# CAHIER DU LAMSADE

Laboratoire d'Analyse et Modélisation de Systèmes pour l'Aide à la Décision
(Université Paris-Dauphine)
Unité de Recherche Associée au CNRS n° 825

## A HYBRID METHOD FOR THE GENERAL ROSTERING PROBLEM WITH A MIXED AND HOMOGENEOUS WORKFORCE

CAHIER N° 151
décembre 1997

Ariane PARTOUCHE [1]

received: July 1997.

[1] *LAMSADE, Université Paris-Dauphine, Place du Maréchal De Lattre de Tassigny, 75775 Paris Cedex 16, France – partouche@lamsade.dauphine.fr – Eurodécision : info@eurodecision.fr*

# Contents

Résumé-Abstract . . . . . i

**1 Introduction** **1**

**2 The General Rostering Problem** **3**

**3 A hybrid method for the general rostering problem** **5**

**4 Phase 1: An ILP model to select the shifts** **7**

**5 Phase 2: Evaluation of the workforce size and number of rosters to build** **8**

**6 Phase 3: Building rosters for each class** **10**
6.1 Assignment phase: an ILP model . . . . . 10
6.1.1 The assignment constraints . . . . . 11
6.1.2 Additional constraints . . . . . 11
6.1.3 Objective function . . . . . 13
6.2 Arrangement phase: a CLP approach . . . . . 14
6.2.1 The CLP formulation . . . . . 14
6.2.2 Introducing soft constraints . . . . . 15
6.2.3 Solving the problem . . . . . 17
6.3 Adjustment phase: a CLP model . . . . . 18

**7 Results and Conclusion** **20**
7.1 Evaluation of overall productivity . . . . . 20
7.2 Evaluation in terms of social satisfaction . . . . . 20
7.3 Evaluation of overall time process . . . . . 20
7.4 Conclusion . . . . . 21

**References** **22**

# Une Méthode Hybride pour le Problème Général de Construction de Grilles


## Résumé

Construire des grilles consiste à planifier les horaires de travail du personnel pour couvrir une demande opérationnelle tout en respectant un ensemble de règles plus ou moins flexibles. Cet article introduit le Problème Général de Construction de Grilles. Nous considérons une population homogène en termes de qualifications et mixte en termes de durées hebdomadaires de travail, affectée à des grilles cycliques multi-vacations. La complexité du problème impose une méthode en trois phases où s'associent différentes techniques de résolution.

Dans une première phase, nous utilisons un modèle de couverture d'ensembles pour sélectionner, dans de larges ensembles de vacations, celles qui couvrent une courbe de charge à moindre coût. Ensuite, nous déduisons l'effectif nécessaire pour effectuer ces vacations et le nombre de grilles à construire. Enfin, nous associons Programmation Linéaire en Nombres Entiers et Programmation Par Contraintes pour placer les vacations sélectionnées et les jours de repos dans des grilles tout en vérifiant des contraintes d'enchaînement et de durée de travail hebdomadaire.


*Mots-clés* : grilles de travail, construction de vacations, Programmation Linéaire en Nombres Entiers, Programmation Par Contraintes


# A Hybrid Method for the General Rostering Problem for a Mixed and Homogeneous Workforce


## Abstract

Rostering is the problem of scheduling manpower resources to meet operational demands while satisfying a variety of work policy constraints and goals. This paper introduces the General Rostering Problem for a workforce composed of full-time and part-time employees that rotate on cyclic rosters, made of multiple shift types. The complexity of this problem forces a decomposed process combining different solution techniques.

We first use a set-covering model to select, among large sets of shifts, the ones that cover fluctuant requirement curves with a minimal cost. Then, we deduce the necessary workforce size and the number of rosters to generate. Finally, we combine Integer Linear Programming and Constraint Logic Programming to assign the selected shifts and days-off into rosters so as to respect working constraints including no isolated days-off, a maximum of $m$ consecutive shifts, a minimum of $\theta$ hours rest between two consecutive shifts and an average working time corresponding to the weekly working rate of the workforce.


*Key-words*: rostering, shift scheduling, Integer Linear Programming, Constraint Logic Programming

# 1 Introduction

Most service delivery systems have to face a demand varying from one period to the next within the day and from one day to the next across the week. For such services, where the operational span frequently extends beyond the employees working time, efficient personnel scheduling is essential to achieve productivity goals and to provide adequate service. Over the last twenty years, labor scheduling problems have motivated a large research effort in many domains including hospital nurses (Warner, 1976; Smith, 1976; Siferd and Benton, 1992; Berrada, 1993), bus drivers (Martello and Toth, 1986; Blais *et al.*, 1990), postal employees (Malhotra *et al.*, 1992), telephone operators (Segal, 1974; Keith, 1979; Henderson and Berry, 1976, 1977), and airport ground crew (Holloran and Byrn, 1986; Chew, 1991; Partouche, 1996).

**The general rostering problem** is one of the most general labor scheduling problem since it includes shift scheduling, days-off scheduling and tour scheduling. The goal of rostering is to generate schedules that meet staffing requirements and respect working rules. The planning horizon of the schedules, the level of flexibility allowed and the diversity in the working constraints lead to a large range of problems.

The classical approach for rostering is to determine first a lower bound on the workforce size so as to satisfy constraints and then to propose an optimal algorithm which constructs rosters using exactly this necessary workforce.

First works on rostering considered only *single shift problems* assuming thus a constant demand all day long. This problem corresponds to the days-off scheduling problem, i.e., the assignment of employees to work-days and days-off on a planning horizon. Brownel and Lowerre (1976), followed by Lowerre (1977), considered the same requirement on each week-day and a lower requirement on weekends and determined the minimum workforce size under different alternative days-off policies: two days-off per week, two adjacent days-off each week, every other weekend off and four days-off every two weeks, ... Baker and Magazine (1977) introduced a cyclic dimension by assuming that individual schedules as well as requirements will repeat over an extended period of time. Baker *et al.* (1979) examined the case having any $A$ out of $B$ weekends off whereas Burns and Carter (1985) supposed a variable demand from day to day.

Some other works integrate a hierarchical workforce (Emmons and Burns, 1991; Hung, 1994; Narasimhan, 1996), i.e., workers are classified into qualification categories and a higher qualified can substitute for a lower one but not vice versa.

When services operate around the clock or when the demand for service fluctuates during the day, multiple shift types must be scheduled (a shift type is defined by a start time, an end time and placement of breaks), leading so to *multiple shift rostering problems.* Multiple shift problems can be characterized by the diversity of shift types allowed, yielding to more or less flexibility. The shift diversity leads to consider restrictions on the changing

of shifts from one day to the next (see Koop (1988)). The main cases of shift change constraints are:

- when a change in shift types can only occur after a day-off (*chain shift change constraint*);
- when there exists an ordering of the shift types such that a shift can only be followed by a shift type with a greater rank in the list (*ordered shift change constraint*).

Several researches only considered three labelled shifts: day, evening and night. This means that requirements may vary during the day, but only on every eight-hour periods. For instance Smith (1976) scheduled nurses with three 8-hours shift types. The shift change constraint is ordered (night, day, evening), that is a minimum of 16 hours rest between two consecutive shifts is imposed and at least 24 hours rest between any shift change.

Laporte *et al.* (1980) introduced cyclic rosters with again three shift types and imposed a chain shift change constraint. They also consider constraints on position of days-off, number of week-ends off and maximum work stretch (i.e., maximum number of consecutive shifts). These conditions enable to explicit all possible sequences of work days and days-off, and to assign a degree of desirability to each one. The problem is then modeled as a linear program in which the total desirability is maximized subject to covering exactly the workload. With the same assumptions, Burns and Koop (1987) proposed an algorithm that combine 'minischedules', sets of weekly assignments of shifts and days-off, to form a roster. Both approaches, since they need to explicit all feasible sequences, would be invalidated whenever flexibility in shifts increases: more shift types and less constrained shift changes.

More recently, Jacques (1993) and Partouche (1996) presented a constraint programming approach to solve a rostering problem considering three shift types and ordered changing shift constraint.

If we can find in the literature many works on the determination of workforce lower bounds and several algorithms to tackle the rostering problem, they are essentially limited to very few shift types and most of them can only be applied to very specific situations. Khoong *et al.* (1994) constitute one of the first work done on general models. A large diversity in shift types is allowed to cover a fluctuant curve of requirements. They present a generic manpower rostering toolkit based on a decomposition of the general rostering problem. The first subproblem is the determination of shifts to meet the requirements. Then, they position days-off into the roster and finally assign the shift types to a particular location in the roster. If the complete method is general enough to be applied in many operational contexts, the heuristic for shift assignment (Khoong, 1993) assumes that the number of shift types is lower than the workforce size and that shift change constraint is ordered. Besides, considerations concerning the total number of hours worked are not debated.

We propose in this paper a complete method for the cyclic rostering of a mixed workforce, which was not envisaged by Khoong *et al.*. We also admit a large number of shift types so as to cover a very varying requirement curve with an excellent productivity. To solve the rostering problem, also including constraints on the total working time, we combine linear programming with constraint logic programming.

The paper is organized as follows. In section 2, we describe the general rostering problem that we consider. Section 3 presents an original approach to decompose a very complex problem into subproblems easier to solve. Sections 4, 5 and 6 propose models and hybrid solution approaches for each subproblem. Finally, some conclusions are provided in section 7.

# 2 The General Rostering Problem

In this paper, we examine the problem of scheduling working hours and days-off in seven-day-a-week operations. We assume that the staff demands and scheduling constraints are the same every week, i.e., are cyclic with a cyclic length of one week. Hence, seven daily requirement curves represent, for each planning period, the need for employees. Our problem belongs thus to a general class of cyclical workforce allocation problems, as defined by Baker (1976). Depending on the problem context, the planning period, also called 'period', can be several hours or ten minutes long.

The requirements are covered by a homogeneous workforce, i.e., all workers have the same qualifications. We consider in the following that the workforce may be mixed, i.e., composed of a full-time and various part-time working classes and that the relative proportion of each class is constrained. Of course, a workforce composed of only one class can be considered as a special case of our problem.

Each workforce class is characterized by a weekly working rate and some specific working rules. For example, full-time workers work 39 hours per week, half-time workers work $\frac{39}{2}$ hours per week, and so on for other classes. Working rules include the working legislation and rules specific to the organization. These rules concern:

- the shifts: minimum and maximum duration, number and position of breaks, ...
- the days-off: average number of days-off on a given horizon, position on week-days and on weekends, minimum number of consecutive days-off,...
- the sequence of shifts and days-off: maximum number of consecutive shifts, minimum time interval between two consecutive shifts, ...

A *tour*, also called 'shift pattern' (Khoong *et al.*, 1994) or 'shift assignment' (Bailey, 1985), is a sequence of shifts and days-off over a planning period stretch. We consider the planning period to be one week from monday to sunday. A *roster* is a set of tours. Rosters can be cyclic or individualized. A cyclic roster consists of an ordered list of tours that is rotated across the employees over time. Individual rosters consist of tours that are unique for each employee. We generate here cyclic rosters.

Table 1 represents an example of a cyclic roster of length 12, for 12 full-time workers. The first week, each employee $e$ works on tour $e$. The following week, employee $e$ will work on tour $e+1$ if $e < 12$ and on tour 1 if $e = 12$, and so on the next weeks. For each working day, start time and end time are precised. 'O' denotes a day-off. 'J–8' corresponds to a particular shift called *joker* for which the length is defined (8 hours here) but start time and end time are not already known. The justification of jokers will be given in section 5.

| T | Mon | Tue | Wed | Thu | Fri | Sat | Sun |
|---|---|---|---|---|---|---|---|
| 1 | 5:10–13:40 | 5:30–13:30 | 10:10–18:40 | J–8 | O | O | J–8 |
| 2 | 5:50–13:50 | 5:50–13:50 | 10:10–20:40 | O | O | J–8 | J–8 |
| 3 | 6:50–14:50 | 6:50–14:50 | O | O | J–8 | J–8 | 6:50–17:20 |
| 4 | 7:50–19:20 | O | O | J–8 | J–8 | 5:50–13:50 | 9:50–20:20 |
| 5 | O | O | 5:50–13:50 | 4:30–15:00 | J–8 | 7:10–15:10 | O |
| 6 | O | 10:30–18:30 | J–8 | 4:30–15:00 | 5:10–13:10 | O | O |
| 7 | 12:10–20:40 | 11:50–20:50 | J–8 | 8:30–20:00 | O | O | 9:50–20:20 |
| 8 | 13:50–21:50 | 15:30–23:30 | O | O | O | 7:10–15:10 | 10:10–21:40 |
| 9 | J–8 | O | O | O | 5:30–13:30 | 9:50–17:50 | 12:50–21:50 |
| 10 | J–8 | O | O | 4:30–16:00 | 6:30–14:30 | 15:10–23:10 | O |
| 11 | O | O | 10:50–20:20 | 10:50–20:20 | 9:10–20:10 | O | O |
| 12 | O | J–8 | 14:10–22:10 | 10:50–20:20 | 15:10–23:10 | O | O |

Table 1: Example of a full time roster

As each tour is assigned to one employee each week, altogether the shifts of all tours must cover the requirements for each period of each day in the week. Moreover, the roster length is bounded. This induces to generate several rosters per working class, the number of rosters depending on the size of the class. Not only each roster built must respect the working rules specific to the class but also there must be as much equity as possible among rosters of the same class.

Therefore, our problem consists in building one or several rosters per working class so as to meet the requirements each day, to satisfy the rules of the class and to ensure equity among rosters of the same class, while minimizing staffing costs.

# 3 A hybrid method for the general rostering problem

In recent years, Constraint Logic Programming appeared as an interesting alternative to Integer Linear Programming in tackling combinatorial optimization problems (VanHentenryck, 1989). Several researchers have compared or combined ILP with CLP for solving difficult problems (Smith *et al.*, 1995; Partouche, 1996; Darby-Dowman *et al.*, 1997; Jacquet-Lagrèze *et al.*, 1997). In particular, CLP proved to be more adapted than ILP to the rostering problem in terms of modeling and solving strategies, as shown in Partouche (1996). On the opposite, Jacquet-Lagrèze *et al.* (1997) who compared an implicit CLP model and a set-covering ILP approach to solve the shift scheduling problem, concluded that the latter was much more efficient.

The purpose here is not to oppose again ILP to CLP, but to propose a method where ILP and CLP are combined. Each approach will be used for the part of the problem where it is more efficient. The combination of the two techniques thus allow the resolution of large-scale rostering problems in real world contexts.

We propose to decompose the general rostering problem in three main phases, as showed in Figure 1.

### Phase 1: Shift scheduling

The first aspect of the problem is to determine the set (or sets) of shifts that will cover the requirement curves while minimizing staffing costs. Since several workforce classes combine to cover the requirements, we consider one set of alternate shifts, defined by the working rules, per class.

We admit that requirements are discontinuous, that is each day can be tackled separately. So we first solve the shift scheduling problem for each day of the week (Phase 1). The modeling of this problem as a particular set-covering has proved to be very efficient when solved by linear programming (Jacquet-Lagrèze *et al.*, 1997) and thus will be used here.

### Phase 2: Workforce size

Relying on resulting shifts as well as rules concerning working time and days-off, we determine for each class, the minimum necessary workforce and the number of days-off requested. For each class, the workforce size is then increased to integrate a certain rate of absenteeism. Finally we deduce the number of rosters to build in each class to assign every employee (Phase 2).

The supplementary workers, added to compensate the absences of some others, are not directly useful to cover the shifts selected in phase 1. Thus, we need to create for them some new shifts and some new days-off, both called *jokers*. The *joker-shifts* will serve to compensate the absence of an employee

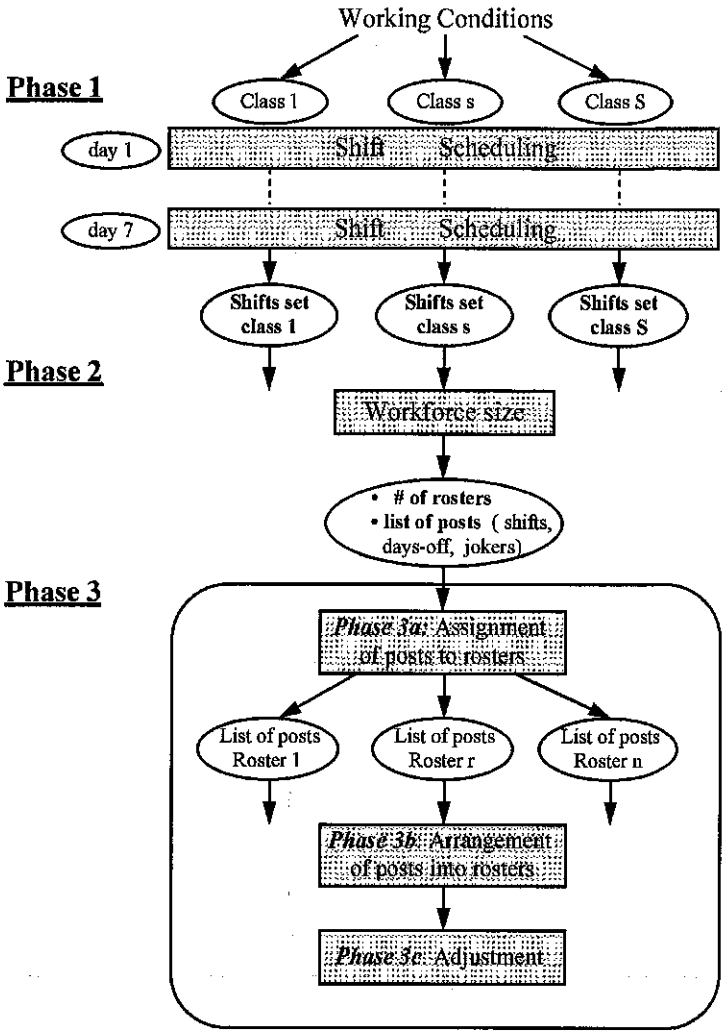


Figure 1: Decomposition into 3 phases

that was supposed to work on a 'real-shift'. The *joker-days-off* ensure that the planning for supplementary employees follow the same rules as the other employees of the class.

### Phase 3: Rostering

The previous phases have determined, for each class, the number of rosters to be generated as well as the list of shifts, the number of days-off and jokers to assign to the rosters. The problem for each class may be tackled independently. Frow now, we use the generic term *posts*, to represent shifts, days-off or jokers. The phase 3 consists then, for each class, in assigning posts to one roster and to one week of the roster.

In a previous study, we showed that CLP was well-adapted to generate a single roster (Partouche, 1996). Therefore, we first tried to tackle directly

this multi-rosters problem with a CLP model. We need then one variable per cell, i.e., per day, per week and per roster. The variable domain corresponds to the list of posts that could be assigned to this cell. If each roster contains $q$ cells and $l$ posts can be assigned to each cell, the size of the search space, for the $n$-rosters problem, is $O(l^{nq})$.

The order of the variables we considered, lead to construct all rosters almost sequentially. Therefore, and even if backtracks between rosters were possible, the choices made to build the first rosters often prevented from generating the last rosters in a reasonable time.

Instead of trying to develop efficient reduction heuristics or search strategies, we took advantage of the fact that each roster is an independent entity to partition the problem into several single rostering problems. If many solutions may be lost from this decomposition, it also permit to reduce the search space size to $O(nl^q)$.

The multi-rosters problem for one class (Phase 3) is thus decomposed into three subphases that combines ILP with CLP models.

**Phase 3a: Assignment** A first linear model shares posts between the rosters of the class. The assignment of a subset of posts to each roster has to be constrained enough to ensure that if a feasible solution exists for the $n$-rosters problem, a feasible solution also exists for the $n$ single-roster problems.

**Phase 3b: Arrangement** The remaining problem is then to arrange the posts in their roster using a constraint programming approach. In order to guarantee that a solution can always be found, some constraints are modeled as *soft binary constraints*, that is we tolerate these constraints not to be always respected but maximize the number of times they are. At this stage, we get rosters still containing jokers.

**Phase 3c: Ajustement** A last CLP model will turn some jokers into shifts and set their duration to adjust the roster average working duration to the weekly working rate of the class.

# 4 Phase 1: An ILP model to select the shifts

In a first phase, we need to determine, for each day of the week, the best set of shifts operated by the mixed workforce, that covers the requirements and minimizes costs. Our model is based on a set covering formulation initially suggested by Dantzig (1954). As in the Dantzig model, the variables $x_j$ correspond to the number of employees assigned to shift $j$, included in a set of feasible shifts. However, since we consider a mixed workforce, we define a particular set $J_s(s \in \mathcal{S})$, $\mathcal{S}$ being the set of classes.

An major component of productivity enhancement is the flexibility in the shifts definition (possible lengths, various start times, ...). Hence, some

given shifts (for instance a 6-hours length shift) could be assigned to several working classes (full-time and different part-times in our example). To avoid the sharing of such shifts among the classes when they heve been selected, we generate them once for each class it could be assigned to. The same shift will thus appear in several sets $J_s$. Therefore, the different sets of shifts form a partition and the output of this phase is expressed independently for each class.

The set-covering model is described as follows:

$$\min \quad \sum_{s\in\mathcal{S}} \sum_{j\in J_s} c_j x_j \tag{1}$$

subject to :

$$\sum_{s\in\mathcal{S}} \sum_{j\in J_s} a_{ij} x_j \geq w_i, \qquad \forall i \in \mathcal{I} \tag{2}$$

$$b^s_{min} \leq \sum_{j\in J_s} x_j \leq b^s_{max}, \qquad \forall s \in \mathcal{S} \tag{3}$$

$$\delta^s_{min} \leq \sum_{j\in J_s} x_j \delta_j \leq \delta^s_{max}, \qquad \forall s \in \mathcal{S} \tag{4}$$

$$x_j \geq 0, \quad \text{integer}, \qquad \forall j \in J_s, \quad \forall s \in \mathcal{S} \tag{5}$$

where $c_j$ is the cost of one employee assigned to shift $j$, $\delta_j$ is the duration of shift $j$, $w_i$ are the requirements for period $i \in \mathcal{I}$, set of periods of the day and $a_{ij} = 1$ if period $i$ is covered by shift $j$, 0 otherwise. The sum of selected shifts from all sets must cover the requirements (Constraints (2)). Additional constraints force a certain shift structure that will follow the conditions concerning the workforce repartition between classes and also anticipate the roster construction: constraints (3) bound the workforce size for each class and constraints (4) bound the total shift duration for each class. We suppose that these bounds are chosen not to lead to infeasibility.

This NP-complete problem can be solved almost optimally by a heuristic rounding the relaxed solution. Once the continuous solution found, we set all the variables whose value is zero and limit the domain of the others (integer or not) to +/- 2. Then we start a branch and bound procedure. The algorithm is stopped when the first integer solution is reached (for more details, see Jacquet-Lagrèze *et al.* (1997)).

# 5 Phase 2: Evaluation of the workforce size and number of rosters to build

In this phase, we use the total amount of hours to be worked and several working rates, to calculate the necessary workforce size of each classe and to decide the number of rosters to build. The integration of absenteeism

introduces enough flexibility to enable the respect of all working policies, while generating rosters in the next phase.

### Workforce size and days-off

The shift scheduling phase has determined, for each day $d \in \mathcal{D} = \{1, \ldots, 7\}$, the number of employees assigned to shift $j$ denoted by $x_j^d$. Let $V_s^d$ be the corresponding number of shifts that has to be performed by each working class $s$, $V_s^d = \sum_{j \in J_s} x_j^d$. From now, we can then treat each class independently.

If we add the duration of all selected shifts on the seven days, we get the total number of hours to be worked in a week by the class. Since this class corresponds to a given weekly working duration ($\Delta^s$ in hours), and since the workforce size can't be lower than the number of shifts to execute each day, we deduce the number of workers, denoted by $W_s^1$, necessary to operate the shifts:

$$W_s^1 = \max\left(\left\lceil \frac{\sum_{d=1}^{7} \sum_{j \in J_s} x_j^d \delta_j}{\Delta^s} \right\rceil, \max_{d \in \mathcal{D}} V_s^d\right) \qquad \forall s \in \mathcal{S} \tag{6}$$

where the symbol $\lceil . \rceil$ is used for the next greatest integer function. Since we know the number of selected shifts each day, we can deduce from the number of necessary workers, the number of days-off each day for each class ($O_s^d$):

$$O_s^d = W_s^1 - V_s^d, \qquad \forall d \in \mathcal{D}, \forall s \in \mathcal{S} \tag{7}$$

### The integration of absenteeism

To ensure that the requirements will be met in spite of absenteeism, represented by a certain rate $\tau^s$, we must increase the necessary number of workers. The needed workforce size then becomes $W_s$:

$$W_s = \left\lceil \frac{W_s^1}{1 - \tau^s} \right\rceil, \qquad \forall s \in \mathcal{S} \tag{8}$$

Note that $W_s$ corresponds also to the number of tours that have to be generated.

As explained in section 3, we create jokers for the supplementary workers. The number of jokers to generate each day ($J_s^d$), including joker-shifts and joker-days-off, is expressed by:

$$J_s^d = W_s - (V_s^d + O_s^d), \qquad \forall d \in \mathcal{D}, \forall s \in \mathcal{S} \tag{9}$$

### Number and length of the rosters

The workforce size of each class $s$ is divided into $n_s$ rosters. For each roster $r \in \mathcal{R}_s = \{1, \ldots, n_s\}$, the length $l_r$ corresponds to the number of weeks after

which a worker operates the same schedule. This period is limited by the law (bounds denoted by $L_{min}$ and $L_{max}$). We thus have to define the number of rosters to generate per class and the length of each roster such that:

$$\sum_{r=1}^{n_s} l_r = W_s \quad \text{with} \quad L_{min} \leq l_r \leq L_{max}, \quad \forall s \in \mathcal{S},\ \forall r \in \mathcal{R}_s \qquad (10)$$

It appears clearly that several solutions are possible. The scheduler may decide to construct either few long rosters or many short ones or any solution in between. This flexibility enables to construct personalized rosters for some employees of a class that need particular working conditions: for instance, some people prefer working early shifts whereas some other ones prefer late shifts, some women do not want to work on wednesdays because their children do not go to school,... Besides these specific employees, all other workers of a same class are assigned to several rosters, well-balanced in terms of the features that make a roster more or less difficult (number of very early mornings, of very late evenings, number of days-off,...).

# 6 Phase 3: Building rosters for each class

We now consider a single class. From the two previous phases, we know exactly the list of shifts, the number of days-off and the number of jokers to be sequenced as well as the number of rosters to be generated. As we explained in section 3, building all rosters with a direct CLP model is illusory and we need to divide the multi-rosters problem into several single roster problems. Phase 3 is decomposed into three sub-phases.

## 6.1 Assignment phase: an ILP model

The first subphase is to assign each post $p$ (shift, day-off or joker) to a particular roster (see Figure 2).

By reducing thus the search space, we also lose many alternate solutions. Hence, this assignment must be conceived to ensure that, if a feasible solution exists for the multi-roster problem, a solution will still exist after the decomposition. Moreover, we strive for equity between rosters, on criteria such as the number of morning shifts, the number of night shifts, the number of days-off each day.

The notations defined in section 5 for a certain class $s$ will now be used without the index $s$: let $W$ be the workforce size, $n$ the number of rosters to build.

The shift set is partitionned, according to the start time of the shift, into three subsets: morning, day and evening. We then define $\mathcal{K}$, the set of post types including morning ($mor$), day ($day$), evening ($eve$), jokers ($jok$) and days-off ($off$).

Let:

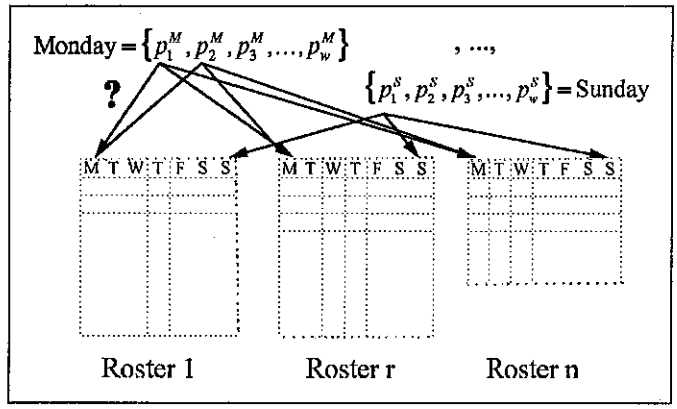


Figure 2: Assignment of posts to rosters

- $P_d^k$ be the set of posts of day $d$ and type $k$ with $|P_d^k| = w_d^k$, ($|.|$ denotes cardinality).
- $P_d$ be the set of posts of day $d$. $P_d = \bigcup_{k \in \mathcal{K}} P_d^k$ and $|P_d| = w_d = W$, $\forall d \in \mathcal{D}$.

### 6.1.1 The assignment constraints

The assignment problem can be easily formulated with a linear program. For all $p \in P_d$, for all $d \in \mathcal{D}$ and $r \in \mathcal{R} = \{1 \ldots n\}$, let the decision variables be:

$$x_{pr} = \begin{cases} 1 & \text{if post } p \text{ is assigned to roster } r \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Constraints (12) ensure that each post is assigned only once:

$$\sum_{r=1}^{n} x_{pr} = 1, \qquad \forall p \in P_d, \ \forall d \in \mathcal{D} \tag{12}$$

Constraints (13) ensure that, for any day, the number of posts assigned to any roster equals the roster length:

$$\sum_{p \in P_d} x_{pr} = l_r, \qquad \forall d \in \mathcal{D}, \forall r \in \mathcal{R} \tag{13}$$

### 6.1.2 Additional constraints

Besides the assignment constraints, the model must be completed by additional constraints that ensure:

1. feasibility
2. equity.

**Feasibility** While sharing posts among rosters, we must also ensure that the arrangement of posts in the roster they have been assigned to, will be feasible. Experiments had shown that one of the most constraining rules while arranging shifts into rosters concerns the changing of shifts, expressed here in the most general way: a minimum time interval $\theta$ must separate two consecutive shifts.

To be able to satisfy this rule in the next phase (3b), we add a set of constraints to balance repartition of shifts depending on their start time and end time. First we note that a working shift on day $d$, ending at time $h$, can be followed on day $d+1$ by another working shift, starting after $h+\theta$, a day-off or a joker. Second, we note that a shift ending at time $h' < h$ can be followed by any post that could follow the shift ending at $h$ or by a shift starting between $h'+\theta$ and $h+\theta$.

Therefore, we impose that, for each end time $h$ on day $d$ concerning $s > 0$ shifts, there must be, on day $d+1$, at least $s$ posts when adding shifts starting after $h+\theta$, days-off and jokers, that have not been already associated with shifts on day $d$ ending after $h$. To model this constraint, we need:

- to define $hsta(p)$ and $hend(p)$ to be start time and end time of post $p \in P$ respectively. If $p$ is a day-off or a joker, by convention $hsta(p) = hend(p) = 0:00$ AM ;
- to rank in increasing order the end times of posts in $P_d$ denoted by $h_d^0, h_d^1, \ldots, h_d^{max}$. $h_d^0$ corresponds to days-off and jokers, $h_d^1$ to the earliest shift end time on day $d$, $h_d^{max}$ to the latest.

The constraints are then written in a recurrent way:

For all $r \in \mathcal{R}$, for all $d \in \mathcal{D}$,

for finish time $h_d^i = h_d^{max}$

$$\sum_{p \in P_d | hend(p) = h_d^{max}} x_{pr} = \sum_{p \in P_{d+1} | hsta(p) \geq h_d^{max} + \theta} x_{pr} + \sum_{p \in P_{d+1}^{off}} x_{pr} + \sum_{p \in P_{d+1}^{jok}} x_{pr} - rest_{h^{max}}^d \tag{14}$$

for all finish time $h_d^i \neq h_d^{max}$ and $h_d^i \neq h_d^0$

$$\sum_{p \in P_d | hend(p) = h_d^i} x_{pr} = \sum_{p \in P_{d+1} | h_d^i + \theta \leq hsta(p) < h_d^{i+1} + \theta} x_{pr} - rest_{h^i}^d + rest_{h^{i+1}}^d \tag{15}$$

where $rest_{h^i}^d$ represents the number of shifts, days-off or jokers that are not necessary on day $d+1$ to follow the shifts on day $d$ finishing at $h_d^i$ and thus, can be used to follow earlier shifts.

These constraints are necessary to guarantee that a solution to the $n$ rostering problems exists but it is not sufficient because of the conjonction of all constraints.

**Equity** The assignment phase should preserve as much equity as possible between rosters corresponding to a same class. For this purpose, we add *soft constraints*, that balance the repartition of each type of post among rosters, proportionally to their length. This balance should be achieved both at a day level (constraints (16)), ensuring a fair repartition of the numbers of posts of type $k$ on day $d$ among all rosters and at an overall level (constraints (17)), ensuring a fair repartition of the total numbers of posts of type $k$ among all rosters.

At the day level, we set:

$$\sum_{p \in P_d^k} x_{pr} = \frac{w_d^k}{W} \times l_r, \qquad \forall d \in \mathcal{D}, \forall r \in \mathcal{R}, \forall k \in \mathcal{K} \tag{16}$$

and at the overall level:

$$\sum_{d=1}^{7} \sum_{p \in P_d^k} x_{pr} = \frac{\sum_{d=1}^{7} w_d^k}{W} \times l_r, \qquad \forall r \in \mathcal{R}, \forall k \in \mathcal{K} \tag{17}$$

Since the fraction of posts of type $k$ that is to be assigned to a roster is not necessarily an integer, we introduce *goal variables* that represent the slack to minimize:

$$\sum_{p \in P_d^k} x_{pr} + s_{dr}^{k-} - s_{dr}^{k+} = \frac{w_d^k}{W} \times l_r, \qquad \forall d \in \mathcal{D}, \forall r \in \mathcal{R}, \forall k \in \mathcal{K} \tag{18}$$

$$\sum_{d=1}^{7} \sum_{p \in P_d^k} x_{pr} + s_r^{k-} - s_r^{k+} = \frac{\sum_{d=1}^{7} w_d^k}{W} \times l_r, \qquad \forall r \in \mathcal{R}, \forall k \in \mathcal{K} \tag{19}$$

where $s_{dr}^{k+}, s_{dr}^{k-}$ correspond to positive and negative slack variables for the repartition of posts of type $k$ on day $d$ for roster $r$, and $s_r^{k+}, s_r^{k-}$ correspond to positive and negative slack variables for the overall repartition of posts of type $k$ for roster $r$.

### 6.1.3 Objective function

Unlike in the classical assignment problem, there is no cost related to the main decision variables $x_{pr}$ since there is no preference or objection for the assignment of a post to a particular roster. The objective represents then a weigthed sum of the goal variables defined in the additional constraints:

$$\min \sum_{k \in \mathcal{K}} \sum_{r=1}^{n} \sum_{d=1}^{7} c_1^k (s_{dr}^{k+} + s_{dr}^{k-}) + \sum_{k \in \mathcal{K}} \sum_{r=1}^{n} c_2^k (s_r^{k+} + s_r^{k-}) \tag{20}$$

where $c_1^k$ and $c_2^k$ correspond to the relative importance of a good repartition of posts of type $k$. Very different costs may permit a lexicographic ranking of the different post types repartition according to the scheduler priorities.

## 6.2 Arrangement phase: a CLP approach

In the previous phase we have assigned posts to rosters. Therefore, we can consider independently each roster with its corresponding 7 sets of posts. Each problem is thus a single roster problem which is solved by a CLP approach.

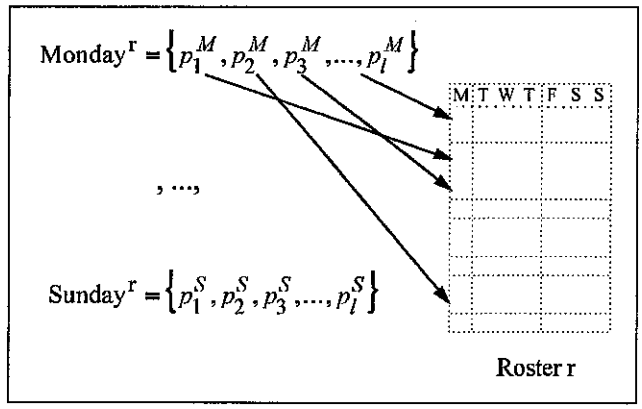


Figure 3: Arrangement of a posts subset into a roster

Unlike in LP, there is currently no standard representation for CLP problems (see Darby-Dowman *et al.*, 1997). We will describe our model as a combination of algebraic form, similar to ILP models, within logic syntax that CLP enables to model.

Considering a roster of length $l$, $t \in T = \{1 \dots l\}$ corresponds to a tour of the roster, i.e., a week. For each day $d \in \mathcal{D}$, a set of $l$ posts has been assigned to this roster. The problem is now to assign each post, which concerns a particular day, to a certain tour. This problem can also be seen as ranking the $l$ posts of each day in a particular order such that all rules concerning the succession of shifts and days-off over the tours are respected (see Figure 3). We consider in the following some typical constraints that are often met in rostering problems.

### 6.2.1 The CLP formulation

The principal variables $y_t^d$ represent the index of post of day $d$ assigned to the tour $t$. The domain of each $y_t^d$ is the set of post indices:

$$y_t^d \in \{1, \dots, l\} \qquad \forall d \in \mathcal{D}, \forall t \in T \tag{21}$$

The constraints stating that there must be exactly one post per day and per tour are automatically satisfied by any solution to the problem since each variable $y_t^d$ must have exactly one value.

Logic programming languages provide some high level constraints called *global constraints*. These constraints embed a conjunction of elementary con-

straints and a constraint handling method for them. In particular, the constraints that all posts of day $d$ must be assigned to a particular tour and only one, can be expressed by a single constraint:

$$y_1^d, y_2^d, \ldots, y_l^d \text{ are all different}, \qquad \forall d \in \mathcal{D} \tag{22}$$

This global constraint abstract the modelling and solving of a set of $\frac{l \times (l-1)}{2}$ binary disequation constraints.

Legal rules impose at most $m$ consecutive working days. This constraint can be interpreted saying that there must be at least one day-off on $m+1$ consecutive days. We need here to introduce new constrained 0-1 variables, $o_t^d = 1$ if there is a day-off on day $d$ tour $t$, 0 otherwise. The relationships between these variables and the principal ones are specified by the $7 \times l$ boolean constraints:

$$o_t^d = 1 \iff \begin{cases} y_t^d = p \text{ and, in the set of posts of day } d, \\ \text{the index } p \text{ corresponds to a day-off} \end{cases} \tag{23}$$

As soon as $y_t^d$ will be instantiated, i.e., gets its value, the value of $o_t^d$ will be deduced. On the other hand, if $o_t^d$ is instantiated first, the domain of $y_t^d$ will be reduced. The order in which the variables are instantiated is a main part of the searching strategy that is developed to solve a CLP model (see 6.2.3).

The maximum-consecutive-working-days constaints are thus expressed by:

$$\sum_{h=0}^{m} o_t^{d+h} \geq 1, \quad \forall d \in \mathcal{D}, \ \forall t \in T, \tag{24}$$

Since rosters are cyclic, note that $o_t^{d+h}$ becomes $o_{t+1}^{d+h-7}$ when $(d+h > 7$ and $t < l)$ or $o_1^{d+h-7}$ when $(d+h > 7$ and $t = l)$.

### 6.2.2 Introducing soft constraints

Scheduling rules to build a roster can be divided in two categories: the hard ones that must always be satisfied (classically law rules) and the flexible ones that can be relaxed (generally rules specific to the organisation that increase employees satisfaction). Whereas *hard constraints* may lead to infeasability, this is not the case with *soft constraints*. Therefore, besides the real flexible rules, it might be useful also to model some 'difficult' hard rules as soft constraints so as to get a first imperfect roster. This roster will be manually corrected afterwards to make it statutory.

In the CLP literature, flexibility in constraints may express the possibility to violate some constraints (Freuder and Wallace (1992)) or to get distance from ideal solutions (Fargier *et al.* (1995); Fortemps (1997)). We consider here the former conception, that is binary soft constraints that can be violated but are to be respected as often as possible.

We introduce two sets of soft constraints. The first one concerns the minimum time interval $\theta$ between two consecutive shifts. Actually, this constraint is hard in the reglementation but we model it as soft to ensure that a roster can be generated. Remember that the additional constraints in the assignment phase were necessary but not sufficient to ensure that the minimum time interval between two consecutive shifts can always be respected. The second set of soft constraints models the unpopularity of isolated days-off.

To deal with the minimum-time-interval constraints, we introduce new 0-1 variables: $c_t^d = 1$ if the constraint is satisfied between the post on day $d$ tour $t$ and the following day, 0 otherwise. The constraint is satisfied if the two consecutive days are worked and the time interval between the two shifts is more than $\theta$ or if, at least one of the two days is a day-off. Since jokers do not correspond to shifts or days-off yet, they are considered here as days-off and we suppose that they can follow any shift. The minimum-time-interval constraint will be considered again while defining start times and end times of joker-shifts.

To define precisely the $c_t^d$ variables, we thus need variables that indicate if the day $d$ on tour $t$ is assigned to a joker or not. The constraint linking the new variables to the original ones are:

$$j_t^d = 1 \iff \begin{cases} y_t^d = p \text{ and, in the set of posts of day } d, \\ \text{the index } p \text{ corresponds to a joker} \end{cases} \tag{25}$$

We also need $hsta(y_t^d)$ and $hend(y_t^d)$, the start time and end time respectively, of the post assigned to day $d$ on tour $t$.

Finally, for all $t \in T$ and for all $d \in \mathcal{D}$

$$c_t^d = 1 \iff \begin{cases} hsta(y_t^{d+1}) - hend(y_t^d) \geq \theta \\ \text{or } o_t^d = 1 \text{ or } o_t^{d+1} = 1 \\ \text{or } j_t^d = 1 \text{ or } j_t^{d+1} = 1 \end{cases} \tag{26}$$

Note that the couple of index $(t, d+1)$ becomes $(t+1, 1)$ when $d = 7$ and $t \leq l$ or $(1,1)$ when $d = 7$ and $t = l$.

Maximizing the number of times the constraints are satisfied leads to the following objective:

$$\max \sum_{t \in T} \sum_{d \in \mathcal{D}} c_t^d \tag{27}$$

The second kind of soft constraint concerns the number of consecutive days-off and more particularly the isolated days-off[1]. We thus minimize the

[1]It hardly appears that the minimum number of consecutive days-off is more than 2 but it could also be modeled as a soft constraint.

number of times isolated days-off appear. Here again we introduce new 0-1 variables $c_t'^d$ that indicate if the day $d$ on tour $t$ is an isolated day-off, i.e., if $d$ is a day-off and both $(d-1)$ and $(d+1)$ are worked.

The constraint variables are defined by:

$$c_t'^d = 1 \iff o_t^d - o_t^{d-1} - o_t^{d+1} > 0, \qquad \forall t \in T, \forall d \in \mathcal{D} \tag{28}$$

Again, special cases must be distinguished when $d = 1$ or $d = 7$, and when $t = 1$ or $t = l$. The resulting objective will then be:

$$\min \sum_{t \in T} \sum_{d \in \mathcal{D}} c_t'^d \tag{29}$$

### 6.2.3 Solving the problem

The key to success of CLP in tackling combinatorial search problems involving constraints lies in the combination of:

1. constraints handling methods, i.e., constraint propagation methods that aim at reducing the search space and
2. search heuristics, i.e., rules for choosing which variable to consider next and which value to assign to it.

**Constraints handling methods** The constraints described above are sufficient to model the problem but we add some simple constraints to reduce the search space. In particular, we bound, per tour, the number of days-off, as well as the number of jokers and also the total working time.

**Search heuristics** Since all variables described above derive from the principal variables $y_t^d$, it is sufficient to instantiate those to arrive at a solution. However, it appears to be more efficient to place the days-off first, i.e., to instantiate first the $o_t^d$ variables that reduce the domains of $y_d^t$. The variables $y_t^d$ are considered next and then the constraint variables $c_d^t$. Value of other variables are deduced by constraint propagation. This search strategy corresponds to the decomposition suggested by Khoong *et al.* (1994) which position days-off into roster before assigning shift. However, in our approach, a bad assignment of days-off can be backtracked since both shifts and days-off are instantiated in the same model.

The domain of variables $y_t^d$ is described in the following order: first the shifts ranked in increasing order of their start times, then the days-off, then the jokers. The choice of the value corresponds to the first available value in the remaining domain.

**Optimization** The optimization in CLP is an iterative process. An additional constraint bounds the criterion by an initial value. If no solution can be found, the value is modified so as to find a first solution. When a solution is found, the value is increased by one (decreased by one) in case of maximization (in case of minimization) and a new solution is searched; when no better solution can be found, the process stops. The last solution found corresponds to the optimum.

In our case, the two criteria, corresponding to the two types of soft constraints, are ranked in a lexicographic order (minimum-time-interval constraint as a main criterion and isolated day-off constraint as a secondary criterion). The optimization process is started with initial values of both criteria set to zero (no soft constraint violated).

## 6.3 Adjustment phase: a CLP model

At this stage, rosters are almost built. However, constraints concerning the total working time have not been considered yet and jokers are still not defined. The purpose of this last phase is precisely to select which jokers are to be transformed in working shifts and determine their durations so that the average working time of the roster corresponds *exactly* to the weekly working time of the class. The remaining jokers become days-off. While changing jokers into shifts or days-off, all rules concerning the sequencing of shifts and days-off, that were considered in the previous phase, must still be taken into account.

$\delta_t^d$ denotes the duration of the post assigned to day $d$ on tour $t$. When the post is a shift or a day-off, its duration is known. However, when the post is a joker, its duration is to determine.

The principal variables of this model are thus $\delta_t^d$, when the post assigned to day $d$ on tour $t$ is a joker. The domain of the variables consists of the set of all possible shift durations from $\delta_{min}$ to $\delta_{max}$ and the value 0. If the value 0 is selected, this means that the joker becomes a day-off:

$$\delta_t^d \in \{[\delta_{min}, \delta_{max}], 0\} \quad \forall d \in \mathcal{D}, \forall t \in T \text{ such that } j_t^d = 1 \tag{30}$$

The constraints of this model are of three kinds. The first constraints impose that the average work duration of the roster equals the weekly working rate $\Delta$:

$$\sum_{t \in T} \sum_{d \in \mathcal{D}} \delta_t^d = \Delta \times l \tag{31}$$

The second type of constraints ensures that there is at least one joker-shift per day so that, any day, an absence could be compensated:

$$\sum_{t \in T | j_t^d = 1} \delta_t^d \geq \delta_{min}, \qquad \forall d \in \mathcal{D} \tag{32}$$

Some other constraints concern the placement of the 'new shifts' or 'new days-off' so as to respect the sequencing rules considered in phase 3b: maximum-consecutive-shifts, minimum-time-interval and no isolated days-off.

1. Since jokers were considered as shifts when we modeled the maximum-consecutive-shift constraint (24), transforming a joker into a shift cannot lead to a violation of this constraint.

2. Since joker-shifts are used to compensate an absence on a 'real shift', their start time and end time are not to be explicited here but only their durations. Thus, the minimum-time-interval constraint cannot be considered.

3. On the opposite, we must avoid generating a day-off while transforming jokers. Several cases are considered, and for instance:

- if a joker is isolated between two shifts, not to create an isolated day-off, it must become a shift, i.e., with a duration over $\delta_{min}$

$$\boxed{\text{SJS} \longrightarrow \text{SSS}}\ [2]$$

$$\text{if } \begin{cases} j_t^d = 1 & \text{and} \\ o_t^{d-1} = 0 & \text{and} \quad o_t^{d+1} = 0 \\ j_t^{d-1} = 0 & \text{and} \quad j_t^{d+1} = 0 \end{cases} \text{ then } \delta_t^d \geq \delta_{min} \tag{33}$$

- if two jokers are isolated between two shifts, not to create an isolated day-off, they must become either a couple of shift or a couple of days-off

$$\boxed{\text{SJJS} \longrightarrow \text{SSSS ou SOOS}}$$

$$\text{if } \begin{cases} j_t^{d-1} = 1 & \text{and} \quad j_t^d = 1 \\ o_t^{d-2} = 0 & \text{and} \quad o_t^{d+1} = 0 \\ j_t^{d-2} = 0 & \text{and} \quad j_t^{d+1} = 0 \end{cases} \text{ then } \begin{cases} \delta_t^{d-1} \geq \delta_{min} & \text{and} \quad \delta_t^d \geq \delta_{min} \\ & \text{or} \\ \delta_t^{d-1} = 0 & \text{and} \quad \delta_t^d = 0 \end{cases} \tag{34}$$

- if a joker is followed by a day-off which is followed by a shift, then the joker must become a day-off

$$\boxed{\text{JOS} \longrightarrow \text{OOS}}$$

$$\text{if } \begin{cases} j_t^d = 1 & \text{and} \quad o_t^{d+1} = 1 \\ o_t^{d+2} = 0 & \text{and} \quad j_t^{d+2} = 0 \end{cases} \text{ then } \delta_t^d = 0 \tag{35}$$

The problem here is only to find a feasible solution. The output of this last model is a complete roster where every start time and end time as well as breaks are clearly defined, except for jokers transformed into shifts.

[2] S denotes a shift, J a joker and O a day-off.

# 7 Results and Conclusion

This method was applied in a real world context for scheduling ground staff in an airport. The demand was described by 7 requirements curves from 4:30 am to 24:00 pm with a 10-minutes step. The staff was composed of three population classes: full-time, 3/4-time and half-time. For each class, a set of about 20,000 shifts, including up to two breaks, was proposed. The roster length had to be between 8 and 12 weeks. A maximum of $m = 6$ consecutive shifts was accepted. The time between two consecutive shifts had to be over $\theta = 11$ hours.

## 7.1 Evaluation of overall productivity

The productivity is defined by the ratio of the useful hours (sum of demands) and the paid hours (sum of the selected shift durations). A first level of productivity is calculated after the shift scheduling phase and appeared to be around 80%.

Thanks to the flexibility brougth by the integration of absenteeism and jokers, this productivity is not degraded during the following phases. As a matter of fact, jokers permit to adjust the average working time of rosters without generating any useless hour. Employee are thus paid exactly for the time they have to work, which is exactly the minimum required to cover the demands and to anticipate absenteeism. Besides, the supplementary workers, introduced to compensate absenteeism, allow all days-off and sequencing rules to be respected.

## 7.2 Evaluation in terms of social satisfaction

At the level considered here, rosters are not assigned to employees by name and thus individual preferences or unavailability are not integrated. The social satisfaction is thus reduced to the respect of soft constraints. As explained, soft constraints guarantee the problem feasibility, but the less soft constraints are violated, the better social satisfaction is.

On a complete test, made of 16 rosters, only one isolated day-off was unavoidable and none interval less than 11 hours between two shifts. This leads us to conclude that the method was bolted enough since the beginning to prepare a good situation while arranging posts in rosters.

## 7.3 Evaluation of overall time process

The whole problem was solved on a PC Pentium. The solver used for linear programming models is CPLEX (1995) and for constraint programming is ILOG Solver (1995).

The shift scheduling phase (Phase 1) needs in average 2 minutes per day to select the best sets of shifts. Phase 2 calculates the workforce size for the three

classes and the number of rosters to build almost immediately. The full-time employees were to be 135 people shared on 12 rosters; the 3/4-time employees were to be 24 people shared on 2 rosters and the half-time employees were to be 18 people shared on 2 rosters.

For each class, the construction of rosters (Phase 3) is independent. Computation time for the LP model to assign posts to rosters (Phase 3a) is always less than a minute. Solving the CLP models to arrange and adjust posts (Phases 3b and 3c) may be immediate or may last up to 1 minute for one roster. If a solution with every soft constraints satisfied can be found (always at least 11 hours between two shifts and no isolated day-off), then the solving process is very quick. Otherwise, it must be shown that the perfect solution does not exist before looking for one with a single constraint unrespected, and so on. The proof of the non-existence of a solution supposes that the tree search has been entirely considered and thus may require a certain time (however less than a minute).

Table 2 summarizes the overall computation time. All together, the overall process takes about 20 minutes. Note that the rosters finally built will be in function as long as the requirements stay the same, i.e., in the air context, one IATA season (about 6 months).

| | ***Phases*** | ***nb of passages*** | ***calc. time*** | ***Total*** |
|---|---|---|---|---|
| | shift scheduling | 7 (days) | 2 min | 14 min |
| | workforce size | 3 (classes) | $\epsilon$ | $\epsilon$ |
| full-time | assignment | 1 | 2 min | 2 min |
| | arrangement | 12 (rosters) | 1 to 11 sec | $<$ 1 min |
| | adjustment | 12 (rosters) | $\epsilon$ | $\epsilon$ |
| 3/4-time | assignment | 1 | 2 sec | 2 sec |
| | arrangement | 2 (rosters) | 13 to 43 sec | $<$ 1 min |
| | adjustment | 2 (rosters) | $\epsilon$ | $\epsilon$ |
| half-time | assignment | 1 | 2 min | 2 min |
| | arrangement | 2 (rosters) | 10 sec | $<$ 1 min |
| | adjustment | 2 (rosters) | $\epsilon$ | $\epsilon$ |
| ***Total*** | | | | 20 min |

Table 2: Overall process computation time

## 7.4 Conclusion

We have proposed in this paper an overall method for scheduling a mixed workforce on rosters. The decomposition into several subproblems, even if complicated, allowed us to build rosters observing all working rules and covering the requirements without deficit and with an excellent rate of productivity.

This method constitutes also an example of a successful hybrid approach that combines linear programming with constraint logic programming.

Further work could be done to try to adapt this method in the case of individualized rosters, i.e., without the assumption of the cyclic demand.

## Acknowledgments

This work was partly funded by the Esprit project 22165 Chic-2.

## References

Bailey, J. (1985), "Integrated days-off and shift personnel scheduling", *Computers and Industrial Engineering* 9/4, 395–404.

Baker, K. R. (1976), "Workforce allocation in cyclical scheduling problems: A survey", *Operations Research Quarterly* 27, 155–167.

Baker, K. R. and Magazine, M. J. (1977), "Workforce scheduling with cyclic demands and days-off constraints", *Management Science* 24, 161–167.

Baker, K. R., Burns, R. N., and Carter, M. W. (1979), "Staff scheduling with day-off and workstretch constraints", *AIIE Transactions* 11/4, 286–292.

Berrada, I. (1993), *Planification d'horaires du Personnel Infirmier dans un Etablissement Hospitalier*, Ph.D. thesis, Université de Montréal.

Blais, J., Lamont, J., and Rousseau, J. (1990), "The Hastus vehicle and manpower scheduling systems at the Société de Transport de la Communauté Urbaine de Montréal", *Interfaces* 20, 26–42.

Brownel, W. S. and Lowerre, J. M. (1976), "Scheduling of workforce required in continuous operations under alternative labor policies", *Management Science* 22, 597–605.

Burns, R. N. and Carter, M. W. (1985), "Work force size and single shift schedules with variable demands", *Management Science* 31/5, 599–607.

Burns, R. N. and Koop, G. J. (1987), "A modular approach to optimal multiple-shift manpower scheduling", *Operations Research* 35/1, 100–110.

Chew, K. L. (1991), "Cyclic schedule for apron services", *Journal of the Operational Research Society* 42, 1061–1069.

CPLEX (1995), *Using the CPLEX$^{TM}$ Callable Library and CPLEX$^{TM}$ Mixed Integer Library.* Cplex Optimization, Inc., Incline Village, Nevada, 4.0 edition.

Dantzig, G. B. (1954), "A comment on Edie's traffic delay at toll booths", *Operations Research* 2, 339–341.

Darby-Dowman, K., Little, J., Mitra, G., and Zaffalon, M. (1997), "Constraint logic programming and integer programming approaches and their collaboration in solving an assignment scheduling problem", *Constraints, An International Journal* 1/3, 245–264.

Emmons, H. and Burns, R. (1991), "Off-day scheduling with hierarchical worker categories", *Operations Research* 39, 484–495.

Fargier, H., Dubois, D., and Prade, H. (1995), "Problèmes de satisfaction de contraintes flexibles : une approche égalitariste", *Revue d'Intelligence Artificielle* 9/3, 311–354.

Fortemps, P. (1997), *Fuzzy Sets for Modelling and Handling Imprecision and Flexibility*, Master's thesis, Faculté Polytechnique de Mons.

Freuder, E. and Wallace, R. (1992), "Partial constraint satisfaction", *Artificial Intelligence* 58, 21–70.

Henderson, W. B. and Berry, W. L. (1976), "Heuristic methods for telephone operator shift scheduling: An experimental analysis", *Management Science* 22, 1372–1380.

Henderson, W. B. and Berry, W. L. (1977), "Determining optimal shift schedules for telephone traffic exchange operators", *Decision Sciences* 8, 239–255.

Holloran, T. J. and Byrn, J. (1986), "United airlines station manpower planning system", *Interfaces* 16, 39–50.

Hung, R. (1994), "Single-shift off-day scheduling of a hierarchical workforce with variable demands", *European Journal of Operational Research* 78, 49–57.

ILOG (1995), *Ilog Solver Reference Manual.* Ilog, Gentilly, France, 3.0 edition.

Jacques, P. (1993), "Solving a manpower planning problem using constraint programming", *Belgian Journal of Operations Research, Statistics and Computer Science* 33/4, 77–85.

Jacquet-Lagrèze, E., Montaut, D., and Partouche, A. (1997), "The shift scheduling problem: Different formulations and solution methods", Cahier du Lamsade 146, Université Paris-Dauphine.

Keith, E. (1979), "Operator scheduling", *AIIE Transactions* 11, 37–41.

Khoong, C. M. (1993), "A simple but effective heuristic for workshift assignment", *Omega* 21/3, 393–395.

Khoong, C. M., Lau, H. C., and Chew, L. W. (1994), "Automated manpower rostering: Techniques and experience", *International Transactions in Operational Research* 1/3, 353–361.

Koop, G. J. (1988), "Multiple shift workforce lower bounds", *Management Science* 34/10, 1221–1230.

Laporte, G., Norbert, Y., and Biron, J. (1980), "Rotating schedules", *European Journal of Operational Research* 4, 24–30.

Lowerre, J. M. (1977), "Work stretch properties for the scheduling of continuous operations under alternative labor policies", *Management Science* 23, 963–971.

Malhotra, M. K., Ritzman, L. P., Benton, W., and Leong, G. K. (1992), "A model for scheduling postal distribution employees", *European Journal of Operational Research* 58, 374–385.

Martello, S. and Toth, P. (1986), "A heuristic approach to the bus driver scheduling problem", *European Journal of Operational Research* 24, 106–117.

Narasimhan, R. (1996), "An algorithm for single shift scheduling of hierarchical workforce", *European Journal of Operational Research* 96, 113–121.

Partouche, A. (1996), "Comparaison de deux approches sur un problème de planification d'horaires : la construction de cycles de travail", Cahier du Lamsade 142, Université Paris-Dauphine.

Segal, M. (1974), "The operator-scheduling problem: A network-flow approach", *Operations Research* 22, 808–823.

Siferd, S. P. and Benton, W. C. (1992), "Workforce staffing and scheduling: Hospital nursing specific models", *European Journal of Operational Research* 60, 233–246.

Smith, B. M., Brailsford, S. C., Hubbard, P. M., and Williams, H. P. (1995), "The progressive party problem: Integer linear programming and constraint programming compared", in: Ilog, editor, *Ilog Solver and Ilog Schedule: first international users' conference*, Abbaye des Vaux de Cernay, France.

Smith, L. (1976), "The application of an interactive algorithm to develop cyclical rotational schedules for nursing personnel", *INFOR* 14, 53–70.

VanHentenryck, P. (1989), *Constraint Satisfaction in Logic Programming*, The MIT Press, Cambridge, Massachusetts, USA.

Warner, D. M. (1976), "Scheduling nursing personnel according to nurse preference: a mathematical programing approach", *Operations Research* 24/5, 842–856.