Modeling and Solving a Crew Rostering Problem with Constraint Logic Programming and Integer Programming

Tallys H. Yunes *Arnaldo V. Moura*
Cid C. de Souza

Relatório Técnico IC–00-04

Março de 2000

Modeling and Solving a Crew Rostering Problem with Constraint Logic Programming and Integer Programming

Tallys H. Yunes* Arnaldo V. Moura Cid C. de Souza†

Abstract

This article describes the crew rostering problem stemming from the operation of a Brazilian bus company that serves a major urban area in the city of Belo Horizonte. The problem is solved by means of Integer Programming (IP) and Constraint Logic Programming (CLP) approaches, whose models are discussed in detail. Lower bounds obtained with a Linear Programming relaxation of the problem are used in order to evaluate the quality of the solutions found. We also present a hybrid column generation approach for the problem, combining IP and CLP over a set partitioning formulation. Experiments are conducted upon real data sets and computational results are evaluated, comparing the performance of these three solution methods.

1 Introduction

The overall *crew management* problem concerns the allocation of trips to crews within a certain planning horizon. In addition, it is necessary to respect a specific set of operational constraints and minimize a certain objective function. Being a fairly complicated problem as a whole, it is usually divided in two smaller subproblems: *crew scheduling* and *crew rostering* [4]. In the crew scheduling subproblem, the aim is to partition the initial set of trips into a minimal set of *feasible duties*. Each such duty is an ordered sequence of trips which is to be performed by the same crew and that satisfies a subset of the original problem constraints: those related to the sequencing of trips during a workday. The crew rostering subproblem takes as input the duties output by the crew scheduling phase and builds a roster spanning a longer period, e.g. months or years.

This article describes the crew rostering problem stemming from the operation of a Brazilian bus company that serves a major urban area in the city of Belo Horizonte. The problem is solved by means of Integer Programming (IP) and Constraint Logic Programming (CLP) approaches, whose models are discussed in detail. Lower bounds obtained with a Linear Programming relaxation of the problem are used in order to evaluate the quality of the solutions found. We also present a hybrid column generation approach for the problem, combining IP and CLP. Experiments are conducted upon real data sets and computational results are evaluated, comparing the performance of these three solution methods.

*Supported by FAPESP grant 98/05999-4, and CAPES.

†Supported by FINEP (ProNEx 107/97), and CNPq (300883/94-3).

Some quite specific union regulations and operational constraints make this problem fairly distinct from some other known crew rostering problems found in the literature [3, 5]. In general, it is sufficient to construct one initial roster consisting of a feasible sequencing of the duties that spans the least possible number of days. The complete roster is then built by just assigning shifted versions of that sequence of duties to each crew so as to have every duty performed in each day in the planning horizon. In other common cases, the main concern is to balance the workload among the crews involved [2, 6, 7]. Although we also look for a roster with relatively balanced workloads, these approaches will not in general find the best solution for our purposes. We are not interested in minimizing the number of days needed to execute the roster, since the length of the planning horizon is fixed in advance. Our objective is to use the minimum number of crews when constructing the roster for the given period. Another difficulty comes from the fact that some constraints behave differently for each crew, depending on the amount of work assigned to it in the previous month. Moreover, different crews have different needs for days off, imposed by personal requirements.

The text is organized as follows. Section 2 gives a detailed description of the crew rostering problem under consideration. Section 3 explains the format of the input data sets used in our experiments. In Sect. 4, we present an Integer Programming formulation of the problem, together with some computational results. A pure Constraint Logic Programming model for the problem is described in Sect. 5, where some experiments are also conducted to evaluate its performance. As one additional attempt to solve the problem, the results achieved with a hybrid column generation approach appear in Sect. 6. All computation times presented in Sects. 4 to 6 are given in CPU seconds of a Pentium II 350 MHz. Finally, we draw the main conclusions in Sect. 7.

2 The Crew Rostering Problem

The duties obtained as output from the solution of the crew scheduling phase[1] must be assigned to crews day after day, throughout an entire planning horizon. This sequencing has to obey a set of constraints that differs from the constraints which are relevant to the crew scheduling problem. This set includes, for example, the need for days off, with a certain periodicity, and a minimum rest time between consecutive workdays.

2.1 Input Data

The set of duties to be performed on weekdays is different from the set of duties to be performed on weekends or holidays, due to fluctuations on customer demand. Therefore, the crew scheduling problem gives as input for the rostering problem a number of distinct sets of duties.

The planning horizon we are interested in spans one complete month. It is important to take into account as input data many features of the month under consideration, such as: the total number of days, which days are holidays and which day of the week is the first day

[1] For more specific information on the scheduling subproblem for this case, see [8].

of the month (the remaining weekdays can be easily figured out from this information). The differences in the number of working days from one month to the next one lead to variations on the number of crews actually working in each month. Consequently, some rules must be observed in order to select the crews that are going to be effectively used. If, say, in month m 40 crews were needed, and in month $m+1$ only 38 will be necessary, how to select the 2 crews that are going to be left out? Furthermore, suppose that, after eliminating those crews that cannot work on the current month for some reason, the company has 50 crews available. Even if the number of crews remains the same, e.g. 40, from one month to the next one, it is important to evenly distribute the work among them. This balance can be obtained considering the number of days each crew has worked since the beginning of the year, for example, or with the aid of another kind of ranking function for the crews. Finally, since some constraints refer to a time window that spans more than one month (see Sect. 2.2) some attributes, for each crew, have to be carried over between successive months.

The input data needed to build the roster for month m is the following:

- The sets of duties D_{wk}, D_{sa}, D_{su} and D_{ho} which have to be performed on weekdays, Saturdays, Sundays and on holidays, respectively;
- The number of days, d, in month m;
- The occurrence of holidays in month m;
- The day of the week corresponding to the first day in month m;
- The whole set of crews, C, employed by the company;
- For each crew $i \in C$:
  - The set of days, OFF_i, in which i is off duty (e.g. vacations, sickness), excluding its ordinary weekly rests;
  - The number of days between the last Sunday i was off duty and the first day of month m (ls_i);
  - A binary flag, wr_i, that is equal to 1 if and only if i had a weekly rest in the last week of month $m-1$;
  - A binary flag, sl_i, that is equal to 1 if and only if i performed a split-shift duty during the last week of month $m-1$;
  - The difference, in minutes, between the last minute i was working in month $m-1$ and the first minute of the first day of month m (lw_i);
- For each duty $k \in D_{\text{wk}} \cup D_{\text{sa}} \cup D_{\text{su}} \cup D_{\text{ho}}$:
  - The start and end times of k (ts_k and te_k, respectively), given in minutes after midnight;
  - A binary flag, ss_k, that equals 1 if and only if k is a split-shift duty;

Table 1: Description of the instances for the experiments

			# Duties			
Name	#Crews	#Days	Week	Sat	Sun	Holy
string	c	d (h)	$ss_{\mathrm{wk}}/tt_{\mathrm{wk}}$	$ss_{\mathrm{sa}}/tt_{\mathrm{sa}}$	$ss_{\mathrm{su}}/tt_{\mathrm{su}}$	$ss_{\mathrm{ho}}/tt_{\mathrm{ho}}$

- A list of all crews in C sorted according to a certain ranking function. This ordering will be used to assign priorities to the crews when identifying the subset of C that is going to work in month m.

2.2 Problem Constraints

The constraints associated to the sequencing of the duties are:

(a) The minimum rest time between consecutive workdays is 11 hours;

(b) Every employee must have at least one day off per week. Moreover, for every time window spanning 7 weeks, at least one of these days off must be on a Sunday;

(c) When an employee performs one or more split-shift duties during a week, his day off in that week must be on Sunday;

(d) In every 24-hour period starting at midnight, within the whole planning horizon, each crew can start to work on at most one duty.

2.3 Objectives

For each month, we are looking for the cheapest solution in terms of the number of crews needed to perform all the duties requested. Additionally, it is desirable to have balanced workloads among all the crews involved, but the models we present in this article are not concerned with this issue yet.

3 The Input Data Sets

Before describing the IP and CLP models for the rostering problem, it is important to understand the format of the instances used in the computational experiments. These instances correspond to actual schedules constructed by a crew scheduling algorithm executed over real world data from the same bus company mentioned in Sect. 1 [8]. Using the duties built during the crew scheduling phase, we have constructed a set of instances ranging from small ones up to large-sized ones, typically encountered by the management personnel in the bus company. The main features of these instances appear in Table 1.
The *Name* is just a string identifying the instance. The number of crews available for the roster, c, appears under the heading *#Crews*. The column *#Days* shows the number of days in the planning horizon in the format d (h), where d is the total number of days and h

indicates how many of those d days are holidays. The next four columns show the number of duties that must be performed in each kind of the possible working days: weekdays, Saturdays, Sundays and holidays, respectively. The format used is ss/tt, where tt is the total number of duties and ss represents how many of the tt duties are split-shift duties. To begin with, we set the following parameters, for every crew i: $OFF_i = \emptyset$, $ls_i = 1$, $wr_i = 1$, $sl_i = 0$ and $lw_i = 660$. This is the same as ignoring any information from the previous month when constructing the roster for the current month.

4 An Integer Programming Approach

Let n be the total number of crews available and let d be the number of days in the current month m. Moreover, let p, q, r and s be the numbers of duties to be performed on weekdays, Saturdays, Sundays and holidays, respectively (i.e. $|D_{\mathrm{wk}}| = p$, $|D_{\mathrm{sa}}| = q$, $|D_{\mathrm{su}}| = r$ and $|D_{\mathrm{ho}}| = s$).

The IP formulation of the rostering problem is based on x_{ijk} binary variables which are equal to 1 if and only if crew i performs duty k on day j. If j is a weekday, k belongs to $\{0, 1, \ldots, p\}$. Analogously, if j is a Saturday, Sunday or holiday, k ranges over $\{0, p+1, \ldots, p+q\}$, $\{0, p+q+1, \ldots, p+q+r\}$ or $\{0, p+q+r+1, \ldots, p+q+r+s\}$, respectively. The duty numbered 0 is a special duty indicating *idleness*. Thus, if $x_{ij0} = 1$ it means that crew i is not working on day j. For modeling purposes, we set $ts_0 = +\infty$, $te_0 = 0$ and $ss_0 = 0$.

Given a day j in m, K_j represents its set of duty indexes, except for the duty 0. For instance, if j is a Saturday then $K_j = \{p+1, \ldots, p+q\}$.

4.1 The Model

The main objective is to minimize the number of crews working during the present month. This is equivalent to maximizing the number of crews which are idle during the whole month. Let us define new variables $y_i \in R^+$, for all $i \in \{1, \ldots, n\}$, which are equal to 1 if $x_{ij0} = 1$, for all $j \in \{1, \ldots, d\}$, and are equal to 0 otherwise. To achieve this behavior for the y_i variables, it is necessary to relate them to the x_{ij0} variables through the following constraints

$$y_i \leq x_{ij0}, \quad \forall\, i,\ \forall\, j \quad . \tag{1}$$

The objective function can then be written as $\max \sum_{i=1}^{n} y_i$. Equations (1) combined with the objective function force a y_i variable to be equal to 1 if and only if crew i is idle during the entire month.

The occurrence of days on which the crews are known to be off duty (e.g. previously assigned holiday periods) is satisfied by setting

$$x_{ij0} \quad = \quad 1, \quad \forall\, i,\ \forall\, j \in OFF_i \quad . \tag{2}$$

The subsequent formulas take care of the feasibility of the roster (see Sect. 2.2).

Constraints (a) are dealt with in two steps. Equation (3) takes care of the assignment of duties for the first day in month m. For the other days, assume that a crew i does duty k on day $j-1$. The set $K'_j[k]$ of other duties that cannot be taken by the same crew i on day j because of the 660-minute minimum rest time is given by $\{k' \in K_j \mid ts_{k'} - (te_k - 1440) < 660\}$. Therefore, (4) guarantees the minimum rest time between successive days in month m.

$$x_{i1k} = 0, \ \forall\, i, \ \forall\, k \in K_1 \mid ts_k + lw_i < 660 \ , \tag{3}$$

$$x_{i(j-1)k} + \sum_{k' \in K'_j[k]} x_{ijk'} \leq 1, \ \forall\, i, \ \forall\, j \in \{2, \ldots, d\}, \ \forall\, k \in K_{j-1} \ . \tag{4}$$

Let us define a *complete week* as seven consecutive days, inside month m, ranging from Monday to Sunday. For every complete week, W, in m, we impose the mandatory day off as

$$\sum_{j \in W} x_{ij0} \geq 1, \ \forall\, i \ . \tag{5}$$

If month m does not start with a complete week, let W' be the set of days in the first week of m up to Sunday. Each crew i with $wr_i = 0$ needs to rest in W' and this is achieved with

$$\sum_{j \in W'} x_{ij0} \geq 1, \ \forall\, i \mid wr_i = 0 \ . \tag{6}$$

The constraint stating that for each period of time spanning 7 weeks each crew must have at least one day off on Sunday can be described as follows. For each crew i such that $ls_i + d \geq 49$, we construct the set T_i containing the Sundays in the first $(49 - ls_i)$ days of m. Then, we impose

$$\sum_{j \in T_i} x_{ij0} \geq 1, \ \forall\, i \mid ls_i + d \geq 49 \ . \tag{7}$$

Together, (5) to (7) represent constraints (b).

Suppose that the first day of month m is not Monday and let j^* be the first Sunday in m. To satisfy constraint (c) for each crew i such that $sl_i = 1$, we must state that

$$x_{ij^*0} = 1 \ . \tag{8}$$

Let S_m be the set of Sundays in m after its 6^{th} day and let P_j be the set of split-shift duties on day j. For these Sundays, we respect constraint (c) with

$$x_{ij0} \geq \sum_{k \in P_{j-r}} x_{i(j-r)k}, \ \forall\, i, \ \forall\, j \in S_m, \ \forall\, r \in \{1, \ldots, 6\} \ . \tag{9}$$

Equation (10) guarantees that each crew is assigned exactly one duty in each day, thus satisfying constraints (d). Additionally, (11) represents the implicit constraint that every

Table 2: Computational experiments with the IP model

Name	#Crews	#Days	# Duties Week	Sat	Sun	Holy	LB	Sol	Time
s01	10	10 (1)	00/04	00/01	00/01	00/01	4	6	0.62
s02	10	15 (2)	00/04	00/01	00/01	00/01	4	7	1.50
s03	10	20 (2)	00/04	00/01	00/01	00/01	4	6	2.00
s04	10	25 (2)	00/04	00/01	00/01	00/01	4	6	4.33
s05	10	30 (2)	00/04	00/01	00/01	00/01	4	8	20.91
s06	10	30 (2)	01/04	00/01	00/01	00/01	4	6	9.06
s07	10	30 (2)	02/04	00/01	00/01	00/01	4	6	10.61
s08	10	30 (2)	03/04	00/01	00/01	00/01	4	7	6.81
s09	10	30 (2)	04/04	00/01	00/01	00/01	4	8	9.21
s10	10	30 (2)	04/04	01/01	00/01	00/01	4	7	5.05
s11	10	30 (2)	04/04	01/01	00/01	01/01	4	8	8.35
s12	15	30 (2)	00/04	00/01	00/01	00/01	4	5	8.90

duty must be performed in each day, except for the special duty 0.

$$x_{ij0} + \sum_{k \in K_j} x_{ijk} = 1, \ \forall\, i, \ \forall\, j \ , \tag{10}$$

$$\sum_{i=1}^{n} x_{ijk} = 1, \ \forall\, j, \ \forall\, k \in K_j \ . \tag{11}$$

4.2 Computational Results

The computational results obtained with the IP model are shown in Table 2. The figures under the heading *LB* come from lower bounds on the value of the optimal solution returned by the linear programming relaxation of the IP model. Notice however that the objective function described in Sect. 4.1 asks for the maximization of the number of idle crews, which is equivalent to minimizing the number of crews needed to compose the roster. For the purpose of comparison with the CLP model, the values in the *LB* and *Sol* columns of Table 2 represent the number of crews actually working, i.e. the total number of crews available minus the value of the objective function. Finding the optimal solution of the instances in Table 2 turned out to be a very difficult task, despite their relatively small size. Hence, the solution value in column *Sol* corresponds to the first integer solution found by the model, for each instance. The linear relaxations and the integer programs were solved with the CPLEX[2] Solver, version 6.5.

Although the computation times are quite small, the gap between the values of the lower bounds and the feasible solutions is noticeable. Further, these values are still not a good

[2]CPLEX is a registered trademark of ILOG Inc.

indication of the quality of the model, since we are dealing with very small instances. Yet, when trying to find integer solutions for instances with tens of duties in a workday, this model performed very poorly and no answer could be found within 30 minutes of computation time. Therefore, we decided to experiment with a pure Constraint Logic Programming formulation of the problem.

5 A Constraint Logic Programming Approach

Having found difficulties when solving the crew rostering problem with a pure IP model, as described in Sect. 4, we decided to try a constraint-based formulation. We used the ECL^{i}PS^e [3] finite domain constraint solver, version 4.2, to construct and solve the model.

5.1 The Model

Let n, d, p, q, r and s be defined as in Sect. 4. The main idea of the CLP model for the rostering problem is to represent the final roster as a bidimensional matrix, X, where each cell X_{ij} ($i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, d\}$) contains the duty performed by crew i on day j.

The X_{ij}'s are finite domain variables whose domains depend on the value of j. As in Sect. 4, the duties obtained from the crew scheduling phase are numbered according to their classification as duties for weekdays, Saturdays, Sundays or holidays. In this model, we will not have the concept of a special duty for idleness, as the duty numbered 0 in the IP model. In fact, we will have, for each day, a set of *dummy duties* which tell that a certain crew is off duty.

It is easy to see that the number of crews needed to construct a roster must be at least the maximum number of duties that may be present in any given day of the current month. Thus, we can state that $n \geq \max\{p, q, r, s\}$. Consequently, as the number of X variables for each day j is equal to n, if the domains of these variables were restricted to be the set of duties for day j, some of them would have the same value in the final solution. As we will see later, modeling can be simplified if we avoid this situation and here comes the need for the dummy duties. Let K_j be defined as in Sect. 4. Moreover, let the total number of duties be calculated as $tnd = p + q + r + s$. The domains of the X_{ij} variables are then defined as

$$X_{ij} \ :: \ K_j \cup \{tnd + 1, tnd + 2, \ldots, tnd + (n - |K_j|)\} \quad \forall\, i,\ \forall\, j \ . \tag{12}$$

If X_{ij} is assigned a duty whose number is greater than tnd, it means that crew i is idle on day j.

Three other sets of variables have to be defined in order to facilitate the representation of the constraints. Let TS, TE and SS be lists of integers defined as follows, $\forall\, k \in \{1, \ldots, tnd\}$: $TS[k] = ts_k$, $TE[k] = te_k - 1440$, $SS[k] = ss_k$. The values of ts, te and ss for the dummy duties are $+\infty$, 0 and 0, respectively. The new variables are called $Start_{ij}$, End_{ij} and $Split_{ij}$

[3] `http://www.icparc.ic.ac.uk/eclipse`.

and relate to the X_{ij} variables through `element` constraints:

$$\begin{aligned} &\texttt{element}(X_{ij},\, TS,\, Start_{ij}) \ , \\ &\texttt{element}(X_{ij},\, TE,\, End_{ij}) \ , \\ &\texttt{element}(X_{ij},\, SS,\, Split_{ij}) \ . \end{aligned}$$

Now we can state the constraints (a) through (d) in the ECL^{i}PS^e notation.

Equations (13) and (14) assure that the minimum rest time between consecutive duties is 11 hours. Note the special case for the first day of month m.

$$Start_{i1} + lw_i \ \geq \ 660, \ \ \forall\, i \ , \tag{13}$$

$$Start_{ij} - End_{i(j-1)} \ \geq \ 660, \ \ \forall\, i,\ \forall\, j \in \{2, \ldots, d\} \ . \tag{14}$$

Similarly to what was defined in Sect. 4.1, we use the concept of a complete week, W_i, for each crew i, as a list of variables $[X_{it}, X_{i(t+1)}, \ldots, X_{i(t+6)}]$, where t is any Monday and $t+6$ is its subsequent Sunday, both in month m. The need for at least one day off during each week is represented by (15), for complete weeks. Notice that this constraint must be repeated for each complete week W_i associated with every crew i. If $wr_i = 0$ and the first day of m is not Monday, we also need to impose (16), for each crew i and initial week W_i'.

$$\texttt{atmost\_less}(6, W_i, tnd+1) \ , \tag{15}$$

$$\texttt{atmost\_less}(|W_i'| - 1, W_i', tnd+1) \ . \tag{16}$$

In Equation (16), $|W_i'|$ denotes the number of elements in list W_i'. We use the predicate `atmost_less`(N, L, V) to state that at most N elements of list L can be smaller than V. This behavior is achieved with the definitions below

```
f_less([],_,[]) :- !.
f_less([X|Y],Val,[B|R]) :- #<(X,Val,B), f_less(Y,Val,R).
atmost_less(N,L,Val) :- f_less(L,Val,BF), atmost(N,BF,1).
```

To satisfy constraints (b), there is one condition missing, besides (15) and (16), which assumes at least one day off on Sunday, every seven weeks, for every crew. For each crew i, if $ls_i + d \geq 49$, then

$$\texttt{atmost\_less}(|L_i| - 1, L_i, tnd+1) \ , \tag{17}$$

where L_i is a list containing the X_{ij}'s associated with the Sundays present in the first $(49 - ls_i)$ days of m.

Constraints (c) also make use of the concept of complete weeks, but do not include Sundays. We denote the reduced complete week W_i^* as the list $[Split_{it}, Split_{i(t+1)}, \ldots, Split_{i(t+5)}]$. Notice that we now consider the *Split* variables instead of the X variables, as when representing constraints (b).

$$Split_{it} + \cdots + Split_{i(t+5)} \ \texttt{\#>}\ 0 \quad \texttt{\#=>} \quad X_{i(t+6)} \ \texttt{\#>}\ tnd, \ \ \forall\, i,\ \forall\, W_i^* \ , \tag{18}$$

$$X_{ik} \quad \texttt{\#>} \quad tnd, \ \ \forall\, i \ . \tag{19}$$

By (18), if one of the $Split_{it}, \ldots, Split_{i(t+5)}$ variables equals 1, then crew i must rest on the next Sunday, which corresponds to $X_{i(t+6)}$. The special case of the first week of m, when the month does not start on Monday and $sl_i = 1$, is dealt with by (19). Here, k stands for the first Sunday of month m.

Our choice of variables already guarantees that each crew starts only one duty per day. But we must also make sure that every duty is assigned to one crew on each day. Because of the dummy duties, this condition can be met easily just by forcing the X_{ij} variables to be pairwise distinct, for each day j:

$$\texttt{alldifferent}([X_{1j}, \ldots, X_{nj}]), \quad \forall\, j \ . \tag{20}$$

Finally, we need to preassign the rest days which are known in advance

$$X_{ij} \ \texttt{\#>} \ tnd, \quad \forall\, i, \ \forall\, j \in OFF_i \ . \tag{21}$$

Labeling is done over the X_{ij} variables using the first-fail principle.

5.2 Computational Results

When compared to the IP model of Sect. 4, this model performed much better both in terms of solution quality and computation time. As can be seen in Table 3, it was possible to find feasible solutions for fairly large instances in a few seconds. Again, no minimization predicate was used and the solutions presented here are the first feasible rosters encountered by the model.

Some special cases deserve further consideration. When providing 15 crews to build the rosters for instances s16 and s17, the model could not find a feasible solution even after 10 hours of search. Then, after raising the number of available crews in these instances to 16 (s16a) and 18 (s17a), respectively, two solutions were easily found. Another interesting observation arises from instance s19. This instance comes from the solution of a complete real world crew scheduling problem. In this problem, the optimal solution for weekdays contains 25 duties, 22 of which are split shifts. As we did not have access to the input data sets for the other workdays, the sets of duties for Saturdays, Sundays and holidays are subsets of the solution given by the scheduling algorithm for a weekday. Instance s19a is made up of the same duties, except that all of them are artificially considered non-split shifts. Notice that the value of the first solution found is significantly smaller for instance s19a than it is for instance s19. This is an indication of how severe is the influence of the constraints (c) introduced in Sect. 2.2. Moreover, we can see from Table 3 that the values of the solutions grow quickly as the number of split-shift duties increases. With this point in mind, we generated two other solutions for the same crew scheduling problem where the total number of duties used was increased in favor of a smaller number of split shifts. These are s20 and s21. Despite the larger number of duties in the input, the final roster for these instances uses less crews than it did for instance s19. This strengthens the remark made by Caprara et al. [4] that, ideally, the scheduling and rostering phases should work cyclicly, with some feedback between them.

Table 3: Computational experiments with the CLP model

Name	#Crews	#Days	# Duties				LB	Sol	Time
			Week	Sat	Sun	Holy			
s01	10	10 (1)	00/04	00/01	00/01	00/01	4	5	0.08
s02	10	15 (2)	00/04	00/01	00/01	00/01	4	5	0.18
s03	10	20 (2)	00/04	00/01	00/01	00/01	4	5	0.23
s04	10	25 (2)	00/04	00/01	00/01	00/01	4	5	0.36
s05	10	30 (2)	00/04	00/01	00/01	00/01	4	5	0.48
s06	10	30 (2)	01/04	00/01	00/01	00/01	4	5	0.52
s07	10	30 (2)	02/04	00/01	00/01	00/01	4	5	0.50
s08	10	30 (2)	03/04	00/01	00/01	00/01	4	6	0.52
s09	10	30 (2)	04/04	00/01	00/01	00/01	4	7	0.52
s10	10	30 (2)	04/04	01/01	00/01	00/01	4	7	0.52
s11	10	30 (2)	04/04	01/01	00/01	01/01	4	7	0.53
s12	15	30 (2)	00/04	00/01	00/01	00/01	4	5	0.90
s13	15	30 (2)	00/10	00/06	00/05	00/05	10	13	1.22
s14	15	30 (2)	03/10	01/06	00/05	01/05	10	13	1.35
s15	15	30 (2)	03/10	03/06	00/05	03/05	10	13	1.36
s16	15	30 (2)	05/10	03/06	00/05	03/05	10	?	> 10 h
s16a	16	30 (2)	05/10	03/06	00/05	03/05	10	16	1.49
s17	15	30 (2)	07/10	03/06	00/05	03/05	10	?	> 10 h
s17a	18	30 (2)	07/10	03/06	00/05	03/05	10	18	1.78
s18	30	30 (2)	00/20	00/10	00/10	00/10	20	25	4.96
s19	50	30 (2)	22/25	12/15	12/15	12/15	25	47	14.46
s19a	40	30 (2)	00/25	00/15	00/15	00/15	25	33	9.36
s20	40	30 (2)	06/26	02/15	02/15	02/15	26	34	10.50
s21	40	30 (2)	00/31	00/20	00/20	00/20	31	36	8.30

6 Proving Optimality

In Sects. 4 and 5, we showed that finding provably optimal solutions for this rostering problem is a difficult task. Moreover, it is possible to see from Table 3 that the lower bound provided by the Linear Programming relaxation of the problem is always equal to the largest number of duties that must be performed on a workday. This is clearly a trivial lower bound and probably not a very good one. We decided then to try another formulation for the problem, so as to find better feasible solutions or, at least, better lower bounds.

6.1 A Hybrid Model

Another possible mathematical model for the rostering problem turns out to be a typical set partitioning formulation:

$$\begin{aligned} \min \ & \sum_{j=1}^{n} x_j \\ \text{subject to } & \sum_{j=1}^{n} a_{ij} x_j = 1, \ \ \forall\, i \in \{1, \ldots, e\} \\ & x_j \in \{0,1\}, \ \ \forall\, j \in \{1, \ldots, n\} \ . \end{aligned}$$

All numbers a_{ij} in the coefficient matrix A are 0 or 1 and its columns are constructed as shown in Fig. 1. Each column is composed of d sequences of numbers, where d is the number of days in the planning horizon. For each $k \in \{1, \ldots, d\}$, the k-th sequence, l_k, contains h_k numbers, where h_k is the number of duties that must be performed on day k. Also, at most one number inside each sequence is equal to 1. The number of lines e, in A, equals $\sum_{k=1}^{d} h_k$.

$$(\overbrace{0 \cdots 0\,1\,0 \cdots 0}^{h_1} \overbrace{0 \cdots 0\,1\,0 \cdots 0}^{h_2} \cdots \overbrace{0 \cdots 0\,1\,0 \cdots 0}^{h_d})^{\mathrm{T}}$$

Figure 1: A column in the coefficient matrix of the set partitioning formulation

Besides having the previous characteristics, a column in A must represent a feasible roster for one crew. More precisely, let $t = (u_1, u_2, \ldots, u_d)$ be a feasible roster for a crew, where u_k, $k \in \{1, \ldots, d\}$, is the number of the duty performed on day k. Remember from Sect. 4.1 that $u_k \in D_k \cup \{0\}$, where D_k may be equal to $\{1, \ldots, p\}$, $\{p+1, \ldots, p+q\}$, $\{p+q+1, \ldots, p+q+r\}$ or $\{p+q+r+1, \ldots, p+q+r+s\}$, depending on whether k is a weekday, a Saturday, a Sunday or a holiday, respectively. For every such feasible roster t, A will have a column where, in each sequence l_k, the i-th number will be equal to 1 ($i \in \{1, \ldots, h_k\}$) if and only if u_k is the i-th duty of D_k. In case $u_k = 0$, all numbers in sequence l_k are set to 0.

With this representation, the objective is to find a subset of the columns of A, with minimum size, such that each line is covered exactly once. This is equivalent to finding a number of feasible rosters which execute the all the duties in each day of the planning horizon.

It is not difficult to see that the number of columns in the coefficient matrix is enormous and it is hopeless to try to generate them all in advance. Hence, we decided to implement a *Branch-and-Price* algorithm [1] to solve this problem, generating columns as they are needed. This approach is considered hybrid because the column generation subproblem is solved by a Constraint Logic Programming model. In our case, this model is a variation of the CLP model of Sect. 5. Now, instead of looking for a complete solution for the rostering problem, we are only interested in finding, at each time, a feasible roster corresponding to a column in A with negative reduced cost. The whole algorithm follows the same basic ideas described in [8].

Table 4: Computational experiments with the hybrid model

Name	#Crews	#Days	# Duties				Opt	Time
			Week	Sat	Sun	Holy		
s01	10	10 (1)	00/04	00/01	00/01	00/01	5	0.95
s02	10	15 (2)	00/04	00/01	00/01	00/01	5	2.19
s03	10	20 (2)	00/04	00/01	00/01	00/01	5	10.57
s04	10	25 (2)	00/04	00/01	00/01	00/01	5	639.75
s05	10	30 (2)	00/04	00/01	00/01	00/01	5	38.12
s06	10	30 (2)	01/04	00/01	00/01	00/01	5	30.60
s07	10	30 (2)	02/04	00/01	00/01	00/01	?	> 1 h

6.2 Computational Results

The best results for the hybrid model were achieved when setting the initial columns of matrix A as the columns corresponding to the first solution found by the CLP model of Sect. 5. Also, the ordinary labeling mechanism worked better than labeling according to the first-fail principle.

With this model, we could find provably optimal solutions for small instances of the rostering problem, as shown in Table 4, where column *Opt* gives the optimal value. This is a noticeable improvement over the pure IP model of Sect. 4, which was not able to find any optimal solution, even for the smallest instances. Besides, when comparing Tables 3 and 4, we can see that the first solutions found by the pure CLP model for instances s01 to s06 are indeed optimal.

This hybrid approach is still under development and there is a lot of work to be done. Nevertheless, we believe that the main reason for the behavior of this model resides on the fact that this formulation leads to a highly degenerate problem. When trying to solve larger instances, the pricing subroutine keeps generating columns indefinitely, with no improvements on the value of the objective function. This is because there are many basic variables with value zero which are replaced by other columns that enter the basis with value zero as well. As a consequence, the linear relaxation of the first node of the *Branch-and-Price* enumeration tree could not be completely solved in the medium and large-sized instances. Thus, in order to obtain better linear programming lower bounds, we need to address those degeneracy problems more closely.

7 Conclusions and Future Work

We have given a detailed description of an urban transit crew rostering problem that is part of the overall crew management process in a medium-sized Brazilian bus company. This problem is rather different from some other bus crew rostering problems found in the literature.

Three main approaches have been applied in order to solve this problem. Initially, a

pure Integer Programming (IP) model was developed, enabling us to find feasible rosters for very small instances. We achieved better results with a pure Constraint Logic Programming (CLP) model, which managed to construct feasible solutions for typical real world instances in a few seconds.

Obtaining better lower bounds on the value of the optimal solution could be helpful in estimating more precisely the quality of the solutions obtained with the pure CLP model. Therefore, following our experience with good quality lower bounds provided by linear relaxations of set partitioning formulations [8], we devised a third approach. The rostering problem was then formulated as a set partitioning problem with a huge number of columns in the coefficient matrix. This integer program was fed into a hybrid column generation algorithm which followed the same ideas presented in [8]. With this attempt, we could find optimal solutions for small instances of the problem. Finding provably optimal solutions for the largest instances is still a difficult task, apparently due to degeneracy problems. We believe that the performance of this third model can be significantly improved if these issues are investigated in more detail. Besides, it may also be possible to improve the labeling strategy with problem specific heuristics, and extract a better performance from the constraint-based column generator.

References

[1] 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. Technical Report COC-9403, Georgia Institute of Technology, Atlanta, EUA, 1993. http://tli.isye.gatech.edu/research/papers/papers.htm.

[2] L. Bianco, M. Bielli, A. Mingozzi, S. Ricciardelli, and M. Spandoni. A heuristic procedure for the crew rostering problem. *European Journal of Operational Research*, 58(2):272–283, 1992.

[3] A. Caprara, M. Fischetti, P. Toth, and D. Vigo. Modeling and solving the crew rostering problem. Technical Report OR-95-6, DEIS, University of Bologna, Italy, 1995. Published in *Software Practice and Experience* number 28, 1998.

[4] A. Caprara, M. Fischetti, P. Toth, D. Vigo, and P. L. Guida. Algorithms for railway crew management. *Mathematical Programming*, 79:125–141, 1997.

[5] A. Caprara, F. Focacci, E. Lamma, P. Mello, M. Milano, P. Toth, and D. Vigo. Integrating constraint logic programming and operations research techniques for the crew rostering problem. Technical Report OR-96-12, DEIS, University of Bologna, Italy, 1996. Published in *Operations Research* number 46, 1999.

[6] P. Carraresi and G. Gallo. A multi-level bottleneck assignment approach to the bus drivers' rostering problem. *European Journal of Operational Research*, 16(2):163–173, 1984.

[7] J. K. Jachnik. Attendance and rostering system. In A. Wren, editor, *Computer Scheduling of Public Transport*, pages 337–343. North-Holland Publishing Co., 1981.

[8] T. H. Yunes, A. V. Moura, and C. C. de Souza. A hybrid approach for solving large scale crew scheduling problems. In *Lecture Notes in Computer Science,* vol. 1753, pages 293–307, Boston, MA, USA, January 2000. Proceedings of the Second International Workshop on Practical Aspects of Declarative Languages (PADL'00).