Open access • Journal Article • DOI:10.1023/A:1011291030731

A Memetic Approach to the Nurse Rostering Problem — Source link

Edmund K. Burke, Peter I. Cowling, Patrick De Causmaecker, Greet Van den Berghe

Institutions: University of Nottingham

Published on: 30 Jul 2001 - Applied Intelligence (Kluwer Academic Publishers)

Topics: Memetic algorithm, Tabu search, Nurse scheduling problem, Heuristics and Heuristic

Applied Intelligence 15, 199–214, 2001

A Memetic Approach to the Nurse Rostering Problem

EDMUND BURKE AND PETER COWLING
School of Computer Science & IT, University of Nottingham, Jubilee Campus, Nottingham, NG8 1BB, UK
ekb@cs.nott.ac.uk
pic@cs.nott.ac.uk

PATRICK DE CAUSMAECKER AND GREET VANDEN BERGHE
KaHo St.-Lieven, Procestechnieken en Bedrijfsbeleid, Gebr. Desmetstraat 1, 9000 Gent, Belgium
patdc@kahosl.be
greetvb@kahosl.be

Abstract. Constructing timetables of work for personnel in healthcare institutions is known to be a highly constrained and difficult problem to solve. In this paper, we discuss a commercial system, together with the model it uses, for this rostering problem. We show that tabu search heuristics can be made effective, particularly for obtaining reasonably good solutions quickly for smaller rostering problems. We discuss the robustness issues, which arise in practice, for tabu search heuristics. This paper introduces a range of new memetic approaches for the problem, which use a steepest descent improvement heuristic within a genetic algorithm framework. We provide empirical evidence to demonstrate the best features of a memetic algorithm for the rostering problem, particularly the nature of an effective recombination operator, and show that these memetic approaches can handle initialisation parameters and a range of instances more robustly than tabu search algorithms, at the expense of longer solution times. Having presented tabu search and memetic approaches (both with benefits and drawbacks) we finally present an algorithm that is a hybrid of both approaches. This technique produces better solutions than either of the earlier approaches and it is relatively unaffected by initialisation and parameter changes, combining some of the best features of each approach to create a hybrid which is greater than the sum of its component algorithms.

Keywords: nurse rostering, personnel scheduling, heuristics

1. Introduction

In Belgian hospitals, the personnel tend to prefer 'ad hoc' schedules where they can express their personal wishes instead of more rigid practices such as regular three-shift schedules that rotate weekly. Moreover, the requirements of modern Belgian hospitals demand a broader variety of services than just morning, day, and night duties. The very time consuming task of constructing schedules that attempt to satisfy both the hospital requirements and the preferences of personnel is still performed manually in many cases. Recently, several methods have been applied to solve the problem automatically.

When we compare the nurse rostering problems described in the literature to the problem we try to tackle, the differences are numerous. In [1, 2] for instance, the planning period is restricted to one week, while the practice in Belgian hospitals is normally to try to plan at least four weeks at a time. The number of possible duties per day generally varies from 6 to 15 in the problems we tackle, whereas in [1, 3, 4] there are only three different duties to be planned. The work described in [1–7] does not consider the variety of constraints that are required for our problem.

In some references [1, 2, 6, 8], we found that higher skill classes can systematically replace the lower ones. Other references [4, 9] did not mention any replacement

possibility among qualification categories at all. In Belgian hospitals, it is the case that only a small number of people from a certain skill category or qualification category (head nurse, regular nurse, caretaker, nurse aid, etc...) can replace somebody from another category (when required). For example, a head nurse would not be expected to replace a caretaker.

A number of potential pitfalls must be addressed when attempting to automatically solve this problem. Unacceptable solutions usually arise when some of the constraints for the roster are contradictory or when the weight parameters for the different constraints create landscapes in the search space which have very deep narrow valleys that are particularly hard to find.

Burke, De Causmaecker, and Vanden Berghe have presented tabu search algorithms and hybridisations [10] that have been implemented in a commercial system entitled Plane. Plane is a scheduling system that has been commercially developed by Impakt[1] and GET[2] to assist the scheduling of personnel in hospitals when the demands for each qualification category can be determined over a fixed period of time and have to fulfil a number of constraints, limiting their assignments. A description of Plane, its problem domain and its system specific and functional requirements can be found in [11]. Plane was first implemented in a hospital in 1995 but the system is still evolving to cope with the new and more complicated real world problems that keep appearing. So far, several hospitals in Belgium have replaced their very time consuming manual scheduling approach by employing this system.

In this paper, we will discuss the results from several new evolutionary approaches, which we have developed for the nurse rostering problem described in Section 2. Any successful solution method must be robust enough to cope with widely varying cost functions and problem instances. Memetic approaches have already been shown to be powerful and robust approaches for solving a range of optimisation problems [12–17]. A memetic algorithm can be defined as a genetic algorithm where a local optimisation is performed before the algorithm moves on to the next generation [18], so that only locally optimal solutions are considered. We demonstrate such an approach for the nurse rostering problem.

The cost function used in each of these algorithms is modular and can deal with all constraints that match the types described in Section 2.2. The user is free to define his own cost function to modify these constraints and the penalties associated with constraint violation.

In Section 2 we discuss our model for the nurse rostering problem. We consider the complex function, which we use for evaluating the schedules in Section 2.3. Section 3 considers tabu search and variants; Section 4 presents several different genetic and memetic algorithms and Section 5 brings together Sections 3 and 4 to consider hybridisations between the two different approaches. In Section 6 we compare and contrast the performance of the algorithms on specific real world problems. We present conclusions in Section 7.

2. Problem Description

In general, a ward consists of about 20 people, having different qualifications and responsibilities. These people are placed into categories based upon their qualifications and job description (such as head nurse, regular nurse, nurse aid, student, etc...). As explained in Section 1 some of the nurses can replace people from another category (depending upon their qualifications). One or two experienced regular nurses will, for instance, have the necessary qualifications to replace the head nurse. This replacement is sometimes necessary to cater for staff shortages, but it is not desirable and will be penalised in the schedule evaluation function as presented in Section 2.3.

Each instance of the nurse rostering problem has a variety of specific hard and soft constraints and a complex evaluation function. We are currently working on a general format in which the detailed description of test problems can be defined.

In the model we use, a solution is represented as a two dimensional matrix, in which the rows represent the personal schedules. For each shift on each day of the planning period there is a column in the matrix. A possible set of shifts is given in Fig. 1, and a part of the solution matrix which might correspond to this set of shifts is given in Fig. 2.

2.1. Hard Constraints

Personnel requirements are expressed in terms of a required number of nurses from each category for each duty of the planning period, which is often one month. These requirements are the only hard constraints in the problem. In practice, the number of required personnel on a certain day is not absolutely strict. Experienced planners know very well when it is reasonable to plan more or less personnel than required. However, because

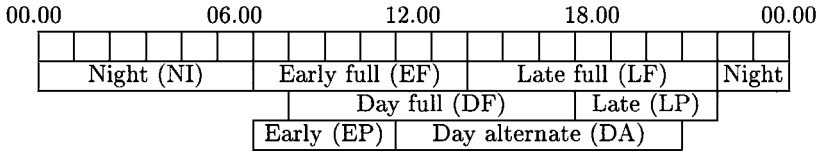

Figure 1. An example of possible shift patterns.

Nurse	Qualification	21st July 1999							22nd July 1999 ···				
		NI	EF	LF	DF	LP	EP	DA	NI	EF	LF	DF	···
A. Nurse	Head Nurse	√									√		···
N. Hance	Staff Nurse					√							···
B. Tarr	Staff Nurse			√							√		···
J. Smith	Caretaker						√					√	···
···	···												···

Figure 2. Matrix representation of a staff roster.

Requirements		21st July 1999							22nd July 1999 ···				
		NI	EF	LF	DF	LP	EP	DA	NI	EF	LF	DF	···
R-min	Head Nurse	1	1	0	0	1	0	1	1	1	1	1	···
R-pref	Head Nurse	1	1	1	0	1	0	1	1	1	1	1	···
R-min	Staff Nurse	3	1	0	0	1	0	1	1	1	1	1	···
R-pref	Staff Nurse	4	1	1	0	1	0	1	1	1	1	1	···
···	···												···

Figure 3. Matrix representation of hard constraints.

there are no clear rules for decisions like this, in our implementation the user can optionally choose to plan the minimum number of required personnel (referred to as R-min in this paper) or the preferred number of personnel (R-pref). A matrix of hard constraints corresponding to the roster matrix given in Fig. 2 is presented in Fig. 3.

Another option is to plan at least the minimal required number of nurses and to add nurses towards the preferred number whenever it does not increase the evaluation function (referred to as R-min-pref, see [10].

If, for one or more days of the planning period, the hard constraints are so strong that no feasible solution exists, planners can opt to relax some of them (R-calc option) until there is some feasible solution. In most cases it is obvious, after a preliminary check, that some of the soft constraints, relating to staff requests etc., cannot be satisfied and we use the evaluation function discussed in Section 2.3 to decide which soft constraints are to be violated.

The following situation, in which a ward consists of 10 people, is a simple example to illustrate how we to calculate that the number of people will not be sufficient to satisfy the demand, where we would need to use the R-calc scheduling option in order to arrive at a feasible solution which satisfied all soft constraints. Suppose that 10 people are available on a particular day, and that the staff who are not working according to a predefined pattern have special requests:

- 2 ask for a day off
- 1 person asks to work the late shift

The patterns for the other staff on that day are as follows

- 2 people have a compulsory day off
- 2 people work a compulsory early shift
- 2 people work a compulsory night shift

Suppose 4 early shifts are required on the day in question. Since, from the 10 people available, 7 are already planned for shifts other than an early shift, the soft constraints above will certainly be violated by the hard constraints. When chosing the R-calc option, the constraint for 4 early shifts that day, which would usually

be a hard constraint, can be relaxed to 3. This relaxation will allow all these soft constraints to be satisfied.

Once the user has decided whether he wants to schedule according to R-min, R-pref or R-calc, the hard constraints can be imposed on the initial schedule. This is simply done by planning all the required duties at random in the schedule, while ignoring the soft constraints such as limitations on the number of shifts each member of staff may work. We modelled the problem and the algorithms in such a way that during the solution search, once an initial feasible solution is found, the hard constraints will never be violated, no matter how bad the evaluation function for the solution is. Our matrix representation of the solution given in Fig. 2 is very helpful for this. Once an initial feasible solution has been found, no new duties can enter the schedule or be removed from it (except for the **SWT** heuristic given in Section 5). Scheduled duties can only move vertically in the matrix, with the number of duties of a certain type on a particular day remaining constant. However, this is complicated by the fact that it is possible, though not desirable, for some staff to work at levels slightly above or below their usual level.

2.2. Soft Constraints

It is very unusual in real world problems to find a schedule that satisfies all the soft constraints. The aim of the algorithms is to minimise the real impact of violations of these constraints. The users of the system specify all the constraints. Certain general constraints are recommended by the hospital (but in certain situations, they may need to be ignored). There are other soft constraints that are normally created by an agreement between the head nurse (or personnel manager) and the individual nurses. At this moment there are more than 30 types of soft constraint. A detailed list of the constraints in Plane can be found at http://www.impakt.be/plane/indexf.htm. Examples of the most typical constraints used include:

— Legal or regulatory requirements:

- **Minimum time between two assignments, depending on the type of duties involved**. It is legally compulsory to have at least 11 hours rest between two duties. The users of Plane can decide to augment or diminish the rest time before and after certain shifts. After the 'wait duty' the rest time is often more than 11 hours. Before and after very short duties (like a morning shift from 8 till 12) a shorter break can be acceptable. It may sound unusual to put this legal requirement among the soft constraints, however it is better to violate the legally compulsory constraint on free time between assignments, than to leave patients unattended.
- **Maximum and minimum number of work hours during the planning period**. These numbers also depend on the work regulations for the hospital and the person involved (e.g. the person may be full time, half time, a night nurse, etc...). Overtime is very common in hospitals so an option is available that does not penalise the use of overtime unless a certain threshold is exceeded.

— Working practices, arrived at by mutual agreement between the head nurse (or personnel manager) and the individual nurses:

- **Maximum number of assignments during the planning period**. For full time nurses this is usually restricted to 20 assignments per 4 weeks. Some hospitals prefer to allow one compensation day per month and limit this constraint to 19. This constraint can be set to 10 for half time nurses.
- **Working full weekends**. This means that when a shift is scheduled on a day of the weekend, an assignment is required for the other day(s) of the weekend. For this constraint, a weekend can be either Saturday and Sunday, or Friday, Saturday and Sunday. In the latter case, there is an extra potential difficulty: a scheduled shift on Friday does not necessarily imply working on Saturday and Sunday but a scheduled shift on Saturday or Sunday requires shifts on Friday and Sunday/Saturday as well.
- **Working according to a predefined pattern**. The length of a pattern can vary from a couple of days to a number of months. At the end of a pattern, the pattern will start all over again (it is independent of the planning period). Each day, a pattern can contain one of the following: a compulsory shift, a compulsory shift of a certain type, a compulsory shift of a certain duration, a compulsory free day, a day on which certain shifts are excluded and 'no restriction on whether something is scheduled or not'.
- **Maximum number of assignments of each duty type during each week, and during the entire planning period**. This constraint can, for

example, limit the number of night shifts per week (not more than 4 night shifts per week) and limit the total number of night shifts per month (not more than 8 per month).
- **Restricting the order in which shifts and free days may and may not be scheduled (for instance never a night duty the day after a free day)**. Some sequences that are highly unwanted (like morning duty after an evening duty) can be excluded with this constraint.
- **Distributing the duty types uniformly over people with the same work regulation**. With this constraint it is possible to schedule the same number (within a certain deviation) of morning, night, waiting shifts, ... for all the full time nurses.

— Ad-hoc personal preferences agreed between the nurse and the head nurse or personnel manager:

- The request for holidays, days off, etc. ...
- The request to work a certain duty on a certain day
- Leave due to illness
- Maternity leave
- Temporary secondment to another ward

It is usually the case that not all constraints can be satisfied at the same time. When a contradiction between constraints occurs, the personal preferences of staff (such as requests for holidays or requests to work a certain duty on a certain day) are usually given more weight than the other constraints.

2.3. Evaluation Function

The evaluation function for this problem is implemented as a series of modules, each corresponding to a soft constraint. The user fixes the parameters and sets the penalty weight per unit violation of the constraint. The evaluation function is the sum of the penalties for each violated soft constraint. We give a simple example to illustrate this. Assume that we have a ward in which the constraint on the maximum number of night shifts per person per week is set to 3 and the corresponding penalty weight to 10. Consider a planning period of 4 weeks and one person of the ward is assigned 3 night duties during week 1, 0 night duties in week 2, 5 night duties in week 3, and 4 night duties in week 4. The total magnitude of the violation of the constraint is 3 (0 for the 1st and 2nd week, 2 for the 3rd week, and 1 for the 4th week). The value of the evaluation function, corresponding to this particular constraint, is 3, the magnitude of constraint violation, multiplied by 10, the penalty weight per unit violation of the constraint. Similar calculations have to be done for each constraint, and the results added to give an overall measure of schedule quality.

3. The Tabu Search Algorithm and Variants

In this section, we present a brief description of the tabu search algorithm and its diversification heuristics which were introduced in [10].

One of the main difficulties in this particular problem is that different categories of qualification can have overlapping rosters. When the head nurse has a day off, for instance, one of the qualified regular nurses can be scheduled to do the head nurse's job on that day. This complicates the hard constraints as illustrated in Fig. 2. In this case, we must avoid allocating other duties for this substitute. Since we opted to plan each qualification category separately (to reduce the complexity of the problem), the partially solved schedules sometimes prevent the planning algorithm from finding good quality solutions for the other qualification categories. This is one of the main reasons why the tabu search algorithms were not always satisfactory and why we decided to investigate a population based approach. In all the algorithms described in this section the qualification categories are planned in the order given by the user, usually starting from the most tightly constrained category (e.g. head nurse) and ending with the most loosely constrained (e.g. student nurse).

The first part of the scheduling algorithm involves the construction of a feasible initial solution. We have defined three possible strategies for initialisation: *current schedule*, which is used particularly to generate a new schedule quickly in response to unforeseen events such as staff illness, *schedule of the previous planning period*, which is useful when the previous and the current planning periods have similar constraints, and *random initialisation*, which starts from an empty schedule. In each case, the schedule is made feasible before starting the algorithms. This feasible solution is obtained by adding or removing appropriate duties from the schedule randomly, without taking into account any of the soft constraints, until the hard constraints are met. In Fig. 2, this corresponds to randomly adding appropriate ticks until we achieve column sums corresponding to R-min or R-pref.

In the simplest tabu search algorithm, the only move we consider is a move of a duty from one person to another on the same day. Essentially we move one tick within a column of the solution representation given in Fig. 2. The move is not allowed if the goal person is not of the right category of qualification or is already assigned to that duty. Hence the hard constraints will still be respected. At each iteration, the best move is selected. If it improves the current solution, it is accepted. If no improving move is found then the best non-improving, non-tabu move is accepted. When a move is made a rectangular area around the original tick and the new tick location are added to the tabu list (see [10]). This simple algorithm turned out not to be powerful enough to produce good solutions for complex problems (see [10]).

Now we will briefly mention some heuristics that can be employed (in conjunction with the tabu search algorithm) to improve the solution.

Diversification 1: Complete Weekend. Although the users of the program can assign a cost parameter to the soft constraint of working complete weekends (see Section 2.2) it is very hard to find schedules which satisfy all of these constraints. Here we diversify by ignoring all soft constraints except those concerning complete weekends. This move often guides the algorithm to an unexplored part of the solution space.

Diversification 2: Consider the Worst Personal Schedule. If the complete weekend function (above) has not changed the schedule it can be beneficial to look at staff having the worst personal schedule (according to the evaluation function). For every person (within the category being scheduled) the effect on the evaluation function is calculated after exchanging a part of his or her schedule with the corresponding part of the worst personal schedule, of a person within that qualification category. The exchanged parts of the schedule always contain full days and the maximum length is half the planning period. After the time consuming process of exploring all possible exchanges, the best exchange is performed, with ties broken randomly. This process often results in a better solution.

Greedy Shuffling: Model Human Scheduling Techniques. There was a problem with the results of the tabu search algorithm because sometimes a human could improve the result by making small changes. The greedy shuffling process calculates all possible 'Diversification 2' moves, above, for each pair of staff from the same qualification category. After listing the gain in the cost function for every possible exchange, the shuffle leading to the best improvement will be performed. Afterwards, the next best improvement in the list is carried out, provided that none of the people involved were already involved in an earlier shuffle. As long as there are improving exchanges in the list, they are carried out. The whole procedure starts over again until none of the possible exchanges improves the cost function. The improvements that can be obtained by employing this procedure and tabu search (described below) are considerable.

After extensive testing of hybrid versions of the tabu search algorithm and the above heuristics, two algorithms were developed. The first one produces schedules when a very short calculation time is required (as it often is when planners must react to unforeseen events such as staff absenteeism). The second algorithm needs more calculation time but generates schedules of a considerably higher quality. Both algorithms are briefly described below.

Tabu Search + Diversification: TS1. The aim of this algorithm is to provide reliable solutions in a very short time. In practice this algorithm has proven to be very useful to check whether the constraints are realistic. For example: "Will it be possible to plan good schedules if every person gets their desired holiday period?" etc.

The algorithm is constructed quite simply from the original tabu search algorithm. If after a number of iterations no improvement is found, the complete weekend diversification step is performed. If the weekend step does not result in a different schedule, the second diversification step is performed. After this diversification step, the original tabu search algorithm is used again, and so on. The calculations stop after a number of iterations without improvement.

Tabu Search + Greedy Shuffling: TS2. This requires more time but the results are considerably better. Anecdotal evidence suggests that the level of satisfaction with schedules produced by this algorithm among users is actually higher than the cost function indicates. The main reason for this is that after the shuffling step the users cannot easily improve the results.

It is important to carry out the greedy shuffling step at the end of the calculations because its real aim is to perform the exchanges that a human user would perform. It is because of the exhaustive search of the large

neighbourhood given by shuffling to reach a local optimum, that this step takes a lot of time. It is very important to calculate this step until there are no further improvements because otherwise the goal of excluding manual improvements to the schedule might be lost. This also holds for the population based algorithms that will be described in the following sections.

4. Genetic and Memetic Algorithms

As mentioned above, the tabu search algorithms do display considerable shortcomings, an important one being that they are not robust enough to handle difficult problems well. This provided the motivation to investigate population based approaches for the same problem. A basic genetic algorithm with just mutation and crossover operators can be employed but to reach convergence it is important to have crossover operators that combine parts of good parent solutions to produce good child solutions. A common difficulty with rostering problems is that the quality of a solution is not necessarily a sum or a combination of the qualities of the partial solutions. We have carried out a number of experiments with several crossover operators, either conserving the "building blocks" as much as possible or repairing the roster when it is destroyed by the crossover.

This section will describe a memetic algorithm that incorporates the tabu search and hybrid tabu search algorithms into a genetic algorithm. The components of the algorithms described in this section and in Section 5 can be seen in Fig. 4. We will describe how the algorithms can be constructed with the components displayed. In the memetic algorithm used for the nurse rostering problem, an initial population consists of N individuals, each of them feasible schedules generated randomly using the initialisation techniques discussed in Section 3. There are several possibilities for recombination to create new offspring. It is very important to organise the recombination so that the child rosters inherit good characteristics of the parent generation. Since the quality of a schedule is the sum of the schedule quality for each person, it is important to get these personal schedules right. The characteristics of the constraints are such that mixing up the scheduled events for a person usually leads to very bad schedules. In the following algorithms, we have used many variants of the recombination operators. Some preserve personal schedules to a great extent, others do not but preserve the position of well-placed events. Each generation requires significant calculation time, so we decided not to plan a large number of generations. Each of our memetic algorithms stops when no improvement arises for two consecutive generations. The variants of the memetic algorithm (described below) contain different recombination mechanisms.

Original memetic algorithm: M In the simplest memetic algorithm, a steepest descent is performed for each individual. The steepest descent algorithm uses the same neighbourhood for the moves as the simplest tabu algorithm where the planning order of nurse categories is as given by the user. After evaluating all the possible moves in the neighbourhood, the best one is performed, unless this best move does not improve the schedule, in which case steepest descent stops. After this step, there is a simple tournament selection of the best individuals for creating offspring. For each pair of parents, two new individuals are created. The first child contains the best personal schedule (referred to as 'row' in the schedule) from the first parent + the best personal schedule from the second parent (different from the first one selected). The other personal schedules are chosen in a pairwise tournament between the rows of the parents. This normally does not result in a feasible schedule, so to make the child schedules feasible, shifts are added or taken away at random (where necessary). This leads to diverse schedules of poor quality, prior to the application of the steepest descent heuristic.

1. *create N different schedules using random initialisation*
   while stop criterion is not reached
   (*stop criterion*: *no improvement during two generations*), *repeat*:
2. *make all the schedules feasible by randomly adding and deleting appropriate shifts* (*ticks in each column in figure 2*)
3. *perform steepest descent on each of the individuals*
4. *select parents from the individuals by tournament selection*
5. *recombine the parents*,
   per pair of parents, *generate two children*:
   child 1:
   best personal schedule (*row*) *from parent* 1
   best personal schedule (*row*) *from parent* 2
   (*or the second best one if the same row is best for both parents*)
   for the other child rows use tournament selection from parent rows
   child 2:
   best personal schedule (*row*) *from parent* 2

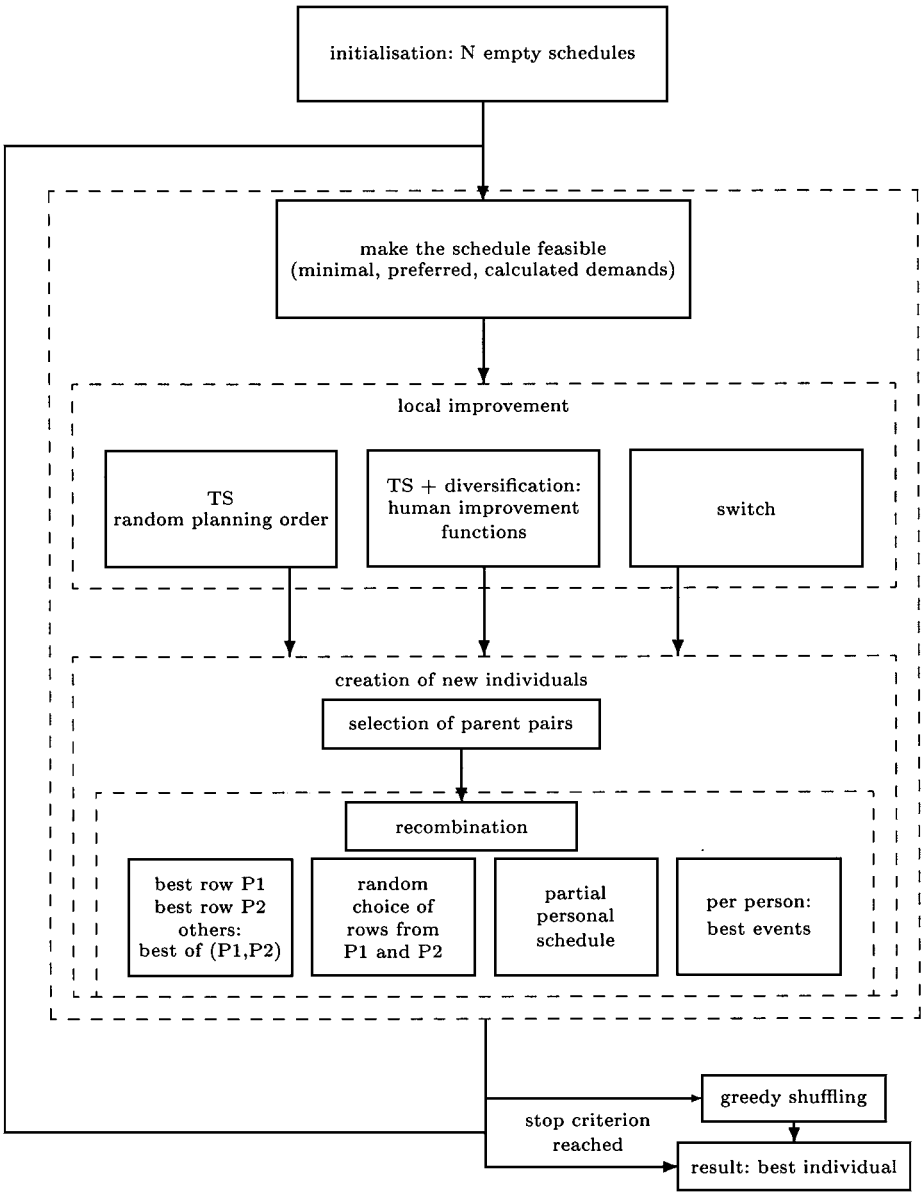

Figure 4. Diagram of the components of the genetic and memetic algorithms for the nurse rostering problem.

best personal schedule (*row*) *from parent* 1
(*or the second best one if the same row is best for both parents*)
for the other child rows use tournament selection from parent rows

Diverse memetic algorithm: DM With the shortcomings of the tabu search algorithms in mind, we decided not to plan the qualification categories according to the planning order chosen by the customers as in algorithm **M** above. In the **DM** algorithm, each time the steepest descent algorithm is performed, the planning order of the qualification categories is randomly chosen for each schedule, causing additional diversity in the population. All other features of the algorithm are the same as for algorithm **M**.

Diverse memetic algorithm with random selection: DMR Here we use the steepest descent approach and other features of the **DM** algorithm, but rather than

choose the rows of the child rosters by tournament selection, each personal schedule is chosen randomly from one of the parents.

Memetic algorithm with string recombination: MSR In this algorithm, a different technique is used to generate offspring. It no longer copies an entire schedule from one of the parents. For each person (row) in the schedule, a time unit (day and shift, corresponding to a column of Fig. 2) is randomly chosen. The part of the schedule between the start of the planning period and this time unit is copied from the first parent and the remainder from the second parent. The procedure is repeated for the second child, except that the first part of each row of this child is taken from the second parent and the remainder from the first parent. Apart from this new recombination operator, this algorithm is the same as the **DM** algorithm.

Memetic algorithm copying the x best events: MEx The difference between the **MEx** algorithm and the **DM** algorithm considered previously, is the way in which child schedules are generated from their parents. Good results were found when copying the 'best placed' events for every person, from the parents to the children. The 'best placed' events are those events that would lead to the worst increase in the cost function when removed. Experiments have been carried out that copy $x = 2, 3, 4$, and 5 events per parent to the child. If the best personal events are the same in both parents, this will lead to only x events in the offspring. Again, the schedules are made feasible by randomly adding the other events, which leads to greater diversity.

5. Combining the Qualities of the Hybrid Tabu Search and the Evolutionary Algorithm

Tabu search algorithms using different initial solutions and randomising the planning order of the qualifications: TSPOP The tabu search algorithm (without hybridisations) and the memetic algorithms do not always lead to excellent solutions for the complicated problems hospital planners have to deal with. Unfortunately, the quality of the solution depends strongly on the initial schedule. This is not to say that the schedule has to be good. On the contrary, very good initial schedules are sometimes hard to improve by the methods considered. The problem is actually that the small move made during the tabu search (Section 3) cannot lead the solution away from some of the local minima. In the commercial version of the program, only the hybrid versions of the tabu search algorithm are used. The users' informal feedback about the hybrid algorithms points to insensitivity to the random seed (initial solution). The strength of the memetic algorithm approach is that many different starting points are taken and a diversity of different schedules is maintained throughout. An even bigger advantage is the possibility of planning the qualification categories in different orders. It might be argued that starting our tabu search from multiple different starting points might produce solutions of comparable quality to the memetic algorithms and hybrids. In order to compare the performance of our tabu search approaches and our memetic approaches and hybrids, given similar time, the **TSPOP** algorithm first produces a population of initial solutions that are (one by one) improved by the **TS1** algorithm (Section 3) except that the ordering of qualification categories is random. Greedy shuffling (also explained in Section 3), is applied to the best individual solution of the population (see also Fig. 4).

Memetic algorithm with human improvement functions: MEH If we combine the extra functions of the tabu search algorithm with the memetic algorithm, either by using them as local improvement functions or by performing them on the best individual of the memetic algorithm, the resulting solutions are much improved. The solutions thus found are of the same quality as, or better than the solutions found with the hybrid tabu search algorithm. Better solutions are found due to the diversification of the algorithm. By starting the calculations from different starting situations, and by changing the planning order of the qualification categories, the probability of finding better solutions is increased. For the tests, we used the **ME4** algorithm from Section 4. We then apply the greedy shuffling step (Section 3) to the best individual obtained with this **ME4** algorithm.

1. *create N random individual solutions*
   while the stop criterion is not reached
   (*stop criterion*: *no improvement during two generations*), *repeat steps* 2 *to* 5:
2. *make all the schedules feasible by adding and removing appropriate shifts at random*
3. *perform the TS1 on each of the individuals*
   (*choose the planning order of the qualifications at random*)
4. *select parents from the individuals by tournament*

5. *recombine the parents as explained in the ME4 algorithm*
6. *perform the greedy shuffling step on the best individual*

Switch: SWT In all the previous algorithms, the number of staff of each type in each shift remained constant once the initial feasible solution was found. Here we add an additional move where, every now and then, the schedule is randomly changed, for a random person and at a random day and shift. In case nothing was scheduled for this person at this time, we introduce a new 'event' in the schedule (i.e. if there is a blank at that row and column in Fig. 2, we insert a tick). In case something was scheduled already, we remove the scheduled event (i.e. if there is a tick at that row and column in Fig. 2, we remove it). The random changes may not violate the hard constraints. We allow a change so long as the number of staff in a given qualification category for each shift is not lower than the minimum number and not higher than the preferred number. Let us for example assume that the number of late shifts for caretakers is minimum 2 and preferably 3 on a certain day in the planning period. Let us also assume that in the current solution 3 late shifts are scheduled on that day. Assume that the 'switch' move, described in this section, randomly picks this day and the late shift for the move. In the case where the randomly chosen person was not assigned to the late duty that day, the algorithm cannot change this because adding a late duty for that person would violate the hard constraints (4 late duties planned instead of the preferred number of 3). The other possibility is that the algorithm picks at random a person who was already scheduled for the late duty. In that case, the late duty will be removed. The removal causes no violation of the hard constraints since the minimal required number of scheduled late duties is still scheduled. With the 'switched' tabu search algorithm, we allow more flexibility for staff to be scheduled at lower or higher levels than usual, where this is permitted, as discussed in Section 2.1. We decided to alternate the ordinary tabu search and the switch function within the **SWT** algorithm, since after a switch the resulting solution is often of poor quality. The **SWT** algorithm is the **ME4** algorithm with this additional change.

6. Results

We tested our algorithms on four difficult real world rostering problems arising in Belgian hospitals. Due to complex confidentiality and operating requirements, gathering each set of problem data required significant amounts of time and effort. Each of the four rostering problems has different characteristics and consistent performance across these four different problems provides strong empirical evidence of performance overall. It is not appropriate to test our algorithms using random data that would not have the same difficulties as those encountered in practice. The results of applying our heuristics to the problems are given in Tables 1–4. We have given the value of the evaluation function and the time taken for the planning of the minimal staff requirements (R-min), planning towards the preferred numbers of staff at each qualification (R-min-pref), and planning the recalculated requirements which will aid satisfaction of the soft constraints (R-calc) as explained in Section 2.1. In all the tables, the column 'Value' shows the value of the evaluation function (cost parameter per soft constraint multiplied by the extent to which the constraint is violated). The column 'Time' contains the calculation times on an IBM Power PC RS6000.

Problem 1 is large with many conflicting soft constraints. It is by far the most difficult problem of this group. Problems 2 and 4 are smaller problems with few soft constraints and are much easier than problem 1. Problem 3 is of intermediate complexity.

The original tabu search algorithm **TS** is very good at producing a reasonable starting solution from a random initialisation in a short time. It is, however, a very slow method if it is restarted several times in order to generate good solutions. Though the results of the hybrid tabu search algorithms **TS1** and **TS2** are considerably better they are often far from optimal. Unacceptable solutions usually arise when the constraints on the problem are contradictory. It is then very hard to find the very narrow valleys in the solution space, which contain good schedules. Giving a very high value to the cost parameter corresponding to a particular constraint does not necessarily guarantee that the solution will be free from violations of this constraint. There are particular difficulties when the requirements for people within a certain qualification category are higher than the number of people available. In the tabu search algorithm every ward is planned qualification category by qualification category. The customers can freely decide upon the order in which this planning is carried out. When planning such a qualification category, the algorithm is free to place the required shifts on every day that the people within the category under consideration (or with

Table 1. Comparison between the algorithms for problem 1.

	R-min		R-min-pref		R-calc	
Problem 1	Value	Time	Value	Time	Value	Time
M (10 generations)	1445	52′33″	1296	52′42″	1103	50′18″
M (100 generations)	1435	4h46′34″	1293	4h59′04″	1100	4h22′41″
DM	1334	52′40″	1298	52′58″	1091	50′28″
DMR	1993	1h00′01″	1829	1h00′20″	1716	59′01″
MSR	1991	31′14″	1806	31′49″	1695	34′51″
ME2	1397	1h24′25″	1266	1h24′36″	1203	1h16′57″
ME3	1300	2h10′18″	1209	2h10′40″	1120	2h13′45″
ME4	1208	2h08′30″	1087	2h08′41″	1003	2h00′43″
ME5	1322	2h02′43″	1199	2h02′52″	1127	1h58′22″
TS	2435	2′05″	2214	2′06″	1928	1′59″
TS1	1341	6′00″	1089	5′59″	929	5′27″
TS2	1264	20′15″	1011	24′39″	809	28′08″
TSR	2893	2′20″	2714	2′21″	2983	2′22″
TS1R	1911	18′33″	1692	35′14″	1573	35′03″
TS2R	1911	34′16″	1691	53′55″	1573	40′09″
TSPOP (12 individuals)	1352	1h38′33″	1089	1h42′55″	736	1h41′47″
TSPOP (24 individuals)	1352	3h40′16″	1083	3h07′30″	746	2h52′02″
MEH	1192	2h22′04″	904	2h28′28″	769	2h31′51″
SWT	1090	1h45′17″	1094	2h20′26″	807	30′14″

this category as an alternative possibility) are available. When this step of the algorithm stops, the shifts planned for this qualification category are frozen. This sometimes causes difficulties in planning the shifts for other qualification categories, because of the overlap caused by staff who may work in more than one qualification category.

Since the test problems consist of wards with different qualification categories, we also carried out experiments to determine the influence of changing the planning order. These results are only significant for the tabu search experiments, in which the user of the system is free to define a planning order for the qualification categories. The other algorithms use a random ordering of these categories. Practical experiments have shown that the best strategy is to plan those categories which are understaffed first. However, determining the ordering of the categories is difficult and requires the human planner's expertise. We present here the results of planning the qualification categoriess in the reverse of the order initially chosen by the experienced planner. We suppose this to be the worst case and our results certainly support this hypothesis. The results for Problem 1 and Problem 2 are in the **TSR**, **TS1R** and **TS2R** rows of Tables 1 and 2 respectively.

We see that, in general, using a poor ordering of qualification categories produces much poorer results. This is particularly true for the difficult problem 1. Surprisingly, the **TS2** algorithm produces slightly (2%) better results for the relatively easy problem 2 when the planning order is reversed in **TS2R**. This behaviour is due to the greedy shuffling step performed at the very end of the **TS2** algorithm's calculation. This step goes through all the qualification categories following the planning order again, at the end of the solution process. It is thus possible to make changes in the schedule of previously planned categories again. The original **TS** tabu search algorithm and **TS1** go through the schedule qualification by qualification, visiting each category only once, with no chance to rectify poor choices later. They are thus more strongly affected by the bad planning order of the qualification categories.

Unless indicated otherwise in the tables, all the memetic algorithms stop after two generations without improvement. This is typically less than 20 generations. All the algorithms used to produce the

Table 2. Comparison between the algorithms for problem 2.

	R-min		R-min-pref		R-calc	
Problem 2	Value	Time	Value	Time	Value	Time
M (10 generations)	1245	4′9″	1245	4′9″	1060	4′12′
M (100 generations)	1245	10′53″	1245	10′53″	1060	10′22″
DM	800	4′10″	800	4′10″	823	4′40″
DMR	1037	2′56″	1037	2′56″	992	2′56″
MSR	1104	2′51″	1104	2′51″	1080	2′59″
ME2	1123	22′02″	1123	22′02″	1130	22′02″
ME3	752	22′16″	752	22′16″	748	21′42″
ME4	698	22′01″	698	22′01″	707	22′10″
ME5	782	21′51″	782	21′51″	769	22′05″
TS	1189	57″	1189	58″	933	1′03″
TS1	843	3′18″	843	3′18″	867	2′14″
TS2	809	6′25″	809	6′25″	588	10′19″
TSR	2614	48″	2614	50″	1584	49″
TS1R	1875	1′04″	1875	1′06″	554	8′21″
TS2R	790	15′10″	790	15′11″	554	9′37″
TSPOP (12 individuals)	885	29′36″	885	29′37″	464	32′34″
TSPOP (24 individuals)	892	50′19′	892	50′21″	457	53′48″
MEH	980	23′54″	980	23′54″	535	24′42″
SWT	992	19′09″	992	19′09″	578	19′26″

Table 3. Comparison between the algorithms for problem 3.

	R-min		R-min-pref		R-calc	
Problem 3	Value	Time	Value	Time	Value	Time
M (10 generations)	567	23′46″	560	23′58″	547	21′20″
M (100 generations)	552	1h37′14″	541	1h37′25″	547	1h15′14″
DM	403	23′57″	402	24′10″	396	24′10″
DMR	636	28′45″	629	28′58″	620	27′51″
MSR	612	27′12″	610	27′30″	604	27′28″
ME2	526	1h17′17″	521	1h17′30″	518	1h11′42″
ME3	472	57′03″	466	57′16″	459	56′54″
ME4	398	1h16′43″	392	1h16′41″	391	1h00′23″
ME5	397	1h09′55″	393	1h10′05″	390	1h04′42″
TS	422	2′05″	418	2′08″	415	2′06″
TS1	398	7′38″	389	7′42″	390	7′38″
TS2	391	13′52″	380	13′56″	377	14′11″
TSPOP (12 individuals)	624	1h35′50″	620	1h48′06″	583	1h32′27″
TSPOP (24 individuals)	608	2h56′11″	608	3h16′12″	583	2h55′00″
MEH	378	1h20′44″	379	1h22′45″	369	1h28′03″
SWT	381	1h17′09″	375	1h30′10″	364	1h26′32″

Table 4. Comparison between the algorithms for problem 4.

	R-min		R-min-pref		R-calc	
Problem 4	Value	Time	Value	Time	Value	Time
M (10 generations)	226	8′47″	226	8′47″	224	8′09″
M (100 generations)	225	33′27″	225	33′27″	224	32′54″
DM	241	9′21″	241	9′21″	237	9′42″
DMR	266	9′33″	266	9′33″	260	10′39″
MSR	273	8′40″	273	8′40″	265	9′15″
ME2	200	42′11″	200	42′11″	205	39′23″
ME3	184	46′31″	184	46′31″	184	46′22″
ME4	186	45′40″	186	45′40″	187	47′19″
ME5	191	42′16″	191	42′16″	190	44′57″
TS	231	52″	231	53″	227	1′03″
TS1	190	2′21″	190	2′23″	189	2′14″
TS2	189	4′25″	189	4′26″	186	4′38″
TSPOP (12 individuals)	269	22′16″	269	23′09″	266	24′28″
TSPOP (24 individuals)	264	39′03″	264	40′21″	263	43′10″
MEH	182	23′54″	182	24′35″	175	24′42″
SWT	179	19′09″	179	19′59″	176	19′26″

test results in Tables 1 to 4 have a population size of 12.

We can see that for each of the problems, the extra benefit from allowing our original memetic algorithm **M** to run ten times longer produces little improvement in the final solution, since this approach does not generate sufficient diversity. However, we see that for problems 1, 2, and 3 the **DM** heuristic produces better rosters than the **M** heuristic, since the random ordering of qualifications gives greater population diversity. However, we can also see that the **DMR** heuristic, which introduces still more diversity through choosing random rows from the parent schedules instead of the best rows in each case has produced results which are worse than those for the **DM** heuristic. The same could be said for heuristic **MSR** which takes an appropriate segment of each row from each parent and has a similar performance. Each of these memetic algorithms has a population size of at least 12, which explains their slower running time. We will consider below the comparison between the memetic approaches and a multistart tabu search approach **TSPOP**.

Copying large parts schedules from high quality parent schedules to the child schedules is ineffective. Since the steepest descent technique used by the memetic algorithms is not powerful enough to improve the children, so that the algorithm performs like a pure genetic algorithm. We have obtained better results by copying small parts (with good qualities) from the parent schedules, so that the degree of freedom after making feasible solutions is high enough to provide diversity. Hence we see that the memetic algorithms **ME2, ME3, ME4**, and **ME5** which are more selective about which parental traits are passed on generate significantly better schedules than the algorithms **M, DM, DMR**, and **MSR**. The algorithms copying the smallest parts of the parent schedules are the **MEx** algorithms. We found that the number **x** of events which are copied per person (row) has a significant influence on final solution quality. The results of copying 2, 3, and 4 events per row are increasingly better but from 5 events on the results get worse again. We believe that these 'best placed' events strongly influence the position of all the other events, so that the freedom of the solution to evolve is restricted to good areas of the search space and the steepest descent heuristic is particularly effective in improving the diverse schedules generated. **ME4** represents the best compromise between a diverse population of solutions and the ability to focus on interesting areas of the search space.

When we compare the best of our memetic algorithms, **ME4**, with **TS2**, the best of our tabu search

algorithms, we see that significantly better solutions are produced by the **ME4** algorithm for problems 1 and 2 and comparable results for problems 3 and 4, at the expense of longer running times. A very important advantage of the memetic algorithms with respect to the tabu search algorithms, however, is the fact that the results of the memetic algorithms are not dependent on the planning order of the qualification categories chosen by the user. The chance of becoming trapped in a local minimum, which is very far from the optimal, is reduced. Problem 1 has a high number of very strict soft constraints, with high cost parameters attached to them. Problem 2, on the other hand, has fewer personnel and duty types and has few soft constraints. Problem 1's search space (only depending on soft constraints) will thus be much hillier and full of traps for algorithms based upon neighbourhood search, so the problem will differentiate more clearly between the algorithms. This explains why the improvements of the hybridisations **TS1** and **TS2** are considerably better for problem 1 than those of the original tabu search algorithms **TS**, and the additional improvements yielded by the memetic approaches.

We can see in the **TSPOP** rows of Tables 1 to 4 that applying the **TS1** algorithm using random ordering of qualification categories starting from a number of different initial solutions (without any recombination or switch) is not effective—showing clearly the dependence upon the human planner's knowledge of the sequencing which must be applied to the ordering of nurse qualification categories. It seems to be a good idea not to do the time consuming greedy shuffling step on every individual but only on the best one. The algorithms make use of a random generator (to create an initial solution and to choose among equally good steps). This explains why some of the experiments with more individuals lead to worse solutions than with fewer individuals. However, the memetic hybrids **SWT** and **MEH** and the memetic algorithm **ME4** demonstrate that the recombination operator consistently improves performance over the **TSPOP** algorithm (which has no recombination) given similar time to solve the problems, and moreover, they do not require the user to specify the order in which nurse qualification categories should be scheduled.

The memetic/tabu hybrid **MEH** shows excellent performance over the more difficult problems 1 and 4 (bettering all other solution methods except the hybrid **SWT** algorithm). This demonstrates the better solutions obtainable and the increased robustness offered by a hybrid approach.

Originally, when planning R-min-pref, we start by planning the minimum personnel requirements and in the end add duties to the schedule whenever this does not introduce new violations of soft constraints. We are inclined to think that better results can be obtained by adding duties while the planning algorithm is still active. The **SWT** algorithm, which was developed to test this, indeed leads to good results for all the examples tested. Note that this algorithm has no greedy shuffling step in the end, in contrast to the **TSPOP** and the **MEH** algorithms.

7. Conclusion

By automating the nurse rostering problem for Belgian hospitals, the scheduling effort and calculation time are reduced considerably from the manual approach that was previously used. The time for automatic schedule generation can be tailored to suit the time available. Fast tabu search algorithms can quickly find reasonably good schedules in response to events such as staff absenteeism. The memetic algorithms are robust enough to produce excellent solutions to hard problems when more time is available. The quality of the automatically produced schedules is much higher than the quality of the manual schedules. The users of Plane often place an emphasis on the higher quality of the solution because the system provides an objective schedule in which all nurses are treated equally and in which the number of violated constraints is very low. Planners are pleased by the ability of the system to generate consistently better solutions than manual planning procedures, demonstrating a high level of robustness. They acknowledge also that experienced planners cannot improve the schedules generated to improve the results manually.

This paper deals with real world problems and some of the algorithms are currently in use. We have described several memetic approaches and compared them to previously published tabu search results [10]. The hybrid tabu search algorithm runs quickly and does produce good solutions but it is highly dependent on the initialisation parameters, requiring the expertise of human planners to judge the correct order of qualification categories and displays a lack of robustness to generate good schedules for all problems. The memetic approaches take much longer to run than the tabu search approaches. Those memetic approaches

which copy only a carefully selected part of each parent schedule to the child schedules use this extra time to good effect to produce better solutions and the dependence on the initialisation and parameter changes is very much reduced. The hybrid memetic algorithms, which combined the basic approach with the hybrid tabu search, provide good solutions in a similar time to the other memetic algorithms. The solutions are significantly better than the best tabu search solution and they are relatively unaffected by initialisation and parameter changes. We believe that these approaches are particularly robust and can handle the variety of instances that occur in the real world.

For many practical scheduling problems, the higher quality of the solutions produced by the hybrid algorithm compared to the simple tabu search algorithm compensates for the increase in calculation time. Different users will choose different algorithms, depending on their opinions and their requirements. The runtime/quality trade-off depends very much on the individual user. Some users are really interested in the lowest possible value of the evaluation function, no matter how long the calculations take, particularly in smaller hospitals where a single planning officer generates the roster for the whole hospital and will not mind if roster generation takes an overnight run. Others, for instance in very big hospitals with many wards to be scheduled by individual head nurses, prefer quick calculations where a slightly lower schedule quality is good enough, since each head nurse may have a very tight window in which to generate a schedule.

Notes

1. Impakt N.V., Ham 64, B-9000 Gent.
2. GET, General Engineering & Technologie, Antwerpse Steenweg 107, B-2390 Oostmalle.

References

1. K. Dowsland, "Nurse scheduling with tabu search and strategic oscillation," *EJOR* to appear.
2. A. Meisels, E. Gudes, and G. Solotorevski, "Employee timetabling, constraint networks and knowledge-based rules: A mixed approach." In *Proc. Practice and Theory of Automated Timetabling*, *First International Conference Edinburgh*, pp. 93–105, 1995.
3. M. Okada, "Prolog-based system for nursing staff scheduling implemented on a personal computer," *Computers and Biomedical Research*, vol. 21, pp. 53–63, 1988.
4. M. Okada, "An approach to the generalised nurse scheduling problem—Generation of a declarative program to represent institution-specific knowledge," *Computers and Biomedical Research*, vol. 25, pp. 417–434, 1992.
5. H. Miller, W. Pierskalla, and G. Rath, "Nurse scheduling using mathematical programming," *Operations Research*, vol. 24, pp. 857–870, 1976.
6. V.M. Trivedi and M. Warner, "A branch and bound algorithm for optimum allocation of float nurses," *Management Science*, vol. 22, no. 9, pp. 972–981, 1976.
7. M. Warner, "Scheduling nursing personnel according to nursing preference: A mathematical programming approach," *Operations Research*, vol. 24, pp. 842–856, 1976.
8. M. Warner and J. Prawda, "A mathematical programming model for scheduling nursing personnel in a hospital," *Management Science*, vol. 19, pp. 411–422, 1972.
9. H. Meyer auf'm Hofe, "ConPlan/SIEDAplan, Personnel Assignment as a Problem of Hierarchical Constraint Satisfaction," in *Proc. PACT 97*, pp. 257–271, 1997.
10. E.K. Burke, P. De Causmaecker, and G. Vanden Berghe, "A hybrid tabu search algorithm for the nurse rostering problem," in *Proc. Second Asia–Pacific Conference on Simulated Evolution and Learning* (SEAL'98) Springes Lecture Notes on AI, vol. 1585, pp. 186–194, 1998.
11. P. Coppens, "Geautomatiseerde personeelsplanning bij het A.Z. Sint-Erasmus," *GET info*, vol. 9, pp. 1–3, 1995.
12. E.K. Burke and J.P. Newall, "A multi-stage evolutionary algorithm for the timetable problem," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 1, pp. 63–74, April 1999.
13. E.K. Burke and A.J. Smith, "Hybrid evolutionary techniques for the maintenance scheduling problem," *IEEE Transactions on Power Systems*, vol. 15, no. 1, pp. 122–128, Feb. 2000.
14. E.K. Burke, J.P. Newall, and R.F. Weare, "Initialisation strategies and diversity in evolutionary timetabling," *Evolutionary Computation Journal* (Special Issue on Scheduling), vol. 6, no. 1, pp. 81–103, 1998.
15. R.W. Cheng and M. Gen, "Parallel machine scheduling problems using memetic algorithms," *Computers & Industrial Engineering*, vol. 33, no. 3–4, pp. 761–764, 1997.
16. P. Moscato and M.G. Norman, "A 'Memetic' appproach for the traveling salesman problem. Implementation of a computational ecology for combinatorial optimization on message-passing systems," in *Parallel Computing and Transputer Applications*, edited by M. Valero, E. Onate, M. Jane, J.L. Larriba, and B. Suarez, IOS Press: Amsterdam, pp. 187–194, 1992.
17. B. Paechter, A. Cumming, M.G. Norman, and H. Luchian, "Extensions to a memetic timetabling system," in *Practice and Theory of Automated Timetabling*, edited by E.K. Burke and P. Ross, Springer Lecture Notes in Computer Science, vol. 1153, pp. 251–265, 1996.
18. N.J. Radcliffe and P.D. Surry, "Formal memetic algorithms," in *Evolutionary Computing*, edited by T. Fogarty, Springer Lecture Notes in Computer Science, vol. 865, pp. 250–263, 1994.
19. W.J. Abernathy, N. Baloff, J.C. Hershey, and S. Wandel, "A three-stage manpower planning and scheduling model—A service-sector example," *Operations Research*, vol. 22, pp. 693–711, 1973.
20. E.K. Burke, P. De Causmaecker, S. Petrovic, and G. Vanden Berghe, "Fitness evaluation for nurse scheduling problems," in *Proc. Congress on Evolutionary Computation 2001 (CEC 2001)*, IEEE Press, pp. 1139–1146, 2001.

21. R.D.T. Farmer and J. Emami, "Models for forecasting hospital bed requirements in the acute sector," *Journal of Epidemiology and Community Health*, vol. 44, pp. 307–312, 1990.
22. P. Gemmel and P. Coppens, "Hospitalia, Tijdschrift verbond der Verzorgingsinstellingen (1996) in the acute sector," *Journal of Epidemiology and Community Health*, vol. 44, pp. 307–312, 1990.
23. B. Paechter, A. Cumming, and H. Luchian, "The use of local search suggestion lists for improving the solution of timetable problems with evolutionary algorithms," in *Evolutionary Computation*, edited by T. Fogarty, Springer Lecture Notes in Computer Science, vol. 993, pp. 86–93, 1995.
24. A.P. Punnen, and Y.P. Aneja, "A tabu search algorithm for the resource-constrained assignment problem," *Journal of the Operational Research Society*, vol. 44, pp. 214–220, 1995.
25. S. Vanderhaeghe, "Dienstroosters in de gezondheidszorg: ongezond?" *Informatiedossier*, *Stichting Technologie Vlaanderen*, 1998.

Edmund K. Burke leads the Automated Scheduling, Optimisation and Planning (ASAP) research group within the School of Computer Science and IT at the University of Nottingham. He is the editor-in-chief of the Journal of Scheduling (published by Wiley) and has published widely in the areas of timetabling and evolutionary computation. He obtained his PhD degree from the University of Leeds. His current research interests include evolutionary computation and hyperheuristics and application to timetabling, production and maintenance scheduling, and space allocation problems.

Peter Cowling is a reader in computer science a member of the Automated Scheduling, Optimisation and Planning (ASAP) research group at the University of Nottingham. He obtained MA and DPhil degrees from the University of Oxford, following which he worked as commercial software consultant. His current research interests include personnel scheduling, heuristics and hyperheuristics, production scheduling and routing and modelling of commercial planning and scheduling problems. He remains active in consultancy and is currently developing a personnel scheduling system with a large bank.

Patrick De Causmaecker obtained his MSc in mathematics at the University of Ghent after which he became a researcher in theoretical physics at the University of Leuven where he obtained his PhD in 1983. After obtaining an additional degree in informatics he became a lecturer at KaHo St.-Lieven where he was responsible for a number of projects, always in collaboration with SME's. Recently he realised a funded research project for the application of agent technology in planning and scheduling environments. Focus of the project is augmenting the user friendliness and the robustness of these systems. He has published and spoken at conferences on operations research and applications of meta-heuristics.

Greet Vanden Berghe holds a degree in metals and materials science of the Universiy of Leuven. She started as a research assistant in KaHo St.-Lieven in 1992 with a software development project in the domain of recycling friendly design. Since 1994 she was involved in the development of a scheduling tool for nursing personnel. She has several publications and has talked at international conferences on operations research, most recently SEAL98. She is preparing a PhD on these subjects.