Analyzing a Family of Formulations for Cyclic Crew Rostering

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 - EI2018-35

Abstract

In this paper, we analyze a family of formulations for the Cyclic Crew Rostering Problem (CCRP), in which a cyclic roster has to be constructed for a group of employees. We derive analytical results regarding the relative strength of the different formulations, which can serve as a guideline for formulating a given problem instance. Furthermore, we propose a column generation approach, which we use to develop an exact Branch-and-Price method, and a heuristic which aims at exploiting the information obtained from the linear relaxation. We conclude by applying our proposed solution method to practical instances from Netherlands Railways. In particular, we show that the computation time depends heavily on the selected formulation, and that the column generation approach outperforms a commercial solver on hard instances.

Keywords: Crew Planning, Roster Sequence, Branch-and-Price, Railway Optimization

1 Introduction

The construction of rosters (often referred to as crew rostering) is an important part of the planning process at a public transport operator. As opposed to many other planning problems at a railway operator (e.g., rolling stock scheduling), the main goal in crew rostering is not to minimize costs. Instead the goal is to maximize the attractiveness of the roster from an employees' point of view. This implies that, for example, the rest time between consecutive working days and the variation of work over a week have to be taken into account when constructing the rosters. Altogether, this leads to a complex optimization problem.

The inclusion of attractiveness in crew planning has shown to be important in practice. In the Netherlands, for example, the incorporation of attractiveness in crew planning was vital in resolving conflicts between the labor unions and Netherlands Railways (NS), the largest railway operator in the Netherlands. An import development in this respect was the introduction of the 'Sharing-Sweet-and-Sour' rules, which aim to increase the quality of work (see Abbink et al. [2005] for a detailed discussion). These rules, for example, assure that 'nice work' (e.g., long distance trips) are equally distributed among all depots. Borndörfer et al. [2017] discuss the importance of attractive work in Germany. They note, for example, that the bus drivers at BVG (Berlin's public transport company) have an average age of around 50 years. The focus on attractiveness, in this case, can be a key instrument for recruiting new (younger) drivers, as the possibility for higher salaries is generally limited.

The focus on the distribution of work is apparent in the use of roster groups, which are groups of employees with similar characteristics (e.g., age, preferences), that operate in cyclic rosters. This implies that, after a certain time period, each employee in a group has done the exact same work, assuring a fair distribution of work within each group. In the Cyclic Crew Rostering Problem (CCRP) the goal is to maximize the attractiveness of a roster constructed for such a group.

In this paper we introduce a family of formulations for the CCRP. This family of formulations includes the assignment formulation proposed in Hartog et al. [2009], and a formulation similar to the one proposed in Breugem et al. [2017] for the Fairness-oriented Crew Rostering Problem (FCRP). The family of formulations is motivated by the poor performance of typical assignment models on difficult instances. Each formulation corresponds to a different partition of the rosters, hence, depending on the constraints, a strong formulation can be obtained by picking a suitable formulation from the family. The partition based on the weeks of the roster is a good example of a possible partitioning.

The contribution of this paper is fourfold. Firstly, we propose a family of formulations for the CCRP, based on the formulation originally proposed in Breugem et al. [2017]. Secondly, we derive analytical results regarding the strength of the proposed formulations. In particular, the derived results give intuition on how to pick a suitable formulation for a given CCRP instance. Thirdly, we develop a column generation approach to solve this model, on which we base an exact Branch-and-Price approach and a heuristic method. Finally, we show the benefit of our approach using practical instances from NS.

The remainder of the paper is organized as follows. In Section 2, we formalize the CCRP. In Section 3, we give an overview of related work, and in Section 4, we introduce our mathematical formulation. In Section 5, we derive analytical results regarding the tightness of the formulation. In Sections 6 and 7, we discuss the Branch-and-Price approach and the heuristic, respectively. Finally, in Section 8, we apply our solution approaches to

practical instances of NS, and the paper is concluded in Section 9.

2 Problem Description

At many public transport operators, the rosters for the employees are constructed for a period of one year, and the timetable for this period is assumed to be the same for every week. This implies, for example, that there are generic Monday tasks, which need to be executed each week. In other words, the tasks that need to be executed on the 3rd of March are identical to the tasks that need to be executed on the 10th of March. This property leads to the use of *cyclic rosters*. In such a roster each employee cycles through the rows (i.e., generic weeks) of the roster, and hence each row in the roster is executed multiple times by different employees.

The goal of the CCRP is to construct a cyclic roster for a group of employees. The input to the CCRP is a *basic schedule*, and a set of duties. The basic schedule specifies the key elements of the roster. The schedule could, for example, specify for each day whether a duty or a rest day needs to be scheduled. Figure 1 gives an example of a basic schedule for three employees. Each row in the basic schedule (indicated on the left) represents a week of work, which is consecutively executed by one of the employees. The duties represent the work that needs to be scheduled. Each duty specifies a sequence of tasks, where a task specifies, for example, that the employee has to drive a train from Amsterdam to Rotterdam at 11:15. The duties can also be divided in categories, based on, e.g., their start time.

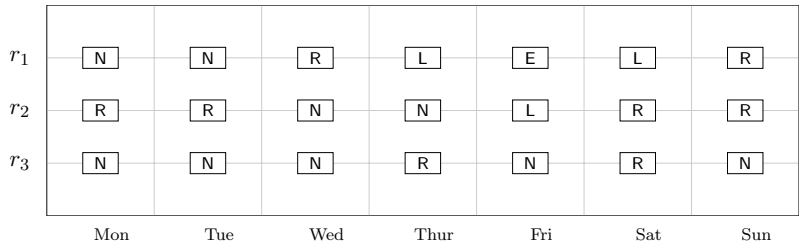


Figure 1: Example of a basic schedule for three employees. Each day specifies either one of three duty types: Early (E), Late (L) or Night (N), or a rest day (R).

Figure 2 shows an example of a cyclic roster for three employees, which fits the basic schedule depicted in Figure 1. Each of the scheduled duties is indicated by its index, and the rows are indicated on the left. The cyclicity of the roster implies that the first employee starts with row r_1, the second employee with r_2, and the third employee with r_3. In the next week, the employees 'cycle': The first employee executes r_2, the second employee executes r_3, and the third employee executes r_1. After three weeks, the cycle is completed, and the process repeats itself. Note that the cyclicity implies that there can

be overlap between different rows: Duty 106 starts on Sunday in the third row, and ends on Monday in the first row.

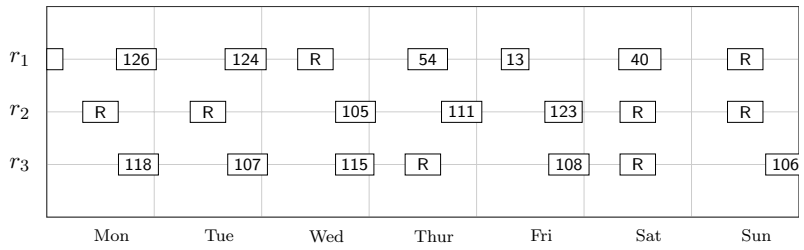


Figure 2: Example of a cyclic roster for three employees.

Two important aspects have to be taken into account when constructing the rosters. Firstly, the roster should be feasible with respect to the labor regulations. For example, there should be sufficient rest time between consecutive duties, and the total amount of work in a row (i.e., in a week of work) cannot be too large. Secondly, the roster should be perceived *attractive* by the employees. Short, although legal, rest times, for example, make the roster unattractive, as employees prefer a proper rest period between two duties. Both the feasibility and perceived attractiveness are expressed using *roster constraints*. Feasibility is modeled using hard constraints, whereas attractiveness is modeled using soft constraints, thereby penalizing unattractive assignments of duties.

The CCRP can now be formulated as follows: Given a set of duties, and a basic schedule, create a cyclic roster, adhering to the basic schedule, such that all duties are scheduled, and the perceived attractiveness of the roster is maximized. The objective in the CCRP is to maximize the attractiveness of the roster from an employees' point of view.

3 Related Work

Crew planning is well-studied in the literature (see e.g., Van den Bergh et al. [2013] for an extensive overview). In mass transit and railway optimization the focus is mainly on cyclic rosters (see Huisman et al. [2005], Caprara et al. [2007]) whereas in, for example, the airline industry the focus is mainly on acyclic rosters (see e.g., Kohl and Karisch [2004]). Generally, crew planning is decomposed into crew scheduling and crew rostering. In the former, the duties (i.e., days of work) have to be constructed (see, for example, Abbink et al. [2005]), whereas in the latter, the rosters are constructed, given the duties.

The crew rostering problem occurs in two variants: either the basic schedules are considered input, or the construction of the basic schedules is part of the planning problem (either in a sequential fashion, or simultaneously). In the former case the focus lies on finding an attractive allocation of the duties, whereas in the latter case also the feasibility

of the basic schedule has to be taken into account (e.g., each number of sequential rows should have a proper weekend) . In this paper, we consider the first variant.

Different models have been proposed to solve the cyclic crew rostering problem. Caprara et al. [1997] develop both a multi-commodity flow model and a set partitioning model for crew rostering. Furthermore, they argue which formulation is more suitable, given the constraint set. They apply these models to the Italian railway case. Freling et al. [2004] develop a Branch-Price-and-Cut algorithm based on a set covering formulation. Hartog et al. [2009] propose an assignment model with side constraints, in which they first optimize the basic schedule, and then the assignment of the duties to these schedules. The latter problem is similar to the crew rostering problem considered in this paper. We note that this type of decomposition was first considered in Sodhi and Norris [2004].

In recent work, Borndörfer et al. [2015] propose both a network flow model and a set partitioning model, for which they propose a heuristic solution approach based on the Lin-Kernighan heuristic. In Borndörfer et al. [2017] an integration of duty scheduling and rostering is proposed. The authors propose a solution method based on Benders decomposition, and show that, by altering the duties (slightly), the attractiveness of the roster can be improved at almost zero cost. Finally, Breugem et al. [2017] consider a variant of the problem in which also the fairness (i.e., the equity of the duty allocation) is taken into account. The authors propose a formulation based on the rows of the roster, which is solved using a Branch-Price-and-Cut approach. The underlying crew rostering problem (i.e., obtained by omitting fairness) relates closely to the one considered in this research.

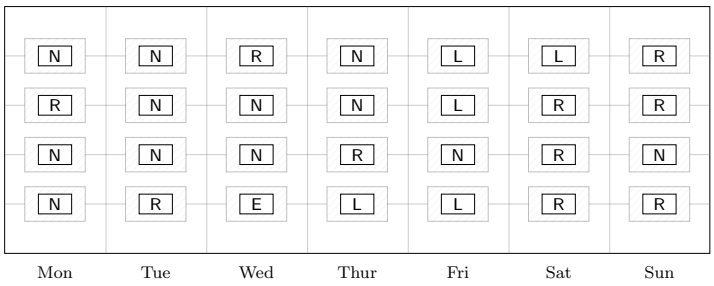
4 Mathematical Model

In this section we propose a family of mathematical formulations for the CCRP. In Section 4.1, we define the concept of clusters and roster sequences. In Section 4.2, we discuss how the roster constraints are modeled, and we introduce the necessary notation in Section 4.3. We conclude with a family of mathematical formulations for the CCRP in Section 4.4. The modeling techniques presented in this section relate closely to those of Breugem et al. [2017], and, to a lesser extent, Hartog et al. [2009].

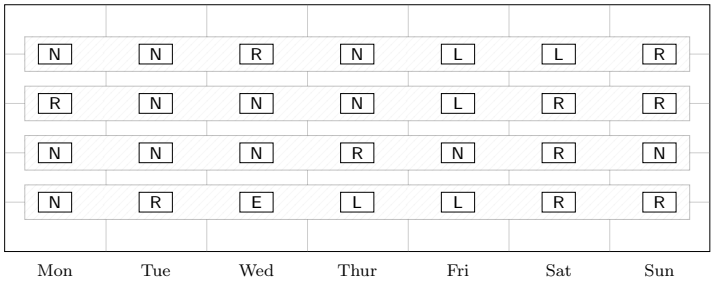
4.1 Clusters and Roster Sequences

The family of formulations is based on different partitions of the basic schedule. That is, we develop a mathematical formulation under the assumption that each basic schedule is partitioned into disjoint subsets, which we call *clusters*. This partition will be referred to as a *clustering* for the respective basic schedule, and should be picked a priori solving the model.

The formulation will have a different structure for each possible clustering, giving rise to the family of formulations. Figure 3 gives an example of two possible clusterings for a basic schedule of four rows. In the clustering in Figure 3a, each cluster contains exactly one of the days in the basic schedule. The clustering in Figure 3b, on the other hand, assigns all days in the same row (i.e., Monday to Sunday) to the same cluster. Note that many more clusterings are possible. One could, for example, also consider a 'weekend' clustering, in which each cluster relates to either Friday to Monday (the 'weekend' days), or Tuesday to Thursday (the 'week' days). Such a clustering can be a good choice when, e.g., the rest time over the weekend is of utmost importance. Generally, days in a cluster do not need to be consecutive.



(a) Day clustering.



(b) Row clustering.

Figure 3: Example of different clusterings. Each highlighted area represents a cluster.

Each cluster is assigned a number of duties simultaneously. Each possible assignment of duties to a cluster is called a *roster sequence*. Formally, a roster sequence specifies a duty or rest day for each day in the cluster, such that the assignment is compatible with the basic schedule, and such that no duty is assigned twice (within the same cluster).

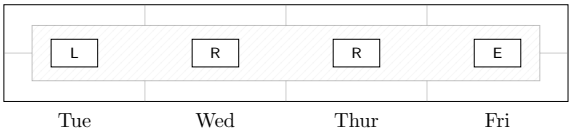


Figure 4: Cluster from Tuesday to Friday.

To illustrate the use of roster sequences, consider the cluster depicted in Figure 4. Two

possible roster sequences for this cluster are depicted in Figure 5. Both roster sequences contain different duties (indicated by the numbers). Note that in this case the second roster sequence has a shorter rest period than the first roster sequence (as duty 124 ends later than duty 126, and duty 58 starts earlier than duty 54), which might be considered undesirable.

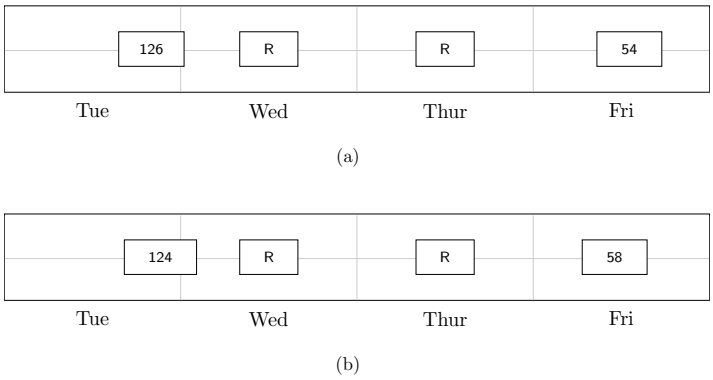


Figure 5: Two possible roster sequences for the cluster of Figure 4.

The goal of a clustering is to model constraints implicitly using the roster sequences. That is, ideally each constraint considers the days in *solely* one of the clusters, and can therefore be taken care of when generating the roster sequences. As an example, consider a constraint in which an employee can have only a maximum amount of work per row. In this case, the clustering of Figure 3b allows to model these constraints implicitly using the roster sequences (i.e., a roster sequence is feasible only if it does not exceed the maximum working time). On the other hand, for the clustering of Figure 3a these constraints have to be modeled explicitly in the mathematical formulation. Another example considers proper rest periods. That is, whenever one or more consecutive rest days are scheduled, there should be a minimum amount of time before the start time of the first duty *after* the rest period and the end time of the last duty *before* the rest period. This rest period could be modeled implicitly in the case of the cluster of Figure 4. It could, for example, be the case that the first roster sequence depicted in Figure 5 has a proper rest period, whereas the second roster sequence has not. In this case, the second roster sequence would not be generated.

4.2 Roster Constraints

We now rigorously show how to model *roster constraints*, i.e., constraints regarding the assignment of duties to the basic schedule. As illustrated in previous examples, the roster constraints define the quality of the roster, and assure feasibility. The former is done using *soft* constraints (i.e., violation results in a penalty), whereas the latter is enforced by *hard* constraints (i.e., violation is never allowed). The roster constraints are modeled similarly

as in Hartog et al. [2009] and Breugem et al. [2017]: Each roster constraint is specified by (i) a coefficient for each assignment of a duty to a day in the basic schedule, (ii) a scalar called the *threshold value*, and (iii) a closed interval called the *violation interval*. Without loss of generality, the interval is assumed to be of the form $[0, u]$, for some $u \geq 0$. The roster constraints enforce that *if* the sum of coefficients of assigned duties exceeds the threshold value, *then* the difference between the sum and the threshold value lies within the violation interval.

Many different constraints can be modeled within the above framework. Consider, for example, a roster constraint which concerns the total workload (i.e., the cumulative length of all duties) in a given row of the roster. In particular, the constraint expresses that the total workload is at most 45 hours. This constraint can be modeled as follows: Let w denote the set of days in the considered row. Furthermore, let D denote the set of duties, and let ℓ_d denote the length of duty d. Let the variable x_{td} indicate whether duty d is scheduled on day t, and let δ model a possible violation. The above constraint can be expressed as

$$\sum_{t \in w} \sum_{d \in D} \ell_d x_{td} \leq 45 + \delta.$$

If no violation is allowed, we have $\delta \in \{0\}$. If, for example, a violation of at most two hours is allowed (i.e., a workload of at most 47 hours), albeit against a penalty, we have $\delta \in [0, 2]$. Note that for this particular constraint, the coefficient for assignment (t, d) is given by ℓ_d if $t \in w$, and 0 otherwise, the threshold value is 45, and the violation interval is the domain of δ.

Roster constraints that only consider assignments to days in a single cluster are modeled implicitly using the set of roster sequences (i.e., infeasible roster sequences are omitted). The above constraint, for example, can be modeled implicitly if there is a cluster containing all days in w (e.g., the row clustering of Figure 3). Otherwise, the constraint has to be modeled explicitly (i.e., the constraint has to be added to the formulation).

4.3 Notation

We are now ready to introduce the notation for the mathematical formulation. Let D denote the set of duties. The basic schedule is represented by a set of days T, where for each day $t \in T$ it is known which duties can be scheduled. An assignment of a duty $d \in D$ to a day $t \in T$ in the basic schedule will be denoted by the pair (t, d).

The set K denotes the set of all clusters (note that these are determined a priori formulating the mathematical model). We define the set S_k as the set of all roster sequences for cluster $k \in K$. Each roster sequence can be seen as a sequence of assignments (t, d). The parameter h^k_{ds} indicates whether roster sequence $s \in S_k$ contains duty d (i.e., duty d appears in one of the assignments describing the roster sequence s). Finally, we define c^k_s

as the penalty associated with roster sequence $s \in S_k$ for cluster $k \in K$.

The set of roster constraints is denoted by P. The coefficient for the assignment (t, d) for roster constraint p is denoted by f_{td}^p. The threshold value for p is denoted by b_p, and the violation interval for p is denoted by $\Delta_p = [0, u_p]$. The penalty corresponding to roster constraint p is denoted by c_p.

Let $P_k \subseteq P$ denote the set of roster constraints fully contained in cluster $k \in K$, and define $P_K = \bigcup_{k \in K} P_k$. The constraints in P_K are exactly those that are modeled implicitly using the roster sequences. The penalty c_s^k associated with roster sequence $s \in S_k$ is the sum of all violations in the roster sequence s, restricted to the roster constraints P_k. Note that the roster constraints in $P \setminus P_K$ need to be modeled explicitly.

4.4 Family of Mathematical Formulations

The CCRP, given a clustering K, can now be formulated as follows. We introduce the binary variable x_s^k, which indicates whether roster sequence $s \in S_k$ is assigned to cluster $k \in K$. Furthermore, we introduce the variable δ_p, for each $p \in P \setminus P_K$, which expresses the violation of the roster constraint $p \in P \setminus P_K$. The variable δ_p is restricted to the violation domain Δ_p. The formulation now reads as follows.

$$\min \quad \sum_{k \in K} \sum_{s \in S_k} c_s^k x_s^k + \sum_{p \in P \setminus P_K} c_p \delta_p \tag{1}$$

$$\text{s.t.} \quad \sum_{s \in S_k} x_s^k = 1 \qquad \forall k \in K \tag{2}$$

$$\sum_{k \in K} \sum_{s \in S_k} h_{ds}^k x_s^k = 1 \qquad \forall d \in D \tag{3}$$

$$\sum_{k \in K} \sum_{s \in S_k} \sum_{(t,d) \in s} f_{td}^p x_s^k \leq b_p + \delta_p \qquad \forall p \in P \setminus P_K \tag{4}$$

$$x_s^k \in \mathbb{B} \qquad \forall k \in K, s \in S_k \tag{5}$$

$$\delta_p \in \Delta_p \qquad \forall p \in P \setminus P_K. \tag{6}$$

The Objective (1) expresses that we minimize the sum of the roster sequence costs, together with the cost of all explicitly modeled roster constraints. Constraints (2) and (3) assure that the duties are assigned correctly to the basic schedules. That is, each cluster is assigned exactly one roster sequence, and each duty is assigned exactly once to a day in the basic schedule. Constraints (4) represent the roster constraints that are modeled explicitly. Finally, Constraints (5) and (6) specify the domains of the decision variables. The family of formulations for the CCRP is now obtained by taking (1)–(6) for all possible clusterings K.

5 Theoretical Analysis

Intuitively, the implicit modeling of the roster constraint violations leads to a tighter linear relaxation. In this section, we prove this rigorously. From hereon, we consider two clusterings K and L such that $P_K \supseteq P_L$ (note that this does *not* necessarily imply that every $\ell \in L$ is contained in some $k \in K$). An example of such clusterings is given in Figure 3, where K and L are the row and day clustering, respectively. Let S_k, for all $k \in K$, and Q_ℓ, for all $\ell \in L$, denote the respective sets of roster sequences for both clusters. For notational convenience, define Ω as the set of all feasible assignments (t, d), with $t \in T$ and $d \in D$, of duties to the days in the basic schedule. Furthermore, we define the operator $[\cdot]^+$ as $[a]^+ = \max\{0, a\}$. Throughout this section, a solution refers to a solution to the linear relaxation. We first prove the following lemma. Intuitively, this lemma states that, given a solution for K, we can construct a solution for L such that each duty is assigned to the same day in both solutions.

Lemma 5.1. *Let $\bar{x}$ be a solution for clustering K. There exists a feasible solution $\bar{z}$ for clustering L, such that*

$$\sum_{k \in K} \sum_{\substack{s \in S_k: \\ s \ni (t,d)}} \bar{x}_s^k = \sum_{\ell \in L} \sum_{\substack{q \in Q_\ell: \\ q \ni (t,d)}} \bar{z}_q^\ell \tag{7}$$

for each $(t, d) \in \Omega$.

Proof. We consider an auxiliary clustering O, defined as the largest clustering which is fully contained in both K and L (see Figure 6). Formally, O is uniquely defined by taking all non-empty subsets $k \cap \ell$, for all $k \in K$ and $\ell \in L$. Let R denote the set of feasible roster sequences for this clustering, and let R_o denote the feasible roster sequences for $o \in O$.

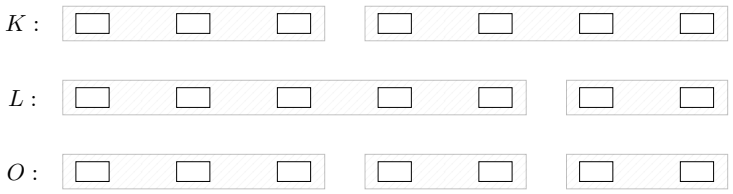


Figure 6: Example of the clustering O, which is the largest clustering contained in both K and L.

Since each cluster $o \in O$ is fully contained in some $k \in K$, we can readily obtain a solution

$\bar{y}$ for O satisfying

$$\sum_{k \in K} \sum_{\substack{s \in S_k: \\ s \ni (t,d)}} \bar{x}_s^k = \sum_{o \in O} \sum_{\substack{r \in R_o: \\ r \ni (t,d)}} \bar{y}_r^o \tag{8}$$

by splitting up each roster sequence for K into smaller roster sequences for O. Furthermore, since each $o \in O$ is also fully contained in some $\ell \in L$, we can obtain a solution $\bar{z}$ satisfying

$$\sum_{\ell \in L} \sum_{\substack{q \in Q_\ell: \\ q \ni (t,d)}} \bar{z}_q^\ell = \sum_{o \in O} \sum_{\substack{r \in R_o: \\ r \ni (t,d)}} \bar{y}_r^o \tag{9}$$

by greedily constructing roster sequences for L given those for O. To be more precise, let $O_\ell \subseteq O$ denote the clusters contained in $\ell \in L$. For each $\ell \in L$, we pick the roster sequence r with smallest non-zero value $\bar{y}_r^o$, say v, over all clusters O_ℓ. This roster sequence is then combined with a roster sequence for each of the other clusters in O_ℓ, to obtain a roster sequence q for cluster ℓ. We set $\bar{z}_q^\ell = v$, reduce $\bar{y}_r^o$ for all involved roster sequences by v, and repeat the procedure until all roster sequences are assigned.

It follows that we can construct a solution $\bar{z}$ that satisfies (7). It remains to show that a solution constructed in this fashion is feasible with respect to the roster constraints. By combining (4) and (7), and doing some algebraic manipulations, it can be shown that $\bar{z}$ is feasible with respect to $P \setminus P_L$ (see Appendix A).

To show that $\bar{z}$ is feasible with respect to the roster constraints in P_L we make the following crucial observation: Since $P_K \supseteq P_L$ it must hold that $P_O \supseteq P_L$. Suppose that this is not true, then there must be a constraint $p \in P_L$ with non-zero coefficient f_{td}^p for multiple clusters in O. By definition of O, however, this would imply that $P_L \setminus P_K \neq \emptyset$, as O is the largest clustering contained in both K and L. This contradicts the assumption that $P_K \supseteq P_L$. Hence, if the constructed solution $\bar{y}$ is feasible with respect to P_O, then a solution $\bar{z}$ created by combining these roster sequences must be feasible with respect to P_L. The feasibility of $\bar{y}$ with respect to P_O, however, follows directly from the feasibility of $\bar{x}$, since $P_O \subseteq P_K$. This concludes the proof. □

It is important to note that Lemma 5.1 does not hold in the other direction. That is, given a solution $\bar{z}$ it is not always possible to construct a solution $\bar{x}$ satisfying (7). We are now able to prove the following theorem.

Theorem 5.1. *Let K and L be two clusterings such that $P_K \supseteq P_L$. Furthermore, let v_K denote the optimal value of the* LP *relaxation using clustering K, and define v_L similarly.*

Let $\bar{x}$ be an optimal solution corresponding to v_K. It holds that

$$v_K \geq v_L + \sum_{p \in P_K \setminus P_L} c_p \pi_p(\bar{x}),$$

where the non-negative coefficients $\pi_p(\bar{x})$ are given by

$$\pi_p(\bar{x}) = \sum_{k \in K} \sum_{s \in S_k} \bar{x}_s^k \left[\sum_{(t,d) \in s} f_{td}^p - b_p \right]^+ - \left[\sum_{k \in K} \sum_{s \in S_k} \bar{x}_s^k \left(\sum_{(t,d) \in s} f_{td}^p \right) - b_p \right]^+ .$$

Proof. Let $\bar{z}$ be a feasible solution for clustering L satisfying (7), obtained using the construction heuristic described in the proof of Lemma 5.1. Note that $\bar{z}$ is feasible for L and hence the objective value of $\bar{z}$ is an upper bound for v_L. Furthermore, note that, by the construction of $\bar{z}$, the cost incurred for the roster constraints P_L is identical for $\bar{x}$ and $\bar{z}$. As a consequence, the difference in objective value between $\bar{x}$ and $\bar{z}$ is exactly the penalty incurred by the roster constraints in $P_K \setminus P_L$. Hence, the difference in the penalty incurred by these constraints is a lower bound on $v_K - v_L$.

First, consider the solution $\bar{x}$. Recall that the constraint violations for each pattern $p \in P_K \setminus P_L$ are modeled implicitly in the roster sequence cost for clustering K. The penalty incurred from roster constraint $p \in P_K \setminus P_L$ is therefore given by

$$c_p \sum_{k \in K} \sum_{s \in S_k} \bar{x}_s^k \left[\sum_{(t,d) \in s} f_{td}^p - b_p \right]^+ .$$

Next, consider the solution $\bar{z}$. Note that for L the constraint violations for all $p \in P_K \setminus P_L$ are modeled explicitly using (4). Hence, the penalty incurred from roster constraint $p \in P_K \setminus P_L$ is given by

$$c_p \left[\sum_{\ell \in L} \sum_{q \in Q_\ell} \bar{z}_q^\ell \left(\sum_{(t,d) \in q} f_{td}^p \right) - b_p \right]^+ .$$

Using that

$$\sum_{\ell \in L} \sum_{q \in Q_\ell} \sum_{(t,d) \in q} f_{td}^p \bar{z}_q^\ell = \sum_{k \in K} \sum_{s \in S_k} \sum_{(t,d) \in s} f_{td}^p \bar{x}_s^k,$$

(see (10) in Appendix A), it follows that the difference in incurred penalty is given by

$$c_p \sum_{k \in K} \sum_{s \in S_k} \bar{x}_s^k \left[\sum_{(t,d) \in s} f_{td}^p - b_p \right]^+ - c_p \left[\sum_{k \in K} \sum_{s \in S_k} \bar{x}_s^k \left(\sum_{(t,d) \in s} f_{td}^p \right) - b_p \right]^+ .$$

The result now follows from summing over all $p \in P_K \setminus P_L$. □

The value $\pi_p(\bar{x})$ represents the error incurred from modeling pattern p explicitly (note that $\pi_p(\bar{x})$ is zero if $\bar{x}$ is integer), opposed to modeling it implicitly (i.e., correctly). Theorem 5.1 can be used as a guideline to pick the 'ideal' set of clusters. Note that Theorem 5.1 does not necessarily state that large clusters are beneficial, but rather that clusters with a (relative) large value of $\sum_{p \in P_K} c_p$ are beneficial.

6 Branch-and-Price

We develop an exact solution approach for the CCRP using Branch-and-Price. We assume the reader is familiar with the general idea behind Branch-and-Price (for detailed surveys on column generation and Branch-and-Price, see e.g., Desaulniers et al. [2006], Desrosiers and Lübbecke [2011]). In a Branch-and-Price approach the solution space is searched using Branch-and-Bound. In each node of the Branch-and-Bound tree the resulting LP-relaxation is solved using column generation. The problem is split into a restricted master problem, using only a subset of the columns, and a pricing problem, aimed at generating profitable columns. The master and pricing problem are solved iteratively, until the found solution to the master problem is provably optimal (i.e., no columns with negative reduced cost can be found).

The master problem is obtained from (1)–(6) by relaxing the integrality constraints on the x_s^k variables. We branch based on the assignments of the duties to the days. That is, we branch if some duty is assigned to multiple days in the basic schedule. Whenever multiple branching decisions exist, we branch on the assignment with the most fractional value. Note that whenever no branching decisions of this kind are present, each duty is assigned to exactly one day, and hence an integer solution is found. The aim of the Branch-and-Price algorithm is to find provably optimal solutions. The node selection strategy in the Branch-and-Bound method is therefore based on the lowest bound, to improve the best-known (i.e., highest) lower bound as quickly as possible.

The reduced cost γ_s^k of a roster sequence $s \in S_k$, for a given cluster $k \in K$ can be expressed as follows. Let μ_k denote the dual variables corresponding to (2), ϕ_d those corresponding to (3), and θ_p those corresponding to (4). The reduced cost γ_s^k can now be expressed as

$$\gamma_s^k = c_s^k - \mu_k - \sum_{d \in D} h_{ds}^k \phi_d - \sum_{p \in P \setminus P_K} \sum_{(t,d) \in s} f_{td}^p \theta_p.$$

Note that by defining the aggregated dual variable $\lambda_{td} = \phi_d + \sum_{p \in P \setminus P_K} f_{td}^p \theta_p$, the reduced cost γ_s^k can be equivalently written as

$$\gamma_s^k = c_s^k - \mu_k - \sum_{(t,d) \in s} \lambda_{td}.$$

The pricing problem for each $k \in K$ can be modeled as resource constrained shortest path problem (RCSPP) with surplus variables on a dedicated graph (see e.g., Irnich and Desaulniers [2005]). For each cluster $k \in K$, we construct a directed layered graph in which each layer corresponds to a day in k, and each vertex corresponds to an assignment (t, d) of a duty to a day in k. Initially, there is an arc between each two assignments in consecutive layers (an arc can be removed when incorporating the roster constraints). Whenever the assignment of a duty to a day is predetermined due to the basic schedule (e.g., a day must be a day off), the vertex corresponding to this layer can be directly incorporated in the arc set. Figure 7 shows an example of a pricing graph for the cluster depicted in Figure 4. In this particular instance, the basic schedule specifies that both Wednesday and Thursday should be a day off, hence the arcs directly connect the possible assignments for Tuesday and Friday.

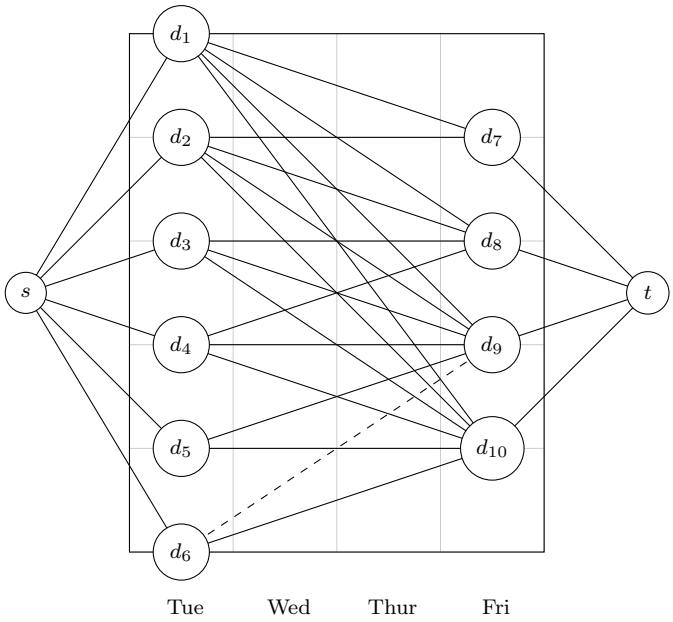


Figure 7: Example Pricing Graph.

The dual costs λ_{td} of the roster sequence are readily incorporated in the arc costs. For the primal cost the violation of each of the constraints in P_k has to be modeled. We distinguish three different cases.

- The first case considers constraints which have non-zero coefficient for only one layer in the graph (e.g., certain assignments are not allowed). These constraints are readily incorporated in the vertex set (if an assignment is not allowed), or in the arc costs (if an assignment is allowed, but penalized). A possible example of such a constraint would be a relatively late duty scheduled just before a rest day: although possibly allowed, such an assignment is not desirable.
- The second case considers constraints that have non-zero coefficients only for two *consecutive* layers in the graph. These constraints are readily incorporated in the

arc set, either by removing an arc, or by adding a penalty to the arc cost. An example of this is shown in Figure 7, where it is not possible to schedule duty d_6 before d_8 (hence the arc is removed), and, furthermore, scheduling duty d_6 before d_9 is allowed, albeit against a penalty (indicated by the dashed arc). Rest time constraints are a typical example of constraints belonging to this category.

- The third and final case considers all other constraints. These constraints can be modeled by defining a resource with consumption f^p_{td} at each vertex, and threshold value b_p. If the constraint is soft, a surplus variable is added to model the violation δ_p. Workload constraints are a typical example of this category.

We solve the RCSPP using an enumerative depth first search approach, where the cost of each path is estimated using Lagrangian distance labels. These labels give a lower bound on the path cost, and have been used for similar problems (see. e.g., Beasley and Christofides [1989], Dumitrescu and Boland [2003], Grötschel et al. [2003], Breugem et al. [2017]). The lower bounds are used to prune the search tree (i.e., part of the tree is pruned whenever the lower bound of a path exceeds the best-known upper bound).

7 Assignment Fixing Heuristic

Initial experiments showed that, even if the linear relaxation is strong, the Branch-and-Price approach fails to solve certain instances in a reasonable amount of time (due to e.g., their size or structure). We therefore propose a heuristic solution method to cope with these more difficult instances, which we will refer to as the Assignment Fixing Heuristic (AFH). In this heuristic we aim at exploiting the information obtained from the linear relaxation. That is, we assume that the optimal solution of the linear relaxation gives information regarding the structure of the optimal integer solution. Note that this heavily depends on the strength of the relaxed formulation. We then restrict the number of possible roster sequences per cluster based on the optimal solution of the relaxation.

The restricted version of the problem is obtained as follows. Let $\bar{x}$ be an optimal solution to the relaxed problem. The set of possible roster sequences for $k \in K$ is initialized with all roster sequences $s \in S_k$ with non-zero coefficient $\bar{x}^k_s$. We then determine, for each cluster $k \in K$ and each day $t \in k$ in the cluster, the η duties that are assigned the largest non-zero coefficients in $\bar{x}^k_s$. That is, we determine the η duties with the largest score α_{td}, defined as

$$\alpha_{td} = \sum_{\substack{s \in S_k: \\ s \ni (t,d)}} \bar{x}^k_s.$$

Denote the resulting set of duties by D_t. If less than η duties have a positive score, only the duties with positive score are selected. We consider a restricted version of the problem, where we only consider the roster sequences for k consisting of duties in D_t.

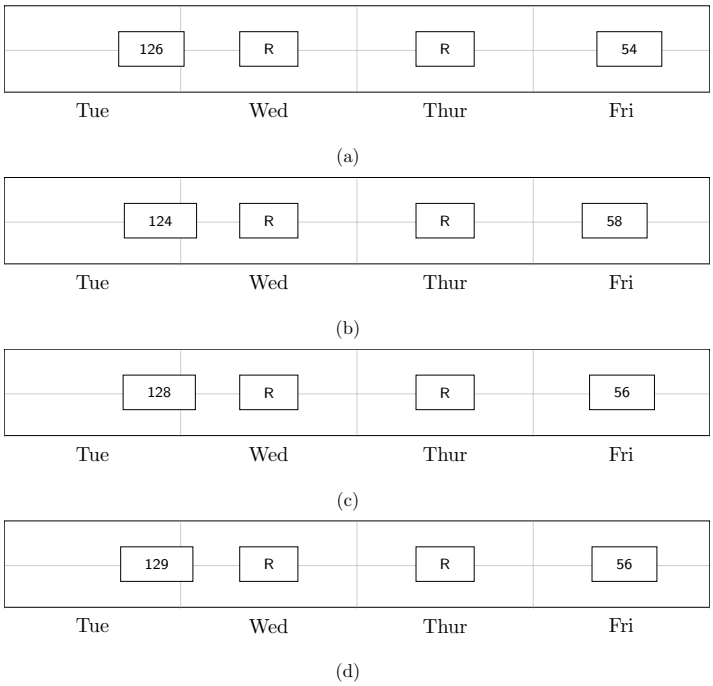


Figure 8: Example heuristic.

To illustrate the selection of duties, consider the four roster sequences depicted in Figure 8. Suppose the four roster sequences are assigned the values 0.25, 0.35, 0.2 and 0.2, respectively, and that $\eta = 2$. For Tuesday, we have $\alpha_{\text{Tue},126} = 0.25$, $\alpha_{\text{Tue},124} = 0.35$, and $\alpha_{\text{Tue},128} = \alpha_{\text{Tue},129} = 0.2$. For Wednesday, we have $\alpha_{\text{Wed},54} = 0.25$, $\alpha_{\text{Wed},58} = 0.35$, and $\alpha_{\text{Wed},56} = 0.4$ (note that duty 56 appears in both the third and the fourth roster sequence). Hence, in this case, all possible roster sequences are generated using duties 126 and 124 for Tuesday and duties 58 and 56 for Friday. For reasonably sized η the resulting problem can be readily solved using a commercial solver. In our experiments we observed that often much less than η duties are selected for each day in the cluster. That is, when the root node solution is close to integer, only a small number of duties per day will have a non-zero coefficient.

8 Computational Experiments

In this section we discuss the computational results. We first discuss the experimental set-up in Section 8.1. That is, we discuss the roster constraints that are taken into account, and the different instances considered. We then turn to the computational results in Section 8.2.

8.1 Experimental Set-Up

We apply our solution approach to different instances based on data from NS. For each instance, the basic schedule specifies the days off. Furthermore, for each duty that is to be scheduled a type is given. The considered types are Early, Late, and Night. The type of each duty is based on the start time of the duty. The following roster constraints are taking into account.

- *Rest Time.* After completing a duty it is required that an employee has a certain minimum time to rest. After a night duty this rest time should be at least 14 hours, otherwise it should be at least 12 hours. Furthermore, we penalize rest times shorter than 16 hours with a penalty of 30.
- *Rest Day.* When rest days are scheduled in the roster, the length of the rest period has to be sufficient. This implies that there is a minimal time enforced between duties scheduled before and after the rest days. The enforced rest time is 6 hours plus 24 hours for each rest day.
- *Red Weekend.* At least once every three rows of the roster there should be a weekend which has a consecutive period of 60 hours off. These so-called red weekends can be determined given the basic schedule. The 60 hour rest period can then be enforced using the roster constraints.
- *Workload.* The total workload in a row is not allowed to exceed 45 hours. Here, the workload of a duty is the difference between the start and end time (i.e., including the meal break).
- *Variation.* The variation constraints assure that the different attributes of work (e.g., duty length, percentage double decker work) are divided equally over the rows. These constraints penalize a positive deviation from the average (measured over all duties) for each row in the roster. In total we consider 10 different variation constraints.

We consider a total of 10 different instances: four 'small' instances of 12 employees and roughly 50 duties, four 'medium' instances of 24 employees and roughly 100 duties, and two 'large' instances of about 50 employees and 200 duties. Each of the instances is obtained by combining multiple roster groups as operated at NS. The properties of the instances are summarized in Table 1.

	Size	E	L	N	Total
1	12	23	11	15	49
2	12	21	12	16	49
3	12	49	0	1	50
4	12	49	0	1	50
5	24	21	36	35	92
6	24	23	35	37	95
7	26	101	0	2	103
8	24	97	0	2	99
9	54	118	47	52	217
10	50	198	0	4	202

Table 1: Characteristics of the instances. For each instance the size (i.e., the number of employees) is specified, along with the number of Early, Late, and Night duties, and the total number of duties.

The instances can be categorized into one of two categories. The instances 1, 2, 5, 6, and 9 represent instances in which all three duty types have to be scheduled. For the other instances the duties consist almost exclusively of early duties. The former category of instances provide more structure compared to the latter ones, since (i) less roster sequences are possible (as the duties are divided over different types), and (ii) the rest time and rest day constraints are expected to be more important for these instances (i.e., if all duties start early, the chance of having a rest time violation is small). The second category is therefore expected to be more difficult to solve if the formulation is not chosen carefully.

8.2 Computational Results

In this section the computational results are discussed in detail. We compare the performance of different clusterings. We then evaluate the performance of the AFH, and we conclude by comparing both the Branch-and-Price approach and the AFH with a state-of-the-art commercial solver. All experiments are done on a computer with a 1.6 GHz Intel Core i5 processor. We use the LP solver embedded in CPLEX 12.7.1 (simply referred to as CPLEX from hereon) to solve the master problems.

8.2.1 Comparison of Formulations

To illustrate the importance of a correct clustering (for the given constraints), we solve the smaller instances for different clusterings using the Branch-and-Price approach. We consider clusterings were each cluster contains a single day, three days, six days, and seven days (i.e., a cluster per row). We denote these clusterings by C_1, C_3, C_6, and

C_7, respectively. Each clustering leads to a different formulation. In particular, the clustering C_1 results in the assignment formulation proposed in Hartog et al. [2009], and the clustering C_7 leads to a formulation similar to Breugem et al. [2017]. We compare the different clusterings on the first four instances (i.e., the smaller instances), as these instances already clearly highlight the relative performance differences of the clusterings.

The results for each of the four smaller instances are shown in Table 2. For each instance, and each clustering, Table 2 shows the bound obtained from the root node relaxation, the best lower bound achieved, the total computation time, and the percentage of roster constraints that could be modeled implicitly. Each run is limited to at most one hour of computation time. For clustering C_1 we generate all roster sequences a priori (as each roster sequence contains only one day). Recall that the Branch-and-Price approach focuses on optimality, i.e., improving the lower bound. This implies that generally either the problem is solved to optimality within the time limit, or no feasible solution is found. Underlined entries in Table 2 denote optimal solutions.

	Clustering	Root Bound	Lower Bound	Time (min)	Imp. Con. (%)
1	C_1	558.0	<u>609.7</u>	1.8	0
	C_3	569.2	<u>609.7</u>	2.4	29
	C_6	581.2	<u>609.7</u>	0.9	49
	C_7	609.7	<u>609.7</u>	0.0	98
2	C_1	654.5	<u>713.8</u>	0.5	0
	C_3	661.0	<u>713.8</u>	0.2	20
	C_6	705.5	<u>713.8</u>	0.1	59
	C_7	713.8	<u>713.8</u>	0.0	95
3	C_1	192.4	210.6	>60	0
	C_3	192.5	208.0	>60	32
	C_6	206.4	224.3	>60	49
	C_7	280.0	<u>283.5</u>	0.1	94
4	C_1	274.0	289.8	>60	0
	C_3	289.5	305.4	>60	38
	C_6	313.5	330.1	>60	59
	C_7	370.2	<u>373.9</u>	0.2	97

Table 2: Results for the small instances using the Branch-and-Price approach for four different clusterings. Underlined entries denote optimal solutions.

The results in Table 2 are in line with Theorem 5.1. That is, there is a clear relation between the percentage of implicit constraints and the bound obtained from the linear relaxation. The benefit of a suitable clustering is most apparent for instances 3 and 4 (i.e., two instances considering only early duties). For these instances the main challenge is to capture the cost incurred from the variation constraints, which only clustering C_7 is able to do. As a result, these instances are solved quickly using C_7, whereas with the

other clusterings the instances cannot be solved within an hour. We conclude by noting that the integrality gaps for C_7 are small, which is a good motivation for the AFH.

8.2.2 Assignment Fixing Heuristic

Next, we consider the performance of the AFH using clustering C_7. Table 3 shows the results for each of the ten instances. The heuristic is run for different values of η (i.e., the maximum number of duties we select per day). For none of the instances more than 10 duties per day had a non-zero coefficient. For each run, Table 3 shows the objective value of the best found solution, the gap with respect to the root bound, the total computation time (including the time for solving the linear relaxation), and the number of columns generated using the heuristic. Recall that the heuristic depends on the solution obtained when solving the linear relaxation. The experiments showed that for some instances the linear relaxation had multiple optimal solutions. In this case we average the results over a total of 10 runs. For these instances, Table 3 shows the standard deviation of each value within brackets.

	η	Best Solution		Gap (%)		Time (min)		Nr. Columns	
	3	609.7	-	0.0	-	0.0	-	12.0	-
1	5	609.7	-	0.0	-	0.0	-	12.0	-
	10	609.7	-	0.0	-	0.0	-	12.0	-
	3	713.8	-	0.0	-	0.0	-	12.0	-
2	5	713.8	-	0.0	-	0.0	-	12.0	-
	10	713.8	-	0.0	-	0.0	-	12.0	-
	3	287.3	-	2.6	-	0.0	-	585.0	-
3	5	287.3	-	2.6	-	0.0	-	691.0	-
	10	287.3	-	2.6	-	0.0	-	691.0	-
	3	380.0	-	2.6	-	0.0	-	1125.0	-
4	5	377.0	-	1.8	-	0.0	-	2554.0	-
	10	374.2	-	1.1	-	0.0	-	3188.0	-
	3	959.2	-	3.5	-	0.0	-	378.0	-
5	5	959.2	-	3.5	-	0.0	-	378.0	-
	10	959.2	-	3.5	-	0.0	-	378.0	-
	3	1034.4	-	3.3	-	0.0	-	266.0	-
6	5	1034.4	-	3.3	-	0.0	-	266.0	-
	10	1034.4	-	3.3	-	0.0	-	266.0	-
	3	505.7	(9.9)	11.4	(1.7)	0.3	(0.0)	1904.4	(35.5)
7	5	479.6	(4.1)	6.6	(0.8)	0.4	(0.0)	8382.3	(250.3)
	10	478.0	(4.9)	6.3	(1.0)	0.5	(0.1)	12782.4	(352.0)
	3	574.7	(5.1)	8.9	(0.8)	0.3	(0.0)	2837.5	(53.3)
8	5	549.6	(3.8)	4.8	(0.7)	0.5	(0.1)	15254.7	901.6
	10	546.7	(2.9)	4.3	(0.5)	0.6	(0.1)	26302.6	(2073.7)
	3	1494.6	(3.9)	4.5	(0.3)	1.1	(0.3)	4186.9	(143.5)
9	5	1474.9	(2.0)	3.3	(0.1)	1.7	(0.5)	15471.9	(899.5)
	10	1474.0	(1.7)	3.2	(0.1)	1.9	(0.4)	21460.5	(2537.1)
	3	720.1	(7.9)	16.9	(0.9)	12.2	(1.4)	5204.2	(55.0)
10	5	672.7	(5.3)	11.1	(0.7)	23.4	(9.5)	31811.4	(994.3)
	10	665.1	(4.7)	10.1	(0.7)	26.5	(5.1)	62212.0	(4362.5)

Table 3: Performance of the AFH for the different instances. For certain instances, multiple solutions to the linear relaxation exist. In this case the results are averaged over 10 runs, and the standard deviation is shown between brackets.

Table 3 shows that the heuristic performs well for all instances. The found gap is reasonably small (from a practical point of view), even for instance 10, which can be regarded as extremely difficult to solve using current state-of-the-art solution methods. As expected, Table 3 shows that the performance of the heuristic improves as η increases. The largest gain is generally observed when increasing η from 3 to 5. For $\eta = 3$, duties with a relatively high coefficient are not added. Note also that in this case the number of columns increases drastically. When increasing η from 5 to 10, the improvement is less. We note,

however, that for these instances all variants could be solved quickly, hence picking the largest value for η is the best choice.

8.2.3 Comparison Commercial Solver

Finally, we compare the performance of the Branch-and-Price approach and the AFH with CPLEX. Here the Branch-and-Price approach and the heuristic are applied to the model resulting from clustering C_7, and CPLEX is used for the model obtained from clustering C_1. Furthermore, the formulation resulting from C_7 is tightened by adding the valid inequalities proposed in Breugem et al. [2017].

The results for the different instances are shown in Table 4. For each run Table 4 shows the root bound resulting from the used clustering, the best obtained lower bound for each method, the objective value of the best found solution, the gap with respect to the best-known lower bound, and the overall computation time (again limited to at most one hour). Note that the AFH does not improve the root bound, hence the lower bound for the AFH is omitted in Table 4. For the sake of clarity, we only display the average value for the AFH (as shown in Table 3).

Table 4 shows that the Branch-and-Price approach outperforms CPLEX for the smaller instances. Both approaches quickly solve instances 1 and 2, but the Branch-and-Price approach is much more efficient for the difficult instances 3 and 4. This can be attributed to the strong linear relaxation obtained using clustering C_7. Both instance 5 and 6 could be solved to optimality within the set time limit using the Branch-and-Price approach and CPLEX. We see that CPLEX outperforms the Branch-and-Price approach on the fifth instance, and both methods perform roughly equally on the sixth instance.

The benefit of the AFH is clearly visible when we consider the final four instances, which are considered the most difficult to solve. For these instances the Branch-and-Price approach stagnates, and CPLEX is not able to obtain a proper gap within the time limit. The heuristic, on the other hand, quickly obtains a good solution for each of the instances. Note that the poor performance of CPLEX is not only due to the weak lower bound: For each of the instances the heuristic finds a solution with a significantly lower objective value. Again, we note that the good performance of the heuristic on these instances is due to the strong linear relaxation obtained using clustering C_7.

Summarizing, the column generation approach allows to efficiently obtain good solutions to each of the ten instances. The smaller instances can be solved to optimality using the Branch-and-Price approach, and for the larger instances a good feasible solution can be obtained using the AFH.

		Root Bound	Lower Bound	Best Solution	Gap (%)	Time (min)
1	CPLEX	558.0	609.7	609.7	0.0	0.0
	B&P	609.7	609.7	609.7	0.0	0.0
	AFH	609.7	-	609.7	0.0	0.0
2	CPLEX	654.5	713.8	713.8	0.0	0.0
	B&P	713.8	713.8	713.8	0.0	0.0
	AFH	713.8	-	713.8	0.0	0.0
3	CPLEX	192.4	283.5	283.5	0.0	31.7
	B&P	280.0	283.5	283.5	0.0	0.1
	AFH	280.0	-	287.3	1.3	0.0
4	CPLEX	274.0	373.9	373.9	0.0	10.9
	B&P	370.2	373.9	373.9	0.0	0.2
	AFH	370.2	-	374.2	0.1	0.0
5	CPLEX	671.4	950.3	950.3	0.0	1.4
	B&P	925.8	950.3	950.3	0.0	13.5
	AFH	925.8	-	959.2	0.9	0.0
6	CPLEX	618.4	1008.5	1008.5	0.0	3.7
	B&P	1000.7	1008.5	1008.5	0.0	3.0
	AFH	1000.7	-	1034.4	2.5	0.0
7	CPLEX	181.0	321.1	518.7	13.2	>60
	B&P	447.8	450.3	-	-	>60
	AFH	447.8	-	478.0	5.8	0.5
8	CPLEX	250.3	399.7	572.2	8.2	>60
	B&P	523.4	525.3	-	-	>60
	AFH	523.4	-	546.7	3.9	0.6
9	CPLEX	917.2	1331.2	1569.2	8.9	>60
	B&P	1426.8	1429.2	-	-	>60
	AFH	1426.8	-	1474.0	3.0	1.9
10	CPLEX	221.4	331.6	1051.3	43.1	>60
	B&P	598.2	598.2	-	-	>60
	AFH	598.2	-	665.1	10.1	26.5

Table 4: Comparison of the Branch-and-Price approach (B&P), the assignment fixing heuristic (AFH), and CPLEX for the different instances.

9 Conclusion

In this paper, we analyzed a family of formulations for the Cyclic Crew Rostering problem (CCRP). This family of formulations is motivated by the poor performance of traditional assignment models for difficult instances. Each formulation has a different structure, which implies that a suitable variant can be picked for a given problem instance. We derived analytical results regarding the relative strength of the different formulations,

which can be used as a guideline to pick a suitable formulation for a given problem instance.

We developed a column generation approach, where columns are generated using a resource constrained shortest path problem (RCSPP) with surplus variables. Based on this approach, we developed an exact Branch-and-Price method. Furthermore, we proposed a heuristic method, which reduces the number of possible duties per day based on the solution of the linear relaxation. By limiting the number of possible duties per day, the number of columns reduces greatly, leading to a tractable problem which can be solved using a commercial solver.

We applied both the Branch-and-Price approach and the heuristic to practical instances from NS. Our experiments showed the importance of picking a suitable formulation for a given problem instance. In particular, we show that a suitable formulation is better able to capture the penalty incurred from the roster constraints. As a result, the column generation approach outperforms a commercial solver using the traditional assignment model. The experiments also showed that the performance of the different methods depends on the structure of the problem instances, which is in line with the derived analytical results.

To further improve the proposed solution methods it would be interesting to embed the heuristic method in, e.g., variable neighborhood search. This would make the heuristic less dependent on the root node solution, as was showed to be the case in our experiments.

Appendix A

To complete the proof of Lemma 5.1 it remains to show that $\bar{z}$ is feasible for the roster constraints in $P \setminus P_L$. Recall that u_p is the upper bound of the violation interval Δ_p. Using (7) we have

$$\sum_{\ell \in L} \sum_{q \in Q_\ell} \sum_{(t,d) \in q} f_{td}^p \bar{z}_q^\ell = \sum_{(t,d) \in \Omega} \sum_{\ell \in L} \sum_{\substack{q \in Q_\ell: \\ q \ni (t,d)}} f_{td}^p \bar{z}_q^\ell \tag{10a}$$

$$= \sum_{(t,d) \in \Omega} \sum_{k \in K} \sum_{\substack{s \in S_k: \\ s \ni (t,d)}} f_{td}^p \bar{x}_s^k \tag{10b}$$

$$= \sum_{k \in K} \sum_{s \in S_k} \sum_{(t,d) \in s} f_{td}^p \bar{x}_s^k. \tag{10c}$$

Hence, for $p \in P \setminus P_K$, it follows that

$$\sum_{\ell \in L} \sum_{q \in Q_\ell} \sum_{(t,d) \in q} f_{td}^p \bar{z}_q^\ell - b_p = \sum_{k \in K} \sum_{s \in S_k} \sum_{(t,d) \in s} f_{td}^p \bar{x}_s^k - b_p \tag{11a}$$

$$\leq u_p, \tag{11b}$$

where (11b) follows from the feasibility of $\bar{x}$. Next, consider some $p \in P_K \setminus P_L$. Since $p \in P_K$, there is a cluster $k' \in K$ such that the coefficients f_{td}^p are non-zero only for this cluster. It follows that

$$\sum_{\ell \in L} \sum_{q \in Q_\ell} \sum_{(t,d) \in q} f_{td}^p \bar{z}_q^\ell - b_p = \sum_{k \in K} \sum_{s \in S_k} \sum_{(t,d) \in s} f_{td}^p \bar{x}_s^k - b_p \tag{12a}$$

$$= \sum_{s \in S_{k'}} \sum_{(t,d) \in s} f_{td}^p \bar{x}_s^{k'} - b_p \tag{12b}$$

$$= \sum_{s \in S_{k'}} \bar{x}_s^{k'} \sum_{(t,d) \in s} f_{td}^p - b_p \tag{12c}$$

$$= \sum_{s \in S_{k'}} \bar{x}_s^{k'} \left(\sum_{(t,d) \in s} f_{td}^p - b_p \right) \tag{12d}$$

$$\leq \sum_{s \in S_{k'}} \bar{x}_s^{k'} u_p \tag{12e}$$

$$= u_p, \tag{12f}$$

where (12d) and (12f) follow from (2), and (12e) follows from the feasibility of the roster sequence s. It follows that $\bar{z}$ is feasible for all $p \in P_K \setminus P_L$, and thus for all $p \in P \setminus P_L$.

References

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

J. E. Beasley and N. Christofides. An algorithm for the resource constrained shortest path problem. *Networks*, 19(4):379–394, 1989.

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.

T. Breugem, T. Dollevoet, and D. Huisman. Is equality always desirable? Analyzing the trade-off between fairness and attractiveness in crew rostering. *Econometric Institute Research Papers*, (EI2017-30), 2017. *Under first revision at Management Science.*

A. Caprara, M. Fischetti, P. Toth, D. Vigo, and P. L. Guida. Algorithms for railway crew management. *Mathematical Programming*, 79(1-3):125–141, 1997.

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.

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

J. Desrosiers and M. E. Lübbecke. Branch-price-and-cut algorithms. *Wiley Encyclopedia of Operations Research and Management Science*, 2011.

I. Dumitrescu and N. Boland. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. *Networks*, 42(3):135–153, 2003.

R. Freling, R. M. Lentink, and A. P. M. Wagelmans. A decision support system for crew planning in passenger transportation using a flexible branch-and-price algorithm. *Annals of Operations Research*, 127(1-4):203–222, 2004.

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.

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.

S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In *Column Generation*, pages 33–65. Springer, 2005.

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

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.

J. Van den Bergh, J. Beliën, P. De Bruecker, E. Demeulemeester, and L. De Boeck. Personnel scheduling: A literature review. *European Journal of Operational Research*, 226(3):367–385, 2013.