graph, by avoiding all 'redundant' start tasks, i.e., by avoiding all start tasks which can be skipped without losing optimality. This can be done as follows. Firstly, we note that all tasks leading to the same start time can be aggregated into one pricing problem: Since the decomposition per start task functions to model the duty length constraint, this can be done while still assuring exactness of the pricing problem. Secondly, we note that, in a similar fashion, any two start tasks leading to the same set of end tasks can be aggregated into one pricing problem. Here, the possible end tasks depend on (i) the duty length constraint and (ii) the specified time window. Given these two rules, the possible start tasks can be aggregated into pricing problems using a simple greedy procedure. In Section 6, we show that this pre-processing framework greatly reduces the number of possible start tasks per pricing graph, and hence the number of RCSPP that need to be solved.

Partial Pricing of Alternative Duties

Secondly, we consider a partial pricing strategy for the alternative duties: For each original duty, we order the set of possible start tasks randomly, and restrict the generation of alternative duties to only those pricing problems corresponding to one of the n first tasks in this list, where n is an a priori set control parameter. Recall that, when solving the pricing problem for start task s, we also allow start tasks starting later than s, i.e., in Figure 6 task s_2 is allowed as start task when pricing for task s_1. This implies that many start tasks are considered even if we price only for a small amount of start tasks, a desirable

property when applying partial pricing. Note, however, that the duties generated for the additional start tasks will be shorter than technically possible (since the duty length constraint is based on an earlier start task), hence, it is still necessary to price for all start tasks to assure no negative reduced columns exist.

The partial pricing strategy is applied in two ways: *heuristically* and *exact*. In heuristic partial pricing, we stop solving pricing problems after the first n start tasks have been considered. The found negative reduced cost columns are returned, and the column generation algorithm continues. The advantage of heuristic partial pricing is that it allows easy control of the time spent on pricing in each iteration. The major downside, however, is that possibly not all negative reduced cost columns are found, i.e., the linear relaxation will most likely not be solved to optimality. Exact partial pricing aims at avoiding the latter: Whenever no negative reduced cost columns are found using heuristic partial pricing, we continue iterating through the list of possible start tasks for each of the original duties, until either the complete list has been considered, or a column with negative reduced cost has been found. This assures that the linear relaxation will be solved to optimality. In heuristic partial pricing, we solve the RCSPP using the heuristic approach (as discussed in Section 5.3), wheres in exact partial pricing we also switch to the exact approach for the RCSPP.

In the iterative selection procedure we apply both heuristic and exact partial pricing to efficiently solve the ICRPP: Exact partial pricing is applied in the root node, to obtain a good initial solution and valid lower bound, and heuristic partial pricing is used in the remaining nodes of the tree, i.e., each time we select an alternative roster and the linear relaxation is resolved. This avoids that we spend much time on exact partial pricing throughout the diving procedure.

Selecting Task-Disjoint Columns

The third and final acceleration strategy focuses on the structure of the column pool. For the ICRPP, and other set-covering type of problems, an optimal solution will most likely have columns with little overlap, i.e., the columns have little tasks in common. By enforcing the returned columns to have little overlap (referred to as being *task-disjont* in Desaulniers et al. [2002]), the column pool will have an 'optimal' structure, without flooding the master problem with a huge number of columns. The level of overlap allowed controls the trade-off between the number of columns in the column pool and the maximum overlap among columns.

Given a set of negative reduced columns, a close to task-disjoint subset is picked as follows. For any two columns x_1 and x_2 a *similarity score* can be computed based on the tasks covered: The similarity score of column x_1 with column x_2 is defined as the number of tasks covered by both columns divided by the total number of tasks covered

by x_1. The higher this score, the more x_1 is similar to (or contained in) x_2. Given the generated columns and an a priori set similarity threshold, we greedily obtain a subset of columns based on most negative cost, whilst assuring that the similarity score with the already selected columns does not exceed the similarity threshold. Note that this procedure assures that the column with most negative reduced cost is always selected. The resulting subset of columns will have little overlap.

6 Computational Experiments

To illustrate the benefit of the integrated approach, we apply our solution approach to practical instances from NS. In Section 6.1, we give a detailed description of the case study. In Sections 6.2 and 6.3 we analyze the computation results: In Section 6.2 we analyze the effect of the acceleration strategies discussed in Section 5.4 and in Section 6.3 we discuss the results of the iterative selection procedure proposed in Section 5.2.

6.1 Case Study

We consider the re-scheduling of crew over a weekend (i.e., Saturday and Sunday), in April 2019, when large-scale maintenance was scheduled around station Leiden (Ledn). The situation is depicted in Figure 7. We construct different instances based on the crew bases Amsterdam (Asd), Lelystad (Lls), The Hague (Gvc), and Rotterdam (Rtd), four major crew bases for which many original rosters became infeasible due to the maintenance activities.

We construct four re-planning instances, each corresponding to a re-planning problem for three of the four crew bases. The characteristics of the four instances are shown in Table 3. The original duties and updated timetable are obtained from the second Saturday in April 2019, a day in which the planned timetable was substantially altered: About 20 duties became infeasible at each of the four crew bases. We extend this to data to the whole weekend by assuming the timetable and original duties for Sunday to be the same. For each instance, we aim at covering all tasks in the (now infeasible) original duties which are still in the plan. The result are four challenging instances, where more than 500 tasks per day are to be covered. Furthermore, to assure sufficient flexibility, the remaining tasks are also added to the problem. This implies that for each of the four instances, there are 2519 possible tasks (i.e., tasks that are not necessarily to be covered but can be used as passenger tasks) and 87514 possible connections per day.

We construct original rosters covering all of the original duties. When constructing the rosters, we assume that each employee works both days in the weekend. The rosters are obtained by solving a stylized rostering problem, where we minimize the occurrence of short rest times and high workload, while assuring that the roster rules (as discussed in

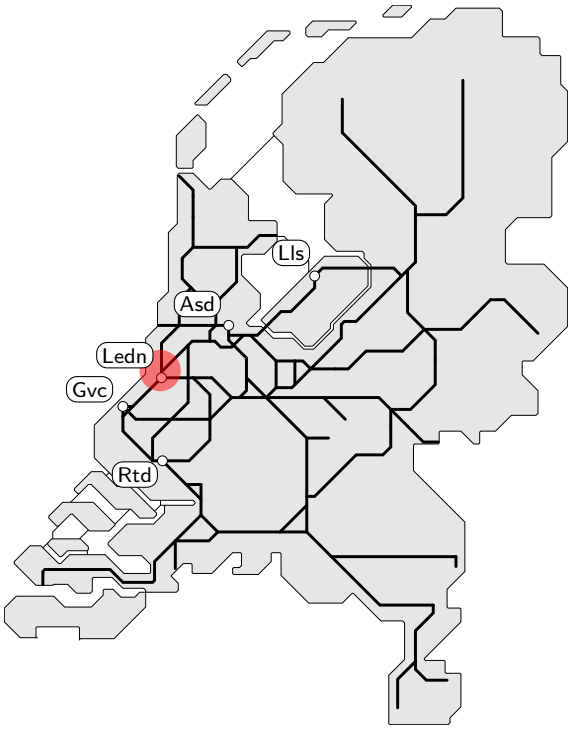


Figure 7: Case study considering scheduled maintenance at Leiden (Ledn), shown together with the crew bases located at The Hague (Gvc), Rotterdam (Rtd), Amsterdam (Asd), and Lelystad (Lls). The black lines show the lines operated by NS.

Section 2) are satisfied. Furthermore, we assure that each roster remains feasible whenever a duty is shifted by at most half an hour.

6.2 Analysis of Acceleration Strategies

The acceleration strategies discussed in Section 5.4 aim at (substantially) improving the time spent in the column generation algorithm. In this section, we discuss the gain from pre-processing the possible start tasks for each original duty, and we analyze the effect of the three control parameters: the number of paths returned per RCSPP, the similarity threshold, and the percentage of pricing problems solved in each iteration.

Table 4 shows the effect of the pre-processing discussed in Section 5.4, i.e., the removal of all pricing problems corresponding to 'redundant' start tasks. For each instance and each maximum allowed shift, Table 4 shows the average number of considered start tasks (averaged over all original duties), and hence RCSPPs to be solved, without pre-processing (Original) and the average number of start tasks with pre-processing, i.e., the average

Instance	Bases	Rosters	Tasks To Cover Per Day	Tasks Per Day	Connections Per Day
1	3	75	649	2519	87514
2	3	64	539	2519	87514
3	3	81	713	2519	87514
4	3	74	640	2519	87514

Table 3: Description of the four re-planning instances. For each instance, the number of involved crew bases and rosters (i.e., employees) is shown. Furthermore, the total number of tasks and possible connections per day are shown, together with the number of tasks per day that need to be covered.

number of non-redundant start tasks (Reduced).

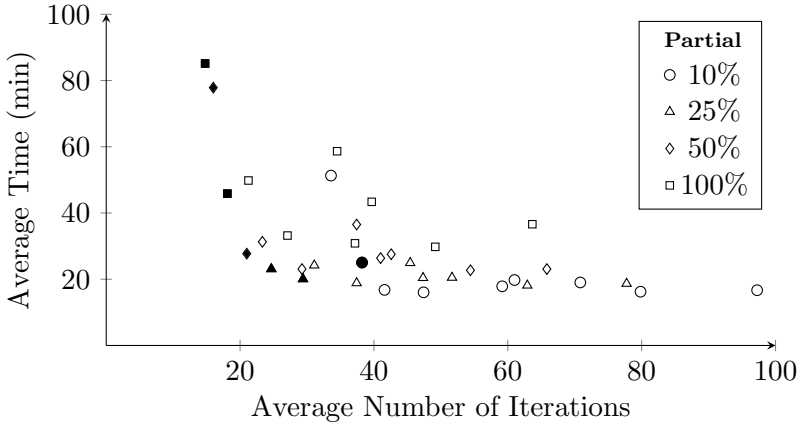
	Instance 1		Instance 2		Instance 3		Instance 4	
Shift	Original	Reduced	Original	Reduced	Original	Reduced	Original	Reduced
00:30	822.1	9.5	551.3	7.4	701.2	8.3	754.3	8.9
01:00	836.5	13.0	562.2	9.3	713.2	11.5	766.3	12.3
02:00	862.7	21.2	582.3	14.0	735.0	18.8	786.7	20.3
05:00	928.8	44.5	633.8	27.4	790.2	39.5	836.4	42.4

Table 4: The effect of pre-processing the possible start tasks. For each instance and each shift, the average number of considered start tasks (averaged over the original duties) before and after pre-processing are shown (indicated by Original and Reduced, respectively).

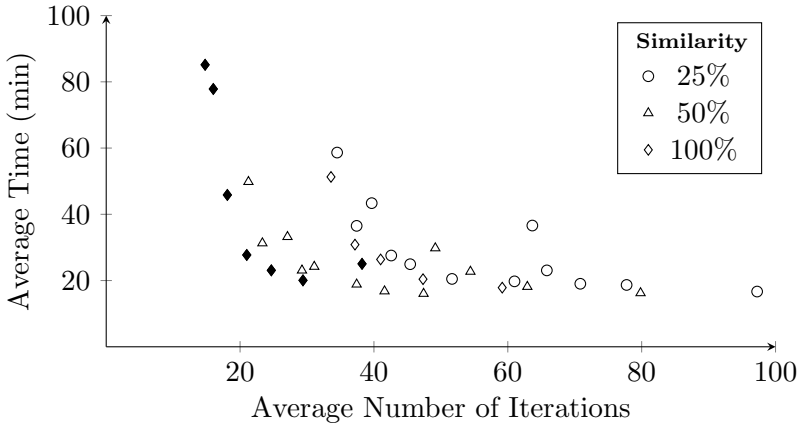
The benefit of pre-processing is clearly visible from Table 4. For all instances and all shift sizes, the reduction greatly reduces the number of start tasks: The reduction ranges from about 95% to 99%. Hence, the number of pricing problems can be reduced by more than a factor 20 if the start tasks are checked for redundancy a prori starting the column generation procedure. For the larger shifts, the number of original start tasks (and the number of start tasks after pre-processing) increases, as expected. As can be seen from Table 4, the effectiveness of the pre-processing slowly decreases when the shift size increases (from about 99% to 95%), which can most likely be explained by the fact that redundancy of a start task is less likely when many different start and end tasks (and hence start and end times) are possible.

We analyze the effect of the different acceleration strategies by considering the solution time for the linear relaxation for different parameter configurations. We consider returning 10, 100, and 250 columns for each RCSPP, a similarity threshold of 25%, 50%, and 100% and the percentage of pricing problems to be solved in partial pricing to be either 10%, 25%, 50%, or 100%. Furthermore, we average the results over three runs, each using a different random seed. This leads to 432 experiments in total. For each individual run, we limit the maximum computation time to 6 hours. Whenever a run exceeds 6 hours, it is not taken into account when computing the average. The results are visualized in Figure 8. For completeness, the full computational results are given in Appendix A.

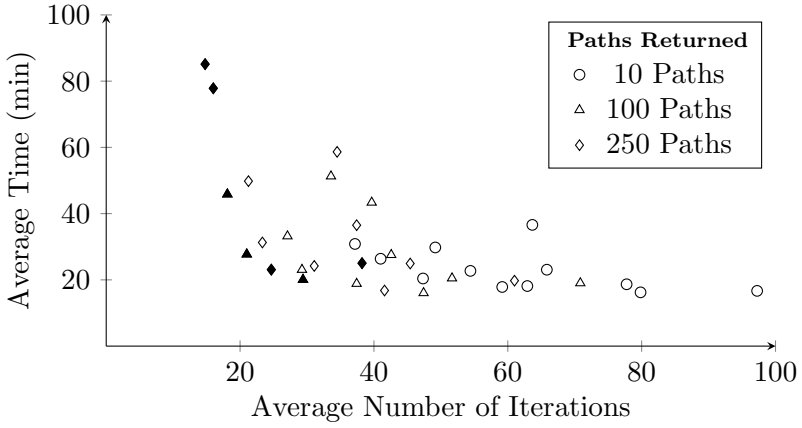
Figure 8a clearly highlights the benefit of partial pricing: The shortest average compu-



(a) Results grouped based on the percentage of pricing problems solved in partial pricing.



(b) Results grouped based on the similarity threshold.



(c) Results grouped based on the number of paths returned per RCSPP.

Figure 8: For each combination of parameters, the average computation time in minutes and the average number of iterations are shown, taken over the four shift lengths (00:30, 01:00, 02:00, and 05:00) and three random seeds. Each subfigure groups the results based on one of the parameters. Filled markers indicate parameters settings for which some runs were not finished within the 6 hour time limit.

tation times are all achieved for partial pricing parameters of 10% and 25%, and there seems to be a clear increasing relation between the percentage of pricing problems solved in each iteration and the average computation time. The gain of using task-disjointness to select columns for the master problem is also visible in Figure 8b: The best performance is achieved for parameter settings using a similarity threshold of less than 100%. Note that a threshold of 100% implies that all generated columns are added to the master problem. As a result, the size of the master problem grows quickly. Whenever the added columns are 'good', the column generation algorithm terminates in only a few iterations. Whenever the columns are 'bad', however, the time spent in the master problem grows quickly. Following this reasoning, it is no surprise that each setting for which not all runs terminated in time, used a similarity threshold of 100%. Figure 8b also seems to indicate that a threshold of 50% outperforms a threshold of 25%. In this case, a 25% threshold might be too strict, implying that too many generated (and useful) columns will not be added to the master problem, and hence more column generation iterations are needed. Finally, the effect of the (maximum) number of paths returned per RCSPP, shown in Figure 8c, seems to be less clear, although returning 10 or 100 paths seems to outperform returning 250 paths. Note, however, that the effect of this parameter is highly intertwined with the other two parameters, e.g., returning 250 paths can perform well accompanied with a low similarity threshold, but can also perform badly whenever accompanied with a 100% threshold and/or a partial pricing parameter of 100%.

Summarizing, the results shown in Figure 8 give insight in the effective usage and possible gain of the acceleration strategies discussed in Section 5.4. Clearly, it is not possible to select the 'best' configuration based on the experiments, as the number of runs per configuration are limited and the running times vary substantially (see Appendix A). It does, however, give insight in 'good' configuration settings. In particular, the experiments highlight the benefit of partial pricing and the focus on task-disjoint columns, together with a suitably picked number of maximum paths to be returned for each solved pricing problem.

6.3 Results Iterative Selection Procedure

In this section, we analyze the benefit of the integrated approach compared to the day-by-day approach. As discussed in Section 2, the start and end of the alternative rosters can shift at most half an hour, but the start and end times of duties within the roster can be changed freely. The maximum allowed shift regulates the latter: Allowing a shift of two hours implies that the end of the alternative duty on Saturday and the beginning of the alternative duty on Sunday can deviate up to two hours from their corresponding original duty. By varying the allowed shift, the trade-off between the number of necessary duties and the deviation from the original rosters can be analyzed. In particular, the solution found using the day-by-day approach (corresponding to a maximum shift of half an hour)

can be compared with the solutions when larger shifts are allowed. Note that every duty should still satisfy the general start and end time rules (i.e., duties start from 04:00 and end before 08:30 the next day), even if the maximum shift would allow otherwise.

We consider four maximum shifts for each instance: 00:30, 01:00, 02:00 and 05:00. Note that larger shifts are also possible, yet we found that these shifts did not lead to a decrease in the root bound (compared to 05:00), and are therefore omitted. The problem for a maximum shift of 00:30 is equivalent to the day-by-day approach, i.e., the problem is decomposable per day. Hence, in this case, we solve two separate CRSPs (instead of the ICRPP). For each instance and each shift size, we average the results over five runs, each for a different random seed. The results are shown in Table 5. The major cost components are a cost of 2100 for a scheduled duty, a cost of 1 for an empty duty, and an additional penalty of 10000 for an additional duty (roughly five times the cost of a normal duty), together with some (small) cost on the connections. In other words, we avoid scheduling additional duties, and try to assign empty duties whenever possible (as assigning an empty duty to an original duty reduces the fixed cost for this duty from 2100 to 1) . We used a partial pricing parameter of 10%, similarity threshold of 50%, and return 100 paths per RCSPP.

	Shift	Objective		Gap (%)		Best Found	Root Bound	Additional		Empty		Time (s)	
1	00:30	343286.0	(4711.9)	4.3	(1.3)	340506.0	328406.0	3.2	(0.4)	6.0	(0.0)	560.0	(65.6)
	01:00	328529.4	(11253.5)	5.5	(3.3)	310009.0	310009.0	2.6	(1.0)	9.4	(0.8)	1127.0	(211.8)
	02:00	330233.2	(7944.9)	8.0	(2.3)	315812.0	303712.0	3.4	(0.8)	13.2	(1.2)	1969.6	(253.5)
	05:00	**327294.6**	(4435.3)	7.8	(1.2)	323714.0	301613.0	3.4	(0.5)	14.6	(0.8)	5281.4	(464.5)
2	00:30	270574.2	(8396.6)	7.8	(2.8)	261314.0	249214.0	1.8	(0.7)	14.2	(0.4)	418.2	(51.0)
	01:00	**258375.4**	(7032.9)	5.0	(2.6)	247115.0	245365.8	1.0	(0.6)	15.4	(0.5)	812.2	(81.8)
	02:00	265437.8	(5837.7)	8.4	(2.0)	257116.0	242917.0	2.0	(0.6)	17.8	(1.2)	965.8	(145.5)
	05:00	259758.2	(8472.8)	6.4	(3.2)	242917.0	242917.0	1.6	(0.8)	18.2	(0.7)	1983.0	(174.4)
3	00:30	375570.2	(8396.6)	5.6	(2.1)	366310.0	354210.0	3.8	(0.7)	10.2	(0.4)	677.4	(39.4)
	01:00	360274.2	(9348.4)	8.0	(2.4)	345814.0	331086.9	3.2	(0.7)	14.2	(1.0)	2936.8	(462.7)
	02:00	**351197.2**	(1570.7)	6.8	(0.4)	349518.0	327417.0	3.0	(0.0)	17.2	(0.7)	3070.4	(625.3)
	05:00	354876.6	(8887.3)	7.7	(2.4)	339517.0	327417.0	3.2	(0.7)	16.6	(0.5)	5924.0	(303.4)
4	00:30	368077.8	(9504.9)	7.9	(2.4)	350718.0	338618.0	6.4	(0.8)	17.8	(0.4)	641.6	(55.3)
	01:00	**334178.4**	(11083.5)	7.5	(3.0)	322918.0	308719.0	4.0	(1.1)	18.4	(1.4)	1923.8	(286.6)
	02:00	337240.4	(7725.3)	9.0	(2.0)	330820.0	306620.0	4.6	(0.8)	20.4	(1.0)	2406.4	(161.9)
	05:00	342080.4	(13671.5)	10.2	(3.5)	328721.0	306620.0	5.0	(1.1)	20.4	(0.5)	5018.2	(308.6)

Table 5: Computational results for the iterative selection procedure. For each instance and each maximum shift, the results are averaged over five different runs, each for a different random seed. The average found solution value, average root gap, best found solution, and root bound are shown. Furthermore, the average number of scheduled additional and empty duties is depicted, together with the average computation time (in seconds). The numbers within brackets denote the standard deviation, when applicable.

The results in Table 5 show a clear benefit from the integrated approach: The solutions found using time shifts larger than 00:30 all improve on the day-by-day approach. Furthermore, in almost all cases, the best found solution is below the root bound for the half hour shift, i.e., the best found solutions are provably better than the best possible solution using a day-by-day approach. The benefit of larger time shifts is most clearly

expressed in the root bounds, where often a major decrease (often about 10%) is visible. Table 5 also shows, however, that the solution quality varies over the different runs. In particular, the objective value of the found solutions are not monotonically decreasing (or non-increasing) in the allowed shift size, something which is clearly the case would the problem have been solved to optimality. This variability seems to be mostly contained in the number of reserve duties in the found solution, the major driver behind the objective value. Table 5 does show that the number of scheduled empty duties tend to increase with larger shift sizes, an indication that the original duties can be used more efficiently, i.e., more tasks can be placed into a single duty.

The computation times shown in Table 5 are in line with the expectations: The running time for the half hour shift (i.e., the combined time of two CRSPs) is substantially lower than the time needed for the ICRPP, since there is no synchronization between the two days in this case. Furthermore, the running times increase in the shift size. Note, however, that the running times for the ICRPP can be considered reasonable: The average computation time never exceeds two hours, and in most cases stays well below one hour.

Summarizing, the integrated approach shows clear potential over the day-by-day approach: Allowing larger shifts leads to provably better solutions. The trade-off between the efficiency (i.e., number of necessary duties) versus the deviation from the original roster, however, is not clear, as the performance of the heuristic varies among runs. From a practical point of view, a small increase in shift (e.g., one hour) seems therefore the most profitable strategy: The solution value decreases substantially, the running time stays within one hour, and the newly scheduled duties stay relatively close to the original duties.

7 Conclusion

In this paper, we introduced the Integrated Crew Re-Planning Problem (ICRPP), an integrated approach for crew re-scheduling over multiple days. In doing so, we extend the Crew Re-Scheduling Problem (CRSP), by allowing more flexibility in the newly assigned duties. The additional complexity of the ICRPP resides in the problem size, as multiple days need to be re-scheduled at once, and in the fact that the feasibility of the rosters should be taken into account explicitly.

We proposed a mathematical formulation for the ICRPP and developed a column generation based heuristic to solve the problem. We applied the approach to four instances, based on data from NS. We analyzed the benefit of an integrated approach and considered the trade-off between the number of necessary duties and the deviation from the original plan. The results show a clear gain from integrating the solution process, yet also show that the performance of the heuristic varies among different runs. From a practical point of view, the integrated approach accompanied with a slightly larger flexibility in

the start and end times seems most profitable: The solution value decreases substantially, the increase in running time is limited, and the deviation from the original schedule will be relatively small.

For further research, the (primal) performance of the column generation heuristic seems most interesting. It is well-known that the incorporation of sophisticated local search methods within the column generation algorithm can substantially improve the performance. Similarly, exact branch-and-price methods could further highlight the potential of the integrated approach. Finally, heuristic approaches that iteratively enlarge the allowed time shift, either in an integrated or sequential way, could be an effective way of obtaining good solutions quickly.

Appendix A Overview Computational Results

Tables 6 and 7 show the computation results used for Figure 8. Table 6 shows the overall computation time and the computation time per shift size, averaged over three random seeds. Table 7 shows the average number of iterations and the average number of iterations per shift size, averaged over three random seeds.

Parameters			Computation Time (s)														
Nr.	Sim.	Perc.	Shift 00:30			Shift 01:00			Shift 02:00			Shift 05:00			Sum		
10	0.25	0.1	268.0	(32.0)	3/3	947.3	(93.1)	3/3	776.0	(176.3)	3/3	2011.3	(157.8)	3/3	4002.7	(105.2)	12/12
10	0.25	0.25	307.7	(2.5)	3/3	1019.3	(134.3)	3/3	1002.3	(111.1)	3/3	2151.7	(231.7)	3/3	4481.0	(246.4)	12/12
10	0.25	0.5	398.3	(9.0)	3/3	1088.3	(40.7)	3/3	1201.7	(184.0)	3/3	2848.0	(63.4)	3/3	5536.3	(279.2)	12/12
10	0.25	1.0	537.3	(4.0)	3/3	1671.7	(232.2)	3/3	1955.3	(254.3)	3/3	4621.0	(455.8)	3/3	8785.3	(430.6)	12/12
10	0.5	0.1	270.3	(91.8)	3/3	786.0	(106.4)	3/3	759.0	(114.0)	3/3	2080.3	(226.1)	3/3	3895.7	(75.5)	12/12
10	0.5	0.25	285.3	(18.1)	3/3	863.0	(126.4)	3/3	962.3	(123.8)	3/3	2242.3	(182.9)	3/3	4353.0	(187.4)	12/12
10	0.5	0.5	338.7	(5.3)	3/3	1127.0	(114.6)	3/3	1220.7	(165.7)	3/3	2764.3	(143.3)	3/3	5450.7	(212.4)	12/12
10	0.5	1.0	469.0	(7.8)	3/3	1493.7	(219.0)	3/3	1632.7	(273.9)	3/3	3552.7	(59.2)	3/3	7148.0	(132.3)	12/12
10	1.0	0.1	188.0	(7.5)	3/3	671.3	(66.3)	3/3	730.7	(65.3)	3/3	2694.7	(199.3)	3/3	4284.7	(174.4)	12/12
10	1.0	0.25	224.7	(2.5)	3/3	784.7	(110.3)	3/3	896.0	(147.4)	3/3	2994.3	(21.8)	3/3	4899.7	(58.0)	12/12
10	1.0	0.5	282.0	(4.3)	3/3	977.0	(121.3)	3/3	1352.0	(253.7)	3/3	3718.7	(135.7)	3/3	6329.7	(485.1)	12/12
10	1.0	1.0	407.7	(6.8)	3/3	1322.7	(150.2)	3/3	1450.3	(137.2)	3/3	4222.0	(122.0)	3/3	7402.7	(144.1)	12/12
100	0.25	0.1	342.0	(24.9)	3/3	827.0	(65.5)	3/3	710.3	(44.2)	3/3	2687.0	(342.2)	3/3	4566.3	(401.9)	12/12
100	0.25	0.25	371.0	(12.2)	3/3	1004.0	(156.3)	3/3	1001.3	(19.7)	3/3	2542.0	(18.4)	3/3	4918.3	(173.9)	12/12
100	0.25	0.5	488.3	(6.9)	3/3	1235.3	(171.1)	3/3	1472.3	(89.4)	3/3	3417.0	(80.8)	3/3	6613.0	(270.1)	12/12
100	0.25	1.0	715.3	(6.1)	3/3	1696.7	(219.0)	3/3	2197.3	(25.4)	3/3	5799.3	(385.0)	3/3	10408.7	(385.7)	12/12
100	0.5	0.1	244.3	(19.6)	3/3	707.0	(117.3)	3/3	753.0	(25.6)	3/3	2145.7	(171.4)	3/3	3850.0	(171.3)	12/12
100	0.5	0.25	276.7	(21.1)	3/3	849.3	(111.5)	3/3	868.7	(30.6)	3/3	2528.7	(269.6)	3/3	4523.3	(347.1)	12/12
100	0.5	0.5	369.3	(18.2)	3/3	955.0	(106.8)	3/3	1163.7	(35.2)	3/3	3050.0	(95.6)	3/3	5538.0	(124.0)	12/12
100	0.5	1.0	543.7	(7.0)	3/3	1362.0	(160.5)	3/3	1703.3	(78.1)	3/3	4357.0	(101.3)	3/3	7966.0	(164.9)	12/12
100	1.0	0.1	222.0	(25.4)	3/3	870.3	(32.8)	3/3	883.3	(96.6)	3/3	10332.3	(7221.3)	3/3	12308.0	(7283.0)	12/12
100	1.0	0.25	349.3	(35.6)	3/3	1114.0	(125.9)	3/3	1167.7	(57.0)	3/3	4141.0	(0.0)	1/3	4011.3	(1825.9)	10/12
100	1.0	0.5	500.7	(9.0)	3/3	1109.3	(151.6)	3/3	1581.7	(65.0)	3/3	7058.0	(0.0)	1/3	5544.3	(3222.4)	10/12
100	1.0	1.0	892.0	(32.6)	3/3	1522.7	(63.8)	3/3	3745.7	(1953.3)	3/3	9026.0	(0.0)	1/3	9169.0	(3709.4)	10/12
250	0.25	0.1	406.7	(88.0)	3/3	1062.0	(173.3)	3/3	896.0	(34.0)	3/3	2376.7	(193.4)	3/3	4741.3	(281.2)	12/12
250	0.25	0.25	531.0	(54.4)	3/3	1220.0	(109.1)	3/3	1247.0	(17.9)	3/3	2986.7	(216.0)	3/3	5984.7	(236.6)	12/12
250	0.25	0.5	730.3	(57.1)	3/3	1654.3	(277.3)	3/3	1887.0	(63.9)	3/3	4491.0	(144.5)	3/3	8762.7	(413.3)	12/12
250	0.25	1.0	1184.0	(114.1)	3/3	2385.0	(282.9)	3/3	3083.3	(73.6)	3/3	7421.7	(500.4)	3/3	14074.0	(772.4)	12/12
250	0.5	0.1	274.0	(45.7)	3/3	819.3	(85.1)	3/3	792.0	(48.0)	3/3	2143.0	(123.3)	3/3	4028.3	(131.4)	12/12
250	0.5	0.25	396.0	(43.8)	3/3	887.0	(117.9)	3/3	1124.3	(36.3)	3/3	3401.0	(937.8)	3/3	5808.3	(1048.3)	12/12
250	0.5	0.5	523.0	(63.1)	3/3	1119.3	(88.7)	3/3	1408.3	(16.3)	3/3	4458.3	(1268.9)	3/3	7509.0	(1332.2)	12/12
250	0.5	1.0	815.3	(93.3)	3/3	1690.0	(227.7)	3/3	2188.0	(69.3)	3/3	7260.7	(2559.7)	3/3	11954.0	(2769.1)	12/12
250	1.0	0.1	419.7	(58.7)	3/3	1198.3	(53.2)	3/3	2889.7	(2208.5)	3/3	0.0	(0.0)	0/3	4507.7	(2211.6)	9/12
250	1.0	0.25	732.7	(16.4)	3/3	1477.3	(91.1)	3/3	1945.0	(151.4)	3/3	0.0	(0.0)	0/3	4155.0	(109.6)	9/12
250	1.0	0.5	1545.7	(173.6)	3/3	2741.3	(480.1)	3/3	6157.0	(4181.5)	3/3	15372.0	(0.0)	1/3	15568.0	(6431.1)	10/12
250	1.0	1.0	2972.7	(80.1)	3/3	4218.3	(519.8)	3/3	8133.7	(2195.0)	3/3	0.0	(0.0)	0/3	15324.7	(2628.5)	9/12

Table 6: Solution time for the root node for different parameter settings. For each configuration and each shift, we considered three different random seeds, and show the average and standard deviation (in brackets) of the computation time over these seeds (taken over the runs that terminated in time), together with the number of runs that finished within 6 hours. The parameters are: the number of paths returned per RCSPP (Nr), the similarity threshold (Sim), and the percentage of pricing problems solved in partial pricing (Perc).

Parameters			Iterations														
Nr.	Sim.	Perc.	Shift 00:30			Shift 01:00			Shift 02:00			Shift 05:00			Sum		
10	0.25	0.1	109.3	(7.6)	3/3	156.0	(7.3)	3/3	67.7	(3.1)	3/3	56.0	(1.6)	3/3	389.0	(1.4)	12/12
10	0.25	0.25	95.0	(0.8)	3/3	115.7	(3.7)	3/3	55.3	(2.5)	3/3	45.0	(0.8)	3/3	311.0	(2.9)	12/12
10	0.25	0.5	85.0	(1.4)	3/3	88.0	(1.6)	3/3	48.7	(1.7)	3/3	41.7	(0.9)	3/3	263.3	(4.5)	12/12
10	0.25	1.0	80.3	(1.2)	3/3	83.0	(2.2)	3/3	47.3	(0.5)	3/3	44.0	(1.6)	3/3	254.7	(1.7)	12/12
10	0.5	0.1	96.3	(19.0)	3/3	122.7	(0.9)	3/3	55.7	(1.7)	3/3	44.7	(1.2)	3/3	319.3	(20.3)	12/12
10	0.5	0.25	81.3	(4.5)	3/3	89.7	(3.3)	3/3	44.7	(1.7)	3/3	36.0	(1.4)	3/3	251.7	(2.5)	12/12
10	0.5	0.5	67.0	(0.0)	3/3	79.3	(2.1)	3/3	39.3	(1.2)	3/3	32.0	(0.8)	3/3	217.7	(3.8)	12/12
10	0.5	1.0	63.3	(0.5)	3/3	67.7	(0.5)	3/3	35.0	(2.2)	3/3	30.7	(0.5)	3/3	196.7	(3.3)	12/12
10	1.0	0.1	67.3	(2.6)	3/3	95.0	(6.4)	3/3	41.3	(0.9)	3/3	33.0	(4.2)	3/3	236.7	(8.6)	12/12
10	1.0	0.25	58.7	(1.7)	3/3	73.0	(2.4)	3/3	32.3	(0.9)	3/3	25.3	(0.5)	3/3	189.3	(4.0)	12/12
10	1.0	0.5	50.7	(0.5)	3/3	61.0	(0.8)	3/3	28.7	(1.2)	3/3	23.7	(1.2)	3/3	164.0	(1.4)	12/12
10	1.0	1.0	47.0	(0.8)	3/3	54.0	(1.6)	3/3	25.7	(1.2)	3/3	22.0	(0.8)	3/3	148.7	(1.2)	12/12
100	0.25	0.1	83.0	(4.9)	3/3	119.7	(8.3)	3/3	44.3	(2.6)	3/3	36.3	(1.2)	3/3	283.3	(7.8)	12/12
100	0.25	0.25	63.3	(3.4)	3/3	78.0	(2.4)	3/3	37.3	(0.5)	3/3	28.0	(0.8)	3/3	206.7	(5.9)	12/12
100	0.25	0.5	54.7	(0.5)	3/3	58.7	(0.5)	3/3	31.0	(0.8)	3/3	26.0	(1.4)	3/3	170.3	(1.9)	12/12
100	0.25	1.0	50.7	(0.9)	3/3	51.3	(0.9)	3/3	30.0	(0.0)	3/3	26.7	(1.2)	3/3	158.7	(1.2)	12/12
100	0.5	0.1	52.3	(4.2)	3/3	83.3	(1.9)	3/3	32.0	(0.8)	3/3	22.0	(0.8)	3/3	189.7	(3.8)	12/12
100	0.5	0.25	42.7	(3.9)	3/3	65.7	(3.7)	3/3	22.3	(0.5)	3/3	19.0	(0.8)	3/3	149.7	(6.2)	12/12
100	0.5	0.5	37.7	(3.1)	3/3	41.0	(1.4)	3/3	20.7	(0.5)	3/3	17.7	(0.5)	3/3	117.0	(4.2)	12/12
100	0.5	1.0	34.3	(0.5)	3/3	36.7	(0.5)	3/3	20.0	(0.8)	3/3	17.3	(0.9)	3/3	108.3	(1.2)	12/12
100	1.0	0.1	35.0	(4.5)	3/3	64.3	(3.8)	3/3	21.0	(2.4)	3/3	14.0	(0.8)	3/3	134.3	(8.1)	12/12
100	1.0	0.25	34.3	(6.6)	3/3	45.0	(6.7)	3/3	15.0	(0.8)	3/3	11.0	(0.0)	1/3	98.0	(12.8)	10/12
100	1.0	0.5	23.7	(0.5)	3/3	29.0	(1.4)	3/3	13.7	(0.9)	3/3	11.0	(0.0)	1/3	70.0	(3.6)	10/12
100	1.0	1.0	21.3	(0.5)	3/3	23.7	(1.7)	3/3	12.0	(0.0)	3/3	10.0	(0.0)	1/3	60.3	(6.2)	10/12
250	0.25	0.1	63.0	(7.8)	3/3	112.3	(7.0)	3/3	39.7	(1.7)	3/3	29.0	(0.8)	3/3	244.0	(9.9)	12/12
250	0.25	0.25	53.0	(5.0)	3/3	76.3	(4.1)	3/3	29.3	(0.9)	3/3	23.0	(0.8)	3/3	181.7	(7.8)	12/12
250	0.25	0.5	45.3	(2.1)	3/3	56.7	(4.1)	3/3	26.3	(1.2)	3/3	21.3	(0.5)	3/3	149.7	(4.5)	12/12
250	0.25	1.0	42.7	(0.5)	3/3	47.3	(1.2)	3/3	25.7	(0.9)	3/3	22.3	(0.5)	3/3	138.0	(0.8)	12/12
250	0.5	0.1	38.0	(3.6)	3/3	86.0	(12.1)	3/3	24.7	(0.5)	3/3	17.7	(0.9)	3/3	166.3	(14.1)	12/12
250	0.5	0.25	36.0	(5.0)	3/3	53.7	(3.3)	3/3	20.0	(1.4)	3/3	14.7	(0.5)	3/3	124.3	(6.8)	12/12
250	0.5	0.5	29.0	(0.8)	3/3	36.0	(2.9)	3/3	14.7	(0.5)	3/3	13.7	(0.9)	3/3	93.3	(4.7)	12/12
250	0.5	1.0	27.0	(0.0)	3/3	29.3	(1.2)	3/3	15.0	(0.8)	3/3	13.7	(0.5)	3/3	85.0	(1.4)	12/12
250	1.0	0.1	32.7	(2.5)	3/3	62.3	(6.1)	3/3	19.7	(5.4)	3/3	0.0	(0.0)	0/3	114.7	(10.2)	9/12
250	1.0	0.25	25.3	(2.6)	3/3	37.3	(3.1)	3/3	11.3	(1.2)	3/3	0.0	(0.0)	0/3	74.0	(5.4)	9/12
250	1.0	0.5	19.0	(0.0)	3/3	21.7	(0.5)	3/3	9.3	(0.5)	3/3	10.0	(0.0)	1/3	53.3	(4.0)	10/12
250	1.0	1.0	17.0	(0.8)	3/3	18.3	(0.9)	3/3	9.0	(0.0)	3/3	0.0	(0.0)	0/3	44.3	(1.2)	9/12

Table 7: Number of column generation iterations when solving the root node for different parameter settings. For each configuration and each shift, we considered three different random seeds, and show the average and standard deviation (in brackets) of the number of iterations over these seeds (taken over the runs that terminated in time), together with the number of runs that finished within 6 hours. The parameters are: the number of paths returned per RCSPP (Nr), the similarity threshold (Sim), and the percentage of pricing problems solved in partial pricing (Perc).

References

E. Abbink, M. Fischetti, L. Kroon, G. Timmer, and M. Vromans. Reinventing crew scheduling at Netherlands Railways. *Interfaces*, 35(5):393–401, 2005.

E. Abbink, D. Huisman, and L. Kroon. *Railway Crew Management*, pages 243–264. Springer International Publishing, Cham, 2018.

C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.

R. Borndörfer, A. Langenhan, A. Löbel, C. Schulz, and S. Weider. Duty scheduling templates. *Public Transport*, 5(1-2):41–51, 2013.

R. Borndörfer, M. Reuther, T. Schlechte, C. Schulz, E. Swarat, and S. Weider. Duty rostering in public transport-facing preferences, fairness, and fatigue. In *CASPT*, 2015.

R. Borndörfer, C. Schulz, S. Seidl, and S. Weider. Integration of duty scheduling and rostering to increase driver satisfaction. *Public Transport*, 9(1):177–191, 2017.

V. Cacchiani, D. Huisman, M. Kidd, L. Kroon, P. Toth, L. Veelenturf, and J. Wagenaar. An overview of recovery models and algorithms for real-time railway rescheduling. *Transportation Research Part B: Methodological*, 63:15–37, 2014.

A. Caprara, L. Kroon, M. Monaci, M. Peeters, and P. Toth. Passenger railway optimization. *Handbooks in Operations Research and Management Science*, 14:129–187, 2007.

J. Clausen, A. Larsen, J. Larsen, and N. J. Rezanova. Disruption management in the airline industry-concepts, models and methods. *Computers & Operations Research*, 37 (5):809–821, 2010.

G. Desaulniers, J. Desrosiers, and M. M. Solomon. Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems. In *Essays and surveys in metaheuristics*, pages 309–324. Springer, 2002.

G. Desaulniers, J. Desrosiers, and M. M. Solomon. *Column Generation*, volume 5. Springer Science & Business Media, 2006.

M. Desrochers and F. Soumis. A column generation approach to the urban transit crew scheduling problem. *Transportation Science*, 23(1):1–13, 1989.

A. T. Ernst, H. Jiang, M. Krishnamoorthy, H. Nott, and D. Sier. An integrated optimization model for train crew management. *Annals of Operations Research*, 108(1-4): 211–224, 2001.

M. Grötschel, R. Borndörfer, and A. Löbel. Duty scheduling in public transit. In *Mathematics-Key Technology for the Future*, pages 653–674. Springer, 2003.

A. Hartog, D. Huisman, E. Abbink, and L. Kroon. Decision support for crew rostering at NS. *Public Transport*, 1(2):121–133, 2009.

J. Heil, K. Hoffmann, and U. Buscher. Railway crew scheduling: Models, methods and applications. *European Journal of Operational Research*, 2019. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2019.06.016.

K. L. Hoffman and M. Padberg. Solving airline crew scheduling problems by branch-and-cut. *Management Science*, 39(6):657–682, 1993.

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

D. Huisman, L. G. Kroon, R. M. Lentink, and M. J. C. M. Vromans. Operations research in passenger railway transportation. *Statistica Neerlandica*, 59(4):467–497, 2005.

C. Joncour, S. Michel, R. Sadykov, D. Sverdlov, and F. Vanderbeck. Column generation based primal heuristics. *Electronic Notes in Discrete Mathematics*, 36:695–702, 2010.

N. Kohl and S. E. Karisch. Airline crew rostering: Problem types, modeling, and optimization. *Annals of Operations Research*, 127(1-4):223–257, 2004.

L. Kroon and M. Fischetti. Crew scheduling for netherlands railways destination: customer. In *Computer-aided scheduling of public transport*, pages 181–201. Springer, 2001.

S. Lavoie, M. Minoux, and E. Odier. A new approach for crew pairing problems by column generation with an application to air transportation. *European Journal of Operational Research*, 35(1):45–58, 1988.

L. Lettovskỳ, E. L. Johnson, and G. L. Nemhauser. Airline crew recovery. *Transportation Science*, 34(4):337–348, 2000.

M. E. Lübbecke. Column generation. *Wiley Encyclopedia of Operations Research and Management Science*, 2011.

M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

M. Mesquita, M. Moz, A. Paias, and M. Pato. A decomposition approach for the integrated vehicle-crew-roster problem with days-off pattern. *European Journal of Operational Research*, 229(2):318–331, 2013.

J. D. Petersen, G. Sölveling, J. Clarke, E. L. Johnson, and S. Shebalov. An optimization approach to airline integrated recovery. *Transportation Science*, 46(4):482–500, 2012.

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

N. J. Rezanova and D. M. Ryan. The train driver recovery problem-a set partitioning

based model and solution method. *Computers & Operations Research*, 37(5):845–856, 2010.

K. Sato and N. Fukumura. Real-time freight locomotive rescheduling and uncovered train detection during disruption. *European Journal of Operational Research*, 221(3): 636–648, 2012.

M. Sodhi and S. Norris. A flexible, fast, and optimal modeling approach applied to crew rostering at London Underground. *Annals of Operations Research*, 127(1-4):259–281, 2004.

M. Stojković and F. Soumis. An optimization model for the simultaneous operational flight and pilot scheduling problem. *Management Science*, 47(9):1290–1305, 2001.

M. Stojković, F. Soumis, and J. Desrosiers. The operational airline crew scheduling problem. *Transportation Science*, 32(3):232–245, 1998.

L. P. Veelenturf, D. Potthoff, D. Huisman, and L. G Kroon. Railway crew rescheduling with retiming. *Transportation research part C: emerging technologies*, 20(1):95–110, 2012.

L. P. Veelenturf, D. Potthoff, D. Huisman, L. G. Kroon, G. Maróti, and A. P. M. Wagelmans. A quasi-robust optimization approach for crew rescheduling. *Transportation Science*, 50(1):204–215, 2014.

C. G. Walker, J. N. Snowdon, and D. M. Ryan. Simultaneous disruption recovery of a train timetable and crew roster in real time. *Computers & Operations Research*, 32(8): 2077–2094, 2005.