International Journal of Computational Intelligence Systems, Vol. 5, No. 1 (February, 2012), 111-125

Anesthesiology Nurse Scheduling using Particle Swarm Optimization

Leopoldo Altamirano [1] , María Cristina Riff [1] , Ignacio Araya [1] , Lorraine Trilling [2]

[1] *Departamento de Informática, Universidad Técnica Federico Santa María,*
Av. España 1680,
Valparaíso, Chile

E-mail: {Leopoldo.Altamirano, Maria-Cristina.Riff, Ignacio.Araya}@inf.utfsm.cl

[2] *Université de Lyon. INSA-Lyon, DISP*
Campus LyonTech La Doua, bt. Léonard de Vinci, 21 Avenue Jean Capelle,
Villeurbanne Cedex, 69621, France

*E-mail:*Lorraine.Trilling@insa-lyon.fr

Received 19 April 2011
Accepted 8 December 2011

Abstract

In this article we present an approach designed to solve a real world problem: the Anesthesiology Nurse Scheduling Problem (ANSP) at a public French hospital. The anesthesiology nurses are one of the most shared resources in the hospital and we attempt to find a fair/balanced schedule for them, taking into account a set of constraints and the nurses' stated preferences, concerning the different shifts. We propose a particle swarm optimization algorithm to solve the ANSP. Finally, we compare our technique with previous results obtained using integer programming.

Keywords: Artificial Intelligence, Heuristic Search, Particle Swarm Optimization, Nurse Scheduling

1. Introduction

Particle Swarm Optimization (PSO) was originally designed as a numerical optimization technique based on swarm intelligence. In the literature, there are a few attempts to exploit its usage in the discrete problem domain,[1] where binary encoding is usually used. Work on the transformation of the working mechanism of PSO to the permutation problem domain, where the representations are highly constrained, has been relatively limited.[2,3,4] This limitation is mainly caused due to the lack of a principled generalization of PSO to guide its adaptation to discrete combinatorial problems such as scheduling problems. In this paper, we design a PSO algorithm for a real world scheduling problem with discrete domains without losing the underlying principles of the original PSO.

The Anesthesiology Nurse Scheduling Problem (ANSP) tackled in this paper is a real world problem occurring in a French public hospital. The problem consists in assigning a group of nurses to a set of working shifts over a given planning period. This assignment has to fulfill several constraints such as work regulation, nurse coverage, competency adequacy and individual preferences. The objective is to optimize fairness among all the nurses. The main characteristic of this group of nurses is that they are organized into a team of equally skilled nurses who attend to various activities during the inpatient care process: preparation and monitoring of inpatients' anesthesia during the surgery, and monitoring the in-

patients when the come round in the recovery room. They are also able to perform activities which overlap with some surgical specialties. These nurses can be assigned to different activities from one day to another. The goal is to find a schedule where the distribution of the tasks is balanced throughout the nursing staff. With the current tendency of grouping together several surgical specialties in one operating, larger groups of nurses are formed. This considerably increases the complexity of the problem and intensifies the need for help provided to head nurses to solve the anesthesiology nurse scheduling problem. Integer programming and constraints programming approaches have been proposed to solve this problem[5] but have shown some limits when the number of nurses exceeds a certain threshold. Moreover, since the balance in the task distribution is a concept that can be difficult to translate into an objective function, we think that including additional orientation during the search could produce better schedules.

The objective of our investigation is twofold: first, to provide a new application of the PSO technique in discrete domains and to show its robustness for solving a problem arising in hospitals; and second, to enhance the quality of the scheduling solutions obtained by using a PSO algorithm during the search. In particular, we investigate how a new evaluation function based on the entropy concept can be used to guide the search. Our new evaluation function is a not linear one, thus, the methods used in our previous work (constraint programming and integer programming based methods) are not able to manage it directly. The new function allow one to obtain, in general, solutions with a better load distribution than those previously reported.

This paper is structured as follows. In the next section we describe the nurse scheduling problem and related work providing solution approaches. This is followed by a description of the real-world problem of scheduling anesthesiology nurses' work. In Section 3 we present the mathematical model of the problem. In Section 4 we introduce the components and mechanisms of the PSO algorithm that we propose. Section 5 presents the experimentations using real data, and finally, Section 6 gives the conclusions of our work.

2. Nurse scheduling problem

2.1. Related work

Assigning nurses to working shifts is a problem that every hospital in the world continuously has to solve. Nevertheless, the specificities of the problem vary from hospital to hospital, and even from ward to ward. Objectives could be to minimize the costs or to maximize the satisfaction of the personnel. The variety of practices among wards leads to a wide range of objectives and problem constraints. Some hospitals use cyclic schedules where the same schedule is repeated as long as the requirements do not change.[6] This requires a shift in patterns and each nurse's cycles, around a selection of patterns. These kinds of schedules are easy to design but may be very rigid and hard to adapt to changes. Other hospitals choose to work with non-cyclic schedules where a new schedule is generated for each scheduling period. This process is more time-consuming but much more flexible in dealing with the variability of demand, the staff requests or unexpected events such as absences.[7,8]

Nurse scheduling problems (NSP) or nurse rostering problems (NRP), which involves the creation of individual schedules, have been widely studied over the last decades. Problem description and models vary from study to study and depend on the need of each hospital. A multitude of solution approaches can be found in the operational research literature, based on, among other things, the wide range of models that can be built. Several recent and complementary bibliographic surveys have been published and give a good overview of the models used and the large range of existing approaches.[8,9,10,11]

Several studies have employed optimization methods to solve the NSP. Some authors use exact methods like linear, integer or mixed integer programming,[12] goal programming[13,14] or constraint programming.[15,16] Many recent papers tackle the NSP with metaheuristic methods such as genetic algorithms,[17] tabu search[13,7,17] or simulated annealing.[18] Vanhoucke and Maenhout[11] respond to the need for a benchmark database (highlighted in

the work of Burke et al.[9] and Cheang et al.[8]), to make the algorithm comparison easier. They also develop a method to characterize the NSP under study according to indicators dealing with preference distribution measures, among others.

In this paper we deal with the ANSP. The objective is to obtain a high-quality schedule where the load is well balanced between the nurses according to a criterion based on the drawbacks of each type of shift. Lorraine et al.[5] model the problem with integer linear programming and as a constraint satisfaction problem. Then, the problem is solved using LINGO and ILOG solvers.

We use the population-based metaheuristic PSO to find good schedules. Unlike linear programming or CSPs solvers, neighborhood-based methods (e.g., population-based and metaheuristic methods), allow us to work with nonlinear functions (in our case, the objective function is highly nonlinear). PSO has shown good results when it has been applied to problems with continuous domains of variables. This motivates us to apply the technique and to observe what occurs when it is applied in problems with discrete domains. Finally, the few parameters that PSO has (only two in our approach), is an important factor to choose PSO instead of other evolutionary technique.

2.2. ANSP Description

The surgical suite is a complex service consisting of two main parts: the operating rooms (ORs), and the post-anesthesia care unit (PACU). Different teams of nurses working in this service have to be scheduled: the operating room nurses, assisting surgeons during surgery; the anesthesiology nurses, taking care of inpatients during surgery or supervising the PACU; the registered nurses, taking care of inpatients during recovery; and finally, the auxiliary nurses, performing the logistic and cleaning tasks.

In this work we deal with the anesthesiology nurses. Anesthesiology nurses can work in a cross section of different surgical specialties and several types of surgery (scheduled cases, ambulatory cases or emergency cases). They work with anesthesiologists during surgery but also during the recovery time. In hospitals, they can perform surgical emergency services, sometimes working around the clock (on-call or on stand-by). They are also are required in the operating suite, where surgical teams have to be ready rapidly to welcome a new patient. Nurses can be assigned to either day or night shifts, and each day to one activity. This daily activity can change from one day to another. In our research we study a French hospital with a surgical suite containing 9 ORs. The part-time and full-time nurses are all equally skilled, and can be assigned to any of the day or night shifts, as shown in Table 1.

Table 1. Type of shifts, start and end time

Type of shift	start-end time
Day shift (DS)	8:00 - 16:00
Emergency day shift (EDS)	8:00 - 20:00
Emergency night shift (ENS)	20:00 - 8:00
PACU supervision shift (PS)	9:00 - 17:00; 11:00 - 19:00

The emergency shifts are on stand-by duty and have to be worked on each day of the week (including week-ends), whereas the other shifts are from Monday to Friday. The shifts involving the supervision of the PACU are considered as equivalent. The anaesthesiology head nurse draws up the anaesthesiology nurses' schedule in a non-cyclical process. First, the head nurse asks the nurses for their preferences concerning the day they would like to be off, and from these preferences he/she tries to plan schedules which satisfy all the constraints listed below. The constraints are related to shift coverage (C1), working time (C2 to C4), and good working practices (C5 and C6).

C1: Each shift on each day requires a specified number of nurses.

C2: Working hours must not exceed 12 hours per day.

C3: Working hours must not exceed 48 hours per week.

C4: A nurse cannot work more than three ENSs during a given week.

C5: A nurse works an EDS (resp. ENS) on Saturday if and only if he/she also works an EDS

(resp. ENS) on Sunday. He/she will then get the following Monday and Tuesday off.

C6: Sequence of activity constraints:

- If a nurse works an ENS on a *weekday* (i.e., one day between Monday and Friday), then the following day is free.
- If a nurse works an EDS on a weekday, then the following day could be either an ENS or free.

The coverage and time-related constraints C1 to C4 are considered as compulsory constraints that have to be respected (hard constraints). Usually, constraints C5 and C6, as well as the constraints of preferences expressed by the nurses, could be optional (soft constraints). Since our objective is to find a good schedule that maximizes the nurses' satisfaction, the required solution must ideally satisfy all the constraints from C1 to C6.

In order to generate the fairest schedule, the most and least preferred shifts have to be distributed among the nurses in a balanced way. To express the shift preferences, a penalty associated to each type of shift on each day of the week is defined. The penalties correspond to rational values ranging from 1 (most preferred shift) to 2 (least preferred shift) and have been generated according to the knowledge of the head nurses and the feedback from the nurses. The penalty values are shown in Table 2.

The PS has more severe constraints than the others since it requires a high level of vigilance during the patient's waking. With emergency night and day shifts, the load is concentrated on only a few days of the week, and these shifts are followed by a day off. Hence, even when ENSs and EDSs are 12-hour shifts, they are preferred to PSs. Finally, the classical DSs are the shifts preferred most, due to the convenience of the start and end working time, and to the work contents. DSs have the lowest penalty.

The best schedule might be the one that minimizes the gap between the penalty of the nurse with the heaviest load and that of the nurse with the lightest load, while satisfying all the constraints. We, however, show in Section 4 how the minimization of an entropy-based function can improve the load distribution among the nurses.

Table 2. Penalty associated to the shifts

Shift	Mon	Tue	Wen	Thur	Fri	Sat	Sun
DS	1	1	1	1	1	-	-
EDS	1.2	1.2	1.2	1.2	1.2	1.4	1.4
ENS	1.4	1.4	1.4	1,4	1.4	1.6	1.6
PS	1.6	1.6	1.6	1.6	1.6	-	-

In the following section we present a mathematical model for the ANSP.

3. ANSP models

In this section we detail two formulations for the ANSP: as a linear program and as a constraint satisfaction problem. These models are used by CPLEX in the experimentation section.

Both models can be expressed by means of the same parameters.

Parameters

N : Number of nurses to be scheduled

H : Number of days of the scheduling period

W : Number of weeks of the scheduling period

K : Number of shifts

b_{jk} : Number of nurses required for shift k on day j

p_{jk} : Penalty associated to shift k on day j

R_i : Working rate of nurse i (1 for full time, < 1 for part time)

T_{max} : Maximum number of hours that a full time nurse can work (48 hours)

n_k : Number of working hours within shift k

3.1. *Integer linear programming model*

The ANSP can be modeled as an integer linear program (ILP) in which the decision variables S_{ijk} are binaries.

Decision variables

$$S_{ijk} = \begin{cases} 1 & \text{if the nurse } i \text{ works shift } k \text{ on day } j \\ 0 & \text{otherwise} \end{cases}$$

Auxiliary variables

P_{max} and P_{min} are respectively the maximum and minimum penalties of all the nurses' schedules. $\forall i = 1..N$:

$$P_{max} \geqslant \sum_{j=1}^{H} \sum_{k=1}^{K} \frac{S_{ijk} p_{jk}}{R_i}$$

$$P_{min} \leqslant \sum_{j=1}^{H} \sum_{k=1}^{K} \frac{S_{ijk} p_{jk}}{R_i}$$

Objective Function

The objective is given by the minimization of the difference between the maximum and minimum nurse penalties:

$$\text{Min } Z = P_{max} - P_{min}$$

Constraints

In the following we give the formulation of the constraints.

Constraints C1 are related to the nurse coverage. The coverage must be exact, without neither shortage nor overcapacity, i.e., $\forall j = 1..H$, $\forall k = 1..K$,

$$\sum_{i=1}^{N} S_{ijk} = b_{jk}$$

Constraints C2 limit the number of daily working hours. This is done by assigning one single shift for each nurse on each day, i.e., $\forall i = 1..N$, $\forall j = 1..H$,

$$\sum_{k=1}^{K} S_{ijk} \leqslant 1$$

Constraints C3 limit the number of weekly working hours. They take into account the working rate R_i of the nurse i, as well as the number of hours n_k included in shift k: $\forall i = 1..N$, $\forall w = 1..W$,

$$\sum_{j=7(w-1)+1}^{7w} n_k S_{ijk} \leqslant T_{max} R_i$$

Constraints C4 limit the number of NDSs($k = 3$) per week: $\forall i = 1..N$, $\forall w = 1..W$,

$$\sum_{j=7(w-1)+1}^{7w} S_{ij3} \leqslant 3$$

First part of constraints C5 corresponds to a logical equality: $\forall i = 1..N$, $\forall w = 1..W$,

$$S_{i,7w-1,k} = S_{i,7w,k}$$

The free Monday and Tuesday constraints (if the shift type on weekend is EDS or ENS) are given by: $\forall i = 1..N$, $\forall w = 1..W-1$, $\forall k = 2,3$,

$$S_{i,7w,k} + \sum_{k'=1}^{K} S_{i,7w+1,k'} \leqslant 1$$

$$S_{i,7w,k} + \sum_{k'=1}^{K} S_{i,7w+2,k'} \leqslant 1$$

Finally, constraints C6 are also divided into two sets of constraints. The first of them indicates that an ENS must be followed by a free day: $\forall i = 1..N$, $\forall j = 1..5$, $\forall w = 1..W$,

$$S_{i,7(w-1)+j,3} \leqslant 1 - \sum_{k'=1}^{K} S_{i,7(w-1)+j+1,k'}$$

The second one corresponds to a disjunction. On a weekday, an ENS must be followed by day off: $\forall i = 1..N$, $\forall j = 1..5$, $\forall w = 1..W$,

$$S_{i,7(w-1)+j,2} \leqslant 1 - \sum_{k'=1}^{K} S_{i,7(w-1)+j+1,k'}$$

or by an ENS:

$$S_{i,7(w-1)+j,2} \leqslant S_{i,7(w-1)+j+1,3}$$

This disjunction can be transformed in the following single set of constraints: $\forall i = 1..N$, $\forall j = 1..5$, $\forall w = 1..W$,

$$S_{i,7(w-1)+j,2} \leqslant 1 - \sum_{\substack{k'=1 \\ k' \neq 3}}^{K} S_{i,7(w-1)+j+1,k'}$$

3.2. Constraints satisfaction problem model

The nurse scheduling problem can also be formulated as a constraints satisfaction problem (CSP) and solved using a constraint programming library (e.g., ILOG[19]).

Decision Variables

Constraint programming allows one to manipulate integer variables. In the CSP formulation of the ANSP, decision variables S_{ij} are integer and correspond to the shift assigned to nurse i on day j. This variable take a value k among the available shifts (1 for DS, 2 for EDS, 3 for NS and 4 for PS). The decision variables also can take the value 0 implying a free day. We also have to add the parameters $n_0 = 0$ and $p_{j0} = 0 \ \ \forall j = 1..H$.

Auxiliary Variables

The definition of P_{min} and P_{max} requires the definition of a set of new variables P_{ij} corresponding to the penalty associated to nurse i on day j: $\forall i = 1..N,\ \forall k = 0..4,\ \forall j = 1..H$,

$$\text{if } S_{ij} = k \text{ then } P_{ij} = p_{jk}$$

Note that the value of P_{ij} depends on the value assigned to S_{ij}. The domain of the variables P_{ij} is $D^P = \{0, 1, 1.2, 1.4, 1.6\}$.

P_{max} and P_{min} are defined as following: $\forall i = 1..N$,

$$\sum_{j=1}^{H} \frac{P_{ij}}{R_i} \leqslant P_{max}$$

$$\sum_{j=1}^{H} \frac{P_{ij}}{R_i} \geqslant P_{min}$$

Objective Function

The objective is the same as in the ILP model, i.e.: Min $Z = P_{max} - P_{min}$.

Constraints

In the CP approach, the constraints are globally the same as in the ILP approach. However, some differences can be noticed due to the method used. For instance, the constraints C2, limiting the working hours per day (i.e., the assignment of only one shift to each nurse on each day), are implicit in the CSP model since a variable can take only one value in its domain.

Constraints C1 related to the nurse coverage: $\forall j = 1..H, \forall k = 1..K$,

$$Card\left(\{i \in \{1,...,N\} | S_{ij} = k\}\right) = B_{jk}$$

where $Card(\star)$ corresponds to the cardinality (number of elements) of the set $\star$.

Constraints C3, limiting the working hours per week, require the introduction of new variables L_{ij} representing the length of the working day for nurse i on day j. These variables take a value in the domain $D^L = \{n_0, ..., n_4\}$ and are defined using *if-then* constraints: $\forall i = 1..N,\ \forall k = 0..4$,

$$\text{if } S_{ij} = k \text{ then } L_{ij} = n_k$$

Thus, constraints C3 become: $\forall i = 1..N,\ \forall w = 1..W$,

$$\sum_{j=7(w-1)+1}^{7w} L_{ij} \leqslant T_{max} R_i$$

Constraints C4 limit the number of night shifts: $\forall i = 1..N,\ \forall w = 1..W$,

$$Card\{j \in \{7(w-1)+1, ..., 7w\} | S_{ij} = 3\} \leqslant 3$$

More details and the definition of constraints C5 and C6 can be found in our previous work.[20] Also in this work, we report the results obtained by these models using several mixed-integer optimization softwares (LINGO, CPLEX, GLPK), and a branch-and-bound algorithm. CPLEX and LINGO have shown the best performances, both using the ILP model. In this paper we then decided to use CPLEX and the ILP model for our experimentations.

In the next section, we show how this problem can be solved using a PSO approach. We also propose an evaluation function, different from $P_{max} - P_{min}$, to guide the search. The results allow us to compare the performance of the different approaches on the same set of problems.

4. A PSO-based algorithm for ANSP

In this section we describe the components of our PSO-based algorithm. Our algorithm works with the same input data as the models above (see Section 3).

A PSO algorithm is a computational method that optimizes a problem by iteratively trying to improve a candidate solution. The algorithm works by having a population (called a swarm) of candidate solutions (called particles). Each particle has a *position* and a *velocity* and is moved around in the search-space in simple movements. These movements are guided by the **LocalBest** (i.e., their own best known position in the search-space) as well as the **GlobalBest** (i.e., the entire swarm's best known position). When improved positions are being discovered these will then come to guide the movements of the swarm. The process is repeated until an ending condition is reached.

In this article we use a simplified variant of the PSO described by Pendersen and Chipperfield.[21] This variant only uses the swarm's best known position to guide the particles (the particle's best known position is forgotten).

In order to apply PSO to the ANSP, we must be careful about the meaning of both the position and the velocity in our context. In a classical PSO, the position vector indicates the values of the variables belonging to a continuous domain.

In a first attempt we defined the position of a particle, using the scheduling of nurses' time of its related solution. However, the behavior of this kind of representation is very different from that of the position vector in continuous domains. It is reasonable to think that small changes of the variable values (particle's position) in continuous domains may imply small variations of the value of the objective function. This is a desirable behavior when we use PSO. In discrete domains, on the other hand, small changes in the solution (e.g., a swap in the scheduling of nurses) can produce unexpected results (much better or much worse). We therefore decided to represent the position of a particle as the *cost* of its related solution. The main problem of this representation is related to the difficulty in changing the position/cost in a desired direction. In Section 4.3 we propose an original and simple idea to solve this problem.

In our representation, the velocity of a particle has a probabilistic meaning and is represented by a single value V. Thus, a high value (resp. low) of V implies a high (resp. low) *probability* of bringing the particle closer to the GlobalBest position.

4.1. Evaluation function

As noted above, in previous researches[20,5] we used the evaluation/cost function $Z = P_{max} - P_{min}$ for computing the fitness of candidate solutions. P_{max} and P_{min} are, respectively, the maximum and minimum penalties among all nurses. The idea was to minimize Z to obtain a fair preference assignment. Now, in order to be more accurate, we change Z to a new entropy-based[a] evaluation function which not only considers the extreme penalty values but also takes into account the inequality of the assignments between P_{max} and P_{min}. Before introducing the evaluation function, we are required to introduce some definitions.

The scheduling of nurses is represented by a matrix $\mathbf{S}$ of size $N \times H$, where N is the number of nurses and H is the number of days. The value of the (i, j) entry of $\mathbf{S}$ (S_{ij}) corresponds to the type of assignment: (1:DS, 2:EDS, 3:ENS, 4:PS or 0:free) of nurse i on day j (similar to the CSP model shown in Section 3.2). Fig. 1 shows a scheduling matrix representing the scheduling of 4 nurses in a week. Consider for example the value of the (3,4) entry equal to 3, which means that nurse 3 works an ENS on

[a]The entropy function is defined by $S = -k_B \sum_{i=1}^{n} (\rho_i ln(\rho_i))$, where k_B is the well-known Boltzmann constant and ρ_i corresponds to a probability in $[0,1]$ such that $\sum_{i=1}^{n} \rho_i = 1$. It is used in thermodynamics. Basically, it measures the amount of uncertainty which remains about a system after its observable macroscopic properties (e.g., temperature, pressure and volume) have been taken into account. The maximum value of S is reached when all the probabilities are equal and it is given by: $S_{max} = k_B ln(n)$. This formula is known as *the most famous equation of statistical thermodynamics.*

Thursdays.

$$\begin{bmatrix} 3 & 2 & 0 & 3 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 2 & 2 \\ 0 & 3 & 0 & \textcircled{3} & 1 & 0 & 0 \\ 3 & 0 & 3 & 2 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} 4.6 \\ 4.4 \\ 4.4 \\ 4.6 \end{matrix}$$

Fig. 1. A scheduling matrix considering 4 nurses and 7 days. Last column shows the penalties associated to each nurse .

Definition 1. (Penalty of a nurse) Consider a solution/matrix $\mathbf{S}$ of size $N \times H$. The penalty P_i of nurse i is defined by the equation:

$$P_i(\mathbf{S}) = \sum_{j=1}^{H} \frac{penalty(j, S_{ij})}{R_i}$$

where $penalty(j, S_{ij}) = p_{jk}$ with $k = S_{ij}$, i.e., the penalty associated to shift S_{ij} on day j.

The last column in Fig. 1 corresponds to the *penalties* associated to each nurse using Table 2 and considering that all nurses are full time ($R_i = 1$).

Definition 2. (Average penalty) Consider a candidate solution/matrix $\mathbf{S}$ of size $N \times H$. The average penalty $\overline{P}$ of the solution is defined by:

$$\overline{P}(\mathbf{S}) = \frac{\sum_{i=1}^{N} P_i(\mathbf{S})}{N}$$

Each nurse having a penalty equal to $\overline{P}(\mathbf{S})$ implies a strict fairness.

Definition 3. (Current work distribution) Consider a candidate solution/matrix $\mathbf{S}$ of size $N \times H$. The current work distribution of the solution is defined by:

$$CW(\mathbf{S}) = \sum_{i=1}^{N} (P_i(\mathbf{S})\ log(P_i(\mathbf{S})))$$

Definition 4. (Ideal work distribution) Consider a candidate solution/matrix $\mathbf{S}$ of size $N \times H$. We define the ideal work distribution IW of the solution as:

$$IW(\mathbf{S}) = \sum_{i=1}^{N} (\overline{P}(\mathbf{S})\ log(\overline{P}(\mathbf{S}))) = -(N\,\overline{P}(\mathbf{S}))\ log(N)$$

Proposition 1. *Consider a candidate solution/matrix* $\mathbf{S}$ *of size* $N \times H$. *The following equation is verified:*

$$CW(\mathbf{S}) \geqslant IW(\mathbf{S})$$

Proof. From thermodynamics' entropy function[22] we deduce that:

$$\sum_{i=1}^{n} (\rho_i log(\rho_i)) \geqslant -log(n) \tag{1}$$

with $\rho_i \in [0,1]$ and $\sum_{i=1}^{n} \rho_i = 1$. Consider $P_T = \sum_{i}^{N} P_i(\mathbf{S})$ the *total penalty* of the candidate solution. We can define a set of values ρ_i between 0 and 1 such that $\sum_{i=1}^{N} \rho_i = 1$ and $P_i(\mathbf{S}) = \rho_i P_T = \rho_i \overline{P(\mathbf{S})} N$ for all i in $\{1..N\}$. Using these equations we can deduce that:

$CW(\mathbf{S}) = \sum_{i=1}^{N} (P_i(\mathbf{S})\ log(P_i(\mathbf{S})))$

$CW(\mathbf{S}) = \sum_{i=1}^{N} \left(\rho_i P_T\ log(\rho_i \overline{P(\mathbf{S})} N)\right)$

$CW(\mathbf{S}) = P_T \sum_{i=1}^{N} \rho_i log(\rho_i) +$
$\qquad P_T \sum_{i=1}^{N} \rho_i \left(log(\overline{P(\mathbf{S})}) + log(N)\right)$

Then, if we replace the first sum by using relation 1:

$CW(\mathbf{S}) \geqslant -P_T\ log(N) + P_T\ log(\overline{P(\mathbf{S})}) + P_T\ log(N)$

$CW(\mathbf{S}) \geqslant \sum_{i=1}^{N} P_i(\mathbf{S})\ log(\overline{P(\mathbf{S})}) = IW(\mathbf{S})$ □

Proposition 1 shows that the function IW corresponds to the lower bound of CW.

Finally, the evaluation function is given by the *fair work distribution*.

Definition 5. (Fair Work Distribution) Given a candidate solution $\mathbf{S}$ with an ideal work distribution IW and a current work distribution CW, the fair work distribution FW is given by:

$$FW(\mathbf{S}) = \alpha * \left(\frac{CW(\mathbf{S})}{IW(\mathbf{S})} - 1\right)$$

with α a constant equal to 10000. We have defined this value in order to amplify the differences between very similar solutions. Note that, by using Proposition 1, $FW(\mathbf{S}) \geqslant 0$. $FW(\mathbf{S}) = 0$ implies a strict fairness of the candidate solution. Thus, our objective is to find a solution/position $\mathbf{S}$ that *minimizes* the value $FW(\mathbf{S})$.

4.2. Initial population

The initial population seeks to obtain particles that satisfy all the constraints. The solution/matrix **S** of each particle is filled by the following greedy procedure.

```
GenerateSolution(in :C, out : S)
  S ← 0_{N,H}
  Initialize_Domains(D, S, C)
  do
    Select a pair (i, j) such that S_ij = 0 and D_{i,j} ≠ ∅
    S_ij ← random value from D_{i,j}
    Forward_Checking(S,D,C,i,j)
  while There exists a pair (i, j) such that S_ij = 0
  and D_{i,j} ≠ ∅
```

First, the matrix is initialized as a zero matrix of dimensions $N \times H$. The `Initialize_Domains` procedure associates each (i, j) entry of **S** with a domain $\mathscr{D}_{i,j}$ initialized with the set of *feasible* shifts, taking into account the set of constraints C (e.g., a part-time nurse working 12 hours a week can work any shift on weekdays but no shift on weekends). Next, a pair (i, j) is randomly selected and assigned with a value of the corresponding domain. A forward-checking procedure is performed. The procedure eliminates from the domains all the values in conflict with one of the constraints in C. For example, if the last assignment was an ENS to nurse *i* on Monday 6, then:

- by using constraint C1: if the number of nurses working ENS on Monday 6 reaches the specified number, then the shift ENS is removed from the domain of all the nurses on that day;
- by using constraint C2: it is removed any shift of the nurse implying a violation of the constraint, if nurse n works it, she/he exceeds her/his working hours;
- by using constraint C6: all the shifts in the domain of Tuesday 7 of nurse *i* are removed; etc.

The procedure finishes when all the domains become empty.

Remark 1. It is possible that the nurses' shift requirement (constraint C1) is not fulfilled. The cost of any solution violating constraint C1 is $+\infty$ (actually C1 is the only constraint that can be violated).

4.3. Moves

In our algorithm we have considered two moves: `Swap-shifts` and `Nearer`. The former is applied for improving the current solution of a particle, and the latter is applied for bringing a particle closer to another one.

1) `Swap-shifts`*(***S***):* Given a scheduling matrix **S**, the procedure randomly selects two nurses i_1, i_2 and a day j. The values of the entries $(i_1,\ j)$ and $(i_2,\ j)$ in **S** are interchanged. If day j corresponds to Saturday (resp. Sunday), the interchange also involves the following day (resp. previous day), i.e., $(i_1,\ j+1)$ and $(i_2,\ j+1)$ would also be interchanged. The values in rows i_1 and i_2 that are *in conflict* with the new values (i.e., they violate one constraint between C2 to C6) are set to 0 (free shift). A greedy procedure is then performed to attempt to satisfy C1 (the do-while loop from the `GenerateSolution` procedure). The move is accepted only if the value of the cost function does not increase.

Consider the scheduling matrix of Fig. 2-a with all nurses working full time ($R_i = 1$ for all $i = 1..4$) and the following shift requirements:

- Monday, Wednesday and Friday: two nurses on PS and one on DS.
- Tuesday and Thursday: two nurses on PS and one on EDS.
- Saturday and Sunday: one nurse on ENS

The move randomly selects nurses 2 and 4 and Sunday for performing the swap. The values corresponding to the weekends are interchanged and the conflicting entries are removed randomly until no constraint (except C1) is violated (Fig. 2-b). As nurse 4 was violating constraint C3, entries (4,1) and (4,3) are set at 0. Finally, the greedy procedure attempts, successfully in this case, to repair the solution setting entries $(3,1)$ and $(1,3)$ at 4 (Fig. 2-c).

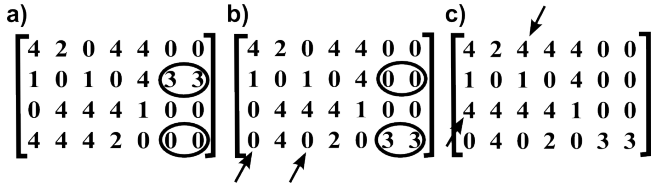

Fig. 2. The swap-shift movement..

2) `Nearer(`$\mathbf{S}, particles, gbest$*):* This move has been designed with the objective of bringing a solution $\mathbf{S}$ closer to the GlobalBest position (*gbest*). Recall that the position of a particle has been defined as the cost of its related solution. Thus, the `Nearer` procedure attempts to replace the solution $\mathbf{S}$ by a solution $\mathbf{S}^*$, such that $gbest \leqslant \texttt{cost}(\mathbf{S}^*) \leqslant \texttt{cost}(\mathbf{S})$. Beginning with a reference solution $\mathbf{S}^r$, randomly selected from the set of particles, the procedure calls to the `Swap-shifts2` method. This method is equivalent to the `Swap-shifts` one but accepts any solution with a cost less or equal to *goal_cost* (not only improving solutions). The solution $\mathbf{S}$ is replaced by $\mathbf{S}^*$ only if the latter is different from $\mathbf{S}^r$ (line 5).

`Nearer`(**in:** *particles*; **in-out:** $\mathbf{S}$)
1: $goal\_cost \leftarrow \texttt{cost}(\mathbf{S})$
2: Select random solution $\mathbf{S}^r \in particles$ such that $\texttt{cost}(\mathbf{S}^r) < goal\_cost$
3: $\mathbf{S}^* \leftarrow \mathbf{S}^r$
4: `Swap-shifts2`($\mathbf{S}^*$, *goal_cost*)
5: **if** $\mathbf{S}^* \neq \mathbf{S}^r$ **then** $\mathbf{S} \leftarrow \mathbf{S}^*$ **end if**

Remark 2. The random selection of the reference solution (line 2) allows us to maintain the population more diversified than if we only select the best solution found as reference.

4.4. PSO − ANSP

The following pseudocode shows our implementation of the PSO for the ANSP. Table 3 summarizes the parameters and variables used by the algorithm.

The algorithm begins by creating the initial set of feasible particles. The GlobalBest solution ($\mathbf{S}^g$) is initialized with the solution of the particle with the best fitness. For each particle, the velocity (V^p) is initialized to 0.

`PSO-ANSP`(**in:** $\phi_g, nb\_particles, maxIter$)
1: $particles \leftarrow$ initializePopulation($nb\_particles$)
2: $\mathbf{S}^g \leftarrow null$
3: **for all** $p \in particles$ **do**
4: **if** $cost(\mathbf{S}^p) < cost(\mathbf{S}^g)$ **or** $\mathbf{S}^g = null$ **then**
5: $\mathbf{S}^g \leftarrow \mathbf{S}^p$
6: **end if**
7: $V^p \leftarrow 0$
8: **end for**
9: $t \leftarrow 0$
10: **repeat**
11: **for all** $p \in particles$ **do**
12: `UpdateVelocity`($V^p, \mathbf{S}^p, \mathbf{S}^g, \phi_g$)
13: **if** $V^p >$ `random`$(0,1)$ **then**
14: `Nearer`($\mathbf{S}^p$, *particles*, $cost(\mathbf{S}^g)$)
15: **end if**
16: `Swap-shifts`($\mathbf{S}^p$)
17: **if** $cost(\mathbf{S}^p) \leqslant cost(\mathbf{S}^g)$ **then** $\mathbf{S}^g \leftarrow \mathbf{S}^p$ **end if**
18: $t \leftarrow t+1$
19: **end for**
20: **until** $t = maxIter$

Table 3. Parameters and variables related to the PSO-ANSP algorithm.

nb_particles:	**user-defined parameter** (integer number greater than 0). It defines the amount of particles in the population/swarm.
ϕ_g:	**user-defined parameter** (belongs to the interval $[0,1]$). It is related to the velocity formula (2).
maxIter:	maximum number of iterations for the algorithm.
particles:	the set of particles.
$\mathbf{S}^g$:	the best solution in the entire swarm (GlobalBest).
$\mathbf{S}^p$:	the current solution of the particle p.
V^p:	a real number in $[0,1]$. It indicates the current velocity of the particle p.
$cost(\mathbf{S})$:	method that computes the cost of a solution $\mathbf{S}$.

Then, an iterative process begins. Each particle is selected, using a round robin principle. The velocity of the particle is updated, using the formula[21]:

$$V^p \leftarrow \omega V^p + \phi_g \, r_g \, diff(cost(\mathbf{S}^p), cost(\mathbf{S}^g)) \quad (2)$$

where ω is the inertia weight, and $r_g \sim U(0,1)$ is a stochastic variable weighted by the user-defined parameter ϕ_g. We define $diff(g,x) := \frac{x-g}{x}$ as a normalized difference, belonging to the interval $[0,1]$, instead of using the classical subtraction $(g-x)$. We have also fixed the ω parameter to 0, so that the velocity depends only on the current and best positions. In our PSO, V^p represents the probability of using the `Nearer` procedure to bring the current particle closer (in cost) to the GlobalBest solution. Thus, the `Nearer` procedure is called if V^p is greater than a random value chosen from a uniform distribution $U(0,1)$. The idea behind our concept of velocity can be straightforwardly explained:

- Solutions with costs far from the best solution found have a high probability of been replaced by better ones, using the `Nearer` procedure.
- Solutions with costs close to the best found are changed with a low probability.

Next, the `Swap-shifts` move is applied in order to improve the current candidate solution of the particle. If the solution related to the particle has a cost less than or equal to the GlobalBest position/cost, then the GlobalBest solution is replaced by the candidate solution of the current particle (line 17).

After *maxIter* iterations the algorithm finishes and the best solution is held in $\mathbf{S}^g$.

Table 4. Parameters of the experimented tests.

			Requirements				
Inst.	N	H	Days	DS	EDS	ENS	PS
P1		7	weekday	2	2	1	2
P5	14	14	weekend	0	1	1	0
P9		28					
P2		7	weekday	7	2	1	2
P6	22	14	weekend	0	1	1	0
P10		28					
P3		7	weekday	10	2	1	3
P7	26	14	weekend	0	1	1	0
P11		28					
P4		7	weekday	20	4	2	4
P8	52	14	weekend	0	2	2	0
P12		28					

5. Results and Evaluation

The hardware platform for the experiments was a PC Intel Corei7-920 with 4GB RAM under the Linux Mandriva 2010 operating system. The algorithm has been implemented in C++.

We have performed various experiments aiming: (1) to ascertain which cost function is better (FW or Z); (2) to find good values for the used-defined parameters and (3) to compare our algorithm and CPLEX.

We used a set of 12 instances related to some requirements of a French hospital.[23] The input parameters of each instance are summarized in tables 4 and 5. Note that instances with the same number of nurses share the same shift requirements. Table 5 reports how a set of N nurses is associated to different working rates. For example, of the instances related to $N = 14$ nurses (i.e., P1, P5 and P9), 5 are full-time nurses (i.e., $R_i = 1$ for $i = 1..5$), 6 are part-time nurses working 80% of the regular weekly working time (i.e., $R_i = 0.8$ for $i = 6..11$), and the last 3 are part-time nurses working 30% of this time (i.e., $R_i = 0.3$ for $i = 11..14$).

Table 5. Working rate of nurses.

N	100%	80%	70%	50%	30%
14	5	6	0	3	0
22	7	8	2	4	1
26	11	9	3	2	1
52	18	15	9	8	2

5.1. *Comparison of different cost functions*

We have compared two different aspects from different cost functions (Z, FW). The first of these aspects is related to the capacity of the cost function to describe the fairness of a schedule. The second aspect is related to the skills of the cost function to guide the algorithm through the search space.

Evaluating the fairness of a schedule

Table 6 shows some information retrieved from two different feasible solutions ($\mathbf{S}_1$ and $\mathbf{S}_2$) for the instance P12. Each row corresponds to a solution, and columns from 2 to 11 report the number of nurses who have a penalty, at the top of the respective column. The last two columns shows the values of three evaluation functions for each assignment (Z, FW).

Note that, even though $\mathbf{S}_1$ has the lower value of Z, by using the FW cost function we consider that $\mathbf{S}_2$ is better. This is because the objective function Z only takes into account the difference between the highest (9.25 in $\mathbf{S}_1$ and 9.42 in $\mathbf{S}_2$) and lowest (8.6 in both solutions) penalties and not the disparity of penalities among all the nurses. We can clearly see this disparity in the number of nurses who have high or low penalties. For example, $\mathbf{S}_1$ has 24 penalties higher than 9.1 or lower than 8.7 and $\mathbf{S}_2$ has only 10. Thus, solution $\mathbf{S}_1$ seems to be more unfair than $\mathbf{S}_2$. This observation is corroborated by the value of FW.

Table 6. Comparison of two solutions on the instance P12.

P12	# of nurses having the given penalty											
	8.6	8.67	8.75	8.8	8.86	9	9.14	9.2	9.25	9.42	Z	FW
$\mathbf{S}_1$	7	2	6	11	5	6	4	5	6	0	0.65	0.105
$\mathbf{S}_2$	1	2	7	17	6	12	2	2	2	1	0.83	0.062

Guiding the algorithm through the space search

Table 7 reports the results obtained by using different cost functions to guide the search ($Z = P_{max} - P_{min}$, Z^+, FW). To keep things simple we have used a basic hill-climbing algorithm HC(fo). The algorithm performs only 10^5 calls to the `Swap-Shifts` procedure and returns the best found solution w.r.t. the given cost function fo. The Z^+ objective function corresponds to an improved variant of the Z function used in our previous work[20]: if two solutions have the same evaluation of Z, Z^+ considers that the best solution is the one that has fewer nurses with *extreme* penalties, i.e., nurses with a penalty equal to P_{max} or P_{min}. For example, consider that the penalties of two solutions $\mathbf{S}_1$ and $\mathbf{S}_2$ involving 5 nurses are $(5.1, 5.1, 5.1, 5.3, 5.4)$ and $(5.1, 5.2, 5.3, 5.3, 5.4)$ respectively. Both have the same evaluation of Z. However, by using Z^+, $\mathbf{S}_2$ is considered better than $\mathbf{S}_1$ because, in $\mathbf{S}_1$, 4 nurses have extreme penalties (5.1 or 5.4) while in $\mathbf{S}_2$ only 2 have extreme penalties.

Each tuple on the table reports how many of ten runs in a given strategy returned a *good evaluation* w.r.t. FW and Z respectively. We say that an evaluation (ev) is good if $ev < 1.2\ best\_ev$, where $best\_ev$ is the best evaluation obtained by the three strategies, using the same random seed.

For instance, when we ran HC(Z^+) on the instance P6, in 9 runs we obtained a solution with a good evaluation of Z, and in only 2 runs we obtained a solution with a good evaluation of FW.

Looking at the total number of good evaluations (# good ev.), we find that HC(Z^+) obtained more good solutions than did HC(Z) w.r.t. all cost functions, specially Z. HC(Z^+) also obtained solutions with a good evaluation of the Z cost function more often than did its competitors. However, we explained in the previous section that Z is not a good way to determine the fairness of a nurses' scheduling. Thus, we prefer solutions with low values of FW. In this respect, the best strategy is precisely the one using the same cost function to guide the search, i.e., HC(FW). It is interesting to note that HC(FW) also obtains solutions with good evaluations of Z.

Table 7. Comparison among different cost functions to guide the search.

	HC(Z^+)		HC(Z)		HC(FW)	
Inst.	FW	Z	FW	Z	FW	Z
P1	4	5	1	4	4	4
P2	8	10	1	3	2	2
P3	4	8	5	10	3	7
P4	0	6	1	4	7	3
P5	9	9	3	3	9	9
P6	2	9	10	10	9	9
P7	6	7	5	6	8	8
P8	2	6	0	4	9	7
P9	10	10	10	10	10	10
P10	0	10	0	10	10	10
P11	4	8	5	7	10	8
P12	0	3	0	1	8	4
# good ev.	**49**	**91**	**41**	**72**	**89**	**81**

5.2. *Tunning ϕ_g*

Recall that ϕ_g is the only user-defined parameter that we have not fixed in the velocity updating formula (2). High (resp. low) values of ϕ_g imply high (resp. low) values of V resulting in a high (resp. low) probability of applying the `Nearer` procedure (line 14 of the `PSO-ANSP` algortithm). The tendency of the `Nearer` procedure to make the swarm converge on the current best particle suggests that ϕ_g should be fixed at a low value.

A few experiments not reported in this article confirmed our intuition. The best results were obtained with ϕ_g belonging to the interval $[0.05, 0.001]$. Thus, in the subsequent experiments we decided to use $\phi_g = 0.01$.

5.3. *Tuning the number of particles*

Table 8 reports the results of the PSO-ANSP algorithm, using a different number of particles (1, 2, 3, 5, 10 and 15), $\phi_g = 0.01$ and FW as the cost function. To make a fair comparison, the number of it-

erations (the *max_iter* parameter) has been fixed at 10^5 for all strategies.

Table 8. Results obtained by PSO-ANSP with different number of particles.

Inst.	HC	**2 part.**	**3 part.**	**5 part.**	**10 part.**	**15 part.**
P1	2.6260 2	2.6139 3	2.0012 2	1.2005 5	**0.8941** **8**	0.9366 7
P2	1.9341 4	1.4859 5	1.8867 1	1.3382 **6**	**1.3117** 5	1.4267 3
P3	2.1079 2	1.6093 3	1.5236 7	1.4958 4	1.2521 7	**1.0903** **8**
P4	3.0846 0	2.3361 **5**	2.5625 2	2.9061 0	**2.3264** 3	2.3336 **5**
P5	0.3482 4	**0.1526** **9**	0.1734 7	**0.1526** **9**	**0.1526** **9**	**0.1526** **9**
P6	0.3174 4	0.2890 6	0.2623 8	0.2634 8	0.2526 9	**0.2451** **10**
P7	0.2267 6	0.2089 6	0.1991 7	0.1951 6	0.1887 7	**0.1766** **9**
P8	0.3480 6	0.3308 5	**0.2880** 7	**0.2879** **9**	0.3085 6	0.3751 1
P9	0.0381 8	**0.0315** **10**	**0.0315** **10**	**0.0315** **10**	**0.0315** **10**	**0.0315** **10**
P10	0.0587 9	0.0514 **10**	**0.0501** **10**	**0.0501** **10**	**0.0501** **10**	**0.0501** **10**
P11	0.0470 7	0.0443 8	0.0431 9	**0.0412** **10**	0.0427 **10**	0.0445 9
P12	0.0529 4	**0.0525** 5	0.0541 1	0.0601 **6**	0.0660 5	0.0778 3
# good ev.	56	75	71	83	**89**	84

The first column shows the name of the instances. Each entry of the table shows the results obtained, using the specified number of particles in a given benchmark. Note that the PSO parameterized with only one particle (HC) is equivalent to a hill-climbing algorithm that uses the `Swap-Shifts` move. The value on the top side of each entry corresponds to the *average cost evaluation* after 10 runs. The value on the bottom side of the entry corresponds to the number of the 10 runs in which the given strategy has obtained a *good evaluation*, i.e., has reached the best cost evaluation (among all strategies) or has been close to it (up to 20%). The last row (# good ev.) shows the total number of times among all the runs that the strategy has obtained good evaluations.

For example, in $P12$ the PSO containing 2 particles obtained the best average of the cost evaluations (0.0525). However, the PSO containing 5 particles obtained good results more often (in 6 of the 10 runs). Looking at the last row we observe that, by fixing the number of particles at 10 we obtain good results more often than with any other strategy (in 89 runs). We therefore decided to use these value in the following experiments.

Note that the PSO, using any number of particles, outperforms the results obtained by the hill-climbing algorithm.

5.4. Comparison between PSO-ANSP and CPLEX

Table 9 shows a comparison between our PSO-ANSP algorithm and the well-known ILOG CPLEX solver, using the ILP model described in Section 3. We do not report the results obtained by the CP model because they are worse than the results obtained by the ILP model in every instance. As the CPLEX solver does not allow us to use non-polynomial functions as the objective function, we used the Z function instead of FW.

The PSO-ANSP uses FW as the cost function (to guide the search), $\phi_g = 0.01$ and 10 particles. For each instance, the CPLEX columns report the last upper bound obtained after 10 and 100 seconds of CPU time w.r.t. the Z and the FW evaluations. In a similar way, the PSO-ANSP columns report the average, best and worst results obtained after 10 runs of the algorithms. The last column of the PSO-ANSP reports the average CPU time spent by one (top of the entry) and 10 (middle of the entry) runs. For example, on instance P1 after ten runs (i.e., 14s of CPU time), the best evaluation found by the PSO-ANSP algorithm evluates $Z = 0.2$ and $FW = 0.0554$.

If we compare CPLEX limited to 10 seconds and the average results given by the PSO algorithm, it is clear that the latter obtains better results than the former in every instance. Furthermore, the worst result obtained by the PSO algorithm is always better than or equal to the results obtained by CPLEX after 10 seconds. The results are similar if we compare CPLEX limited to 100 seconds and the best result obtained by the PSO algorithm (obtained in less than 100 seconds). Only in instance P2, does CPLEX outperforms the PSO algorithm.

Table 9. Result comparison between the ILOG CPLEX solver (by using the ILP model) and the PSO-ANSP algorithm on the twelve instances.

	CPLEX			PSO-ANSP			
Inst.	Z	FW	t		Z	FW	t
P1	0.55	2.7963	10s	av	**0.32**	**0.8941**	1.4s
	0.35	1.8192	100s	best	**0.20**	**0.0554**	14s
				worst	0.35	0.0979	
P2	0.80	6.2433	10s	av	**0.48**	**1.3117**	1.7s
	0.38	**0.6123**	100s	best	0.42	0.8736	17s
				worst	0.55	1.8999	
P3	0.60	2.2504	10s	av	**0.54**	**1.2521**	1.9s
	0.35	0.7258	100s	best	**0.30**	**0.6638**	19s
				worst	0.60	1.9880	
P4	1.00	12.0630	10s	av	**0.80**	**2.3264**	3.1s
	1.00	12.0630	100s	best	**0.55**	**1.6942**	31s
				worst	1.00	2.9353	
P5	0.90	1.6210	10s	av	**0.22**	**0.1526**	2.0s
	0.40	0.3048	100s	best	**0.20**	**0.1442**	20s
				worst	0.40	0.2287	
P6	0.62	0.8002	10s	av	**0.35**	**0.2526**	2.5s
	0.33	0.2859	100s	best	**0.33**	**0.2440**	25s
				worst	0.53	0.3298	
P7	0.93	1.1479	10s	av	**0.36**	**0.1887**	2.9s
	0.43	0.3764	100s	best	**0.33**	**0.1693**	29s
				worst	0.43	0.2401	
P8	1.40	2.9952	10s	av	**0.54**	**0.3085**	5s
	0.85	1.7228	100s	best	**0.33**	**0.2323**	50s
				worst	0.76	0.3719	
P9	0.65	0.2789	10s	av	**0.20**	**0.0315**	3.0s
	0.25	0.0433	100s	best	**0.20**	**0.0315**	30s
				worst	0.20	0.0315	
P10	0.82	0.2652	10s	av	**0.40**	**0.0501**	4.1s
	0.40	0.1195	100s	best	**0.40**	**0.0501**	41s
				worst	0.40	0.0501	
P11	1.00	0.3769	10s	av	**0.44**	**0.0427**	4.6s
	0.60	0.2355	100s	best	**0.40**	**0.0407**	46s
				worst	0.50	0.0468	
P12	2.00	1.1334	10s	av	**0.40**	**0.0660**	8.8s
	1.05	0.4902	100s	best	**0.40**	**0.0617**	88s
				worst	0.40	0.0700	

Remark 3. Though the search in the PSO algorithm is guided by the *FW* function, the obtained solutions evaluate *Z* better than CPLEX.

6. Conclusions

In this paper we propose an application of the particle swarm optimization algorithm in the real-world problem ANSP. We have defined a new evaluation function *FW* to guide the search of our algorithm. *FW*, similarly to the function $Z = P_{max} - P_{min}$, searches for a minimum of unfair assignments among the nurses. However, *FW* allows for a better discrimination among the possible solutions.

We have designed a PSO algorithm, requiring only two user-defined parameters (ϕ_g and $nb\_particles$), for this combinatorial problem. The results obtained by our algorithm outperforms the results obtained by the ILOG CPLEX solver.

In a future work we intend to apply our approach to larger ANSP instances. Furthermore, the good results shown by PSO in the ANSP make us think that the technique can also be applied successfully in other real world problems with discrete domains of variables.

Acknowledgments

This work is supported by the Fondecyt Project 1120781. María Cristina Riff is partially supported by the Centro Científico-Tecnológico de Valparaíso (CCTVal), N^{o}FB0821. Ignacio Araya is supported by the UTFSM Researcher Associated Program.

References

1. T. Gong and A. Tuson . "Particle Swarm Optimization for Quadratic Assignment Problems-A Forma Analysis Approach," *International Journal of Computational Intelligence Research*, Vol.**4**, N. **2**, 177–185 (2008).
2. M. Clerc. "Particle Swarm Optimization," *ISTE Publishing Company* (2006).
3. M. Gunther and V. Nissen, "A Comparison of Neighbourhood Topologies for Staff Scheduling with Particle Swarm Optimisation," *German Conference on Artificial Intelligence - KI*, pp. 185–192 (2009).
4. K. Martin and M. Wright, "Using Particle Swarm Optimization to Determine the Visit Times in Community Nurse Timetabling," *Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling* (2008)
5. L. Trilling , A. Guinet and D. Le Magny . "Nurse Scheduling using Integer Linear Programming and Constraint Programming," *Proceedings of the 12th IFAC International Symposium*, Elsevier , Vol.**3**, Saint-Etienne, France, 651–656 (2006).
6. E.S. Rosenbloom and N. F. Goertzen . "Cyclic nurse scheduling," *European Journal of Operation Research*, **31**, 19–23 (1987).
7. C. Valouxis and E. Housos. "Hybrid optimization techniques for the workshift and rest assignment of nursing personnel," *Artificial Intelligence in Medicine*, **20**, 155–175 (2000).

8. B. Cheang, H. Li, A. Lim and B. Rodrigues ." Nurse Rostering Problems - A Bibliographic Survey," *European Journal of Operational Research*, **151**, 447–460 (2003).
9. E. K. Burke, P. D. Causmaecker, G. V. Berghe and H. V. Landeghem . "The state of the art of nurse rostering," *Journal of scheduling*, **7**, 441–499 (2004).
10. M. Vanhoucke and B. Maenhout. "NSPLib–a tool to evaluate (meta-)heuristic procedures, Operational research for health policy: making better decisions," *proceedings of the 31st meeting of the European working group on operational research applied to health services*, 151–165 (2007).
11. M. Vanhoucke and B. Maenhout . "On the characterization and generation of nurse scheduling problem instances", *European Journal of Operational Research*, Vol.**196**, N.**2**, 457–467 (2009).
12. B. Jaumard , F. Semet and T. Vovor . "A generalised linear programming model for nurse scheduling," *European Journal of Operational Research*, **107**, 1–18 (1998).
13. I. Berrada, J. A. Ferland and P. Michelon . "A multiobjective approach to nurse scheduling with both hard and soft constraints," *Socio-Economic Planning Sciences*, **30**, 183–193 (1996).
14. M.N. Azaiez and S.S. Al Sharif . "A 0-1 goal programming model for nurse scheduling," *Computers & Operations Research*, **32**, 491–507 (2005).
15. G. Weil, K. Heus, P. Franois and M. Poujade. "Constraint Programming for Nurse Scheduling," *IEEE Engeneering in Medicine and Biology*, **14**, 417–422 (1995).
16. S. Adbennadher and H. Schlenker . "Nurse scheduling using constraint logic programming," *Proceedings of AAAI/IAAI,* (1999).
17. U. Aickelin and K. A. Dowsland . "An indirect Genetic Algorithm for a nurse-scheduling problem," *Computers and Operations Research*, **31**, 761–778 (2004).
18. M. J. Brusco and L. W. Jacobs . "Cost analysis of alternative formulations for personnel scheduling in continuously operating organisations," *European Journal of Operational Research*, **86**, 249–261 (1995).
19. ILOG . *ILOG Solver 4.4 Reference Manual*, Gentilly, France (1998).
20. L. Trilling ,A. Guinet and D. Le Magny . "Planification des infirmiers anesthesistes : analyse comparative des performances de diffèrents solveurs," *Journal of Logistique and Management*, Vol.**15**, Issue 1, 5–16 (2007).
21. M. E. H. Pedersen and A. J. Chipperfield, "Simplifying particle swarm optimization," *Applied Soft Computing*, **10**, 618-628 (2010).
22. L. Boltzmann, "Lectures on gas theory," *Translated by Stephen G. Brush (1964) Berkeley: University of Califonia Press*, New York (1995).
23. L. Trilling, "Aide la dcision pour le dimensionnement et le pilotage de ressources humaines mutualises en milieu hospitalier," *PhD Thesis*, INSA de Lyon. (2006).
24. I. Blöchliger. "Modeling staff scheduling problem. A tutorial," *European Journal of Operational Research*, 533–542 (2004).
25. A. T. Ernst, H. Jiang, M. Krishnamoorthy and D. Sier . "Staff scheduling and rostering : A review of applications, methods and models," *European Journal of Operational Research*, **153**, 3–27 (2004).
26. A. Guinet . "Scheduling independent jobs on uniform parallel machines to minimize tardiness criteria," *Journal of Intelligent Manufacturing*, **6**, 95–103 (1995).
27. J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," *1997 IEEE International Conference on Computational Cybernetics and Simulation*, **5**, 4104–4108 (1997).