Open access • Proceedings Article • DOI:10.1109/CEC.2001.934319

Fitness evaluation for nurse scheduling problems — Source link

Edmund K. Burke, P. De Causmaecker, Sanja Petrovic, Greet Van den Berghe

Institutions: University of Nottingham

Published on: 27 May 2001 - Congress on Evolutionary Computation

Topics: Nurse scheduling problem, Evaluation function, Fitness function and Resource allocation

Fitness Evaluation for Nurse Scheduling Problems

Edmund K. Burke
University of Nottingham
School of Computer
Science & IT
Nottingham NG8 1BB, UK
ekb@cs.nott.ac.uk

Patrick De Causmaecker
KaHo St.-Lieven
Information Technology
Gebr. Desmetstraat 1, 9000
Gent, Belgium
patdc@kahosl.be

Sanja Petrovic
University of Nottingham
School of Computer
Science & IT
Nottingham NG8 1BB, UK
sxp@cs.nott.ac.uk

Greet Vanden Berghe
KaHo St.-Lieven
Information Technology
Gebr. Desmetstraat 1, 9000
Gent, Belgium
greetvb@kahosl.be

Abstract- When applying evolutionary algorithms to difficult real-world problems, the fitness function routinely needs evaluating for a very high number of intermediary cases.
This paper is concerned with real-world nurse rostering problems with highly constrained resources. We consider a particular approach, which allows for a quick evaluation and is general enough to deal with other kinds of resource planning problems with time-related constraints. The model developed for this approach handles the constraints in a modular way and the addition of new constraints is relatively straightforward. Simple constraints (such as those affecting the personal wishes of employees) and global constraints (such as balancing the workload among people) can be formulated easily using this approach. Our approach can also handle very complex time-related constraints as well as conditions that are related to previously planned work. Moreover, it provides clear feedback about violation of constraints.
The approach has been implemented successfully in a nurse rostering program entitled "Plane" which is used in hospitals all over Belgium. It can tackle a high number of specific and modifiable constraints of a very different nature. The benefits from this approach (in terms of software requirements) are small memory use and a computationally simple, single evaluation function allowing for the simultaneous rostering of several hospital wards at the same time.

1 Introduction

The work presented in this paper describes the evaluation method employed in Plane, a personnel rostering software system developed for the Belgian hospital market (Burke et. al., [2, 3]). Plane has been developed in conjunction with Impakt[1] and GET[2]. One of the main motivations for developing this software was the discovery that none of the available personnel or nurse rostering tools on the market could cope with the high number of very specific constraints that Belgian hospitals have to deal with.
Employee rostering problems basically consist of assigning a number of tasks (or shifts) to personnel with different skills over a defined period of time. The assignment is usually subject to a large number of constraints. The automation of personnel rostering has attracted the attention of researchers since the sixties (cf. Hung, [8]). The problems are often restricted to imposing constraints limiting the total work and constraints limiting consecutive assignments (Chan and Weil, [5]), (Chiarandini et. al., [6]), (Meisels and Lusternik, [9]), (Meyer auf'm Hofe, [10]). Both (Aickelin and Dowsland, [1]) and (Dowsland, [7]) work with a limited number of shift type patterns whose value is predefined.
The majority of approaches described in the literature aim at producing weekly rotating three-shift schedules (Aickelin and Dowsland, [1]), (Chan and Weil, [5]), (Dowsland, [7]), and (Weil et. al., [12]). Plane, however, attempts to solve rostering problems of larger complexity. It attempts to schedule people of different skill categories so that the demands of the hospital can be met during a particular planning period. The search algorithms implemented in Plane are based on tabu search (Burke et. al., [2]). Experiments with genetic and memetic approaches (Burke et. al., [3]) led to better quality timetables at the expense of computational time.
The aim of this paper is to present an approach for the evaluation of personnel schedules. The method is fast and requires only a very simple algorithm, which is extremely useful for evaluating all neighbourhood solutions for the metaheuristics applied. The evaluation method can be applied in other timetabling or scheduling problems with time-related constraints. Section 2 lists the requirements of Plane which have an immediate effect on the cost function. In Section 3, the main constraints imposed on the personnel schedules are explained. We introduce the evaluation approach in Section 4 and explain that new constraints can easily be defined by only designing a template with numbers. In Section 5 the evaluation method is demonstrated using a simplified example of a personnel roster. Some concluding remarks are presented in Section 6.

2 Requirements

The objectives of Plane, which were formulated after an extensive market research effort, led us to the formulation of the evaluation method described in this paper.
The first set of requirements is functional. We want to pro-

[1] Impakt N.V., Ham 64, B-9000 Gent

[2] GET, General Engineering & Technology, Antwerpse Steenweg 107, B-2390 Oostmalle

duce a solution satisfying all personnel demands and respecting the constraints imposed on the resources as much as possible. To evaluate the latter we construct a fitness function expressing the degree to which the constraints are satisfied.
Among the other objectives, there is a list of system specific requirements: conceptual consistency and continuity, graceful degradation, pertinent behaviour, explanatory power, and extendibility.
The presented evaluation approach is aimed at meeting all of these requirements. The criterion of graceful degradation requires the system to come up with a reasonable schedule when (very often) no solution satisfying all the constraints can be found. Even in the middle of the search process, the system should be prepared to provide the user with the best schedule found when it is prompted for a solution. The scheme selected for Plane is one in which the personnel demands are fulfilled in the initial solution and remain fulfilled while trying to improve on the resource constraints. Any obtained solution must be explanatory, in that all constraint violations are explicitly shown to the user. A feature of the system is that all constraints on personal schedules are treated equally, both in the model and in the user interface. The approach provides a method for handling all characteristics of the constraints such as cost parameters, minimum and maximum values, and consecutiveness in a modular way.

3 Constraints

3.1 Terminology

Planning Period
The planning period is the time interval over which the staff have to be scheduled. Plane allows the user to define a planning period, which usually consists of 4 weeks.
Skill Category
This determines a class of staff who have a particular level of qualification, skill or responsibility. For example, a particular skill category might be a class of *junior* nurses. Staff in this category would not normally be allowed to be allocated to a *ward manager*'s shift. However, it might be the case that we could allocate someone in the *ward manager*'s category to a *junior* shift on a given day. It is usually possible to allocate senior staff to a junior position on any given day but not the other way round. However, in practice, very senior staff are usually reluctant to stand in for junior staff. It is also the case that, in practice, a regular (not a junior) nurse will temporarily stand in for a head nurse.
Shift Type
Shift types are hospital tasks with a well-defined start and end time. The ward manager can set the details of the shift types in order to make them match the activities.
Time Units
Time Units represent time intervals of minimum allocation. They are defined to handle the personnel constraints. In the approach used in Plane, where personnel demands and schedules make use of shift types, each shift type has a corresponding time unit. The time units are ordered according to the start times of the shift types they represent. When two shift types have the same start time, the first time unit will match the shift type with the earliest end time. The time units defined for this approach do not represent consecutive or separate periods but they will very often overlap in time. For the nurse rostering problem considered in this paper, the number of time units equals the number of shift types times the number of days in the planning period. In the evaluation method (described in Section 4), time units play a particularly important role.
Personnel Demands
Personnel demands express the number of personnel needed for every skill category and for every shift during the entire planning period. Plane provides other formulations for the personnel demands as well but since they only affect the initialisation and the algorithmic calculations, we will not discuss them in this paper (see (Burke et. al., [2, 3]) for more details).

3.2 Hard constraints

Hard constraints are those that must be satisfied at all costs. Plane is organised so that the personnel demands form the hard constraints. The system does not allow a user to define personnel demands which would need a larger number of staff than are available. In addition, the system does not allow any violation of the hard constraints during the course of the scheduling process. A schedule satisfying the hard constraints is a solution in which for every day of the planning period, the required personnel for each skill category are assigned to the shift types needed.

3.3 Soft constraints

Soft constraints are those that are desirable but which may need to be (usually will need to be) violated in order to generate a workable solution. Indeed, in most real-world cases it is simply not possible to satisfy all soft constraints. All constraints on the personal schedules are categorised as soft constraints. The cost function sums all violations on these constraints for all the personnel members in the solution.
Belgian healthcare institutions have a tradition of evaluating the personnel schedules on a large number of criteria. Due to the, sometimes unpredictable, character of the workload, the schedules have to be very flexible. The constraints are divided into a few categories, affecting certain groups of personnel members. A full description of all the constraint types in use in Plane is given on the following web page http://www.cs.nott.ac.uk/∼gvb/constraints.ps. In this paper, we restrict the description to the constraints most used in practice.

3.3.1 Hospital constraints

Personnel scheduling in the hospitals we deal with is organised per ward. We consider a ward to be a group of personnel working together in the same location (e.g. a certain floor in

a hospital) or having similar activities (e.g. the ambulance team). There are certain rules which hold for the entire hospital. Underneath these global rules, each ward can define its own local rules, such as:
- Minimum time between two assignments
- Allow use of an alternative skill category in certain situations

3.3.2 Constraints defined by the work regulation

Every personnel member has a contract with the hospital, called the work regulation or work agreement. There are different work regulations for full time personnel members, half time and night nurses. In the real-world there are many hospitals which allow for a personal work agreement per nurse. This enables them to formulate personal constraints such as 'Every Wednesday afternoon should be free', 'Work a weekend every two weeks', etc... When defining the work regulation, either of the following constraints can be defined or made idle.
- Maximum number of assignments in the planning period
- Minimum/Maximum number of consecutive days
- Minimum/Maximum number of hours worked
- Minimum/Maximum number of consecutive free days
- Maximum number of assignments per day of the week
- Maximum number of assignments for each shift type
- Maximum number of a shift type per week
- Number of consecutive shift types
- Assign 2 free days after night shifts
- Assign complete weekends
- Assign identical shift types during the weekend
- Maximum number of consecutive working weekends
- Maximum number of working weekends in a 4-week period
- Maximum number of assignments on bank holidays
- Restriction on the succession of shift types
- Patterns enabling specific cyclic constraints
- Balancing the workload among personnel

3.3.3 Personal constraints

When individual personnel members have an agreement with the personnel manager or head nurse, the following constraints can be put into action:
- Day off; shifts off
- Requested assignments
- Tutorship (people not allowed to work alone)
- People not allowed to work together

4 The Evaluation Approach

Our method evaluates the group of very different constraints that is outlined above. More generally, the method also provides a technique to calculate the extent to which constraints on the schedule are violated. The main ideas of the approach, as well as some guidelines to translate real-world constraints into the model, are explained in this section.

		From	Till
M	morning shift	06:45	14:45
L	late shift	14:30	22:00
N	night shift	22:00	07:00

Table 1: The shift types

The evaluation method allows for the evaluation of the solution per resource (in our nurse rostering example, a resource is a person). The solution of every resource will be evaluated against a schematic representation of the constraints (Section 4.5) in order to determine the value of the cost function for the solution.
To facilitate the explanation of the evaluation method, we consider a simple example consisting of a one-week planning period for a ward with 5 people. The number of shift types in use is restricted to the morning (M), late (L), and night (N) shifts presented in Table 1. All personnel members have the same work agreement. This implies that their personal schedules are all subject to the same set of soft constraints. Each requested shift can be assigned to any of the nurses because they all belong to the same skill category.
Table 2 shows a personnel schedule especially constructed to demonstrate the method. The rows in the table present the schedules associated with each of the personnel members. The schedule for the previous planning period is also presented in the table. The previous solution is used for defining the start values of the constraints to be evaluated.

4.1 Formal description of the evaluation method

The ideas, which form the basis of the evaluation method, will be presented formally in this section. The time units introduced in Section 3.1 are basic concepts in the description of this method. Suppose there are D days in the planning period and that the problem consists of Sh shift types then $D * Sh$ time units are used. The set of time units is denoted by T. As an example, the schedule for 5 people presented in Table 2 is translated into a time unit schedule in Table 3. In this example, $D = 7$ and $Sh = 3$. Since the example consists of only 3 possible shift types, every day in the real-world planning is represented by 3 columns in the time unit schedule.
We introduce *numberings* as templates which are put on each personal schedule in order to evaluate constraints in a uniform way. Instead of writing a separate algorithm for the evaluation of each constraint, we designed the numberings so that all constraints can be evaluated using a single algorithm. The evaluation of every personal schedule is performed in one go, starting from the first time unit for which the person is scheduled and ending at the last. For some easy constraints, very simple numberings suffice. When we come to complications involving weekends, night work, etc., the numberings are constructed in order to allow for sufficient abstractions from the real-world details of the problem.
Definition 1 *A* **numbering** N_i *is a mapping of the set of time*

	Previous planning period							Current planning period						
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun
P1						M	M	M	M	L	L	N		
P2	M	L	L				N	N		N	L	L		
P3			L	L				M	M	M	M	M	M	M
P4						L	L	M		L	N	N	N	
P5	N	N	N	N	N			M L	L	L	L			

Table 2: Shift type solution for 5 people (P1,. . . , P5) and 1 week; M,L, and N being the shift types introduced in Table 1

	Previous planning period																			
P1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*	-	-	*	-	-
P2	*	-	-	-	*	-	-	*	-	-	-	-	-	-	-	-	-	-	-	*
P3	-	-	-	-	-	-	-	*	-	-	*	-	-	-	-	-	-	-	-	-
P4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*	-	-	*	-
P5	-	-	*	-	-	*	-	-	*	-	-	*	-	-	*	-	-	-	-	-

	Current planning period																				
P1	*	-	-	*	-	-	-	*	-	-	*	-	-	-	*	-	-	-	-	-	-
P2	-	-	*	-	-	-	-	-	*	-	*	-	-	*	-	-	-	-	-	-	-
P3	*	-	-	*	-	-	*	-	-	*	-	-	*	-	-	*	-	-	*	-	-
P4	*	-	-	-	*	-	-	-	*	-	-	*	-	-	*	-	-	*	-	-	-
P5	*	*	-	-	*	-	-	*	-	-	*	-	-	-	-	-	-	-	-	-	-

Table 3: Time unit representation of the solution in Table 2: "*" denotes 'used' (scheduled) and "-" denotes 'idle' (free)

	Soft constraints	V	C	N
1	maximum assignments	6	1	N_1
2	minimum assignments	2	1	N_1
3	maximum consecutive days	4	1	N_1
4	minimum consecutive days	2	1	N_1
5	maximum consecutive free days	8	1	N_1
6	minimum consecutive free days	2	1	N_1
7	maximum assignments per day	1	1	N_1
8	maximum night shifts	3	1	N_2
9	minimum consecutive night shifts	2	1	N_2
10	work full weekends	y	1	N_3

Table 4: Some soft constraints; the column V denotes the value, C denotes the cost and N denotes the numbering associated with the constraint

units to a set of numbers i.e.

$N_i : T \rightarrow \{-M, -M+1, \ldots, 0, 1, \ldots, M-1, M, U\}$

where i=1, . . . , I and I is the total number of numberings. M *is a positive integer and* U *is a symbol introduced to represent the time units for which the numbering is undefined.*

The mapping need not be into or onto, nor need it conserve the sequence. A set of 10 constraints of a different nature was selected from the real-world constraints to explain how the approach covers the personnel rostering problem (see Table 4). Table 5 presents 3 numberings denoted by N_1, N_2, and N_3, created for the schedule presented in Table 2. The value for M in both N_1 and N_2 is 6 and M is 1 in N_3. Each numbering is assigned to one or more constraints. When constraints are related to days, for example, the numbering will consist of increasing numbers for the time units corresponding to the days (as in numbering N_1). In Table 5, the appropriate numbers identifying the previous planning period are also shown. These numbers will be used for the initialisation of the evaluation method. In fact, the 3 presented numberings are sufficient to evaluate all 10 real-world constraints given in Table 4. The values in the numbering depend on the nature of the real-world constraints. Numberings provide the possibility of implementing irregular concepts such as days off, bank holidays, etc. All numberings are potentially susceptible to the same set of numbering constraints, introduced later on in this section. One of the main aims of the approach presented in this paper is the reduction of the effort of implementing new real-world constraints to designing a proper numbering. The following definitions allow us to be more specific.

Definition 2 *A* **personal schedule** *for person p is a mapping*

$$S_p : T \rightarrow \{used, idle\}.$$

In the personal schedule, an *event* occurs at every time unit when the person is assigned to a shift (or when S_p has value "used"). At an event, each numbering associated with the personal schedule will be checked against its *constraints*. The events are generated following the order of the time units and will be evaluated in that sequence in the algorithm (see Fig. 1).

Definition 3 *For a given personal schedule* S_p *an* **event** *is a time unit* e *for which* $S_p(e) = used$*. Denote by* E_{S_p} *the set of all events that are induced by* S_p*.*

Denote by T_N the set of time units for which the numbering N does not have value U (undefined). Denote by E_{N,S_p} the set of events of S_p which are defined for the numbering N. In other words, $E_{N,S_p} = T_N \cap E_{S_p}$.

The basic idea of the evaluation method is to go through the set E_{N,S_p} for each personal schedule of person p and consider the values $N(e)$ of each event in E_{N,S_p}. The number of constraint types per numbering is limited to 8 (see later on in this section).

	Previous planning period																				
N_1	-7	-7	-7	-6	-6	-6	-5	-5	-5	-4	-4	-4	-3	-3	-3	-2	-2	-2	-1	-1	-1
N_2	U	U	-7	U	U	-6	U	U	-5	U	U	-4	U	U	-3	U	U	-2	U	U	-1
N_3	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	-3	-3	-3	-2	-2	-2

	Current planning period																				
N_1	0	0	0	1	1	1	2	2	2	3	3	3	4	4	4	5	5	5	6	6	6
N_2	U	U	0	U	U	1	U	U	2	U	U	3	U	U	4	U	U	5	U	U	6
N_3	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	0	0	0	1	1	1

Table 5: Numberings used for expressing the real-world constraints of Table 4

	N_1	N_2	N_3	
Constraints	**(Real-world constraint)**			**Penalty costs**
max_total	6 (1) 1	3 (8) 1		cost_max_total
min_total	2 (2) 1			cost_min_total
max_pert	1 (7) 1			cost_max_pert
min_pert				cost_min_pert
max_between	8 (5) 1			cost_max_between
min_between	2 (6) 1			cost_min_between
max_consecutive	4 (3) 1			cost_max_consecutive
min_consecutive	2 (4) 1	2 (9) 1	2 (10) 1	cost_min_consecutive

Table 6: Constraint values and penalty costs for each of the 3 numberings in Table 5; the numbers between brackets refer to the corresponding real-world constraint of Table 4

4.2 Numbering Constraints and Values

A formal description of the numbering constraints and their values is given here. A numbering constraint is a condition, which is checked against its value, during or at the end of the evaluation. Numbering constraint values (between brackets in Table 6) are derived from the real-world constraints' values (denoted by V in Table 4) as presented in the left part of the columns.

Max_total is an upper limit for the number of events

$$\#E_{N,S_p} \leq max\_total$$

The real-world constraints given in Table 4 are translated into 8 numbering constraints presented in Table 6. In the real-world constraints 1 and 8 presented in Table 4 max_total has value 6 and 3 (see Table 6). The other real-world constraints in Table 4 are not evaluated with the max_total constraint.

Min_total is a lower limit for the number of events

$$\#E_{N,S_p} \geq min\_total$$

The real-world constraint 2 in Table 4 makes use of min_total in numbering N_1.

Max_pert is an array of size M representing for each number in the numbering the maximum number of events that can be mapped to it.

Min_pert is an array of size M which is similar to max_pert except that it represents a minimum instead of a maximum. None of the constraints in Table 4 makes use of the min_pert constraint. In more realistic rostering problems, however, the constraint is used to evaluate real-world constraints such as patterns, requested assignments and balancing the workload (see Section 3.3).

For convenience, we introduce a new operator:

Definition 4 *Two numbers a and b (where $a \leq b$) are said to be* **consecutive** *with respect to a numbering N if and only if for every number m in $\{a,\ldots,b\}$ the numbering N maps an event in E_{N,S_p} to m.*

This allows us to introduce four additional constraints:

Max_consecutive is the maximum number of consecutive events. The constraint max_consecutive is used in numbering N_1 for the 3rd real-world constraint of Table 4 and its value is 4.

Min_consecutive is the minimum number of consecutive events. 3 different real-world constraints are evaluated with this constraint, and so 3 different numberings are required. Constraints 4, 9 and 10 use numberings N_1, N_2 and N_3 respectively. In the example, the value of min_consecutive is 2 for all the numberings.

Max_between is the maximum gap between two non-consecutive events a and b i.e. $b - a \leq max\_between$. The 5th constraint in Table 4 is evaluated with Max_between. It has value 8.

Min_between is the minimum gap between two non-consecutive events a and b i.e. $b - a \geq min\_between$. For constraint number 6 in Table 4 the value of min_between is 2 and the numbering is N_1.

4.3 Counters

A counter is a variable, which is initiated at the beginning of the evaluation and which changes during the procedure in order to calculate the constraint violations. Some real-world constraints can be handled with a single counter, for others a counter array is required. The counters will be adjusted during the course of the evaluation and checked against the values of the constraints. The real-world constraints described in Section 3.3 can be evaluated using not more than 8 different constraint types. The counters are: total, consecutive, pert, and last; respectively representing the total number of events for the numbering, the number of consecutive events, the number of events per value in the numbering and the number of the last evaluated event. The *pert* counters are used to count certain scheduling features for different time periods (e.g., count night shifts in weekends). Of the real-world constraints in Table 4, only the 7th uses max_pert as a constraint. For every value of the numbering N_1, max_pert is set to 1.

The constraints introduced above can all be evaluated by one single algorithm. In Section 5 we show, using the schedule of Table 2 and the real-world constraints presented in Table 6 as an example, how the evaluation approach is implemented.

Copy the start values (Fig. 2) into the numberings.
Find first element e of T belonging to E_{S_p}.
DO
- *Update all the numbering counters in the intermediate evaluation (Fig. 3).*
- *Find next element e of T belonging to E_{S_p}.*

WHILE the end of the solution is not reached.
Perform a final evaluation on all numbering constraints (Fig. 4).
Communicate the results to an output device.

Figure 1: Overview of the fitness evaluation

4.4 Cost Parameters

The fitness function is completely modifiable. The approach allows for the establishment of weight factors adapted to the needs of the schedulers. Any violation of a constraint will contribute to the overall value of the cost function in proportion to the weight factor.
In the evaluation approach, weight factors are denoted by the term $cost\_$ followed by the type of the numbering constraint as defined in Section 4.1. For the demonstration, all cost parameters for the real-world constraints of Table 4 (denoted by C) are set to 1, as presented in the right part of Table 6.

4.5 Evaluation mechanism

Every personal schedule is evaluated separately. The procedure can be presented schematically as in Fig. 1. An evaluation starts with the initialisation of the numbering counters. This initialisation sets the start values induced by the solution of the previous planning period. The initialisation procedure

Denote by P_p the personal schedule of the previous planning period for person p.
FOR i=1,...,I (I is the total number of numbering)
numbering_initialised=False
consecutive=0
last_nr=N_i(t) t: the time unit for the smallest value of N_i
max_nr=N_i(t) t: the time unit for the highest value of N_i
Find last element e of T belonging to E_{P_p}
DO
$nr = N_i(e)$
$nr = nr - max\_nr - 1$
IF $(nr \neq U)$
IF $(nr = last\_nr - 1)$ *THEN*
$consecutive = consecutive + 1$
ELSE IF $(nr < last\_nr - 1)$ *THEN*
numbering_initialised=True
last_nr=nr
Find previous element e of T belonging to E_{P_p}.
WHILE $(\neg numbering\_initialised)$
i=i+1
Save the results.

Figure 2: Pseudo code for the initialisation algorithm

is described in Fig. 2. The start values are only important for the constraint types max_consecutive, min_consecutive, max_between and min_between. After the initialisation phase, the evaluation will go from one event to another adjusting the counters for all the numberings. The procedure is

FOR i=1,...,I
$nr = N_i(e)$
IF $(nr \neq U)$
$total = total + 1$
IF $(nr = last\_nr + 1)$ *THEN*
$consecutive = consecutive + 1$
ELSE IF $(nr > last\_nr + 1)$ *THEN*
IF $(consecutive < min\_consecutive)$ *THEN*
$penalty\_min\_consecutive = penalty\_min\_consecutive + cost\_min\_consecutive * (min\_consecutive - consecutive))$
IF $(consecutive > max\_consecutive)$ *THEN*
$penalty\_max\_consecutive = penalty\_max\_consecutive + cost\_max\_consecutive * (max\_consecutive - consecutive)$
IF $(nr - last\_nr - 1 < min\_between)$ *THEN*
$penalty\_min\_between = penalty\_min\_between + cost\_min\_between * (min\_between - (nr - last\_nr - 1))$
IF $(nr - last\_nr - 1 > max\_between)$ *THEN*
$penalty\_max\_between = penalty\_max\_between + cost\_max\_between * ((nr - last\_nr - 1) - max\_between)$
$pert[nr] = pert[nr] + 1$
$last\_nr = nr$
i=i+1

Figure 3: Pseudo code for the intermediate evaluation

schematically presented in Fig. 3. Suppose the number corresponding to the event is n (different from U) then the value of total will be increased by 1, as will the value of pert[n]. Depending on the relationship between n and the number of the last event encountered, either consecutive will be increased by 1 or an intermediate evaluation on the 'between' and 'consecutive' counters will be performed. The details of this intermediate evaluation are presented in Fig. 3. When the evaluation has reached the last event in the planning period, a final evaluation on the constraints is required (Fig. 4). This provides the values of all the violations on the constraints for the schedule. Since the violation values are stored in appropriate data structures, called $penalty\_x$ where x is one of the numbering constraints (see Fig. 4), the quality of the schedule in terms of each particular constraint can easily be traced back. This approach reduces the difficulty of defining proper cost parameters considerably because the impact of changes to the parameters is immediately visible in the value of all the constraints' violations. In Section 5 the whole procedure is executed on a particular personal schedule from the example in Table 2.

Initial			P1	Event 1			Event 2			Event 3			Event 4			Event 5		
N_1	N_2	N_3		N_1	N_2	N_3	N_1	N_2	N_3	N_1	N_2	N_3	N_1	N_2	N_3	N_1	N_2	N_3
-1	U	-2	last	0	U	-2	1	U	-2	2	U	-2	3	U	-2	4	4	-2
0	0	0	total	1	0	0	2	0	0	3	0	0	4	0	0	5	1	0
2	0	2	consecutive	3	0	2	4	0	2	5	0	2	6	0	2	7	1	2
0	0	0	pert[0]	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
0	0	0	pert[1]	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0
0	0	0	pert[2]	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0
0	0	0	pert[3]	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0
0	0	0	pert[4]	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
0	0	0	pert[5]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	pert[6]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
penalty_max_total				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
penalty_min_total				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
penalty_max_pert				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
penalty_min_pert				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
penalty_max_between				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
penalty_min_between				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
penalty_max_consecutive				0	0	0	0	0	0	0	0	0	0	0	0	3	0	0
penalty_min_consecutive				0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 7: Evaluation procedure for person 1 (P1)

5 Example

5.1 Demonstration of the method

In this section, a demonstration is given to show how the soft constraints from Section 3.3 are formulated and evaluated using this approach. All the data is presented in Tables 1 - 6 in Section 4). We will follow the entire evaluation procedure of Fig. 1 for one single personnel member. The results for this person are presented in Table 7. The left part of the table shows the initial values for the counters. The values can be stored in memory since they will not change when evaluating new solutions. Some counters in the numbering do not reflect any real-world constraints in this particular example (see empty fields in Table 6). They hardly affect the evaluation method; their impact on calculation time and memory is very low indeed. For each numbering, only one value for each counter and penalty is stored in the memory.
From left to right in the table, the chronological updating of the counters is illustrated. Note that the values of the counters can change each time a new event is found. The personal solution we choose for this explanation (P1) consists of 5 assignments (events) in the planning period. No penalty is created during the intermediate evaluation phase (see Fig. 3). After the last event was found, the evaluation goes to the final evaluation phase of the algorithm (see Fig. 4). During this part of the evaluation, a penalty is created for the max_consecutive constraint in numbering N_1 and one for min_consecutive in N_2.
Following the evaluation for the P4 solution step by step, the second event already creates a penalty in the intermediate evaluation phase of Fig. 3. The min_between constraint of numbering N_1 is violated when going from event 1 (where the corresponding number is 0) to event 2 (the corresponding number is 2). Since the $(nr = last\_nr + 1)$ condition of Fig. 3 is not fulfilled, the intermediate evaluations are executed. One extra violation occurs during the final evaluation because the min_consecutive constraint of N_2 is violated.

5.2 Real-world issues

The following figures give an idea of the importance of a quick evaluation scheme for the solutions of the nurse rostering problem tackled in Plane. In a hospital, all wards (the number of wards can be hundreds) have access to the software system. An average ward consists of 20 people, has 6 different shift types and 30 different soft constraints per personal schedule. The length of the most encountered planning period is 4 weeks. An iteration in the evolutionary algorithms described in (Burke et. al., [2]) requires approximately 100 evaluations of the cost function. On an IBM RS6000, it takes about one minute to perform 300 iterations.
The evaluation approach is suitable for other timetabling and scheduling problems such as (Burke et. al., [4]) and (Paechter et. al., [11]), especially when evolutionary algorithms are being employed.

6 Conclusions

The described approach for the formulation of the fitness function in a personnel rostering environment has proven to be very powerful.
The evaluation method introduced in this paper was originally developed to enable easy extendibility of real-world nurse rostering problems and to provide a quick and explanatory mechanism. During the course of the development, special constraints and customer requirements forced us to elaborate on the basic idea of the evaluation mechanism. All the extra difficulties can still be tackled with the evaluation method discussed in this paper. Explaining all these details is beyond the scope of this paper.
The method makes use of one simple evaluation function,

FOR i=1,...,I

IF $(total > max\_total)$ *THEN*

$penalty\_max\_total = penalty\_max\_total + cost\_max\_total * (total - max\_total)$

IF $(total < min\_total)$ *THEN*

$penalty\_min\_total = penalty\_min\_total + cost\_min\_total * (min\_total - total)$

IF $(consecutive > max\_consecutive)$ *THEN*

$penalty\_max\_consecutive = penalty\_max\_consecutive + cost\_max\_consecutive * (consecutive - max\_consecutive)$

IF $(consecutive < min\_consecutive)$ *THEN*

$penalty\_min\_consecutive = penalty\_min\_consecutive + cost\_min\_consecutive * (min\_consecutive - consecutive)$

$\forall t : in N_i$

IF $(pert[t] > max\_pert[t])$ *THEN*

$penalty\_max\_pert = penalty\_max\_pert + cost\_max\_pert * (pert[t] - max\_pert[t])$

IF $(pert[t] < min\_pert[t])$ *THEN*

$penalty\_min\_pert = penalty\_min\_pert + cost\_min\_pert * (min\_pert[t] - pert[t])$

IF$(N_i(0) + N_i(T_n) - N_i(last\_event) > max\_between)$*THEN*

$penalty\_max\_between = penalty\_max\_between + cost\_max\_between * ((N_i(0) + N_i(T_n) - N_i(last\_event) - max\_between)$

i=i+1

Figure 4: Pseudo code for the final evaluation of the algorithm

independent of the number and character of the constraints imposed on the system. Although the problem is complex, the current approach enables a quick evaluation of intermediate solutions in the search. This approach is fast and thus especially interesting for meta-heuristic applications. The memory used to model and evaluate the constraints is very limited. A numbering requires memory for the numbers (many constraints can be handled by the same numbering) and for the numbering constraint counters. Since these constraint counters change in the course of the evaluation, they keep track of the value of the cost function without requiring extra memory. Both the time and memory savings are important for the practical use of Plane. A hospital generally buys one licence and the head nurses or personnel planners can log on from their own office to execute the planning at any given time.

The evaluation method also provides a very structural technique which can handle new constraints. Moreover it can take start values for constraints related to previous planning periods into account without interfering with the evaluation method. The modular nature of the approach allows the system to provide some feedback. This functionality assists the user of the software with the interpretation of the quality of the result.

Bibliography

[1] Aickelin, U., Dowsland, K.: Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem, Journal of Scheduling, Volume 3 Issue 3, 2000, 139–153

[2] Burke, E.K., De Causmaecker, P., Vanden Berghe, G.: A Hybrid Tabu Search Algorithm for the Nurse Rostering Problem, X. Yao et al. (Eds.): SEAL'98, LNCS 1585, 187-194, 1999

[3] Burke, E.K., Cowling, P., De Causmaecker, P., Vanden Berghe, G.: A Memetic Approach to the Nurse Rostering Problem, Applied Intelligence special issue on Simulated Evolution and Learning (to appear)

[4] Burke, E.K., Newall, J.P., Weare, R.F.: A Memetic Algorithm for University Timetabling, Practice and Theory of Automated Timetabling, First International Conference, Edinburgh, 1995, 241–250

[5] Chan, P., Weil, G.: Cyclical Staff Scheduling Using Constraint Logic Programming, Proceedings of the 3rd international PATAT conference, ISBN 3-00-003866-3, 2000, 261–276

[6] Chiarandini, M., Schaerf, A., Tiozzo, F.: Solving Employee Timetabling Problems with Flexible Workload using Tabu Search, Proceedings of the 3rd international PATAT conference, ISBN 3-00-003866-3, 2000, 298–302

[7] Dowsland, K.: Nurse scheduling with Tabu Search and Strategic Oscillation. European Journal of Operations Research (106), 1998, 393–407

[8] Hung, R.: Hospital Nurse Scheduling. JONA, Volume 25, Number 7/8, 1995, 21–23, Lippincott-Raven Publishers

[9] Meisels, A., Lusternik, N.: Experiments on Networks of Employee Timetabling Problems, PATAT II, Second International Conference Toronto, 1997, 130–141

[10] Meyer auf'm Hofe, H.: Solving Rostering Tasks as Constraint Optimization, Proceedings of the 3rd international PATAT conference, ISBN 3-00-003866-3, 2000, 280–297

[11] Paechter, B., Rankin, R.C., Cumming, A.: Improving a Lecture Timetabling System for University-Wide Use, Practice and Theory of Automated Timetabling II, Second International Conference, Toronto, 1997, 156–165

[12] Weil, G., Heus, K., Francois, P. et al: Constraint Programming for Nurse Scheduling, IEEE Engineering in Medicine and Biology, 1995, 417–422