A Tour Scheduling Problem with Fixed Jobs: use of Constraint Programming

Tanguy Lapègue · Damien Prot · Odile Bellenguez-Morineau

Received: date / Accepted: date

Abstract This paper presents a constraint programming approach to solve a specific scheduling problem arising in a company specialized in drug evaluation and pharmacology research. The aim is to build employee timetables covering the demand given by a set of fixed tasks. The optimality criterion concerns the equity of the workload sharing. A solution to this problem is the assignment of all tasks whose resulting working shifts respect tasks requirements as well as legal and organizational constraints. Scheduling problems usually consider a fixed set of shifts which have to be assigned to a given number of employees whereas in our problem shifts are not fixed and must be deduced from the task assignment.

Keywords Tour Scheduling Problem · Fixed Job Scheduling Problem · Constraint Programming

1 Introduction

Personnel scheduling problems tackle the difficult task of building employee rosters respecting legal and organizational constraints in order to satisfy the demand. These problems are of tremendous importance for services oriented companies, especially for those working around the clock. Consequently, many researchs have been carried out into this area (see [12] for an overview). These complex and highly constrained problems proved to be very difficult to solve in a satisfactory way and even more to optimality.

In this paper, we present a real-world problem which arises in a company specialized in drug evaluation and pharmacology research. The problem at hand is to build fine rosters which respect legal and organizational constraints. This task is currently hand-performed on a weekly basis by the chief nurse, which is very time-consuming.

The remainder of the paper is organized as follows: section 2 is devoted to the description of the problem, section 3 presents some related works, in section 4 we

LUNAM Université, École des Mines de Nantes, IRCCyN (UMR CNRS 6597), 44307 Nantes Cedex 3, Nantes (France),
E-mail: {tanguy.lapegue, damien.prot, odile.morineau}@mines-nantes.fr

propose a modelling of our problem, section 5 describes some branching strategies which aim at finding quickly a good solution. Our approach is validated by experimental results in section 6.

2 Problem description

The company carries out clinical studies on behalf of pharmaceutical laboratories which need to control the impact of newly developed drugs on the human body. The company recruits volunteers who are hospitalized during the whole study so that nurses could perform each required task on each volunteer. Laboratories deliver to the company a very specific study procedure which contains a description of all the clinical tasks to be performed, along with their relative starting time and duration. Based on this protocol, the company needs to assign these tasks to qualified and available employees. The individual plannings resulting from this task assignment have to respect a set of legal and organizational constraints, and also, up to a point, they are expected to be as fair as possible, which makes the scheduling task very difficult and time-consuming for the chief nurse.

During a week, employees divide their working time between three kinds of job:

1. **Clinical tasks** are fixed by the protocol and must be performed at the given starting minute, which means that the granularity of the problem drops to the minute. These tasks are fixed whenever during the day of the week and the hour of the day. The chief nurse has to assign one employee to every clinical task.
2. **Compulsory administrative tasks** are also fixed but they are already assigned before the shift-building procedure. This kind of tasks, such as meetings and trainings, are counted as working time, but not as clinical working time, and they could be assigned to more than one employee (in case of meetings within the company). These assignments lead to fixed periods of clinical unavailability that have to be included in designed shifts.
3. **Free administrative tasks**, such as medical reports writing, are not fixed and do not require any specific skills. Each employee has a specific set of administrative tasks to do during the week, and they are free to work on it whenever they want, provided they are not already assigned in the mean time to clinical or compulsory administrative tasks. Consequently, these tasks do not appear in the final timetable.

One characteristic of the problem lies in the lack of fixed shifts: employees could start and end their day whenever it is necessary, provided the resulting shift sequences respect the set of hard constraints due to work regulation, company organization, and nurses agreements. Consequently, working days refer to working periods and they may overlap two calendar days if needed. The main constraints are summarized in the following.

Organizational constraints

- **HC 1:** Employees cannot perform tasks which require unmastered skills.
- **HC 2:** Employees cannot perform tasks while unavailable.
- **HC 3:** Every clinical task must be assigned to one employee.
- **HC 4:** The assignment of compulsory administrative tasks must be respected.
- **HC 5:** Employees must finish a task before starting another one.

Work regulation / nurses agreements constraints

- **HC 6:** The daily working load must not exceed 10 hours.
- **HC 7:** The weekly working load must not exceed 48 hours.
- **HC 8:** The duration of a working day must not exceed 11 hours.
- **HC 9:** The duration of a rest period must not be less than 11 hours.
- **HC 10:** The duration of the weekly rest must not be less than 35 hours.
- **HC 11:** Series of consecutive working days must not exceed 6 days.
- **HC 12:** Depending on their working days, nurses have different breaks.

During the task assignment process, the chief nurse takes into account this whole set of hard constraints which might lead to dead end, meaning that the problem admits no solution. In this case, the chief-nurse strengthen the workforce with externals, who abide by the same rules as regular workers.

Objective function

Hard constraints must be satisfied, but they only ensure the feasibility of schedules. In order to build fine schedules, the chief nurse takes into account multiple criteria based on equity among employees. In this paper, we consider the most important of these criteria: we want to share the workload resulting from clinical and writing tasks in a fair way. However, some nurses are less confident with the writing of medical reports than others. As a consequence, the chief nurse associates to each employee a weekly targeted clinical load: nurses who are less confident with writing tasks have a higher targeted clinical load in order to counterbalance the higher administrative load of nurses who are more confident with writing tasks. The idea of our objective is to fit with this targeted clinical load in order to get "fair" schedules. As a consequence our objective is to minimize the difference between the highest and the lowest nurse's gap value which is defined as the difference between the clinical targeted time and the clinical assigned time. A small difference between the highest gap and the smallest gap means that the workload is well balanced among nurses.

Dimensions of the problem

A typical problem involves 200 tasks, whose durations range from 5 minutes to 4 hours, which have to be assigned to about 20 nurses whose set of skills is closed to 30 different skills. One characteristic of our problem lies in the time granularity which drops to the minute over a scheduling horizon of a week. This might look like an excess of precision, but it is very important for the company to follow scrupulously the given protocol. To highlight this fact, one can state the use of synchronised clocks in the whole building.

Problem studied

As a first step towards the resolution of this industrial problem, we propose to focus on the problem of designing schedules and assigning tasks to the regular workforce, i.e. externals are not taken into account. By doing this, we simulate the first step of the hand-performed resolution process which is done by the chief nurse. In order to provide as much information as possible regarding to needs on externals, we relax the constraint **HC 3**, which allows us to find solutions with an incomplete assignment of tasks to workers.

Constraints related to nurses breaks are particularly complicated because they deal with several kind of breaks which depends on the kind of shift performed:

- some of them have to occur on a given time window whereas others are not time constrained,
- some of them are included in the working time whereas others are not,
- some of them last a few minutes whereas others last one hour.

From our point of view, ensuring these breaks would require heavy constraints without improving much schedules. This may be explained in two steps:

1. because of the fixed start and end of clinical tasks, it is highly probable that short breaks will be respected automatically,
2. the chief nurse pointed out that nurses are very flexible regarding to their breaks, and they are free to exchange some tasks in order to improve their schedules.

Consequently, in this paper, we decided to put aside the constraint **HC 12**. However, warnings are displayed in a post-processing step to alert the chief nurse about these breaks.

In this paper, compulsory administrative tasks are considered to be fixed in time because they are scheduled by the chief nurse before the assignment of clinical tasks, which is the part of the problem we focus on. It would have been possible to give some freedom to these tasks in order to schedule them during the resolution, but it would have make the problem much harder to solve. Consequently, we consider them as data, which is exactly the way it is done in the company.

3 Related work

Over the years, many approaches have been proposed in order to model and solve Personnel Scheduling Problems (see [12] for an overview). Mathematical models, usually based on the Dantzig set-covering formulation, often achieve the lowest cost solutions but they are difficult and time-consuming to implement. In particular, specific constraints and objectives may be difficult to express easily. Consequently, research often focus both on simplified problems and general methods. In [3] for instance, the authors proposed a general mathematical model covering several personnel scheduling problems. Another important trend has been to develop implicit modeling in order to tackle more complex problems. In [2] for instance, the authors present an implicit integer linear programming formulation for the inclusion of meal/rest-break flexibility. On the contrary, metaheuristics, such as Tabu Search, Simulated Annealing and Genetic Algorithms, offer the opportunity to incorporate problems specificities. These approaches do not guarantee the optimality of the solution, but they are quite robust and they could be adapted even to the smallest problem specificity. They have been successfully tested to varying kind of real-world problems (see for instance [5] and [8]). Constraint Programming (CP) offers a promising alternative: CP is very close to a form of declarative programming, and consequently, it offers very powerful tools to state complex problems and produce flexible implementations, which is desirable in a business context. For instance, the Nurse Rostering Problem (NRP), and also the Crew Rostering Problem (CRP), which are some of the most constrained personnel

scheduling problems, belong to the set of problems that could be effectively stated as a constraint satisfaction problem (see [21] and [9]).

Tour Scheduling Problem
Personnel scheduling problems could be adressed in two steps: the first step, referred to as the days-off problem, aims at assigning days-off to workers whereas the second one, referred to as the shift scheduling problem, consists of assigning shift sequences to workers. The combined version of these two problems is referred to as the Tour Scheduling Problem ([17]). This last problem offers a wide range of combinations which enables substantial improvements in the labor utilization. Consequently, many approaches have been proposed and tested (see [1] for an overview).

Loucks and Jacobs ([16]) present a heuristic approach to the dual problem of Tour Scheduling and Task Assignments involving workers who differ in their availabilities and qualifications. In this problem, two ranked objectives are considered: the first one is the minimization of the total man-hours of overstaffing, and the second one is the minimization of the sum of the squared differences between the number of tour hours scheduled and the number targeted for the workers. However, the approach requires to divide the horizon into one-hour periods, which is far too big for our own problem. More generally, to our knowledge, the Tour Scheduling Problem with one-minute periods has not been studied yet. This may be explained in three steps, as mentioned by [16]: first of all, working with one-minute periods means that the demand is known with a one-minute precision, which makes no sense if the demand comes from forecasting methods. Besides, managers would be under the obligation to make efforts so that employees respect their schedules. Finally, managers often prefer to keep some flexibility in schedules in order to cope with potential delays, and other uncertainties.

Fixed Job Scheduling Problem
Given a set of jobs along with their fixed starting times and processing times, the Fixed Job Scheduling Problem (FJSP), also known as the Interval Scheduling Problem, consists of deciding whether or not to accept a job, knowing that chosen jobs have to be assigned to available ressources (see [15] for an overview). The assignment part of our problem could consequently be seen as a FJSP, where every job has to be accepted. In this last case, deciding whether a feasible schedule exists is NP-Complete ([15]).

As pointed out in [15], the FJSP constitutes the core of a variety of applications, such as the well-known Crew Rostering Problem. For instance, in [9] the authors propose a mixed approach based on CP and OR techniques to solve a specific CRP. However, they consider only the rostering step, meaning the sequencing of the given duties, and not the scheduling step, meaning the generation of duties. Moreover, many CRPs integrate some constraints on the feasible sequences of tasks in order to take into account the geographical location of the tasks, which is not relevant in our case.

However, CRPs also consider the scheduling of ground station personnel, which is closer to our problem since the geographical location is absent. In [11] the authors present a decision support system designed for the aircraft maintenance departement of KLM Royal Dutch Airlines. However the scheduling problem of KLM is different from our own problem in several points: first of all, ground station personnel operates in a four-shift system with fixed shifts. Besides, members of the same team are assigned to

the same shifts. In [4] the authors present an approach in two steps based on column generation and simulated annealing in order to schedule the work of the ground station personnel in airport. In order to solve this problem, they used a 15-minutes period interval on a weekly horizon, which is too coarse-grained for our problem. Moreover, they assume that shifts of the same tour are the same, which means that employees are assigned to the same shift during the whole week.

Nurse Rostering Problem
The core of NRPs is to assign shifts to nurses over a scheduling period so that the resulting rosters respect a set of constraints and cover the demand. Most NRPs are NP-Complete [13] and real-world applications, due to their large set of specific constraints, are very challenging and hard to solve.

Our problem clearly shares some characteristics with the NRP as described in [6]: constraints **HC 1** and **HC 2** are common personnel constraints of the NRP, constraints **HC 3** to **HC 5** can be seen as coverage constraints and constraints **HC 6** to **HC 11** are common work regulation constraints of the NRP. However, the core of the problem is different since both the demand and the work assignment are different: in the NRP, the demand is given by a number of nurses for each skill category and for each demand period, whereas in our problem, it is given by a set of tasks with any possible starting and ending times. Moreover, in the NRP, the work assignment corresponds to a shift assignment with usually a very limited number of shifts (see for instance [7], [14] and [10]), whereas in our problem, the work assignment is given by a tasks assignment from which shifts have to be deduced. In addition to these differences, NRPs usually consider patterns to penalize or favour, such as consecutive night shifts, stand-alone day-off shift or complete weekends, which are not taken into account in our problem. Consequently, our problem is much closer to the Tour Scheduling Problem ([17]) and from the Fixed Job Scheduling Problem ([15]) than from the NRP.

The problem of designing schedules by taking into account skills, availabilities and work regulation constraints in order to cover personnel requirements has been widely studied in several business environnements (NRP, TSP). In these problems, personnel requirements are usually given for a set of fixed time slots/shifts, whereas in our problem personnel requirements are given by a set of fixed tasks which cannot be preempted. Translating our personnel requirements into the classical time index representation would lead to allow preemption which is not possible. On the contrary, the problem of assigning fixed tasks to ressources (FJSP) do not take into account some basic constraints related to work regulations, such as the minimal resting time between two worked days. Finally, the design of schedules for ground stationned personnel seems to be the closest problem in the related literature ([4]). However, in practice, additionnal constraints related to the internal organization of airports are often taken into account. Besides, even if the equity among workers may be an interesting objective in these kind of problems it is also often important to minimize iddle times in order to improve the productivity of workers, whereas in our problem it is required to let some iddle time so that nurses could work on their free administrative tasks.

On the whole, even if some related problems are relatively close to the one we consider, none of them allow to grasp its full complexity (the time granularity which drops to the minute, the lack of fixed shifts and the optimality criterion are good

Table 1: Data

Data	Definition
$\mathcal{N} = \{1, .., N\}$	Set of nurses
$\mathcal{D} = \{1, .., D\}$	Set of days
$\mathcal{T} = \{1, .., T\}$	Set of clinical tasks
$\mathcal{A} = \{T+1, .., A\}$	Set of compulsory administrative tasks
$\forall n \in \mathcal{N}, \mathcal{A}_n \subset \mathcal{A}$	Set of tasks assigned to the nurse n
$\mathcal{O} = \{O_1, .., O_{\text{card}(\mathcal{O})}\}$	Set of sets of overlapping tasks
$\forall t \in \mathcal{T} \cup \mathcal{A}, s[t]$	Starting minute (over the week) of task t
$\forall t \in \mathcal{T} \cup \mathcal{A}, e[t]$	Ending minute (over the week) of task t
$\forall n \in \mathcal{N}, Wo[n]$	Targeted clinical working time over the week
$\forall n \in \mathcal{N}, Lo[n]$	Number of worked days since the previous day-off
$\forall n \in \mathcal{N}, U[n] = \{u_0, .., u_k\}$	Periods of unavailability for employee n

Table 2: Variables

Variables	Definition
$Tu \subset \mathcal{T}$	Set of unassigned tasks
$\forall n \in \mathcal{N}$	For every employee n
$Ta[n] \subset \mathcal{T} \cup \mathcal{A}$	Set of weekly assigned tasks
$Bs[n] \in [\![0; 7980]\!]$	Start of the weekly break
$Ww[n] \in [\![0; 2880]\!]$	Weekly working load
$Go[n] \in [\![-2880; 2880]\!]$	Gap between $Ww[n]$ and $Wo[n]$
$\forall n \in \mathcal{N}, \forall d \in \mathcal{D}$	For every employee n and every day d
$Da[d][n] \subset \mathcal{T} \cup \mathcal{A}$	Set of daily assigned tasks
$At[d][n] \in [\![0; 660]\!]$	Daily attendance time
$Wd[d][n] \in [\![0; 600]\!]$	Daily working load
$Sh[d][n] \in \{\text{Off; Worked}\}$	Daily assigned shift
$Fi[d][n] \in \{\omega\} \cup \{s[t], t \in \mathcal{T} \cup \mathcal{A}\}$	Start of the daily working period
$La[d][n] \in \{\alpha\} \cup \{e[t], t \in \mathcal{T} \cup \mathcal{A}\}$	End of the daily working period

examples). Consequently, we propose in the following a dedicated method based on Constraint Programming. The use of CP is motivated by recent works which highlight its ability to tackle highly constrained problems with very specific objectives.

4 Constraints Modeling

Data and variables notations related to our model are presented in Tables 1 and 2. The main idea of the model is to assign tasks to nurses using set variables. Since every task is fixed, the search of a solution amounts to finding a weekly set of tasks for each nurse: the vector Ta gives, for each nurse, the set of assigned tasks. However, most of the constraints deal with daily work instead of weekly work, which requires additional variables. Consequently, the matrix Da gives, for each nurse and for each day, the set of assigned tasks. In order to check constraints over shift sequences, the matrix Sh, gives for each nurse and day whether it is a working day or a day-off. Matrices Fi and La stand for the starting and ending time of each day of each nurse. Domains

of Fi (respectively La) correspond to the starting (respectively ending) time of tasks. In order to deal with days off, domains of Fi (respectively La) are completed with a constant $\omega = 8 \times 24 \times 60$ (respectively α = -24 $\times$ 60). The matrix At gives for each nurse and for each day the minimal attendance time (i.e. the attendance time resulting from clinical and compulsory administrative tasks). Finally, the vector Bs gives for each nurse the starting time of the weekly break.

Based on these variables, organizational and legal constraints could then be written. Some constraints refer directly to legal or organizational constraints whereas others simply ensure the consistency between variables. The first set will be referred to as *business constraints* whereas the second set will be denoted by *channeling constraints*. Finally, some legal constraints could be ensured from the creation of the variables, by reducing their domain of definition. This last set of constraints will be referred to as *preprocessing constraints* since once stated, they do not impact the solver anymore. In the following, *preprocessing constraints*, *business constraints* and finally *channeling constraints* are explained.

Preprocessing constraints

The domains of Ta and Da are reduced during the creation of variables by comparing both the mastered skills of each employee with the required skills of each task and the starting and ending times of each task with the availabilities of each employee. Basically, a task t which requires the skill s will be removed from the domain of the variable $Ta[n]$ if the nurse n does not master s. Besides, if the processing interval of t, which is given by $[s[t]; e[t]]$ overlaps any periods of unavailibility of employee n, which are given by $U[n]$, then t will also be removed from $Ta[n]$. The same process holds for Da. Moreover, the domain of the variables of Da could be even more reduced by taking into account the day of the week: tasks which are fixed on Wednesday, for instance, could not be performed on Monday, and so on. More precisely, a working period d could gather every task whose starting time belongs to the interval : $[60\times(6+24\times d); 60\times(6+24\times(d+1))]$, which corresponds to a period of 24 hours starting every day at 6 am. Consequently, employees who start working around 9 pm a day could finish their work at 6 am the following day. Thus, constraints **HC 1** and **HC 2** are verified from the creation of the problem, without any cost (i.e. the solver will not have to check these constraints during the resolution). The bounds of working period intervals have been fixed to correspond to the earliest starting time and the latest ending time applied by the company. Domains of Wd and Ww variables are also bounded from their creation, so that employees could neither be assigned to more than 10 hours of work over a day (**HC 4**), nor to more than 48 hours of work over the week (**HC 5**).

Business constraints

Constraints (1a) to (1c) aim at assigning the exact number of required employees to each task (**HC 3**). More precisely, constraint (1a) aims at assigning at least one employee to every task. Constraint (1b) ensures that tasks which have to be assigned to one employee, are not assigned to several employees. Constraint (1c) ensures that employees do not perform the same task several days. Constraint (2) ensures that the assignment of compulsory administrative tasks is respected (**HC 4**). Constraint (3) prevents employees from starting a new task before finishing the previous one (**HC 5**). Constraints (4) and (5) aim respectively at respecting the daily maximum attendance time (**HC 8**) and the minimum resting time between two working days (**HC 9**). The

constraint (6a) aim at building a weekly break of at least 35 hours (**HC 10**) by ensuring that no work is assigned to employees on an interval of 35 hours. Constraint (7) ensures that the maximal number of consecutive worked days does not exceed 6 days (**HC 11**).

$$Tu \cup \bigcup_{n \in \mathcal{N}} Ta[n] = \mathcal{T} \cup \mathcal{A} \tag{1a}$$

$$\forall (n_1, n_2) \in \mathcal{N}^2 \,|\, n_1 \neq n_2, \quad Ta[n_1] \cap Ta[n_2] = \mathcal{A}_{n_1} \cap \mathcal{A}_{n_2} \tag{1b}$$

$$\forall n \in \mathcal{N}, \forall (d_1, d_2) \in \mathcal{D}^2 \,|\, d_1 \neq d_2, \quad Da[d_1][n] \cap Da[d_2][n] = \emptyset \tag{1c}$$

$$\forall n \in \mathcal{N}, \quad \mathcal{A}_n \subset Ta[n] \tag{2}$$

$$\forall d \in \mathcal{D}, \forall n \in \mathcal{N}, \forall O_i \in \mathcal{O}, \quad \text{card}(O_i \cap Da[d][n]) \leq 1 \tag{3}$$

$$\forall d \in \mathcal{D}, \forall n \in \mathcal{N}, \quad At[d][n] \leq 60 \times 11 \tag{4}$$

$$\forall d \in \mathcal{D} \setminus \{D\}, \forall n \in \mathcal{N}, \quad Fi[d+1][n] - La[d][n] \geq 60 \times 11 \tag{5}$$

$$\forall n \in \mathcal{N}, \forall d \in \mathcal{D}, \quad (Fi[d][n] \geq Bs[n] + 35 \times 60) \vee (La[d][n] \leq Bs[n]) \tag{6a}$$

$$\forall n \in \mathcal{N}, \quad \text{card}\{d \in [\![0; 6 - Lo[n]]\!] \,|\, Sh[d][n] = \text{Off}\} \geq 1 \tag{7}$$

Channeling constraints

Constraint (8) ensures that the daily and the weekly assignments of each employee are consistent. Constraints (9) and (10) ensure that the weekly working load and the daily working load of each employee correspond to the task assignment. Constraint (11) ensures that the matrix Sh is consistent with the matrix Da. Constraints (12a) and (12b) aim at finding the beginning and the end of the working days of each employee when daily assigned sets are not empty. Empty sets, which refer to days-off, are handled by constraints (13a) and (13b). This last point might need some deeper explanations: first of all, empty sets have to get a starting and ending times because of the construction of Fi and La. In order to respect constraints **HC 8** and **HC 9**, Fi and La are assigned respectively to the end and the beginning of the week. This setting is also consistent with constraint (6a). Constraint (14) aims at computing the daily attendance time of each employee: for non-empty sets, the attendance time corresponds to the difference between the end and the start of the corresponding day, for empty sets, the attendance time corresponds to 0. Finally, constraints (15a) to (15d) calculate the objective value. In constraint (15a) we substract compulsory administrative time from the weekly workload, in order to keep only the clinical workload.

$$\forall n \in \mathcal{N}, \quad \bigcup_{d \in \mathcal{D}} Da[d][n] = Ta[n] \tag{8}$$

$$\forall n \in \mathcal{N}, \quad Ww[n] = \sum_{t \in Ta[n]} e[t] - s[t] \tag{9}$$

$$\forall n \in \mathcal{N}, \forall d \in \mathcal{D}, \quad Wd[d][n] = \sum_{t \in Da[d][n]} e[t] - s[t] \tag{10}$$

$$\forall n \in \mathcal{N}, \forall d \in \mathcal{D}, \quad \text{card}(Da[d][n]) > 0 \Leftrightarrow Sh[d][n] = \text{Worked} \tag{11}$$

$$\forall n \in \mathcal{N}, \forall d \in \mathcal{D}, \quad \text{card}(Da[d][n]) > 0 \Rightarrow Fi[d][n] = \min_{t \in Da[d][n]} s[t] \tag{12a}$$

$$\forall n \in \mathcal{N}, \forall d \in \mathcal{D}, \quad \text{card}(Da[d][n]) > 0 \Rightarrow La[d][n] = \max_{t \in Da[d][n]} e[t] \tag{12b}$$

$$\forall n \in \mathcal{N}, \forall d \in \mathcal{D}, \quad \text{card}(Da[d][n]) = 0 \Leftrightarrow Fi[d][n] = \omega \tag{13a}$$

$$\forall n \in \mathcal{N}, \forall d \in \mathcal{D}, \quad \text{card}(Da[d][n]) = 0 \Leftrightarrow La[d][n] = \alpha \tag{13b}$$

$$\forall d \in \mathcal{D}, \forall n \in \mathcal{N}, \quad At[d][n] = \max(La[d][n] - Fi[d][n], 0) \tag{14}$$

$$\forall n \in \mathcal{N}, \quad Go[n] = (Ww[n] - \sum_{t \in \mathcal{A}_n} e[t] - s[t]) - Wo[n] \tag{15a}$$

$$\forall n \in \mathcal{N}, \quad G\text{min} \leq Go[n] \tag{15b}$$

$$\forall n \in \mathcal{N}, \quad G\text{max} \geq Go[n] \tag{15c}$$

$$Obj = 10\ 000 \times card(Tu) + G\text{max} - G\text{min} \tag{15d}$$

Choice of the model

This model is oriented from a task assignment point of view, meaning that our concern is to assign tasks to nurses. Based on this assignment, additional information (such as starting times, ending times, daily load, weekly load, etc...) could then be easily deduced. Another way of modeling the problem is to assign one nurse to every task. However, a lot of constraints which are easily written with task sets, seem more difficult to write with such a model. Consequently, we decided to focus on the task assignment instead of the nurse assignment. Another common way of modeling personnel scheduling problems with constraint programming is based on the use of a general matrix giving the activity of each nurse (on lines) for each time period (on rows). For instance, The Nurse Rostering Problem has been stated and solved by constraint programming in such a way ([21]). However, this kind of approach requires to consider identical time periods, or slots. Since the time granularity of our problem drops to the minute over a planning horizon of a week, the use of identical time periods would lead either to an explosion of the number of variables or to some approximations on the duration of the tasks, which is not desirable. Consequently, our approach, based on task assignment seems to handle the complexity of the problem in a more promising way.

5 Variable and value strategies

When using constraint programming in order to build a good solution, it is very important to implement a dedicated search strategy. Basically, CP aims at finding a satisfying

solution, not the optimal one. However CP could still be used to find the optimum by solving the given problem in an iterative way: at each iteration, the cost of the solution is constrained to be less (when minimizing) than the cost of the last solution.

The aim of a dedicated search strategy is to avoid backtracking time in the huge space of unfeasible and unsatisfying solutions, by using the specificities of the problem. Once the filtering process is over, if every variable is not yet instantiated, the solver has to branch on a variable. The variable ordering strategy (VarOS) aims at choosing the most promising variable in order to branch on this variable, and the value ordering strategy (ValOS) aims at starting the branching with the most promising value. This value is chosen from the domain of the chosen variable. In our model, set variables are represented by two sets: the kernel and the envelope. The first one represents the set of values which belong to every solution whereas the second one refers to the set of values which belong to at least one solution. Consequently, the kernel of a set variable is a subset of its envelope. The chosen value must belong to the envelope, but not to the kernel. This set of possible values is called the open domain.

Concerning set variables, the *default* strategy select the variable with the smallest open domain, and among this domain, it selects the smallest value. However, this strategy do not use the specificities of the problem. Consequently, two VarOS and five ValOS are proposed and discussed in the following.

VarOS Among the variables of the vector Ta:

- Choose the variable corresponding to the nurse whose numerical difference between the assigned clinical working time and the targeted clinical working time is the smallest. The idea of this VarOS, which will be referred to as LW (*Less Working*), is to start assigning tasks to the less working nurse, as soon as possible, in order to improve the solution.
- Find the variable corresponding to the nurse whose weekly working load is the highest among those which are under a fixed limit, controlled by a parameter $\bar{l}$, expressed in minutes. More precisely, among the variables respecting the following inequality:

$$W_w[n] - W_o[n] \leq \bar{l}$$

  choose the variable which maximizes $(W_w[n] - W_o[n])$. If such a variable does not exist, choose a variable randomly. This strategy, which will be referred to as MW (*Most Working*), could be used in two different ways depending on the value of $\bar{l}$. For instance, setting $\bar{l}$ to a positive value allow the solver to keep assigning tasks to nurses who are already above their targeted clinical load, in order to keep room for further assignments and consequently, avoid dead ends. Consequently, this will not lead to well-balanced solutions, but the idea is to evaluate the number of feasible solutions which could be found by this way. On the contrary, setting $\bar{l}$ to a negative value aim at ensuring a minimum working load for each nurse. Even if setting $\bar{l}$ to a negative value is also a way of finding good solutions, there is an important difference between LW and MW: LW tries to improve the solution at every node whereas MW allow the solver to deteriorate the solution, hoping to improve it in

the end.

ValOS Among the *open domain* (referred to as $O(Ta[n_c])$) of the chosen variable:

- Find the task which produces the highest increase of work density. The work density of a day corresponds to the working load of this day divided by the attendance time of the day. If the highest increase of work density is higher than a given limit controlled by a parameter $\underline{\delta}$, then choose the corresponding task, else choose the first possible task. More formally, for each possible task t, we can compute the new daily working load $W_d^{t+}[d][n_c]$ and the new attendance time $At^{t+}[d][n_c]$ resulting from the addition of t to the corresponding working day of n_c. We refer to the increase of density produced by t as $\delta[t]$:

$$\delta[t] = \frac{W_d[d][n_c]}{At[d][n_c]} - \frac{W_d^{t+}[d][n_c]}{At^{t+}[d][n_c]} \in [-1; 1]$$

  Among tasks whose corresponding $\delta[t]$ is above $\underline{\delta}$, choose the task t which maximizes $\delta[t]$. If such a task does not exist, choose the first possible task. The idea of this ValOS, which will be referred to as *ID* (*Increase Density*), is to produce compact schedules, in order to avoid wasting time.
- Choose the task which belongs to the biggest set of overlapping tasks. The idea of this ValOS, which will be referred to as *BO* (*Biggest Overlap*), is to assign as soon as possible tasks which are fixed on activity peaks. Since employees cannot work simultaneously on overlapping tasks, choosing such a task may also be interesting because of the filtering process which may lead to many deductions (the employee could not anymore perform the other overlapping tasks).
- Choose the task which can be performed by the smallest number of nurses. The idea of this ValOS, which will be referred to as *LN* (*Lack of Nurses*), is to avoid backtracking procedure by avoiding dead ends.
- Find the task with the biggest duration. If this duration is higher than a given limit controlled by a parameter $\underline{p}$, then choose this task, else choose the first possible task. More precisely, among the tasks which respect the following inequality:

$$e[t] - s[t] \geq \underline{p}$$

  choose the longest task. The idea of this ValOS, which will be referred to as *BT* (*Biggest Task*), is to assign biggest tasks as soon as possible.

Mixed Strategy

Based on these simple heuristics we implement a more complex one, which will be referred to as *Mx* (*Mixed*). The idea of this ValOS, which is presented in detail in Algorithm 1, is to use the previously described heuristics in a combined way. More precisely, the global idea of *Mx* is to avoid dead ends by using *Lack of Nurses* and *Biggest Overlap*, then, if possible choose an interesting task by using *Biggest Task* and *Increase Density*, otherwise, choose a task which has a big impact on the search by using *Biggest Overlap*.

In the first step of *Mx* (lines 1 to 5), we compare the number of employees available for a task t with the number of tasks overlapping t. The idea is to avoid the assignment of employees available for t to other tasks. In the second step (lines 6 to 10), we compare

the highest processing time with the parameter $\underline{p}$. The idea of this threshold is to focus on the duration of tasks only when it is a highly distinguishing criterion. In the third step (lines 11 to 15), we compare the highest increase of density with the parameter $\underline{\delta}$. Again, this comparison aims to avoid making a decision on a hardly distinguishing criterion. Finally (lines 16 to 18), we choose the task which corresponds to the highest activity peak.

Algorithm 1 Mx ValOS: combine simple heuristics to choose the most promising task

Require:
$\underline{p} \in \mathbb{N}$, $\underline{\delta} \in [-1; 1]$
$\forall t \in O(Ta[n_c]), available[t]$: the number of available employees for task t
$\forall t \in O(Ta[n_c]), overlapping[t]$: the number of tasks overlapping task t
$findIndex(value, vector)$: returns the index of $value$ in the given $vector$.

1: **for all** $t \in O(Ta[n_c])$ **do**
2: **if** $available[t] \leq overlapping[t]$ **then**
3: **return** t
4: **end if**
5: **end for**
6: $maxP \leftarrow \max_{t \in O(Ta[n_c])} e[t] - s[t]$
7: $tMaxP \leftarrow findIndex(maxP, p)$
8: **if** $maxP \geq \underline{p}$ **then**
9: **return** $tMaxP$
10: **end if**
11: $maxD \leftarrow \max_{t \in O(Ta[n_c])} \delta[t]$
12: $tMaxD \leftarrow findIndex(maxD, \delta)$
13: **if** $maxD \geq \underline{\delta}$ **then**
14: **return** $tMaxD$
15: **end if**
16: $maxO \leftarrow \max_{t \in O(Ta[n_c])} overlapping[t]$
17: $tMaxO \leftarrow findIndex(maxO, overlapping)$
18: **return** $tMaxO$

6 Experimental results

We implemented our model with Choco, a Java Constraint Satisfaction Problem Solver [20]. Each instance has been runned on an Intel Core i3 (3.06 GHz) with a time limit of 5 minutes under *default* (see section 5) and dedicated ordering strategies, in optimization. The time limit has been fixed to 5 minutes because the company would like to use the method as a simulation tool, which requires great responsiveness.

6.1 Instances generation

In order to test our model, we have generated 720 instances gathered in 24 sets of 30 instances. Each set of instances corresponds to a specific combination of three parameters: the number of tasks, the kind of skills required and the tightness of the workload compared to the work capacity. The number of tasks ranges from 100 to 400 which

allows us to check the behavior of the method for normal activity (200/300 tasks) and extreme cases (100/400 tasks). Moreover we have generated two kinds of skills requirements: the first one considers only common skills, meaning that each skill is mastered by most of the nurses whereas the second one considers also rare skills, meaning that some skills are mastered by only a few nurses. For each instance the targeted clinical workload of each employee is generated in order to fit to the global workload (+/- 5h). The tightness is defined as the average clinical working time of nurses: a usual tightness does not exceed 700 minutes. The first set of instances has a tightness of 600 minutes, which corresponds to a usual workload. The second one has a tightness of 800 minutes which corresponds to a big workload. The tightness of the last set of instances amounts to 1000 minutes which aims at testing the limit of the model. Finally, the task distribution and profile are also based on realistic data: tasks are distributed all along the week, with a density peak around 8 a.m. and three kinds of tasks, with specific probabilities of occurrence:

1. small tasks ranging from 5 to 15 minutes, with a probability of occurrence of 5%.
2. medium tasks with a processing time close to one hour, with a probability of occurrence of 65%.
3. big tasks ranging from 2 to 5 hours, with a probability of occurrence of 30%.

Then, a simple procedure computes the required number of workers along with their personal data (skills, availabilities, etc...). It ensures that the tightness of the instance is respected but it does not ensure the feasibility of the instance. Consequently some instances do not admit a complete assignment. In the following, we define the size of an instance as its number of tasks.

6.2 Parameters design

Parameter $\bar{l}$ (in minutes) has been tested with values: -180, 0 and 180. Increasing $\bar{l}$ leads some ValOS, such as BO, to a higher number of complete assignments and a smaller number os unassigned tasks. However this improvement is obtained at the expense of the mean equity value. Moreover, some strategies such as BT and Mx do not profit from this increase of $\bar{l}$, on the contrary, it leads only to worsen the equity.

Parameters $\underline{p}$ and $\underline{\delta}$ have been tested separately within the BT (*Biggest Task*) and the ID (*Increase Density*) ValOS respectively.

Parameter $\underline{\delta}$ has been tested from -1 to 0 with a range of 0.1. We did not try to set $\underline{\delta}$ to strictly positive values, because it seems unlikely to be possible to use this strategy during the whole search. Results show no significant differences for these various settings, which means that making a big increase of density is not so important compared to completing working days (whether it is by increasing or decreasing the density). In order to get a well balanced Mx strategy, we set $\underline{\delta}$ to 0, but when used inside ID we set it to -1.

Parameter $\underline{p}$ (in minutes) has been tested with values 5, 30, 50, 120, 180 and 240 which enables us to consider either every task or only subsets of tasks. Increasing $\underline{p}$ leads to a higher number of complete assignments and a smaller number of unassigned tasks, but it worsen the equity. The best solutions (considering equity) are obtained with values 5 and 30. Above 30 the equity value starts to decrease. Consequently, we set $\underline{p}$ to 30. By doing this, we put aside small tasks whose durations range from 5 to

30 minutes, but when used inside BT we set it to 5.

6.3 Results analysis

In order to evaluate our results, which are presented in detail in Table 3, we used 4 indicators:

1. "Complete": the number of solutions with a complete assignment.
2. "Equity": the mean equity value of best solutions, in minutes (over complete assignments only).
3. "Left": the average number of unassigned tasks.
4. "Time": the mean computation time of best solutions, in seconds (over complete and incomplete assignments).

Table 3: Results (time limit: 5 min)

Strategy	Indicator	Size				
		100	200	300	400	All
LW BO	Complete	27/180	43/180	45/180	52/180	167/720
	Equity	304	301	295	293	297
	Left	6	8	9	11	8
	Time	44	23	23	41	32
LW BT($\underline{p} = 5$)	Complete	30/180	43/180	64/180	61/180	198/720
	Equity	49	41	39	34	39
	Left	5	6	7	6	6
	Time	22	26	19	36	26
LW ID($\underline{\delta} = -1$)	Complete	57/180	89/180	104/180	108/180	358/720
	Equity	248	268	290	297	280
	Left	4	5	3	3	4
	Time	60	27	32	39	37
LW LN	Complete	54/180	75/180	90/180	93/180	312/720
	Equity	227	261	271	277	263
	Left	4	5	5	5	5
	Time	70	30	31	36	39
LW Mx($\underline{p} = 30, \underline{\delta} = 0$)	Complete	33/180	50/180	64/180	72/180	219/720
	Equity	42	43	40	36	40
	Left	5	6	6	6	6
	Time	35	5	16	37	23
LW All	Complete	71/180	101/180	116/180	124/180	412/720
	Equity	119	146	121	117	126
	Left	3	3	2	2	2
	Time	109	53	50	53	66
MW($\bar{l} = -180$) BO	Complete	43/180	56/180	60/180	64/180	223/720
	Equity	389	471	500	540	483

Strategy	Indicator	Size				
		100	200	300	400	All
	Left	5	8	7	8	7
	Time	73	40	54	95	66
MW($\bar{l} = -180$) BT($\underline{p} = 5$)	Complete	41/180	60/180	70/180	66/180	237/720
	Equity	286	363	390	414	372
	Left	5	7	5	5	6
	Time	96	58	59	78	71
MW($\bar{l} = -180$) ID($\underline{\delta} = -1$)	Complete	79/180	102/180	126/180	125/180	432/720
	Equity	354	439	514	544	476
	Left	4	5	3	3	4
	Time	75	74	73	106	83
MW($\bar{l} = -180$ LN	Complete	68/180	82/180	96/180	112/180	358/720
	Equity	349	477	551	579	505
	Left	4	5	4	4	4
	Time	101	89	94	100	96
MW($\bar{l} = -180$) Mx($\underline{p} = 30, \underline{\delta} = 0$)	Complete	45/180	63/180	75/180	77/180	260/720
	Equity	312	369	375	419	376
	Left	5	6	5	5	5
	Time	66	69	77	75	73
MW($\bar{l}$=-180) All	Complete	92/180	109/180	135/180	144/180	480/720
	Equity	283	365	414	458	380
	Left	3	3	2	2	2
	Time	146	129	127	147	137
MW($\bar{l} = 180$) BO	Complete	59/180	71/180	78/180	81/180	289/720
	Equity	802	1056	1113	1185	1056
	Left	4	6	6	7	6
	Time	151	109	148	142	138
MW($\bar{l} = 180$) BT($\underline{p} = 5$)	Complete	63/180	85/180	96/180	97/180	341/720
	Equity	903	1102	1217	1284	1150
	Left	5	6	5	4	5
	Time	146	138	134	140	139
MW($\bar{l} = 180$) ID($\underline{\delta} = -1$)	Complete	91/180	108/180	128/180	128/180	455/720
	Equity	783	1003	1132	1213	1054
	Left	3	4	3	3	3
	Time	141	143	135	148	142
MW($\bar{l} = 180$) LN	Complete	90/180	108/180	127/180	144/180	469/720
	Equity	763	989	1122	1190	1043
	Left	4	4	3	3	4
	Time	168	155	139	162	156
MW($\bar{l} = 180$) Mx($\underline{p} = 30, \underline{\delta} = 0$)	Complete	69/180	88/180	101/180	107/180	365/720
	Equity	908	1104	1212	1282	1149
	Left	5	6	5	4	5
	Time	140	143	139	141	141
MW($\bar{l} = 180$) All	Complete	103/180	123/180	139/180	152/180	517/720
	Equity	732	952	1063	1152	975

Strategy	Indicator	Size				
		100	200	300	400	All
	Left	2	2	1	2	2
	Time	222	226	223	226	224
All	Complete	106/180	127/180	142/180	153/180	528/720
All	Equity	205	260	210	238	228
	Left	2	2	1	1	2
	Time	256	246	244	251	249

Table 3 gives for each size the value of the various indicators depending on the branching strategy. Results obtained with $\bar{l} = 0$ are not presented because they stand between those obtained with $\bar{l} = -180$ and $\bar{l} = 180$. The column "All" gives for each configuration the sum of the indicator "Complete" along with the mean values of indicators "Equity", "Left" and "Time". The line *LW-All* (respectively *MW-All*) gives the results which can be found by using in parallel each ValOS with *LW* (respectively *MW*).

Results regarding to feasibility

The *default* strategy gives similar results on each size: it finds around 70 instances with a complete assignment with a mean equity of 1600 minutes in 90 seconds. Compared to the *default* strategy, dedicated strategies find more solutions, especially with the *MW* VarOS, which illustrates the importance of using dedicated strategies.

Generally speaking, the number of solutions is bigger with the *MW* VarOS than with the *LW* strategy, which is coherent with the idea of these strategies. The *ID* ValOS gives the highest number of solutions for both VarOS, which means that building compact schedules is a good lead to get a feasible solution. On the contrary, *BO* gives few solutions for both strategies. This comes from the data specificities: the activity peak turns around 8 a.m. every day, consequently, by systematically choosing these tasks over the others we set to many shifts around the same time slot, which makes the assignment of night tasks much more difficult. The average number of unassigned tasks turns around 5 which is quite small.

On the whole, small instances turned out to be more difficult than bigger instances, which comes from the variations of the number of employees: instances with 100 tasks which corresponds to a small industrial activity have a smaller number of available workers than instances with 400 tasks which corresponds to a big activity. On the whole, tested ValOS perform differently on each instance, which means that they can complete one another, at least to some extent. Consequently, a practical way of finding solutions is to solve the problem in series with various ValOS, which is highlighted by the lines *LW-All*, *MW-All* and *All-All*. For instance, using each ValOS in parallel with *LW* increases the number of found solutions by 54. Another way to make up for the lack of solutions is to combine simple heuristics, as we do with *Mx*.

Instances with rare skills are harder to solve than those with only common skills. On average the relative difference of the number of complete assignments between these two sets turns around 10 to 20%. However, the average number of unassigned tasks does not change between these two sets. Instances with a high tightness are much

more difficult to solve than those with a common tightness. On the whole, 96%, 85% and 39% of the instances with a respective tightness of 600, 800 and 1000 minutes are solved with a complete assignment. This means that the method encounters difficulties to cope with heavy workload. However the number of unassigned tasks do not increase much.

Results regarding to equity
On average, between first solutions and best solutions, the improvement of the objective is not very big, which may be explained by the choice of the method: CP aims more at finding a solution rather than improving a solution. Consequently it is very important to find a good solution from the beginning, especially for the biggest instances. *LW-BT* finds the best results regarding to the equity, but only on a very small subset of instances. The *LW-Mx* strategy finds more solutions than the *LW-BT* strategy, and their value is relatively close to the best known value. Consequently, this heuristic is a successful combination of the more simple heuristics.

6.4 Operational point of view

On the whole, the method is quite fast: the first solution is found within a few seconds and the best solution is found in less than 3 minutes, which fits the requirements of the company. Given that some instances are not feasible without externals, the method finds a relatively good number of instances with a complete assignment and the number of unassigned tasks is very small. Depending on the running configuration, the number of complete assignments along with the value of the equity may be very different: *LW-BT* and *LW-Mx* strategies perform very well on equity but they have some difficulties to find complete assignments. On the contrary, *LW-ID* and *LW-LN* strategies perform relatively well on the assignment but not so well regarding to the equity.

7 Concluding remarks

We have presented a CP model along with several branching strategies in order to solve a real-world problem which shares similarities both with the Tour Scheduling Problem and the Fixed Job Scheduling Problem. For the sake of clarity we did not mention the problem into its full complexity, but this model could be extended in order to deal with additional legal constraints such as break constraints but also more complex equity objectives such as the distribution of night or weekend shifts which are also important for the company. This would lead to the interesting problem of defining a good solution when facing multiple objectives. This approach deals simultaneously with the task assignment problem and the design of personnel scheduling. We intend to compare this approach with a sequential one which deals with the design of personnel schedules before the task assignement problem.

We believe that our model could be improved in two ways. The first one is to work on the branching strategies which clearly have a big impact on results. For instance, it may be interesting to design a more complex variable ordering strategy, able to mediate between the need to improve a solution and the need to keep room to avoid

dead ends. The second lead is to strengthen the filtering process by using redundant constraints: set variables are a powerful tool which makes constraints easy to write, but they suffer from a weak filtering. Consequently, adding integer variables and channeling constraints may increase the filtering and therefore the performances of the model.

Our results show that branching strategies can successfully complete one another by being used, either in series or in a combined way. A more thorough study of the impact of parameters would allow to get sharpened conclusions, but the main conclusions remain: many results are good enough to be used by the company whereas others need to be improved. Since the model encounters some difficulties to improve a solution, it may be interesting to use another method such as a Large Neighbourhood Search ([19]), as a following of our method. Since the number of unassigned tasks is on average very small, it may be much more efficient to explore some targeted neighbourhood rather than following the basic tree exploration in order to find complete assignments. In order to evaluate in a better way the quality of our approach, future work may also focus on the search of optimal solutions and lower bounds regarding to the equity and the number of externals required to perform each tasks. However our optimal criterion makes this task difficult. Working on the sequential approach (shifts are fixed) makes this task a bit easier. In this context, we proposed two lower bounds for the equity value ([18]). However they do not correspond to the lower bound for the general problem studied in this paper and hence need to be adapted which seems very challenging.

References

1. Alfares, H.K.: Survey, Categorization, and Comparison of Recent Tour Scheduling Literature. Annals of Operations Research 127, 145–175 (2004)
2. Bechtold, S.E., Jacobs, L.W.: Implicit modeling of flexible break assignments in optimal schift scheduling. Management Science 36(11) (1990)
3. Brucker, P., Qu, R., Burke, E.K.: Personnel scheduling: Models and complexity. European Journal of Operational Research 210(3), 467–473 (2011)
4. Brusco, M.J., Jacobs, L.W., Bongiorno, R.J., Lyons, D.V., Tang, B.: Improving personnel scheduling at airline stations. Operations Research 43(5), 741–751 (1995)
5. Burke, E.K., Cowling, P.: A Memetic Approach to the Nurse Rostering Problem. Applied Intelligence 15(3), 199–214 (2001)
6. Burke, E.K., De Causmaecker, P., Berghe, G.V., Landeghem, H.V.: The state of the art of nurse rostering. Journal of Scheduling 7, 441–499 (2004)
7. Burke, E.K., Li, J., Qu, R.: A hybrid model of integer programming and variable neighbourhood search for highly-constrained nurse rostering problems. European Journal of Operational Research 203(2), 484–493 (2010)
8. Cai, X., Li, K.N.: A genetic algorithm for scheduling staff of mixed skills under multi-criteria. European Journal Of Operational Research 125, 359–369 (2000)
9. Caprara, A., Focacci, F., Lamma, E., Mello, P., Milano, M., Toth, P.: Integrating Constraint Logic Programming and Operations Research Techniques for the Crew Rostering Problem. Software Practice and Experience (28), 49–76 (1998)
10. Cheng, B.M.W., Lee, J.H.M., Wu, J.C.K.: Speeding Up Constraint Propagation By Redundant Modeling. In: 2nd Int. Conf. on Principles and Practice of Constraint Programming (1996)
11. Dijkstra, M.C., Kroon, L.G., Salomon, M., Van Nunen, J.A.E.E., van Wassenhove, L.N.: Planning the Size and Organization of KLM's Aircraft Maintenance Personnel. Interfaces 24(6), 47–58 (1994)

12. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Sier, D.: Staff scheduling and rostering: A review of applications, methods and models. European Journal of Operational Research 153(1), 3–27 (2004)
13. Garey, M.R., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman (1979)
14. Jaumard, B., Semet, F., Vovor, T.: A generalized linear programming model for nurse scheduling. European Journal Of Operational Research 2217(97) (1998)
15. Kolen, A.W.J., Lenstra, J.K., Papadimitriou, C.H., Spieksma, F.C.R.: Interval Scheduling: A Survey. Naval Research Logistics 54, 530–543 (2007)
16. Loucks, J.S., Jacobs, R.F.: Tour Scheduling and Task Assignment of a Heterogeneous Work Force: A Heuristic Approach. Decision Sciences 22(4), 719–738 (1991)
17. Mabert, V.A., Watts, C.A.: A Simulation Analysis of Tour-Shift Construction Procedures. Management Science 28(5), 520–532 (1982)
18. Prot, D., Lapègue, T., Bellenguez-Morineau: Lower bounds for a fixed job scheduling problem with an equity objective function. In: 25th European Conference on Operational Reserach (2012)
19. Shaw, P.: Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In: Principles and Practice of Constraint Programming (CP'98), volume 1520 of LNCS. pp. 417–431 (1998)
20. Team, C.: choco: an open source java constraint programming library. Research report, Ecole des Mines de Nantes (2010)
21. Weil, G., Heus, K., Patrice, F., Poujade, M.: Constraint Programming for Nurse Scheduling. Engineering in Medicine and Biology Magazine, IEEE 14(4), 417–422 (1995)