# HIBISCUS: A Constraint Programming Application to Staff Scheduling in Health Care

Stéphane Bourdais, Philippe Galinier, and Gilles Pesant

Département de génie informatique,
École Polytechnique de Montréal,
C.P. 6079, succ. Centre-ville
Montreal, Canada H3C 3A7
{sbs,philipg,pesant}@crt.umontreal.ca

**Abstract.** This paper presents a constraint programming model and search strategy to formulate and solve staff scheduling problems in health care. This is a well-studied problem for which many different approaches have been developed over the years but it remains a challenge to successfully apply any given instance of a method to the various contexts encountered. We show how the main categories of rules involved may be expressed using global constraints. We describe a modular architecture for heuristic search. The resulting flexible and rather general constraint programming approach is evaluated on benchmark problems from different hospitals and for different types of personnel.

## 1 Introduction

In many industries and public services work is carried out on a continuous basis, twenty-four hours a day, seven days a week. Health care workers (nurses and doctors) are in such a situation. Managing hospital personnel is a perpetual balancing act between three very important, yet often contradictory, objectives: high quality care provided to patients, good working conditions for the staff, and low costs. The first corresponds to the mission of the health care sector, the second contributes significantly to personnel retention, and the third ensures that we make the best of limited financial resources. Warner [19] recognizes three levels of decision making in managing a nursing staff, and much of it applies equally to doctors:

*i.* The staffing decision (strategic): dimensioning care units by determining the amount of staff required for each skill.
*ii.* The scheduling decision (tactical): building the actual schedules that specify when each staff member works and what task is performed, in a scheduling horizon spanning from one week to several months.
*iii.* The allocation decision (operational): readjusting daily because of unforeseen events such as illness or an increase in demand.

We focus here on the second level and consider the number of care units, the amount of staff and the demand fixed from the first level. Since financial costs

are primarily influenced by that first level, the two objectives that remain are the quality of the care provided and working conditions for the staff. Such objectives do not readily lend themselves to formalization. The problem is to some extent fuzzy in the sense that optimality is not easily defined through a formula. In practice, human judgement is required to make a choice from a set of candidate schedules meeting predefined criteria.

There are two main families of solutions in staff scheduling: rotating schedules and personalized schedules. When the personnel is interchangeable, rotating schedules, a repeating pattern of sequences of work and rest days alternating over a few weeks, are particularly well adapted. In effect, everyone has an identical schedule but that is out of phase with the others, thus ensuring fairness among the staff. When members of the personnel have individual restrictions or preferences that must be taken into consideration, such as unavailabilities due to other activities or particular skills, rotating schedules become inappropriate. Personalized schedules for each member of personnel are then preferred. That latter family is typical for doctors and even for nurses.

Staff scheduling is a well-studied problem for which many different approaches have been developed over the years (see for example [6] for a recent survey in the health care sector). In particular, constraint programming has been applied to nurse scheduling. A semi-automated system is described in [1]. A step by step procedure is implemented in [9] for a French hospital, possibly requiring manual adjustments during the scheduling process. Redundant modeling is used in [8] to generate one-week schedules for nurses in a Hong-Kong hospital. Another system written in CHIP is currently used by a French hospital [7]. Finally, [12] combines constraint programming and local search to produce schedules for several German hospitals.

Staff scheduling in health care remains a challenging problem: most of the time it is still solved by hand through a lengthy process, and where automated systems are involved they tend to be strongly linked to a particular context. There are significant differences between hospitals or even between care units of a hospital in the rules governing the schedules, due to government or union regulations but also to local traditions emerging over the years. Nevertheless, the rules encountered fall into a few categories common to all contexts (see Sect. 2).

We make two contributions in this paper. First, we describe a flexible and rather general constraint programming approach to the staff scheduling problem in health care (doctors or nurses), the HIBISCUS software. Though this may not be a new domain of application for constraint programming, we believe its widespread use of global constraints and the different contexts to which it is successfully applied are indeed novel in this area. Second, we propose a modular architecture for heuristic search that combines ranking information on variable/value pairs from the individual constraints present in the model.

In the rest of the paper: Section 2 describes the staff scheduling problem in health care; Section 3 lays down the constraint programming model for HIBISCUS; Section 4 develops its search heuristics, including the modular architecture

combining information from the constraints; Section 5 presents and discusses the results on real data sets; Section 6 summarizes the contributions and identifies future research directions.

## 2 Staff Scheduling in Health Care

Scheduling is about assigning resources to tasks (here, staff members to work shifts) in time, while respecting various constraints (here, the rules below). The resulting work schedules consist of sequences of work shifts of several types separated by rest periods. A sample schedule is given at Fig. 4.

### 2.1 The Rules

We present the main categories of rules for this problem, based on the relevant literature and on our own experience with many hospitals in the Montreal area [6, 11, 2, 10, 18, 4, 5, 17]. Concrete examples of such rules are given in Sect. 3.

**Demand (DEM).** A sufficient number and variety of shifts must be staffed throughout the scheduling horizon in order to guarantee minimum coverage.

**Availability (AVA).** A given staff member, according to his qualifications, full/part time status, vacation, and outside responsibilities, is not available at all times. We distinguish *preassignments* (AVA1), *forbidden assignments* (AVA2), and *candidate vacation days* (AVA3) from which a certain number must be selected.

**Distribution (DIS).** Many rules aim at a fair distribution of shifts among staff members and at balanced individual schedules. We distinguish *individual workloads* (DIS1), constrained to lie in a given range over the whole scheduling horizon or subsets of it to encourage a uniform workload, *balance for a certain type of shift among the staff* (DIS2), either evenly or according to some criterion such as seniority, *distribution of weekends off* (DIS3) across the scheduling horizon for individual staff members, and *relative proportion of certain types of shifts* (DIS4) in individual schedules.

**Ergonomics (ERG).** This is the largest and most heterogeneous category. Various rules ensure a certain level of quality for the schedules produced and may be specified either globally for the staff or only for certain individuals. We distinguish *patterns of shifts over certain days* (ERG1) such as alternating between two types of shifts on weekends, *length of stretches of shifts of identical type* (ERG2) to avoid working too few or too many days in a row on a certain shift, *patterns of stretches* (ERG3) such as forward rotation (going from day shifts to evening shifts to night shifts to day shifts again), *patterns of stretches of a given length* (ERG4) that ask for at least so many consecutive shifts of a certain type right after shifts of another type, and *preferences and aversions* (ERG5).

Rules regarding demand and availability are always hard. Some distribution and ergonomic rules may be soft.

## 3 HIBISCUS Constraint Programming Model

Let $H$ denote the index set for successive days of the planning horizon, $P$ for staff members, and $Q$ for possible shifts, including off and vacation. A shift type is a set of shifts sharing a particular property (e.g. morning shifts, work shifts). More generally, a shift type may include any subset of $Q$. Considering a partition $\alpha$ of $Q$ into $m$ classes, $\alpha = \{Q_1, \cdots, Q_m\}$, the type of a shift $q$ with respect to $\alpha$, denoted by $t_\alpha(q)$, is the index $i$ such that $q \in Q_i$.

### 3.1 Variables

For each staff member $i \in P$ and each day $j \in H$, we define an *assignment variable* $\mathtt{X}_{ij} \in Q$ that indicates which shift is assigned to $i$ on day $j$. For short, we use $\mathbf{X}_{i\bullet}$ (respectively $\mathbf{X}_{\bullet j}$) to represent successive assignment variables $\langle \mathtt{X}_{i1}, \mathtt{X}_{i2}, \ldots, \mathtt{X}_{i\ |H|} \rangle$ associated to staff member $i$ ( respectively $\langle \mathtt{X}_{1j}, \mathtt{X}_{2j}, \ldots, \mathtt{X}_{|P|j} \rangle$ associated to day $j$).

In addition to assignment variables, we define as needed some auxiliary variables in order to implement the constraints of the problem. In particular, considering a partition $\alpha$ as before, it is possible to define an auxiliary variable $Y \in \{1, 2, \ldots, m\}$ that indicates the type of an assignment variable $X_{ij}$ as follows: $Y = t_\alpha(X_{ij})$.

### 3.2 Constraints

Most availability constraints (AVA) such as preassignments and forbidden assignments are easily modeled as unary constraints ($\mathtt{X}_{ij} = q$ or $\mathtt{X}_{ij} \neq q$). We rather focus on the higher arity constraints required. Again, this is a nice application domain for constraint programming because it showcases several of the global constraints developed over the years. We first review those higher arity constraints that are needed, establishing a notation and concentrating on their semantics, filtering capability, and computational complexity. We then go back to the rules presented in Sect. 2 and, using real cases, provide instances together with the way we model them using the following constraints.

**SUM constraints.** Consider a vector $\mathbf{U} = \langle \mathtt{U}_1, \mathtt{U}_2, \ldots, \mathtt{U}_n \rangle$ of integer variables and an additional integer variable S. Constraint SUM($\mathbf{U}$, S) guarantees that S is the sum of variables $\mathtt{U}_i$ ($1 \leq i \leq n$). The filtering algorithm we use only checks bound consistency and its time complexity is linear in $n$.

**EXTENSION constraints.** Given a vector $\mathbf{U} = \langle \mathtt{U}_1, \mathtt{U}_2, \ldots, \mathtt{U}_n \rangle$ of finite domain variables, constraint EXTENSION($\mathbf{U}$, $\mathcal{T}$) defines in extension the set $\mathcal{T}$ of admissible $n$-tuples for $\mathbf{U}$. The filtering algorithm that maintains generalized arc-consistency is exponential in $n$. For that reason, we use this type of constraint only when a small number of variables is involved.

`DISTRIBUTION` **constraints [15].** Consider vectors $\mathbf{U} = \langle \mathtt{U}_1, \mathtt{U}_2, \ldots, \mathtt{U}_n \rangle$ of finite domain variables, $\mathbf{C} = \langle \mathtt{C}_1, \mathtt{C}_2, \ldots, \mathtt{C}_m \rangle$ of integer variables, and $\mathbf{V} = \langle v_1, v_2, \ldots, v_m \rangle$ of values. Constraint `DISTRIBUTION`($\mathbf{C}$, $\mathbf{V}$, $\mathbf{U}$) guarantees that each $\mathtt{C}_j$ equals the number of variables in $\mathbf{U}$ whose value is $v_j$. Its filtering algorithm achieves generalized arc consistency and runs in polynomial time. This constraint could be implemented using $m$ ordinary cardinality constraints. However, it is possible to reduce the complexity of filtering by treating simultaneously the $m$ possible values in a global constraint.

`SLIDING_DISTRIBUTION` **constraints [16].** Consider vector $\mathbf{V} = \langle v_1, v_2, \ldots, v_m \rangle$, vector $\mathbf{U} = \langle \mathtt{U}_1, \mathtt{U}_2, \ldots, \mathtt{U}_n \rangle$ of finite domain variables taking their values in $V$, and two vectors of $m$ integers $\underline{\lambda}$ and $\overline{\lambda}$. Constraint `SLIDING_DISTRIBUTION`($\mathbf{U}, V, \underline{\lambda}, \overline{\lambda}, w$) guarantees that in each subsequence of $\mathbf{U}$ of length $w$, value $v_k$ ($1 \leq k \leq m$) appears between $\underline{\lambda}_k$ and $\overline{\lambda}_k$ times. This constraint is conceptually equivalent to `DISTRIBUTION` constraints expressed on each position of a sliding window but treating them all at once improves the filtering capability.

`STRETCH` **constraints [14].** Consider a set $V = \{v_1, v_2, \ldots, v_m\}$ and a sequence $\mathbf{s} = \langle s_1, s_2, \ldots, s_n \rangle$ whose elements belong to $V$. We call *stretch* of type $v \in V$ in $\mathbf{s}$ any maximal subsequence $\langle s_i, s_{i+1}, \ldots, s_j \rangle$ of elements all equal to $v$. The length of the stretch equals $j - i + 1$. Consider a vector $\mathbf{U} = \langle \mathtt{U}_1, \mathtt{U}_2, \ldots, \mathtt{U}_n \rangle$ and let $\underline{\lambda}$ and $\overline{\lambda}$ be two vectors of $m$ integers. Constraint `STRETCH`($\mathbf{U}$, $V$, $\underline{\lambda}$, $\overline{\lambda}$) guarantees that the length of any stretch of type $v_k$ ($1 \leq k \leq m$) lies between $\underline{\lambda}_k$ and $\overline{\lambda}_k$. The difference with the previous constraint is that we are counting *consecutive* variables of a certain value. The filtering algorithm of the constraint exhibits a running time that is quadratic in $m$ and linear in $\underline{\lambda}$ and $\overline{\lambda}$.

`PATTERN` **constraints [13].** We call $k-$pattern any sequence of $k$ elements such that no two successive elements have the same value. Consider a set $V = \{v_1, v_2, \ldots, v_m\}$ and a sequence $\mathbf{s} = \langle s_1, s_2, \ldots, s_n \rangle$ of elements of $V$. Consider now the sequence $\langle v_{i_1}, v_{i_2}, \ldots, v_{i_\ell} \rangle$ of the types of the successive stretches that appear in $\mathbf{s}$. Let $\mathcal{P}$ be a set of $k-$patterns. Vector $\mathbf{s}$ satisfies $\mathcal{P}$ if and only if every subsequence of $k$ elements in $\langle v_{i_1}, v_{i_2}, \ldots, v_{i_\ell} \rangle$ belongs to $\mathcal{P}$. For example, let $\mathbf{s} = \langle a, a, b, b, b, a, c, c \rangle$ and $\mathcal{P} = \{(a, b, a), (a, b, c), (b, a, c)\}$. The sequence of the types of the successive stretches that appear in $\mathbf{s}$ is $\langle a, b, a, c \rangle$. Since $(a, b, a)$ and $(b, a, c)$, the two subsequences of three elements in $\langle a, b, a, c \rangle$, both belong to $\mathcal{P}$, $\mathbf{s}$ satisfies $\mathcal{P}$. Constraint `PATTERN`($\mathbf{U}$, $\mathcal{P}$) guarantees that any assignment for $\mathbf{U}$ satisfies $\mathcal{P}$. The filtering algorithm of constraint `PATTERN` is low polynomial.

### 3.3 Instances of the rules

*Demand Constraints (`DEM`).* A demand constraint makes it possible to control the number of staff members that are present at a particular period of a day. A so-called period is simply defined by an interval of time in the day (for example, between noon and 4pm). Observe that staff member $i$ is present at a particular period on day $j$ if and only if shift $X_{ij}$ covers the considered period.

Let us consider the following simplified example where there are five shifts in $Q$: $q_1$ from midnight to 8am, $q_2$ from 8am to 4pm, $q_3$ from 8am to 8pm, $q_4$ from 4pm to midnight, and $q_5$ (off). Therefore, the day is partitioned according to four time periods $p_1$ =[midnight, 8am], $p_2$ =[8am, 4pm], $p_3$ =[4pm, 8pm] and $p_4$ =[8pm, midnight]. Note that $p_1$ is covered by shift $q_1$, $p_2$ by $q_2$ and $q_3$, $p_3$ by $q_3$ and $q_4$, and $p_4$ by $q_4$. Consider vectors $\underline{q} = (3, 5, 6, 3)$ and $\overline{q} = (6, 8, 9, 6)$ indicating the minimum and maximum workforce required for each period ($\underline{q}_\ell \leq$ workforce at $p_\ell \leq \overline{q}_\ell$). In order to express the constraint, we first introduce a vector $\mathbf{M} = \langle \mathtt{M}_1, \ldots, \mathtt{M}_5 \rangle$ of auxiliary variables to represent the number of occurrences of each shift during day $j$. Then, we state

$$\mathtt{DISTRIBUTION}(\mathbf{M},\ Q,\ \mathbf{X}_{\bullet j}),$$

$$\mathtt{SUM}(\langle \mathtt{M}_1 \rangle,\ \mathtt{S}_1),\ \ \mathtt{SUM}(\langle \mathtt{M}_2, \mathtt{M}_3 \rangle,\ \mathtt{S}_2),\ \ \mathtt{SUM}(\langle \mathtt{M}_3, \mathtt{M}_4 \rangle,\ \mathtt{S}_3),\ \ \mathtt{SUM}(\langle \mathtt{M}_4 \rangle,\ \mathtt{S}_4),$$
$$3 \leq S_1 \leq 6,\ \ 5 \leq S_2 \leq 8,\ \ 6 \leq S_3 \leq 9,\ \ 3 \leq S_4 \leq 6.$$

*Workload constraints (DIS1).* A workload constraint is defined by a 5-tuple $(i, j_{\text{beg}}, j_{\text{end}}, \underline{h}, \overline{h})$ and imposes that the number of hours worked by staff member $i$ over the time period (set of successive days) $[j_{\text{beg}}, j_{\text{beg}+1} \ldots j_{\text{end}}]$ lies between $\underline{h}$ and $\overline{h}$.

Let us consider the following example where there are seven shifts in $Q$: off (that lasts 0 hours); D4 and E4 (4 hours); D6 (6 hours); D8, N and E8 (8 hours). We consider the workload constraint with $(i, j_{\text{beg}}, j_{\text{end}}, \underline{h}, \overline{h}) = (i, 15, 21, 30, 35)$ that requires staff member $i$ to work between 30 and 35 hours over the third week of the horizon (from day 15 to day 21). A straightforward way to express the constraint is to state $30 \leq h(X_{i\,15}) + h(X_{i\,16}) + \cdots + h(X_{i\,21}) \leq 35$, where $h(\mathtt{X})$ represents the duration of the shift assigned to $\mathtt{X}$. Note that, in this case, simple bound consistency would normally be applied.

We use in HIBISCUS the following more powerful way to express the constraint. Let $\alpha_{\mathtt{DIS1}} = \{Q_1, \ldots, Q_p\}$ be the partition of $Q$ into classes of shifts having the same duration and $h_1, \ldots, h_p$ be the durations of shifts in $Q_1, \ldots, Q_p$. In our example, there are four classes and the possible durations of the shifts are 0, 4, 6, and 8. Let $\mathcal{T}$ represent the set of tuples $(m_1 \ldots m_p)$ such that $\underline{h} \leq \sum_{1 \leq k \leq p} h_k m_k \leq \overline{h}$ and $\sum_{1 \leq k \leq p} m_k = j_{\text{end}} - j_{\text{beg}} + 1$. For the example, $\mathcal{T} = \{(m_1, m_2, m_3, m_4) : 30 \leq 0 * m_1 + 4 * m_2 + 6 * m_3 + 8 * m_4 \leq 35$ and $m_1 + m_2 + m_3 + m_4 = 7\} = \{(3, 0, 0, 4), (3, 0, 1, 3), \ldots\}$. We introduce auxiliary variables $\mathbf{Y} = \langle \mathtt{Y}_1, \ldots, \mathtt{Y}_{j_{\text{end}} - j_{\text{beg}} + 1} \rangle$, where $\mathtt{Y}_k = t_{\alpha_{\mathtt{DIS1}}}(\mathbf{X}_{i j_{\text{beg}} + k - 1})$ $(1 \leq k \leq j_{\text{end}} - j_{\text{beg}} + 1)$ and multiplicity variables $\mathbf{M} = \langle \mathtt{M}_1, \ldots, \mathtt{M}_p \rangle$, where $\mathtt{M}_k$ $(1 \leq k \leq p)$ will count the number of variables in $\{X_{i j_{\text{beg}}}, \ldots, X_{i j_{\text{end}}}\}$ whose assigned value belongs to class $Q_k$. To express the constraint, we state

$$\mathtt{DISTRIBUTION}(\mathbf{M},\ \langle 1, \ldots, p \rangle,\ \mathbf{Y}), \quad \mathtt{EXTENSION}(\mathbf{M},\ \mathcal{T}).$$

Note that this way of expressing the constraint ensures generalized arc consistency. The filtering algorithm is exponential in $p$ but this is not a problem in practice since the number of possible durations never exceeds five in the instances encountered.

In some cases, workloads are expressed not on calendar weeks but on a sliding window of a given number of days. For example in any nine consecutive days, five or six must be worked:

$$\texttt{SLIDING\_DISTRIBUTION}(\mathbf{X}, \langle \mathsf{off}, \mathsf{day}, \mathsf{evening}, \mathsf{night} \rangle, \langle 3, 1, 1, 1 \rangle, \langle 4, 9, 9, 9 \rangle, 9).$$

*Distribution of weekends off (DIS3).* Consider a staff member $i \in P$ who may work at most two weekends in a row. Note that to work on a weekend means that the considered staff member performs a work shift on Saturday or Sunday, or even performs a Friday work shift that overlaps Saturday, for example a night shift from 8pm to 4am. We introduce auxiliary variables $\mathbf{W} = \langle \mathtt{W}_1, \mathtt{W}_2, \ldots \rangle$ with $\mathtt{W}_k \in \{\mathsf{y}, \mathsf{n}\}$ indicating whether weekend $k$ is worked at all ($\mathsf{y}$) or not ($\mathsf{n}$) by staff member $i$. Let $H_k \subset H$ be the set of days that correspond to weekend $k$ and $Q_j \subset Q$ the set of work shifts for day $j$. We then state

$$\mathtt{W}_k = \mathsf{y} \quad \Leftrightarrow \quad \bigvee_{j \in H_k} \bigvee_{v \in Q_j} (\mathtt{X}_{ij} = v)$$

$$\texttt{STRETCH}(\mathbf{W},\ \langle \mathsf{y}, \mathsf{n} \rangle,\ \langle 1, 1 \rangle,\ \langle 2, \infty \rangle).$$

*Length of stretches of shifts of identical type (ERG2).* Consider a staff member $i \in P$ who may work at least two but at most seven $\mathsf{day}$ shifts in a row or $\mathsf{evening}$ shifts in a row, and at least three but at most six $\mathsf{night}$ shifts in a row. This rule can be expressed as

$$\texttt{STRETCH}(\mathbf{X}_{i\bullet},\ \langle \mathsf{day}, \mathsf{evening}, \mathsf{night}, \mathsf{off} \rangle,\ \langle 2, 2, 3, 1 \rangle,\ \langle 7, 7, 6, \infty \rangle).$$

Consider another rule that says that stretches of night shifts must be at least fourteen days apart. We introduce partition $\alpha_{\texttt{ERG2}} = \{Q_1 = \{\mathsf{night}\},\ Q_2 = Q \setminus Q_1\}$ and auxiliary variables $\mathbf{Y} = \langle \mathtt{Y}_1, \ldots, \mathtt{Y}_{|H|} \rangle$, where $\mathtt{Y}_k = t_{\alpha_{\texttt{ERG2}}}(\mathtt{X}_{ik})$ $(1 \leq k \leq |H|)$. Since class $Q_2$ includes any shift that may separate two stretches of night shifts, this rule can be expressed as

$$\texttt{STRETCH}(\mathbf{Y},\ \langle 1, 2 \rangle,\ \langle 1, 14 \rangle,\ \langle \infty, \infty \rangle).$$

*Patterns of stretches (ERG3).* Consider a staff member $i \in P$ and the set of shifts $Q = \{\mathsf{day}, \mathsf{evening}, \mathsf{night}, \mathsf{off}\}$. In order to impose homogeneous stretches of work and forward rotation (recall from Sect. 2), we introduce the set $\mathcal{P}$ of 3-patterns $\{\mathsf{odo}, \mathsf{oeo}, \mathsf{ono}, \mathsf{dod}, \mathsf{eoe}, \mathsf{non}, \mathsf{doe}, \mathsf{eon}, \mathsf{nod}\}$ (with $\mathsf{d}$=$\mathsf{day}$, $\mathsf{e}$=$\mathsf{evening}$, $\mathsf{n}$=$\mathsf{night}$, and $\mathsf{o}$=$\mathsf{off}$). The first three patterns in $\mathcal{P}$ ensure that the stretches are homogeneous: for example, a $\mathsf{day}$-stretch is preceded and followed by rests and not any other type of work shift. The last six patterns allow us to move from one work stretch to another of the same type or one type forward. We then state

$$\texttt{PATTERN}(\mathbf{X}_{i\bullet},\ \mathcal{P}).$$

*Patterns of stretches of a given length (ERG4).* Any staff member $i \in P$ working three consecutive night shifts must then have at least three days off. Such constraints can be expressed using `STRETCH` and `EXTENSION` constraints. Because of space limitations, we can only outline this combination. An auxiliary variable $\mathtt{Y}_{ij}$ is linked to each $\mathtt{X}_{ij}$, with a domain containing two values corresponding to night, two others to off, and one to all the other shifts. The first "night" value is reserved to represent stretches of three nights and the first "off" value to represent stretches of three days off. A set of pairs $\mathcal{T}$ is then built to only allow the right transitions from one shift to another. Finally the following constraints are stated:

$$\mathtt{STRETCH}(\mathbf{Y},\ \langle \mathsf{3nights}, \mathsf{3offs}, \mathsf{night}, \mathsf{off}, \mathsf{others}\rangle,\ \langle 3,3,1,1,1\rangle,\ \langle 3,3,2,\infty,\infty\rangle),$$

$$\mathtt{EXTENSION}(\langle \mathtt{Y}_{ij}, \mathtt{Y}_{ij+1}\rangle,\ \mathcal{T}) \quad (1 \leq j \leq |H| - 1).$$

## 4 HIBISCUS Search Heuristics

This section describes the search heuristics used to complement the model just described. In an initial phase, vacations are taken care of. Vacation days are often preassigned or are otherwise considerably restricted in their location. Consequently they generally have little impact on the search for good solutions and are fixed first.

### 4.1 Problem Decomposition

Next consider our decision variables $\mathtt{X}_{ij}$ arranged in a matrix whose rows correspond to staff members and columns to days of the planning horizon, thus offering a natural graphical representation of a schedule. Note that most of the constraints identified in Sect. 2 link variables horizontally, on individual schedules, while demand constraints (`DEM`) link them vertically. The many contexts we examined may be partitioned in two classes: those whose vertical constraints are tight (a precise demand as opposed to one lying in a certain range) while the (horizontal) workload constraints (`DIS1`) are loose (within a range); those whose vertical constraints are loose but whose workload constraints are tight (a set number of hours per week). The former typically corresponds to emergency room physicians and the latter to nurses.

So while the types of constraints encountered may be the same, the relative tightness of some has an impact on which dimension of the problem is harder to solve and hence on which search strategy offers better chances of success. For these two classes, which to our knowledge have always been treated separately in the literature, we use different decomposition strategies but using the same model. They are *decomposition by day* (i.e. column by column) and *decomposition by individual schedule* (i.e. row by row) (see Fig. 1). The decision of which one to use is solely based on which constraints are tight.


**Fig. 1.** Two decompositions of the staff scheduling problem.

Either case induces subproblems that must be tackled in a certain order. Days are ordered chronologically to help the satisfaction of horizontal ergonomic constraints and individual schedules are ordered randomly for lack of a significantly better ordering.

### 4.2 Variable/Value Ordering Heuristic

We describe a simple framework for the definition of heuristics to select variable/value pairs whose main asset is its modular structure following that of the constraint programming model for the problem at hand. To each constraint of the model (hard or soft) we may associate an *incentive heuristic* whose goal is to favor, among uninstantiated variables, value assignments that bring us closer to satisfying that constraint. Each incentive heuristic associates a score to potential assignments and those scores are then combined to dynamically select the next variable/value pair (see Fig. 2).

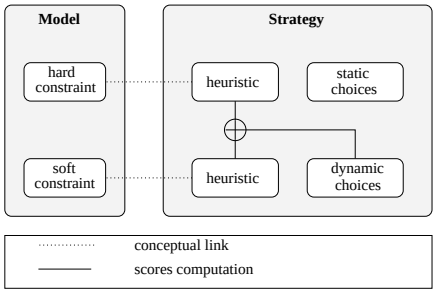


**Fig. 2.** Modular architecture of the variable/value ordering heuristic.

Incentive heuristic $h$ acts on a subset $\mathbf{X}^h = \{\mathtt{X}_1^h, \mathtt{X}_2^h, \ldots \mathtt{X}_k^h\}$ of the variables of the problem. Given the current domains of those variables, function

$$\phi_h : \mathtt{X}_1^h \times D_{\mathtt{X}_1^h} \cup \cdots \cup \mathtt{X}_k^h \times D_{\mathtt{X}_k^h} \rightarrow \mathbb{R}$$

assigns numerical scores to variable/value pairs. Let $\mathcal{H}$ denote the set of incentive heuristics present and $\mathbf{X}$ the set of our decision variables $\mathtt{X}_{ij}$. For each potential assignment $(\mathtt{X}, v)$ we compute a weighted sum of the individual scores

$$\pi(\mathtt{X}, v) = \sum_{h \in \mathcal{H} \,:\, \mathtt{X} \in \mathbf{X}^h} \omega_h \cdot \phi_h(\mathtt{X}, v).$$

We then select the assignment with the largest $\pi()$ value.

An incentive heuristic can be designed for every type of constraint in the model in order to bring its contribution to the variable/value ordering heuristic. In what follows, we detail one such example and outline another.

**Incentive Heuristic for Demand Constraints (DEM).** As we saw in Sect. 3.2, demand is broken down for each period of a day. To ease the exposition and without loss of generality, we present the heuristic for the demand on one given period and express it in the slightly more general context of a cardinality constraint. Let $\mathbf{X}$ represent a set of $m$ variables, $D_{\mathtt{X}}$ the current domain of a variable $\mathtt{X}$, $T$ the subset of values of interest, and $\underline{\nu}$ and $\overline{\nu}$ two natural numbers. The number of times variables from $\mathbf{X}$ are assigned a value from $T$ must be at least $\underline{\nu}$ and at most $\overline{\nu}$.

We introduce $\underline{n}$, the number of variables that necessarily take their value in $T$, and $\overline{n}$, the number of variables that necessarily take their value outside $T$. These are computed as

$$\begin{aligned} \underline{n} &= |\{\mathtt{X} \in X \;:\; D_{\mathtt{X}} \subseteq T\}| \\ \overline{n} &= |\{\mathtt{X} \in X \;:\; T \cap D_{\mathtt{X}} = \emptyset\}|. \end{aligned}$$

We also propose $\tilde{n} = \frac{m - \overline{n} + \underline{n}}{2}$ as a rough estimate of the final number of values from $T$ in $\mathbf{X}$.

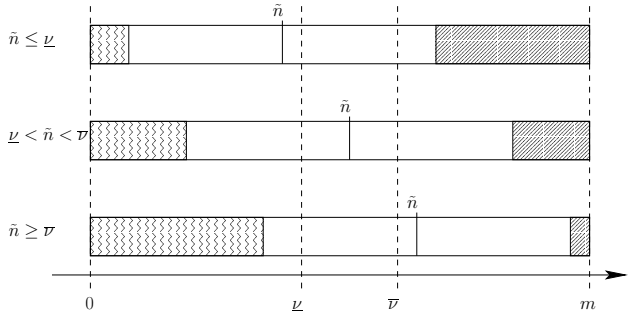


**Fig. 3.** Three cases of the incentive heuristic for the demand constraint.

Three cases may arise, which we give below with the corresponding score function and which we also illustrate at Fig. 3.

If $\tilde{n} < \underline{\nu}$, we wish to encourage assignments from $T$:

$$\phi_h(\mathtt{X}, v) = \begin{cases} 1 \text{ if } v \in T, \\ 0 \text{ otherwise.} \end{cases}$$

If $\underline{\nu} \leq \tilde{n} \leq \overline{\nu}$, we remain neutral:

$$\phi_h(\mathtt{X}, v) = 0.$$

If $\tilde{n} > \overline{\nu}$, we wish to discourage assignments from $T$:

$$\phi_h(\mathtt{X}, v) = \begin{cases} 0 \text{ if } v \in T, \\ 1 \text{ otherwise.} \end{cases}$$

An incentive heuristic for workload constraints (`DIS1`) can be designed along the same lines since it is a straightforward generalization to weighted cardinality in which weights correspond to the duration of shifts.

## 5 Experimental Results

The "Optimization of Health Care Management" research team in Montreal has been collecting data from several hospitals in order to create a set of benchmarks on which to evaluate several approaches (column generation, 0-1 linear programming, tabu search, constraint programming) developed by different members of the team over the years. Some initial results have already been obtained and an extensive comparison should soon be possible. In this section, we present some experiments performed on real-world instances from that evolving benchmark: they originate from three hospitals in Montreal.

### 5.1 Data sets

Table 1 presents some characteristics of these instances. The first four columns respectively give the name of the instance, the number of staff members, the number of days in the scheduling horizon, and the number of shifts. CHILD and BC are nurse scheduling instances originating from two different units of the same hospital; ERMGH is a nurse scheduling instance from a different hospital; HSC is a physician scheduling instance from still another hospital. A complete description of the instances may be found in [3]. Note that the number of decision variables equals $|P| \cdot |H|$ and the size of all domains equals $|Q|$. Therefore, these instances are quite large.

### 5.2 Experiments

For the experiments reported here, the search heuristic was kept simple by using a single incentive heuristic at a time, thereby not requiring the setting of weights $\omega_h$ for the combination of different influences. We will eventually investigate the interplay between several incentive heuristics.

**Table 1.** Instances tested.

| Origin | $\|P\|$ | $\|H\|$ | $\|Q\|$ | success rate | avg time | std dev |
|---|---|---|---|---|---|---|
| CHILD | 41 | 42 | 5 | 80% | 9.5 | 15.8 |
| BC | 54 | 42 | 12 | 70% | 8.4 | 5.1 |
| ERMGH | 41 | 42 | 4 | 80% | 78.3 | 192.7 |
| HSC | 18 | 28 | 8 | 100% | 25.0 | 53.5 |

For decomposition by individual schedule, recall that the order in which the subproblems are solved is chosen randomly, whereas for decomposition by day, subproblems are solved in chronological order. At this early stage of experimentation with incentive heuristics, chronological variable ordering for the former decomposition and smallest-domain-first variable ordering for the latter perform slightly better than the fully-fledged variable/value ordering heuristic, and so they are used to produce the results reported here. However, the incentive heuristic approach does yield the best results for value ordering and is therefore retained in that capacity. In order to better evaluate the robustness of our algorithm and to offer several candidate solutions to the decision maker, we break ties at random during value ordering.

HIBISCUS was implemented using the ILOG Solver C++ constraint programming library. In our experiments, we performed ten runs of one hour each (on a Sun Ultra-10, 440 MHz). Each run finishes with a success (when a feasible solution is found) or a failure.

Table 1 presents the results obtained. The fifth column gives the success rate based on the ten runs. The sixth and seventh columns respectively give the average computation time and the standard deviation in seconds for the successful runs. In every instance, the success rate is high and the majority of solutions are found within a few seconds.

| $n^\circ$ | Mo | Tu | We | Th | Fr | Sa | Su | Mo | Tu | We | Th | Fr | Sa | Su | Mo | Tu | We | Th | Fr | Sa | Su | Mo | Tu | We | Th | Fr | Sa | Su |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | - | - | - | - | - | - | - | - | D | D | - | - | A | B | H | H | - | B | - | B | A | - | D | - | B | - | - | - |
| 2 | - | E | E | - | - | - | - | - | - | - | - | - | - | - | - | C | H | H | H | - | - | - | E | - | D | - | - | - |
| 3 | - | F | - | B | - | - | - | - | C | - | B | - | B | A | - | F | - | E | - | - | - | - | F | - | A | - | - | H |
| 4 | - | C | - | A | E | E | E | - | - | - | F | F | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 5 | - | D | C | C | H | - | H | - | E | E | E | - | - | - | F | D | C | C | - | H | H | - | C | C | C | H | - | - |
| 6 | - | A | - | D | - | H | - | - | F | - | D | - | - | - | B | - | - | D | - | - | - | - | B | - | - | A | - | - |
| 7 | - | B | - | H | C | D | C | - | B | - | - | A | - | - | - | B | - | F | - | A | B | - | A | H | - | B | - | - |
| 8 | - | - | - | - | - | - | - | - | A | - | C | - | H | H | D | - | D | - | E | E | E | - | - | - | F | F | - | - |
| 9 | F | - | D | - | - | A | B | E | - | H | - | B | - | - | A | - | F | - | F | - | - | H | H | E | - | - | H | - |
| 10 | H | H | - | F | B | - | - | H | H | - | A | - | - | - | - | - | - | - | - | - | - | D | - | - | E | - | B | A |
| 11 | E | - | H | - | F | - | - | D | - | C | - | D | C | D | C | - | B | - | - | - | - | C | - | D | - | - | A | B |
| 12 | D | - | F | - | D | C | D | C | - | F | - | H | - | - | - | - | - | - | - | - | - | B | - | - | H | E | E | E |
| 13 | C | - | B | - | - | B | A | F | - | B | - | - | - | - | E | E | E | - | - | - | - | F | - | F | - | D | C | D |
| 14 | B | - | - | E | - | - | - | B | - | - | H | C | D | C | - | A | A | - | B | - | - | E | - | B | - | - | - | - |
| 15 | G | - | G | - | C | - | - | G | - | G | - | G | - | - | G | - | G | - | G | - | - | G | - | G | - | G | - | - |
| 16 | A | - | A | - | A | - | - | A | - | A | - | E | E | E | - | - | - | A | A | - | - | A | - | A | - | C | D | C |
| 18 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | D | C | D | - | - | - | - | - | - | - |
| 19 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | C | D | C | - | - | - | - | - | - | - |

**Fig. 4.** A sample schedule for HSC. "-" corresponds to off.

### 5.3 Discussion

Warner [19] proposes a short check-list to compare staff scheduling approaches:

- ☐ *Coverage* : Are the tasks adequately covered by the schedule provided? Otherwise the hospital must call on additional personnel.
- ☐ *Quality* : How good are the working conditions of the staff, based on union rules, hospital rules, and individual preferences?
- ☐ *Stability* : Are individual schedules fairly homogeneous? Then staff members can more easily organize their social and family life.
- ☐ *Flexibility* : How elegantly does the approach deal with changes from one scheduling horizon to the next (personnel turnover, going from full time to part time, vacation, change of preferences, etc.)?
- ☐ *Fairness* : Are the tasks equally shared among the staff, while taking into account varying status (seniority, full time vs part time)?
- ☐ *Cost* : What amount of resources (human, time, computing power) were required to build the schedule?

*Coverage* is a requirement of the vast majority of the methods proposed, including ours, and therefore it is necessarily achieved. *Quality* and *fairness* depend on how the corresponding rules are handled. Some methods, typically local search but also others, associate penalties to rule violations and try to minimize their weighted combination. We treat most of those rules as (hard) constraints, possibly with some amount of tolerance built in: it is therefore easier to guarantee a certain level of quality and fairness. The danger is that the instance becomes over-constrained and some of the recent work on soft constraints could prove useful. Currently, the search heuristic described in Sect. 4 allows us to add incentives for preferences (`ERG5`) and to aim for a given target in cases where the constraint has been loosened somewhat by a certain tolerance. Being able to enforce weekly workloads and special rules for weekends off helps in achieving *stability* in the resulting schedules. As for *flexibility*, the fact that schedules have been produced for different hospital contexts, with similar categories of rules but often very distinct instantiations of those rules, shows that our method goes beyond adapting from one scheduling horizon to the next in a given context. Nevertheless, we believe that the weakest part of HIBISCUS is currently its search strategy: it could be made more robust. To its defense, we clearly have not yet explored its full potential in combining incentives. As for *cost*, human cost is very low since no assistance is required and computational cost is also low.

## 6 Conclusion

This paper presented a constraint programming model and search strategy to formulate and solve staff scheduling problems in health care. HIBISCUS turns out to be quite flexible and able to adapt to many different situations encountered in staff scheduling. This is first due to its global constraints, such as constraints on stretches that make it possible to capture the large number of rules present

in this type of application. Another promising feature is the flexible technique proposed in order to introduce search heuristics. We are presently generalizing the introduction of heuristics in HIBISCUS. Again note that, in real life, there is generally no evaluation function to optimize, but simply preferences generally expressed in a fuzzy way, such as: to favour or discourage some particular shifts for a staff member, to balance the types of shifts assigned to a staff member, to balance unpopular shifts between staff members, and so forth. The heuristic approach explored in HIBISCUS seems to be well suited to deal with this kind of soft constraint.

## Acknowledgements

The authors would like to thank the other members of the "Optimization of Health Care Management" research team for sharing their insight on the data and the anonymous referees for their constructive comments. This work was partially supported by the Canadian Natural Sciences and Engineering Research Council under grants OGP0218028 and 227247-99.

## References

1. S. Abdennadher and H. Schenker. INTERDIP - An Interactive Constraint Based Nurse Scheduler. In *The First International Conference and Exhibition on The Practical Application of Constraint Technologies and Logic Programming - PACLP99*, 1999.
2. H. Beaulieu. Planification de l'horaire des médecins dans une salle d'urgence. Master's thesis, Université de Montréal, Canada, 1998.
3. S. Bourdais. Génération automatique d'horaires dans le milieu hospitalier. Master's thesis, École Polytechnique de Montréal, Canada, 2003.
4. I. Buzon and S. D. Lapierre. A Tabu Search Algorithm to Schedule Emergency Room Physicians. Technical report, Centre de Recherche sur les Transports, Montréal, Canada, 1999.
5. I.E.B. Cantera. La confection des horaires de travail des médecins d'urgence résolue à l'aide de la recherche tabou. Master's thesis, Université de Montréal, Canada, 2001.
6. M. W. Carter and S. D. Lapierre. Scheduling Emergency Room Physicians. *Health Care Management Science*, 4:347–360, 2001.
7. P. Chan, K. Heus, and G. Weil. Nurse Scheduling with Global Constraints in CHIP: GYMNASTE. In *Conference on Practical Applications of Constraint Technology*, London, UK, 1998.
8. B.M.W. Cheng, J.H.M. Lee, and J.C.K. Wu. A Constraint-Based Nurse Rostering System Using a Redundant Modeling Approach. In *Eight International Conference on Tools with Artificial Intelligence - TAI 1996*, pages 140–148. IEEE Computer Society Press, 1996.
9. S. J. Darmoni, A. Fajner, N. Mahe, A. Leforestier, M. Vondracek, O. Stelian, and M. Baldenweck. Horoplan: computer-assisted nurse scheduling using contraint-based programming. *Journal of the Society for Health Systems*, 5:41–54, 1995.

10. F. Forget. Confection automatisée des horaires de médecins dans une salle d'urgence. Master's thesis, Université de Montréal, Canada, 2002.
11. P. Labit. IRIS : Amélioration d'une méthode de génération de colonnes pour la confection d'horaires d'infirmières. Master's thesis, École Polytechnique de Montréal, Canada, 2000.
12. H. Meyer and A. Hofe. Nurse rostering as constraint satisfaction with fuzzy constraints and inferred control strategies. In E.C. Freuder and R.J. Wallace, editors, *Constraint Programming and Large Scale Optimisation Problems*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science. 2000.
13. G. Pesant. The Pattern Constraint. In preparation.
14. G. Pesant. A Filtering Algorithm for the Stretch Constraint. In *Principles and Practice of Constraint Programming - CP 2001*, pages 183–195. Springer-Verlag LNCS 2239, 2001.
15. J.-C. Régin. Generalized Arc Consistency for Global Cardinality Constraints. In *Proc. of AAAI-96*, pages 209–215. AAAI Press/MIT Press, 1996.
16. J.-C. Régin and J.-F. Puget. A Filtering Algorithm for Global Sequencing Constraints. In *Principles and Practice of Constraint Programming - CP 1997*, pages 32–46. Springer-Verlag LNCS 1330, 1997.
17. L.-M. Rousseau, G. Pesant, and M. Gendreau. A Hybrid Algorithm to Solve a Physician Rostering Problem. In *Second Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, Paderborn, Germany, 2000.
18. M. Saadie. Planification de l'horaire des médecins dans une salle d'urgence par la programmation par contraintes. Master's thesis, Université de Montréal, Canada, 2003.
19. D. M. Warner. Scheduling Nursing Personnel According to Nursing Preference: A Mathematical Programming Approach. *Operations Research*, 24:842–856, 1976.