Optimized Rostering of Workforce Subject to Cyclic Requirements

François Ramond and David De Almeida
SNCF - Direction de l'Innovation et de la Recherche,
40 avenue des Terroirs de France, 75611 Paris Cedex 12, France
E-mail: francois.ramond@sncf.fr, david.de_almeida@sncf.fr

Stéphane Dauzère-Pérès
Ecole des Mines de Saint-Etienne
Centre Microélectronique de Provence
880 avenue de Mimet, 13541 Gardanne, France
E-mail: dauzere-peres@emse.fr

Hanif D. Sherali
Virginia Tech, Grado Department of Industrial and Systems Engineering,
250 Durham Hall, Blacksburg, VA 24061, USA
E-mail: hanifs@vt.edu

Keywords: optimization, rostering, workforce

Received: June 23, 2012

SNCF is a large railway transportation company that operates 365 days a year and 24 hours a day. In order to schedule a certain category of workers at train stations and ticket selling points, rosters are designed to cover a cyclical demand. However, the highly combinatorial nature of the rostering problem makes it very difficult to solve it manually, and experts spend a huge amount of time to make them legally feasible and to improve a certain number of preference criteria. This paper presents a mixed-integer programming model to address the cyclical rostering problem using patterns corresponding to feasible blocks of seven days and assigning them to each week of the roster. Some valid inequalities are presented to improve the linear relaxation of the model and thereby enhance computational performance. Implementation results are presented, including comparisons with an alternative daily-variables model

Povzetek: Opisano je optimirano cikli?no razporejanje delavcev z aplikacijo za podjetje SNCF.

1 Introduction

Like many public transportation companies, the French national railways (SNCF, for Société Nationale des Chemins de fer Franąis) require some work to be performed 365 days a year and 24 hours a day. Because of complex legal issues, planning of human resources is very difficult to implement. People responsible for human resources in different operational units spend a considerable amount of time preparing timetables. Yet the plan they finally obtain is rarely optimal with respect to preferences of workers and unions or with respect to costs. Indeed, this kind of problem is highly combinatorial, and so far, no software-based solution approach has been implemented at SNCF. The lack of automation in producing timetables was put under the spotlight when a new regulation of work schedules in 1997 reduced the overall time of work over a year for all employees and required most timetables to be redesigned.

Nevertheless, many papers dealing with personnel scheduling have been published in the open literature, focusing on different problems specific to several fields [20]. Hospitals, public services (firefighting and police units), and airline and railway companies are among the most studied domains. These organizations share the characteristic of being operated 365 days a year and 24 hours a day, which makes workforce scheduling particularly fastidious and justifies the effort to design effective decision-support systems.

Two approaches have been considered for workforce scheduling: the first approach aims at minimizing the costs of production through the number of employees required to perform a certain amount of work [1, 2, 3, 4], whereas the goal of the second approach is to actually schedule the work performed by a certain number of employees with respect to a set of operational constraints, while minimizing costs. Some research is also focused on integrating these two phases into a single stage to produce either cyclic or non-cyclic rosters [14].

In the case of the second approach, Beaumont [5] develops a mixed-integer programming (MIP) model to design cyclic rosters of length one year, including four or five holiday weeks and a certain number of rostered off-days. Constraints on minimal and maximal lengths of work stretches and rest periods are explicitly expressed, and the objective function aims at minimizing costs related to workload coverage and acceptability of the roster. Freling et al. [17]

present a similar problem, which is divided into four parts. A first module enables the feasibility checking of given rosters, a second one is responsible for the generation of feasible rosters, and a third one evaluates each roster with respect to its cost and preferential criteria. Finally, the fourth module selects the best quality rosters through mathematical programming based methods (set partitioning problem).

Problems of workforce scheduling relative to nurses focus more specifically on the satisfaction of employee preferences. Thus, Miller et al. [26] consider two sets of constraints in their integer program, namely hard constraints defining the feasibility set, and soft constraints whose violation is permitted but penalized by an associated cost in the objective function. Sherali et al. [30] develop a mixed-integer program for the resident scheduling problem (RSP) at the *St John Hospital and Medical Center* and exploit the inherent network structure of the problem to design a solution procedure. The advantage of this methodology lies in its capacity to propose compromise solutions when the MIP model turns out to be infeasible. For surveys on nurse rostering problems, we refer the interested reader to [10] and [7]. In a recent paper, Glass and Knight [19] study the nurse rostering problem structure, and propose an mixed-integer programming approach validated on four benchmark problem instances. Another contribution is a methodology for handling continuity between rostering periods.

The problems encountered in the airline and railway industries are, in general, divided into two sub-problems: the Crew Scheduling Problem (CSP) [35] and the Crew Rostering Problem (CRP) [11, 16, 22, 23]. The CSP deals with the design of *pairings*, which are sequences of tasks and rest periods lasting typically 24 to 72 hours, and starting and ending at the same domicile location. The CRP can be seen as the natural consequence of the CSP because its aim is to assign pairings to specific employees and to sequence them over a longer term planning horizon, typically one to four months. Caprara et al. [8, 9] address the Crew Scheduling Problem and the Crew Rostering Problem for the railway industry. In [9], a procedure is proposed to combine the CSP and the CRP in an iterative fashion, through the computation of the Lagrangian cost of potential pairings. De Pont [12] discusses the construction of rosters for Dutch railway operators. Other research in the context of the airline industry is concerned with the integration of aircraft routing and crew scheduling (see for instance [24] and [25]). This is not relevant in our case since we are interested in designing rosters for so-called sedentary workers.

Similar problems to those cited above can be found in urban transportation companies. The problem studied by Townsend [34] for the bus drivers of "London Regional Transport" is of certain interest in the sense that it has similarities with the design of rosters at SNCF. A solution procedure based on the utilization of pre-built patterns of one, four, or five weeks is proposed. This procedure is problem-specific and cannot be used here. However, the use of pre-built patterns, such as the use of pairings in the CRP, is worthwhile since it enables a higher level of abstraction in the formulation of the model. We refer the reader to [15] for an annotated bibliography of personnel scheduling and rostering.

Regarding commercial software, a few rostering packages are available on the market. The software modules developed by Quintiq are among the most popular ones used in the industry and rely heavily on Operations Research techniques (see [28]). The rostering problem described in this paper is, however, too specific to be solved using a generic software tool.

The present paper makes the following specific contributions:

1. We describe the employee rostering problem faced at SNCF for a class of workers, and discuss related specific labor rules and work restrictions along with employee and management performance criteria for assuring high quality rosters.

2. We design an MIP model using special weekly pattern blocks composed of feasible compositions of work stretches and rest periods, and we further enhance the solvability of the model by incorporating two classes of valid inequalities. The proposed model is structured to facilitate a direct implementation using a commercial MIP software package (we used CPLEX [21] for this purpose).

3. We present computational results based on real data at SNCF and provide comparisons against an alternative daily-variable model. Some practical implementation guidelines are also discussed.

The remainder of this paper is organized as follows. Section 2 explains the rosters to be generated. Section 3 develops the formulation designed to solve the problem. Some experimental results are presented in Section 4. Section 5 concludes the paper with some perspectives for future research.

2 Problem Definition

A roster is basically a table (see Figure 1) whose rows correspond to work cycles; there are as many rows in the roster as employees in the team. Each row or work cycle is a sequence of work stretches (sequence of consecutive working days associated with different shifts - for example, morning, evening, or night) and rest days. Once designed and validated, the roster is used until a major change arises such as, for example, an evolution of the requirements in terms of personnel or a modification of labor policies or union rules.

The idea behind a roster is that every worker in the team begins on a different row and then progresses cyclically through the rows of the roster. Hence, the first worker begins on the first row, and then continues for the next cycle according to the second row, and so on. Likewise, the second worker begins with the second row and proceeds cyclically down the roster, returning back to the first row. As a

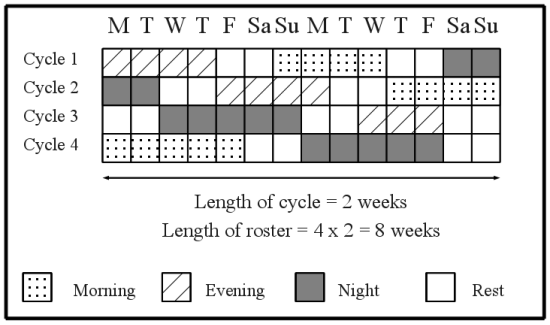

Figure 1: Example of a roster with three daily shifts and cycle length of two weeks

result, the work covered by the team of workers is cyclical, with a period equal to the length of the cycle of the roster. The work covered by a given employee is also cyclical, with the associated period being the total length of the roster (product of the length of the cycle with the number of rows).

In practice, for economic reasons, the workload is not fully covered by the employees in the roster. Certain *reserve employees* are typically called to cover the shifts that are not covered by the team scheduled in the roster. Although such reserve employees are called on a regular basis by a certain number of rosters corresponding to their skills and experience, their work is not scheduled through the use of rosters.

Once the phase of roster design is complete, the process of scheduling is executed using this roster. This has one major advantage: it does not require any modification from the human resources managers, except for slight adjustments to take production disturbances into account and to plan the holidays of the employees. It also guarantees a certain equity between employees in a given roster, since they all share the same plan (although with different starting points). Finally, workers enjoy transparency with respect to their work schedules. However, major changes cannot be made to the schedules without having to build a new roster; hence leading to a lack of flexibility.

2.1 Labor rules

Rosters have to comply with various rules described in [32, 33]. Recall that, in this paper, rosters are constructed for sedentary workers that operate, for instance, at train stations and ticket selling points. To clarify the following, a distinction is made between *periodic* and *daily* rests: a periodic rest designates the off-day(s) between work stretches, whereas a daily rest is the rest period between two consecutive workdays within a work stretch.

A work stretch cannot last less than three days or longer than five days. Periodic rests must last one, two, or three consecutive off-days.

Furthermore, there are two categories of employees: one is given 114 off-days per year, and the other one is given 118 off-days. The category to which an employee belongs depends on the difficulty of working conditions (and in particular, on the length of night work); all employees that share a given roster, however, must belong to the same category, and so, the data for any instance specifies the particular targeted number of off-days per year for each employee.

Each employee must have at least 12 consecutive pairs of Saturday-Sundays off, and at least a total of 22 Sundays off combined with an adjacent Saturday off or an adjacent Monday off (called *weekends*). For instance, a roster providing yearly 12 Saturday-Sundays off and 10 Sunday-Mondays off (hence, 12+10=22 weekends) satisfies these rules.

Finally, the daily rest between two consecutive workdays must last 12 hours at a minimum. Also, starting with the first off-day of periodic rest there should be a break for at least 36 hours. If this rule cannot be respected, the rest period must last at least 24 hours and the reduction below these 36 hours must be replaced at the latest during the second periodic rest that follows. In any case, the length of the second and third off-days cannot last less than 24 hours.

2.2 Preference criteria

Aside from the constraints cited above, the quality of a roster is evaluated by employees and managers according to several criteria such as:

1. The number of single off-days (i.e., non-consecutive to other off-days) : to be minimized.
2. The range-width of consecutive Saturday-Sundays off over the different cycles of the roster : to be minimized.
3. The range-width of weekends over the different cycles of the roster : to be minimized.
4. The peaks in the use of reserve teams, i.e., the maximal number of calls to reserve employees over the days of the roster : to be minimized.

3 MIP Formulation of the Problem

The formulation of the roster design problem we propose uses a set of pre-built patterns of one week and generates rosters within which each day is either a work day or an off-day (daily shifts are not considered). Section 3.1 discusses four important assumptions, and the MIP model itself is presented in Section 3.2.

3.1 Model assumptions

1. Approximations: Labor rules set the annual number of days of periodic rest to 114 or 118 days, which corre-

sponds to a proportion of $\frac{114}{365}$ or $\frac{118}{365}$ of off-days. However, this proportion can rarely lead to an integer in the case of real-life rosters. For instance, a roster of length 8 weeks (56 days) for employees of Category 1 (114 off-days) must include at least $\frac{114\times56}{365} \approx 17.49$ off-days. This number is rounded down (17 in the example) because it is always preferable for a manager to assign an off-day that was rostered as a workday than to assign a workday that was rostered as an off-day. Indeed, managers can perform slight changes on the rosters throughout the year to meet labor rules by calling reserve employees. On the other hand, numbers concerning the minimum proportions of Saturday-Sundays off and weekends will be rounded up.

2. Workload: We assume that the required workload is constant over each week, which therefore facilitates the use of cyclical schedules.

3. Personnel capacities: We assume that personnel capacities are not limited, meaning that whatever the work requirements, there are enough employees to cover the workload. This remark concerns both rostered employees and reserve employees.

4. Shifts: Typically, each workday is partitioned into three shifts : Morning, Evening, and Night. In practice, it is desirable (but not necessary) to assign employees to particular shifts over each work stretch by varying the type of shift in order; for example, Morning, then Night, and then the Evening shift. This facilitates satisfying daily rest constraints and is also desirable from the viewpoint of work rhythm and health perspectives. Note that the proposed model built on one-week patterns ignores such shift considerations, and focuses mainly on determining work stretches comprising workdays and off-days. We assume that managers would determine shift assignments along with any subsequent tweaking of the generated schedules as necessary at a later stage. (Section 4.2 provides additional discussion, including possible enhancements in the proposed model.)

3.2 Formulation using one-week patterns

The proposed Mixed-Integer Programming (MIP) model is inspired by research performed in the airline industry and uses one-week patterns that can be considered as "blocks" composed of work stretches and rest periods. They are built such that, within a pattern, the rules relative to the length of work periods (3 to 5 days) and rest periods (1 to 3 days) are respected. Some constraints on the succession of patterns ensure that these rules are also respected when coupling two adjacent patterns. A limited number of patterns is used, which represents the exhaustive set of all different "types" of weeks respecting labor rules from the basic work-or-rest point of view. Thus, these patterns only include two types of days, which are work days and rest days. Such patterns are assumed to be generated *a priori* using historical experience and managerial insights. Column generation approaches, as for example reviewed in [18] and [13] could be used alternatively – we recommend such an investigation for future research. The set of patterns used in our model is denoted by P.

3.2.1 Model parameters

An instance of the problem is fully defined by the following parameters:

- The number of daily shifts (nds) used in the roster. This is a positive integer.
- The work requirements in number of employees ($req_{s,d}$) for each shift $s \in S = \{0, 1, ..., nds - 1\}$ and each day of the week $d \in DW = \{1, 2, ..., 7\}$ (1 for Monday to 7 for Sunday). Note that for the proposed one-week patterns model, only the aggregate requirement $\sum_{s\in S} req_{s,d}$ for each day $d \in DW$ is of relevance.
- The number of cycles in the roster, corresponding to the number of employees in the team: ncr ($CR = \{1, 2, ..., ncr\}$ denotes the set of cycles).
- The number of weeks in a cycle: nwc ($WC = \{1, 2, ..., nwc\}$ denotes the set of weeks of a cycle).
- The number of weeks in the roster: $nwr = nwc \times ncr$ ($WR = \{1, 2, ..., nwr\}$ denotes the set of weeks of the roster, indexed consecutively in order of occurence over the cycles of the roster).
- The number of days in a cycle of the roster: $ndc = 7 \times nwc$ ($DC = \{1, 2, ..., ndc\}$ denotes the set of days of a cycle, indexed consecutively in order of occurence over the cycle).
- The number of days in the roster: $ndr = 7 \times nwr$ ($DR = \{1, 2, ..., ndr\}$ represents the set of days in the roster).
- The minimum numbers of off-days, Saturday-Sundays off, and weekends off to be included in the roster, respectively: $minNbOd$, $minNbSatSun$, and $minNbWkendOff$.
- The nature of day d of pattern p, $pattern_{p,d}$, is equal to 0 if d is a rest day and equal to 1 if d is a work day. The index d varies between 1 (Monday) and 7 (Sunday).
- The binary parameter $potMonSingle_p$ is equal to 1 if Monday of pattern p is a rest day and Tuesday of the same pattern p is a work day, and 0 otherwise. In this case, Monday of pattern p is a *potential* single off-day.
- The parameter $potSunSingle_p$ is equal to 1 if Sunday of pattern p is a rest day and Saturday of the same pattern is a work day.
- The number of single off-days within pattern p (i.e., from Tuesday to Saturday) is represented by $withinSingle_p$.
- The parameter $pattern_{p,67}$ is equal to 1 if both Saturday and Sunday of pattern p are rest days, 0 otherwise. Note that these last four parameters are computed while constructing the actual MIP model.
- $cannotFollow_p$ are sets that are used to determine successive pairs of patterns that would violate the rules stating that work stretches vary "between three and five days" and

rest periods "between one and three days":

$$cannotFollow_p = \{p' \in P : p' \text{cannot be selected just after } p \text{ in a roster}\}.$$

3.2.2 Principal decision variables

The principal decision variables of this model decide whether pattern p is associated with week w of the roster:

$$x_{p,w} = 1 \text{ if pattern } p \text{ is associated with week } w, \text{ and 0 otherwise.}$$

3.2.3 Auxiliary decision variables

- $satSun_w$ is equal to 1 if Saturday and Sunday of week w of the roster are both rest days, and 0 otherwise.
- $wKend_w$ is equal to 1 if $satSun_w$ is equal to 1 *or* if Sunday of week w and Monday of week $w+1$ are both off-days, and 0 otherwise.
- likewise, $sunMon_w$ is equal to 1 if Sunday of week w and Monday of week $w+1$ are both off-days, and 0 otherwise.
- $monSingle_w$ (respectively $sunSingle_w$) is equal to 1 if Monday (respectively Sunday) of week w is a single off-day, and 0 otherwise.
- $nSingle_w$ is an integer variable equal to the number of single off-days of week w of the roster.
- res_d is an integer variable equal to the number of reserve employees used to fully cover the workload on day d of a cycle.
- $nSingleOffDays$ is an integer variable associated with the number of single off-days.
- $minSatSun$ and $maxSatSun$ are integer variables associated respectively with the minimal and maximal numbers of consecutive Saturday-Sundays off over all cycles, and $diffMinMaxSatSun$ denotes their difference.
- $minWkend$ and $maxWkend$ are, likewise, integer variables associated respectively with the minimal and maximal numbers of weekends off over all cycles of the roster, and $diffMinMaxWkend$ denotes their difference.
- Other dependent variables are evident through the constraint definitions below.

3.2.4 Constraints of the formulation

The unique choice of pattern for each week of the roster is imposed by:

$$\sum_{p \in P} x_{p,w} = 1, \quad \forall w \in WR. \tag{1}$$

The constraint on the admissible succession of patterns is enforced by (in light of (1)):

$$x_{p,w} + \sum_{p' \in cannotFollow_p} x_{p',w+1} \leq 1, \quad \forall w \in WR, \quad \forall p \in P. \tag{2}$$

The number of Mondays that are single off-days is defined by the following constraint, which imposes that $monSingle_w$ is equal to 1 if $potMonSingle_p = 1$ (for pattern p associated with week w) and if the adjacent Sunday is a work day:

$$\begin{aligned} monSingle_w &\geq \sum_{p \in P} (x_{p,w} \cdot potMonSingle_p) \quad (3) \\ &+ \sum_{p \in P} (x_{p,w-1} \cdot pattern_{p,7}) \\ &\quad -1, \forall w \in WR. \quad (4) \\ monSingle_w &\geq 0, \qquad \forall w \in WR. \quad (5) \end{aligned}$$

The same types of constraints are used to determine the number of Sundays that are single off-days, $sunSingle_w$. Then, the number of single off-days in week w is given by the sum of single off-days within pattern p associated with week w, plus Monday or Sunday if these are single off-days:

$$nSingle_w \geq \sum_{p \in P} (x_{p,w} \cdot withinSingle_p) + monSingle_w + sunSingle_w, \quad \forall w \in WR. \tag{6}$$

Accordingly, the total number of single off-days in the roster is given by:

$$nSingleOffDays = \sum_{w \in WR} nSingle_w. \tag{7}$$

Constraint (8) determines if Saturday or Sunday of week w are off-days, and, if both are off-days, sets $satSun_w$ to 1.

$$satSun_w = \sum_{p \in P} x_{p,w} \cdot pattern_{p,67}, \forall w \in WR. \tag{8}$$

Constraints (9) – (14), expressed for Sunday and Monday, enable to define if Sunday of week w and the consecutive Monday of week $w+1$ are off-days, in which case $sunMon_w$ is set equal to 1. It is more complex to express here, compared with the above Saturday-Sunday constraint, because the value of the variable depends on the choice of patterns for both weeks w and $w+1$.

$$sun_w = \sum_{p \in P} x_{p,w} \cdot pattern_{p,7}, \forall w \in WR. \tag{9}$$

$$mon_w = \sum_{p \in P} x_{p,w} \cdot pattern_{p,1}, \forall w \in WR. \tag{10}$$

$$sunMon_w \geq 1 - sun_w - mon_{w+1}, \forall w \in WR. \tag{11}$$

$$\begin{aligned} sunMon_w &\leq 1 - sun_w, \forall w \in WR, \quad (12) \\ sunMon_w &\leq 1 - mon_w, \forall w \in WR, \quad (13) \\ sunMon_w &\geq 0, \forall w \in WR. \quad (14) \end{aligned}$$

The definition of weekends is then obtained by (15) to (18):

$$wKend_w \geq satSun_w, \forall w \in WR. \quad (15)$$
$$wKend_w \geq sunMon_w, \forall w \in WR. \quad (16)$$
$$wKend_w \leq satSun_w + sunMon_w, \quad (17)$$
$$\forall w \in WR. \quad (18)$$

Note that Equation (19) helps further tighten the LP relaxation, besides enforcing $wKend_w \leq 1$.

$$wKend_w \leq 1 - sun_w, \forall w \in WR. \quad (19)$$

The number of consecutive Saturday-Sundays off in each cycle, as well as the difference between the maximum and the minimum values over all cycles, are determined through Constraints (20) to (23).

$$satSunCycle_c = \sum_{w \in WC} satSun_{nwc\cdot(c-1)+w}, \forall c \in CR. \quad (20)$$

$$maxSatSun \geq satSunCycle_c, \forall c \in CR. \quad (21)$$

$$minSatSun \leq satSunCycle_c, \forall c \in CR. \quad (22)$$

$$diffMinMaxSatSun = maxSatSun - minSatSun. \quad (23)$$

Similar constraints are used to compute $wKendCycle_c$, $minWkend$, $maxWkend$, and $diffMinMaxWkend$ for weekends.

The respecting of labor rules on the number of consecutive Saturday-Sundays off, on the number of weekends, and on the total number of off-days over the roster is ensured via Constraints (24), (25), and (26), respectively.

$$\sum_{w \in WR} satSun_w \geq minNbSatSun. \quad (24)$$

$$\sum_{w \in WR} wKend_w \geq minNbWkendOff. \quad (25)$$

$$\sum_{\substack{w \in WR \\ p \in P}} \sum_{d=1}^{7} x_{p,w}.pattern_{p,d} = ndr - minNbOd. \quad (26)$$

Reserve calls (res_d) on each day d of a cycle are equal to the total requirements minus the shifts assigned to employees in the roster, as expressed by the following constraint (in which $\mathrm{mod}^+ 7$ designates an integer between 1 and 7, computed via modulo 7, except that a modulo value of 0 is replaced by 7):

$$res_d = \sum_{s \in S} req_{s,(d \bmod^+ 7)} - \sum_{\substack{c \in CR \\ p \in P}} x_{p,(c-1)\cdot nwc+\lceil d/7 \rceil} \cdot pattern_{p,(d \bmod^+ 7)}, \quad \forall d \in DC. \quad (27)$$

The maximum number of reserve calls over all days d of a cycle is then bounded by res_d.

$$maxRes \geq res_d, \forall d \in DC. \quad (28)$$

Finally, all auxiliary variables are automatically explicitly restricted to be binary or integer variables, where the principal decision variables $x_{p,w}$ are required to be binary-valued.

$$x_{p,w} \in \{0,1\}, \qquad \forall p \in P, \forall w \in WR. \quad (29)$$

3.2.5 Objective function

The objective function is a weighted sum (with suitable positive weights A, B, C, and D prioritizing the different terms) of the criteria described in Section 2.2, expressing the desirability of the roster from the point of view of both employees and managing staff:

$Minimize$:
- $A \cdot nSingleOffDays$ (see Constraint (7))
- $+B \cdot diffMinMaxSatSun$ (see Constraint (23))
- $+C \cdot diffMinMaxWkend$ (see statement after Constraint (23))
- $+D \cdot maxRes$ (see Constraint (28)).

3.2.6 Model symmetry

Note that the model possesses inherent symmetries due to its cyclical nature that could be inhibited to achieve greater computational efficiency (see [31], for example). In particular, to address this issue, we tried (see [29]) to add some constraints expressing the fact that the first pattern of the first cycle should be the one having the lowest index among all patterns assigned to the beginning of cycles. Some preliminary tests revealed that these symmetry-defeating restrictions did not help much in reducing the computational times, so we did not go further in this direction. However, we advocate a further investigation of this issue for future research.

3.3 Formulation using daily variables

Another formulation to address the cyclical rostering problem at SNCF was also developed based on daily variables that determine, for each day d of the roster, whether a rest period or a work period of any permitted duration, and any required shift among Morning, Night, and Evening, begins on day d (see [6] and [29]). This formulation turned out to be computationally prohibitive due to the large number of integer variables, and we refer the interested reader to [29] for details.

4 Model Refinements and Experimental Results

All computations were performed on a PC equipped with a 2.4 GHz processor, 512 MB memory, and using the ILOG CPLEX 8.0 optimization software with default settings (although more efficient versions of CPLEX presently exist [21], this was the software available to us at the time of the study). The method used to solve the MIP model is the Branch & Cut algorithm [27] implemented within CPLEX and the runs were executed until an optimal solution was found.

Some additional refinements were made to the model for improving its computational performance. First, we relaxed all auxiliary varibles, earlier defined as integer variables. Integer variables are, in general, known to be difficult to deal with. However, this extensive use of integer variables is not necessary since many of them are simply defined as intermediate variables that depend only on the principal variables of the formulation. Indeed, these variables automatically take on integer values at optimality.

Furthermore, we introduced some valid inequalities (or cuts) in the model before calling CPLEX to solve the problem. These cuts simply deal with the values that can be taken by some variables under given conditions:

1. We explicitly imposed $maxRes \geq 1$ if the number of employees in the roster is *a priori* known to be insufficient to cover the entire workload.

2. $diffMinMaxSatSun$ and $diffMinMaxWkend$ cannot be equal to 0 if the number of consecutive Saturday-Sundays off (respectively, weekends off) that is specified to be included in the roster is not a multiple of the number of cycles. For example, if one wants to include five Saturday-Sundays off and nine weekends off in a four-cycle roster, it is not possible to obtain an equal number of Saturday-Sundays and weekends off in each cycle. Hence, we specifically impose $diffMinMaxSatSun \geq 1$ and $diffMinMaxWkend \geq 1$ in such cases.

Our test set was comprised of $432 = 16 \times 27$ realistic instances constructed by composing :

- 16 combinations of employee requirements and numbers of weeks per cycle corresponding to real-world instances that vary between 2 and 4 employees per day, and between 2 and 5 weeks per cycle; along with:

- 27 $(= 3 \times 3 \times 3)$ realistic sets of values for the minimum annual number of: (a) off-days (114, 118, or 122, where this third case was introduced for experimental purposes); (b) consecutive Saturday-Sunday off-days (10, 12, or 14); and (c) number of weekends(19, 22, or 25).

Table 1 displays results on the computational times (in seconds) required to obtain optimal rosters with the different cuts. We tested the aforementioned 432 instances for each configuration : without the two cuts mentioned above, with cut (1), with cuts (2), and with cuts (1) and (2). The columns of this table provide the average and maximum CPU times over all the test problems, and the number of problems whose computation requires more than 1 s., 10 s., and 100 s., and the mean CPU times in these three categories.

The results presented in Table 1 show that cut (1), by itself or in combination with cut (2), does not help the resolution process as it increases the mean CPU time. More recent versions of CPLEX might be able to handle such lower-bounding restrictions more efficiently. However, adding cut (2) to the model leads to major improvements in the computational effort: the mean CPU time with (2) is smaller by a factor of more than five, and there is only one instance requiring more than 100 s. In general, cuts alter LP relaxation solutions of different nodes of the branch-and-bound tree, thereby affecting the choice of branching variables, and consequently result in varying effects on overall performance. In our runs, for three particular instances, where the root node relaxation was unaffected (although other node relaxations can still be affected after branching), cut (1) increased the CPU times for two instances from 7.7 to 9.1 seconds and from 55.8 to 255.0 seconds, respectively, but decreased it for a third instance from 195 to 109.5 seconds. On average, cut (1) increased the overall effort as indicated in Table 1.

Since the best results were obtained by adding cuts (2) to the formulation, this refinement was included in the model for further experiments.

We also observe from Table 1 that the computational times to solve the roster generation problem vary widely. Many parameters of the formulation impact the CPU times but the computational times are mainly dependent on two of them that determine the number of integer variables: the number of weeks in the roster (nwr) and the number of cycles in the roster (ncr). Tables 2 and 3 present mean CPU times obtained on sets of instances of the problem having varying values of nwr and ncr. The general trend is an exponential increase of the computational times with respect to these two parameters.

	Mean CPU	Max CPU	Nb of problems			Mean CPU		
			>1s.	>10s.	>100s.	>1s.	>10s.	>100s.
No cuts	11.3	498.5	167	66	9	28.7	66.7	278.9
Cut (1)	12.7	423.9	143	73	14	37.9	70.5	240.1
Cuts (2)	2.1	121.2	75	12	1	11.4	55.4	121.2
Cuts (1) & (2)	12.8	385.7	162	84	21	33.5	61.9	184.7

Table 1: Number of problems solved and CPU times (in s.) by introducing cuts within the formulation using one-week patterns

nwr	CPU
4	0.09
6	0.12
8	0.72
9	0.13
10	0.34
12	11.48
16	1,022.64
20	1,640.15

Table 2: Mean CPU times (in s.) with respect to nwr for the formulation using one-week patterns

ncr	CPU
2	0.34
3	0.15
4	304.83
8	5,108.80

Table 3: Mean CPU times (in s.) with respect to ncr for the formulation using one-week patterns

4.1 Computational comparison with the formulation using daily variables

For the formulation using daily variables mentioned in Section 3.3, over the 432 tested instances, 360 were infeasible and, among the remaining 72 feasible instances, there was no case where this model required less time than the formulation using one-week patterns. For some particularly hard to solve instances (where ncr equals 4 and nwc equals 3 – hence, the rosters comprise 12 weeks overall), the required CPU time ranged from 1,498 s. to 7,972 s. with the daily variables model, whereas the CPU times ranged from 1 s. to 98 s. with the formulation using one-week patterns. We refer the interested reader to [29] for further details on this daily-variables model.

4.2 Practical use of the model

The formulation using one-week patterns reaches optimal solutions much faster than the formulation using daily variables - 10 to 20 times faster on average. Therefore, if multiple solutions are to be produced quickly, for example for the sake of comparison of different feasible rosters, the formulation using one-week patterns is well adapted.

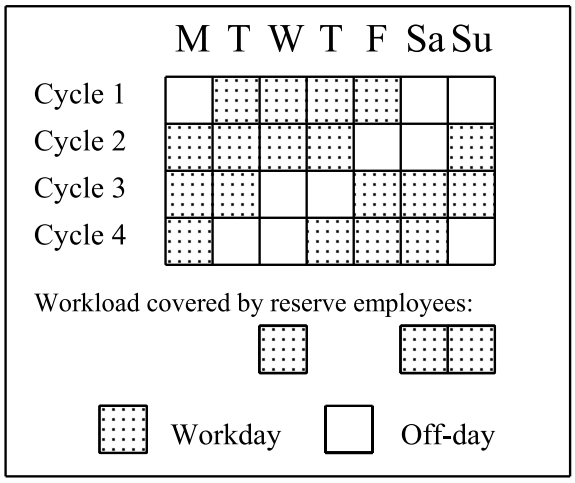

Figure 2: A valid roster for the formulation using one-week patterns

One should keep in mind that the formulation using one-week patterns specifies only the nature of each day of the roster as either a workday or an off-day. This has a significant impact on computational times but also has consequences on the usability of the rosters. For example, consider the roster presented in Figure 2. This roster has 10 off-days, one Saturday-Sunday off-day, and two weekend off-days. Its work stretches are three to five days long, and the rest periods are between one and three days. Hence, it is valid with respect to work regulations. However, the sequence of assigned shifts Morning $\rightarrow$ Night $\rightarrow$ Evening $\rightarrow$ Morning, and so on, one type for each work stretch, which is commonly implemented for three-shift rosters, cannot be respected here because the roster has five work stretches, which is not a multiple of three. On the other hand, the formulation using daily variables directly handles such restrictions on sequences of shifts.

The pattern-based formulation recognizes a large number of rosters as valid, but these rosters do not necessarily correspond to certain expectations of workforce managers. A way to resolve this issue would be to use one-week patterns that specify for each workday the type of shift that is worked. If we consider three shifts that alternate with each other according to the sequence Morning, then Night, and then Evening, and so on, the number of such patterns is exactly three times the number of "basic" patterns (indeed,

for each basic pattern, there are three possible choices for the first workday, and the nature of the following workdays derives from this initial choice). This new set of patterns is still of manageable size for the MIP formulation using patterns, and could be used instead of the original set of patterns to produce more precise rosters. Moreover, the pattern-based formulation can be useful to design other types of rosters, for example, rosters having only one or two different shifts.

Finally, we comment on handling infeasibilities in the generated solutions in practice. In such cases, we advise the user to try another computational run with a different number of weeks per cycle. This is acceptable from a practitioner's point of view since roster specialists know that some numbers of weeks per cycle fit better certain requirements that need to be covered (although there is no formal proof of this), and thus accordingly, they manually adapt the roster width to meet these requirements.

5 Summary and Conclusions

The research described in this paper on the rostering problem at SNCF has enabled the design of a model using one-week patterns, producing rosters where each day is defined as being either a workday or an off-day. As discussed in Section 4, it can be solved very fast and allows users to generate several rosters for the sake of comparison, or to obtain "long" rosters within reasonable computational times.

5.1 From theoretical models to an industrial tool

This study was initiated by SNCF to develop a decision-support system that would be available to all workforce managers of the railway company. A prototype of this system was designed based on the proposed formulation, and has been used online by several SNCF workforce managers via the company intranet. The system is basically composed of a set of dynamical Web pages. Once problem instances are submitted and optimal solutions are found, all relevant pieces of information are written in a database and displayed in a graphical manner on a Web page.

The feedback obtained from the users is very encouraging. The consensus is that the prototype is very helpful and saves a great deal of time. Also, the solutions that are proposed are sometimes quite different from the ones that workforce managers would have found manually, thus enlarging the field of possibilities. Of course, efforts are needed to improve the user interface and the reliability of the system, but the prototype proved the need and interest for such a decision-support system. The next step is the industrialization of the tool to make it available to all workforce managers of the company.

5.2 Recommendations for future research

Although the formulations designed in this study are capable of designing feasible rosters in acceptable times, the following features can be investigated to further improve the models:

- Use more specific patterns in which the shifts are additionally assigned to workdays, and introduce new sequence constraints in the formulation using one-week patterns to produce more detailed and ready-to-use rosters.

- Implement column generation to derive patterns, or use longer patterns, that are dynamically generated by column generation, to reduce computational times.

Acknowledgement

This work has been supported in part by the National Science Foundation under Grant No. DMI-0094462.

References

[1] K R Baker. Scheduling a full-time workforce to meet cyclic staffing requirements. *Management Science*, 20(12):1561–1568, 1974.

[2] K R Baker and M J Magazine. Workforce scheduling with cyclic demands and day-off constraints. *Management Science*, 24(2):161–167, 1977.

[3] J F Bard, C Binici, and A H DeSilva. Staff scheduling at the united states postal service. *Computers & Operations Research*, 30(5):745–771, 2003.

[4] J J Bartholdi III, J B Orlin, and H D Ratliff. Cyclic scheduling via integer programs with circular ones. *Operations Research*, 28(5):1074–1085, 1980.

[5] N Beamont. Using mixed-integer programming to design employee rosters. *Journal of the Operational Research Society*, 48(6):585–590, 1997.

[6] C Bentz. Conception des roulements pour le personnel de l'escale. SNCF Internship report (unpublished), 2002.

[7] E K Burke, P De Causmaecker, G Vanden Berghe, and H Van Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7(6):441–499, 2004.

[8] A Caprara, M Monaci, and P Toth. A global method for crew planning in railway applications. *Computer-Aided Transit Scheduling*, 505:17–36, 2001.

[9] A Caprara, P Toth, D Vigo, and M Fischetti. Modeling and solving the crew rostering problem. *Operations Research*, 46(6):820–830, 1998.

[10] B Cheang, H Li, A Lim, and B Rodrigues. Nurse rostering problems–a bibliographic survey. *European Journal of Operational Research*, 151(3):447 – 460, 2003.

[11] H Dawid, J König, and C Strauss. An enhanced rostering model for airline crews. *Computers and Operations Research*, 28(7):671–688, 2001.

[12] G De Pont. *Personalized crew rostering at Netherlands railways*. PhD thesis, University of Tilburg, The Netherlands, 2006.

[13] G Desaulniers, J Desrosiers, and M M Solomon. *Column generation*. Springer Science, New York, NY, 2005.

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

[15] A T Ernst, H Jiang, M Krishnamoorthy, B Owens, and D Shier. An annotated bibliography of personnel scheduling and rostering. *Annals of Operations Research*, 127(1):21–144, 2004.

[16] T Fahle, U Junker, S E Karisch, N Kohl, M Sellmann, and B Vaaben. Constraint programming based column generation for crew assignment. *Journal of Heuristics*, 8(1):59–81, 2002.

[17] R Freling, N Piersma, A P Wagelmans, and A van de Wetering. Rostering at a dutch security firm. Technical Report ERS-2001-37-LIS, Erasmus Research Institute of Management (ERIM), The Netherlands, June 2001.

[18] M Gamache, F Soumis, G Marquis, and J Desrosiers. A column generation approach for large-scale aircrew rostering problems. *Operations Research*, 47(2):247–262, February 1999.

[19] C A Glass and R A Knight. The nurse rostering problem: A critical appraisal of the problem structure. *European Journal of Operational Research*, 202(2):379 – 389, 2010.

[20] E D Gunes. Workforce scheduling. Technical report, Bilkent University, Ankara, Turkey, April 1999.

[21] IBM ILOG. IBM ILOG CPLEX Optimizer - High-performance mathematical programming solver for linear programming, mixed integer programming, and quadratic programming, 2012. See `www-01.ibm.com/software/integration/optimization/cplex-optimizer`.

[22] J König and C Strauss. Rostering-integrated service and crew efficiency. *Information Technology and Tourism*, 3(1):27–39, 2000.

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

[24] M Lohatepanont and C Barnhart. Airline schedule planning: Integrated models and algorithms for schedule design and fleet assignment. *Transportation Science*, 38(1):19–32, 2004.

[25] A Mercier and F Soumis. An integrated aircraft routing, crew scheduling and flight retiming model. *Computers and Operations Research*, 34(8):2251–2265, 2007.

[26] H E Miller, W P Pierskalla, and G J Rath. Nurse scheduling using mathematical programming. *Operations Research*, 24(5):857–870, 1976.

[27] G L Nemhauser and L A Wolsey. *Integer and Combinatorial Optimization*. John Wiley and Sons, New York, second edition, 1999.

[28] Quintiq. Rail planning and scheduling solution - planning for success in a changing market, 2012. Commercial brochure available online at `http://connect.quintiq.com/wfo-rail-rail-rostering-buna.html?keyword=\%2Brail\%20\%2Brostering\&matchtype=b\&creative=18828281748\&gclid=CKiUvJv8xrACFUZN4AodX1jwXQ`.

[29] F Ramond. Optimized rostering of workforce subject to cyclic requirements. Master's thesis, Virginia Tech, USA, October 2003.

[30] H D Sherali, M H Ramahi, and Q J Saifee. Hospital resident scheduling problem. *Production Planning and Control*, 13(2):220–233, 2002.

[31] H D Sherali and J C Smith. Improving discrete model representations via symmetry considerations. *Management Science*, 47(10):1396–1407, 2001.

[32] SNCF. *Human Resources Department – Réglementation du travail RH-0077*, Paris, France, February 2000.

[33] SNCF. *Human Resources Department – Réglementation du travail RH-0677, Instruction d'application du décret n°99-1161 du 29 décembre 1999*, Paris, France, December 2000.

[34] W Townsend. An approach to bus-crew roster design in London regional transport. *Journal of the Operational Research Society*, 39(6):543–550, 1988.

[35] P H Vance, C Barnhart, E L Johnson, and G L Nemhauser. Airline crew scheduling: A new formulation and decomposition algorithm. *Operations Research*, 45(2):188–200, 1997.