Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 153 (2004) 28–40
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

A greedy-based neighborhood search approach to a nurse rostering problem

F. Bellanti, G. Carello *, F. Della Croce, R. Tadei

DAI, Politecnico di Torino, Turin, Italy

Abstract

A practical nurse rostering problem, which arises at a ward of an Italian hospital, is considered. The nurse rostering problem is a typical employee timetabling problem, where each month it is required to generate the nursing staff shifts subject to various contractual and operational requirements. These requirements may be in conflict especially for those months in which manpower is reduced due to seasonal holidays. It is required to consider both holidays planning and parametric contractual constraints, but no cyclic schedules and corresponding weekly patterns. A local search approach is introduced which is based on a neighborhood operating on partial solutions completed by means of a greedy procedure so as to avoid the generation of infeasible solutions. Both a tabu search procedure and an iterated local search procedure are proposed. The solutions computed by the proposed procedures compare well with respect to a straightforward lower bound and strongly outperform those manually generated in the hospital ward. The proposed approach is now currently used on site.

Keywords: Nurse rostering; Tabu search; Iterated local search

1. Introduction

This work deals with a nurse rostering problem, which occurs at the intensive care unit of a hospital located in Turin, Italy. At the end of each month, the nurses' shifts for the following month must be scheduled. The problem consists in optimally assigning a working shift or a day off (rest) to each nurse on each day, according to several contractual requirements, such as, for instance, a limit on the maximum number of consecutive working days, and operational requirements, such as, for instance, the minimum number of nurses needed on each shift. In addition, it is required to handle these contractual and operational requirements together with the nurses' holidays requests.

The problem belongs to the family of timetabling problems [6,7]: more specifically it is a typical employee timetabling problem [12]. Many papers have been published on the nurse rostering problem since the pioneering works of Warner [13] and Miller [11]. Currently, the proposed approaches are based mainly on

* Corresponding author.
E-mail address: giuliana.carello@polito.it (G. Carello).

0377-2217/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0377-2217(03)00096-1

constraint programming and metaheuristic procedures. Among the metaheuristics recently proposed, we refer here to the tabu search (TS) procedure proposed in [8] and to the genetic algorithm proposed in [1]. Both papers, however, use weekly patterns, which are not foreseen in this case. We refer also to [2] for a TS approach and to [3] for a variable neighborhood search approach. In particular, in [8] the TS procedure uses strategic oscillations and chain neighborhood while generating weekly schedules such that all coverage constraints are satisfied, as many nurses' preferences as possible are respected and the produced schedules are reasonably "fair". Since the number of possible day shifts and night shifts in a given week for a given nurse are known, one can enumerate all the possible feasible patterns for each nurse. A penalty is associated to each nurse on each pattern taking into account all nurses' preferences, the working shifts history and the quality of the pattern. The TS oscillates between feasible (meeting the covering constraints) and infeasible solutions. After a feasible local minimum is found, the search jumps to an infeasible region and tries to minimize the covering violations. However, in the real case considered in this paper, as monthly schedules are requested, it is not possible to enumerate all the feasible monthly patterns for each nurse. Also, the use of fixed weekly patterns would significantly limit the solution space. Further, the above approach requires a complete patterns enumeration where a penalty must be assigned to each pattern and this aspect strongly reduces the user-friendliness of the method. Finally, it is required to handle covering and contractual requirements provided in a parametric way such that they can be modified with respect to different wards environments. For the above reasons, both the approaches in Refs. [1,8] could not be adapted to the considered problem. With respect to the problem considered in [2,3], we notice that the *hard* constraints there correspond to *soft* constraints here (see Section 2) and vice versa. In this paper, a local search approach, based on a neighborhood that operates on partial solutions, is proposed to solve the considered nurse rostering problem: the partial solutions are then completed by means of a greedy procedure in order to avoid the generations of infeasible neighbors.

The article proceeds as follows. In Section 2 the problem is described. In Section 3 the modeling step is presented. In Section 4 the solution approach is described and two local search procedures, namely a TS algorithm and an iterated local search (ILS) procedure are presented. Section 5 is devoted to computational testing. Section 6 concludes the paper with final remarks.

2. Problem description

The considered problem may be expressed as follows. It is required to schedule monthly the nurses' shifts. A working shift or a day off must be assigned to each nurse in each day of the month. In each day there are three working shifts: the morning shift from 7 a.m. till 3 p.m., the afternoon shift from 3 p.m. till 11 p.m. and the night shift from 11 p.m. till 7 a.m. of the day after. According to the ward policy, nurses may require to have a rest on a particular day: this type of rest is referred to as *requested day off*. Further, they may also ask not to be assigned to a specific working shift in a given day of the week, e.g. a nurse may ask not to be assigned to a night shift on Friday. This second type of request is referred to as *desiderata*. The monthly schedule must also satisfy several contractual and operational requirements.

The main contractual requirements are:

(C1) The number of days off per month must be equal to a predefined value provided by the ward management.
(C2) The nurses' requirements related to holidays and requested days off must be satisfied.
(C3) A nurse cannot work consecutively for more than K days.
(C4) The night shifts must be allocated in sets of minimum L and maximum M consecutive days.
(C5) After a set of night shifts, there must be at least N days off.
(C6) An interval of at least P days must occur between two night shifts sets.

In the above requirements, K, L, M, N and P are parameters chosen by the ward management.
The main operational requirements are:

(O1) A minimum number of nurses must be guaranteed for each working shift. This parameter, provided by the ward management, may differ from shift to shift and from day to day.
(O2) A nurse assigned to an afternoon shift in a given day must not be assigned to a morning shift in the following day.
(O3) A balanced assignment of morning, afternoon and night working shifts must be guaranteed among the nurses.
(O4) Working shifts and days off during the week-ends must be evenly assigned.
(O5) The so-called nurses' *desiderata* should be satisfied as much as possible.

The ward management provided also the following optional advices to be possibly respected:

(A1) Try to assign a set of morning shifts before the first day of a holidays period and a set of night shifts after a holidays period.
(A2) Try to assign two days off after K consecutive working days.
(A3) Try to avoid the assignment of a night shift before a requested day off.
(A4) Try to allocate night shifts in sets of $M - 1$ days, avoiding however the end of the set on Saturday by adding, when necessary, one more night shift.
(A5) Try to allocate day shifts in sets of three consecutive days.

3. Modeling the problem

Operational and contractual requirements may be in conflict especially in those months where manpower is reduced because several nurses are on holiday, such as July or August. Because of this conflict, it is not possible to derive feasible solutions if both contractual and operational requirements are considered as problem constraints. After several interactions with the ward management, it was decided to consider contractual requests C2–C6 as constraints of the model, while contractual request C1 and all the operational requirements O1–O5 were handled as objectives. The advices A1–A5 were also handled as (minor) components of the objective function. The ward management was giving a high level of importance to the coverage of the working shifts with particular emphasis on the night shifts. To this extent, it was decided to add the following further operational constraint:

(O6) A deviation of at most one unit is allowed from the requested number of nurses in the night shifts (namely, if in a night shift four nurses are requested, at least three nurses must be assigned to make the solution feasible).

With reference to the day shifts coverage requirements, the so-called multiple covering violations (namely the violations involving both morning and afternoon of the same day or those related to deviations of more than one unit in a given shift) are strongly penalized.

In summary, the following objectives were considered:

(F1) Minimize the deviation of the total number of days off per month from a predefined value provided by the ward management (contractual requirement C1).
(F2) Minimize multiple covering violations of day shifts (operational requirement O1).
(F3) Minimize unitary covering violations of day shifts (operational requirement O1).

(F4) Minimize unitary covering violations of night shifts (operational requirement O1).
(F5) Minimize a linear combination of the operational requirements O2–O5 and the advises A1–A5.

The ward management decided that the relevance of the objectives was directly proportional to the order in which they are listed above according to a lexicographic Lex (or hierarchical) approach. Hence, first the minimization of F1 is considered, then the minimization of F2 and so on, subject to the constraints given by the contractual requirements C2–C6 and the operational requirement O6, namely

$$\begin{aligned} &\text{Lex min} && \{(F1),(F2),(F3),(F4),(F5)\} \\ &\text{subject to} && (C2),(C3),(C4),(C5),(C6),(O6). \end{aligned}$$

It is possible to formulate an integer programming model of this problem. However, this could include several thousands variables and constraints for a ward with 20 nurses. The model is not presented here (we refer to [4] for details) as it is not relevant for the remainder of the paper. Further, the LP relaxation of the model did not provide good bounds, nor it was possible to solve the problem to optimality by applying efficient MIP techniques due to CPU time limits. A heuristic approach has then been used.

4. Solution approach

A neighborhood search approach is proposed to solve the problem. The initial solution is computed by means of a greedy algorithm. Then a local search step is applied. A complete solution for this problem defines for each day of the month and for each nurse the corresponding shift. Due to the various problem constraints, moves operating on generic shifts can easily lead to infeasible solutions and often may change only slightly the objective function value, particularly because of the lexicographic order of the objective function terms. Consider the following example showing the current schedule with respect to a pair of nurses i, j and several consecutive days of a month.

Nurse i:	*m*	*n*	*n*	*n*	*n*	*rs*	*rs*	*m*	*a*	*a*	*a*	*rs*
Nurse j:	*m*	*m*	*rn*	*n*	*n*	*n*	*rs*	*rs*	*m*	*a*	*a*	*a*
Day	1	2	3	4	5	6	7	8	9	10	11	12

Here, *n* represents a night shift, *m* a morning shift, *a* an afternoon shift, *rn* a day off requested by the nurse or a holiday, *rs* a day off generated by the scheduler.

Assume that the contractual requirements correspond to those shown in Table 1.

Consider a move swapping in the same day the shift of nurse i with the shift of nurse j. If the swap takes place in days 1, 4, 5, 7, 10 or 11, the solution is still feasible and the objective value remains unchanged. If it takes place in days 9 or 12, the solution is still feasible, the covering violations remain unchanged and only the last term F5 of the objective function is modified. If it takes place in day 3, it

Table 1
Contractual requirements parameters

Parameter	Symbol	Value
Maximum number of consecutive working days	K	5
Minimum number of consecutive night shifts	L	2
Maximum number of consecutive night shifts	M	4
Minimum number of days off after a set of night shifts	N	2
Minimum number of days between two night shifts sets	P	6

leads to an infeasible solution since it removes a requested day off (constraint C2). Finally, if the swap takes place in days 2, 6 or 8, it leads to an infeasible solution as constraint C5 is violated. Consider now a move related to a shift of a single nurse. Removing a day off or a night shift can easily lead to infeasible solutions where constraints C3, C4 or C5 are violated. According to these observations, we decided to operate on partial solutions instead of complete solutions for the neighborhood generation. With respect to the constraints set, the night shifts are definitely the most critical ones as they impact directly on the operational constraint O6 and the contractual constraints C3–C6 and indirectly on constraint C2. The partial solution is then represented by holidays, requested days off and night shifts. This partial solution is then completed by assigning the day shifts by means of the greedy algorithm applied for finding the initial solution.

4.1. Initial solution

After an initialization step setting holidays and requested days off, a greedy algorithm examines all the days trying to guarantee the requested coverage for each shift. This is done by selecting for each given shift the best candidate to be assigned to that shift. In order to evenly assign shifts, the best candidate for a shift in a given day is chosen in a myopic way as the nurse being currently assigned to the least number of shifts of that type. The greedy algorithm is divided into three main subroutines.

AssignNightShift

```
for (all days of the month) {
   OrderNurses;
   do {
      ChooseBestCandidate;
      if (it does not violate constraints) {
         AllocateNightShift;
         AllocateDayOff;
      }
   } until ((requested coverage is reached) or (all nurses have been checked))
}
```

For each day, the nurses are sorted in increasing number of night shifts already assigned in the last two months (the current one and the previous one). The best candidate is the nurse currently assigned to the least number of night shifts. The subroutine assigns to this nurse $M-1$ consecutive night shifts and N days off, if it is possible to do so without violating constraints, else the second best candidate is considered and so on.

AssignDayShift

```
for (all days of the month) {
   OrderNurses;
   do {
      ChooseBestCandidate;
      if (it does not violate constraints) {
         AllocateDayShift;
      }
   } until ((requested coverage is reached) or (all nurses have been checked))
}
```

This subroutine considers morning (afternoon) shifts as follows. For each day, the nurses are sorted in increasing number of morning (afternoon) shifts already assigned in the last two months (the current one and the previous one). The best candidate is the nurse currently assigned to the least number of morning (afternoon) shifts. The subroutine assigns to this nurse a set of consecutive morning (afternoon) shifts, if it is possible to do so without violating constraints, else the second best candidate is considered and so on.

CompleteSolution

```
AssignAllShifts;
if (Number of days off != fixed value)
   ForceDayOff;
if (Number of days off == fixed value) {
   ImproveCoverage;
   RemoveAM;
}
```

At the end of the above subroutines, the solution may be not complete as some nurses may still be not yet assigned to a shift. The solution is then completed by setting the remaining free shifts (*AssignAllShifts*) in such a way to respect (whenever possible) all the above mentioned requirements.

Given a complete solution, if the number of days off per month is not equal to the predefined value provided by the ward management, procedure *ForceDayOff* forces either days off or working days in order to fulfil this requirement. Then, if the requested number of days off in a month is satisfied for all the nurses, the solution is further improved by means of two procedures, *ImproveCoverage* and *RemoveAM*. Procedure *ImproveCoverage* tries to reduce day shift coverage violations by swapping a night shift with a day shift whenever this does not violate neither the minimum coverage on the night shift nor contractual requirements. Procedure *RemoveAM* removes afternoon shift-morning shift sequences whenever possible. Alternatively, if the requested number of days off per month is not reached, a new solution is built by means of a new iteration of the set of multistart procedures described below.

4.2. Multistart

In order to generate different initial solutions, the greedy algorithm is inserted into three different multistart procedures embedded each other.

The first multistart procedure searches for the requested number of nurses v for the night shifts in as many days as possible. An initial attempt is performed to set v nurses for all days of the month. If this attempt fails, a dichotomic search is applied in order to satisfy this requirement as much as possible.

The second multistart procedure relates to requirement C1, i.e., it tries to guarantee that the number of days off per month of each nurse is equal to a predefined value provided by the ward management. Days off are strictly linked to night shifts by requirement C5. By modifying the night shifts assignment, placement and cardinality of days off are then modified. The procedure operates by partially randomizing the order in which the nurses are sorted for the night shifts in *AssignNightShift*.

The third multistart procedure searches for an improved solution by partially randomizing the order in which the nurses are sorted for the day shifts in *AssignDayShift*.

The best of the solutions generated by the three embedded multistart procedures is taken as the starting solution for the local search step.

4.3. Local search step

Given the initial solution, a neighborhood search step is applied. Its main features described below are the solution representation and the proposed neighborhood. Both a TS procedure and an ILS procedure are proposed.

4.3.1. Solution representation

As mentioned before the neighborhood search operates on partial solutions. Given the current complete solution, morning and afternoon shifts are eliminated in order to derive a partial solution. In a partial solution the fixed data are just the requested days off and night shifts. An example of partial solution (always considering a subset of consecutive days in a month) for a single nurse i is given below:

Nurse i:	*rn*	0	0	*n*	*n*	0	0	0	*rn*	*rn*	0	0
Day	1	2	3	4	5	6	7	8	9	10	11	12

where 0 represents a day in which no shift has been assigned to nurse i yet. Any partial solution is then completed by means of the procedures *AssignDayShift* and *CompleteSolution* used in the greedy algorithm.

4.3.2. Neighborhood

Given the current complete solution, the corresponding current partial solution is derived. Then, a neighbor partial solution is generated by applying one of the following four operations:

(1) A set of night shifts belonging to the current partial solution is moved from a nurse to another by respecting the contractual constraints. Consider the following example with four nurses where the contractual parameters L, M are such that $L = 2$ and $M = 4$. Given the current partial solution below:

Nurse i:	0	0	0	0	*n*	*n*	*n*	0	0	0	0	0
Nurse j:	0	0	0	0	0	0	0	0	0	0	0	0
Nurse k:	0	0	*n*	*n*	0	0	0	0	0	0	0	0
Nurse l:	0	0	0	0	0	0	0	*n*	*n*	0	0	0
Day	1	2	3	4	5	6	7	8	9	10	11	12

(1)

the set of three nights from day 5 to day 7 may be moved from nurse i to nurse j, but not to nurse k or l, because it would violate the contractual constraint C4: a feasible neighbor partial solution generated by moving the considered set from i to j is

Nurse i:	0	0	0	0	**0**	**0**	**0**	0	0	0	0	0
Nurse j:	0	0	0	0	***n***	***n***	***n***	0	0	0	0	0
Nurse k:	0	0	*n*	*n*	0	0	0	0	0	0	0	0
Nurse l:	0	0	0	0	0	0	0	*n*	*n*	0	0	0
Day	1	2	3	4	5	6	7	8	9	10	11	12

(2) The first night of a set is moved from one nurse to another, by respecting the contractual constraints. Given the first partial solution (1), it is possible to move the first night of nurse i to nurse k without violating constraints. The neighbor partial solution is

Nurse *i*:	0	0	0	0	**0**	*n*	*n*	0	0	0	0	0
Nurse *j*:	0	0	0	0	0	0	0	0	0	0	0	0
Nurse *k*:	0	0	*n*	*n*	***n***	0	0	0	0	0	0	0
Nurse *l*:	0	0	0	0	0	0	0	*n*	*n*	0	0	0
Day	1	2	3	4	5	6	7	8	9	10	11	12

(3) The operation symmetric to the previous one, namely the last night of a set is moved from one nurse to another.

(4) A new night shift is assigned to a nurse as first or last night shift of a set, by respecting the contractual constraints. Given the current partial solution (1), it is possible to add a night shift to nurse k as the first night of a set without violating constraints. The corresponding neighbor partial solution is

Nurse *i*:	0	0	0	0	*n*	*n*	*n*	0	0	0	0	0
Nurse *j*:	0	0	0	0	0	0	0	0	0	0	0	0
Nurse *k*:	0	***n***	*n*	*n*	0	0	0	0	0	0	0	0
Nurse *l*:	0	0	0	0	0	0	0	*n*	*n*	0	0	0
Day	1	2	3	4	5	6	7	8	9	10	11	12

Let $\mathscr{I}$ be the number of nurses. The first three moves generate $O(\mathscr{I}^2)$ neighbors, while the last move generates $O(\mathscr{I})$ neighbors.

4.3.3. Tabu search procedure

TS is a widely known metaheuristic that improves upon the typical steepest descent local search approach accepting, when necessary, cost-increasing solutions. This allows to escape from local minima so that other parts of the search space can be explored. We refer to [9,10] for detailed description of the procedure. For the considered problem the main TS parameters are set as follows:

- *Neighborhood*: a neighbor is generated by considering one of the four operations in 4.3.2. The whole neighborhood is obtained by considering all pairs of nurses for operations 1–3 and all nurses for operation 4.
- *Tabu move*: a move is memorized by indicating the nurses altering their night shifts, the days involved and which of the above four operations is applied.
- *Tabu list*: A static tabu list was considered with constant length. Experimental results indicated that the best results were obtained with tabu list length = 6.
- *Aspiration criterion*: A standard aspiration criterion was used: a candidate tabu move is accepted if it improves upon the currently best available solution.
- *Stopping criterion*: The algorithm is stopped after 50 iterations without improvement. Also in this case this value was set experimentally.

4.3.4. Iterated local search procedure

ILS is another widely known advanced search approach that improves upon the classic multistart descent approach. An ILS procedure iteratively applies runs of the pure descent local search procedure by restarting each time from an initial solution obtained by modifying somewhat (the "kick") the previous local optimum. We refer to [5] for a discussion on ILS approaches. For the considered problem, the main ILS parameters are set as follows:

- *Kick*: given the current local minimum, all four operations in 4.3.2 are applied together twice (also in this case this value was set experimentally) on nurses selected randomly. The new initial solution is therefore 8 moves far from the considered local optimum.
- *Stopping criterion*: The algorithm is stopped after 200 iterations (also in this case this value was set experimentally).

5. Computational results

The proposed procedures were implemented in C code and tested on a Pentium IV at 2 GHz with 512 MB RAM both on real life and random generated instances with 20 up to 60 nurses. The procedures were tested with different number of multistart iterations and stopping criteria. The best trade-off between solution quality and computational effort was obtained with 20 multistart iterations and 50 neighborhoods without improvement for the TS procedure and with 5 multistart iterations and 200 descents for the ILS procedure.

Several real life instances provided by the ward management were tackled. The results of four months only, July to October, are reported here as the other months of the year behave quite similarly. Parametric values of the contractual requirements defined by the ward management are shown in Table 1. Shifts coverage requirements are shown in Table 2. The months from July to September involve 20 nurses while October involves 21 nurses.

For the real life instances, the solutions generated by the proposed procedures were compared with those manually generated by the ward management. In order to evaluate the TS and the ILS procedures, their solutions were also compared to the solutions obtained by a simple greedy multistart and by a steepest descent local search. To further assess the algorithm effectiveness, a straightforward lower bound was computed on the covering requirements violations as follows. The covering violations are not split into multiple day shift, single day shift and night shift violations but are considered as a whole. A simple bound on this value is given by the difference between the total monthly requirement on the three shifts and the available working shifts taking into account holidays and total amount of days off as requested by the ward management. The results are shown in Table 3. The first entry refers to the considered month. The second entry is related to the solution procedure applied. MS stands for manual solution. GMS indicates the solution obtained by the greedy algorithm with 20 multistart iterations. LSS indicates the solution obtained by a steepest descent local search with 20 multistart iterations. TSS represents the final solution obtained by the TS procedure with 20 multistart iterations where for each iteration the stopping criterion is given by 50 neighborhoods without improvement. Finally, ILSS represents the final solution obtained by the ILS procedure with 5 multistart iterations where for each iterations 200 descents are performed. The third entry indicates the multiple day shift covering violations (MDCV), i.e. the number of coverage violation occurring in a day in which another violation takes place, and the fourth entry indicates the single day shift covering violations (SDCV). Suppose that in a day there are three people on the afternoon shift instead of

Table 2
Coverage requirements

Shift	Requested number of nurses
Morning shift from Monday to Friday	5
Morning shift in the week-end	4
Afternoon shift from Monday to Friday	4
Afternoon shift in the week-end	4
Night shift from Monday to Friday	4
Night shift in the week-end	4

Table 3
Testing the algorithm on real instances

Month	Solution of algorithm	MDCV	SDCV	NCV	ORV	CV bound	[s]
July	MS	0	11	31	840	40	
	GMS	0	9	31	122		13
	LSS	0	9	31	92		32
	TSS	0	9	31	70		507
	ILSS	0	9	31	94		296
August	MS	3	17	31	620	50	
	GMS	1	18	31	156		31
	LSS	0	19	31	121		33
	TSS	0	19	31	101		408
	ILSS	0	19	31	116		350
September	MS	0	14	30	345	42	
	GMS	0	12	30	161		15
	LSS	0	12	30	123		32
	TSS	0	12	30	106		503
	ILSS	0	12	30	122		359
October	MS	0	0	2	789	0	
	GMS	0	0	4	165		20
	LSS	0	0	0	100		41
	TSS	0	0	0	74		676
	ILSS	0	0	0	80		359

four and four people on morning shift instead of five: in this case both a single violation and a multiple violation are reported. The fifth entry refers the night shift covering violations (NCV). The sixth entry is related to the other requirements violations (ORV) such as, for instance, afternoon shift/morning shift sequences, that constitute the fifth objective introduced in Section 3. The seventh entry is related to the lower bound on the covering violations (CV bound), where the following inequality always holds:

$$\text{CV bound} \leqslant \text{MDCV} + \text{SDCV} + \text{NCV}.$$

The last entry represents the CPU time required. Notice that the requirement on the total number of days off was always satisfied and therefore it is not reported here.

From Table 3 we conclude that the greedy solutions improves upon the one manually computed by the ward management in all months except from October. These results are further strongly improved by the local search approaches where the TS procedure reaches the best performances. For these instances we notice that a pure descent algorithm iterated several times in a multistart context already reaches pretty good results (for instance, in October LSS with 20 multistart iterations reaches a better solution quality than ILSS with 5 multistart iterations and in much less CPU time). All CPU times are rather small (less than 12 minutes in the worst case). Notice also that all local search approaches compare favorably with the bound on the coverage requirement.

The proposed procedures were tested also on randomly generated instances with 20, 40 and 60 nurses. For each ward size, 4 instances were generated where, in order to produce different scenarios, the nurses on holidays were set in a different way from month to month. [1] The considered contractual requirements

[1] Instances are available upon request from authors.

parameters were the same as in the real life instances (see Table 1), while coverage requirements were scaled up with the ward size in order to guarantee a constant manpower-coverage requirements ratio. For these instances the same entries of Table 3 are available except for the entry related to the manual solution that was not available.

Table 4 shows the results related to 20 nurses instances, whilst Tables 5 and 6 are related to 40 and 60 nurses instances respectively.

Table 4
Testing the algorithm on randomly generated instances with 20 nurses

Instance	Solution of algorithm	MDCV	SDCV	NCV	ORV	CV bound	[s]
20-1	GMS	0	6	31	240	30	18
	LSS	0	4	27	163		33
	TSS	0	3	27	144		216
	ILSS	0	2	28	148		349
20-2	GMS	6	19	31	238	56	8
	LSS	3	22	31	191		14
	TSS	2	23	31	179		222
	ILSS	2	23	31	220		309
20-3	GMS	0	0	0	293	0	2
	LSS	0	0	0	231		11
	TSS	0	0	0	192		207
	ILSS	0	0	0	185		352
20-4	GMS	0	1	3	213	0	11
	LSS	0	0	3	123		14
	TSS	0	0	2	81		172
	ILSS	0	0	2	80		331

Table 5
Testing the algorithm on randomly generated instances with 40 nurses

Instance	Solution of algorithm	MDCV	SDCV	NCV	ORV	CV bound	[s]
40-1	GMS	0	4	31	401	18	68
	LSS	0	0	18	260		403
	TSS	0	0	18	181		3059
	ILSS	0	0	18	199		4174
40-2	GMS	12	24	31	389	67	16
	LSS	8	28	31	303		205
	TSS	7	29	31	288		2653
	ILSS	5	31	31	337		4355
40-3	GMS	0	0	1	485	0	15
	LSS	0	0	1	485		80
	TSS	0	0	0	317		2412
	ILSS	0	0	0	265		3668
40-4	GMS	2	5	5	358	0	16
	LSS	0	1	2	210		289
	TSS	0	0	2	171		2414
	ILSS	0	0	3	130		3050

Table 6
Testing the algorithm on randomly generated instances with 60 nurses

Instance	Solution of algorithm	MDCV	SDCV	NCV	ORV	CV bound	[s]
60-1	GMS	0	2	11	565	2	147
	LSS	0	0	9	426		1678
	TSS	0	0	4	284		17,896
	ILSS	0	0	2	251		14,681
60-2	GMS	15	24	31	443	70	30
	LSS	10	29	31	467		794
	TSS	9	30	31	417		12,564
	ILSS	8	31	31	369		24,456
60-3	GMS	5	6	6	738	0	28
	LSS	0	0	1	556		1206
	TSS	0	0	0	507		10,991
	ILSS	0	0	0	410		16,277
60-4	GMS	2	4	3	498	0	28
	LSS	0	0	3	357		1368
	TSS	0	0	3	310		12,305
	ILSS	0	0	4	163		16,713

From these tables we notice that, as the ward size increases, the TS and ILS procedures tend to outperform more and more the simple multistart local search approach. For the largest instances the CPU time becomes more important but still reasonably limited (less than eight hours in the worst case). Between TS and ILS there is not a clear winner, though ILS performs on the average slightly better than TS. With respect to the bound on the coverage requirement, the results are also satisfactory.

6. Conclusions

A neighborhood search approach was proposed for a real life nurse rostering problem where the local search step works on partial solutions completed then by means of a greedy procedure. Both a TS procedure and an ILS procedure were derived based on the proposed approach. Both the TS and the ILS procedures showed a very good behavior both in terms of solution quality and CPU time requirement. The obtained solutions strongly improve upon the corresponding solutions manually computed by the ward management on real instances and compare favorably with respect to a straightforward lower bound. Randomly generated instances confirm the good behavior of these procedures. A software has been developed which is currently used in the hospital ward. This software handles the nurses planning of a whole year by iteratively applying the monthly nurses staff scheduling procedure.

References

[1] U. Aickelin, K.A. Dowsland, Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem, Journal of Scheduling 3 (2000) 139–153.
[2] E.K. Burke, P. De Causmaecker, G. Vanden Berghe, A hybrid tabu search algorithm for the nurse rostering problem, in: Simulated Evolution and Learning—Second Asia Pacific Conference on Simulated Evolution, 1999, pp. 187–194.
[3] E.K. Burke, P. De Causmaecker, S. Petrovic, G. Vanden Berghe, Variable neighbourhood search for nurse rostering problems, in: 4th Metaheuristics International Conference Proceedings, Porto, July 2001, pp. 755–760.
[4] G. Carello, An integer programming model for the nurse rostering problem, Internal Report, D.A.I., Politecnico di Torino, 2001.

[5] R.K. Congram, C.N. Potts, S.L. van de Velde, An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem, Informs Journal on Computing 14 (2002) 52–67.
[6] D. De Werra, An introduction to timetabling, European Journal of Operational Research 19 (1985) 151–162.
[7] D. De Werra, The combinatorics of timetabling, European Journal of Operational Research 96 (1997) 504–513.
[8] K.A. Dowsland, Nurse scheduling with tabu search and strategic oscillation, European Journal of Operational Research 106 (1998) 393–407.
[9] F. Glover, Tabu search. Part I, ORSA Journal on Computing 1 (1989) 190–206.
[10] F. Glover, Tabu search. Part II, ORSA Journal on Computing 2 (1990) 4–32.
[11] H.E. Miller, Nurse scheduling using mathematical programming, Operations Research 24 (1976) 857–870.
[12] R. Nanda, J. Browner, Introduction to Employee Scheduling, Van Nostrand Reinhold, New York, 1992.
[13] D.M. Warner, Scheduling nursing personnel according to nursing preference: A mathematical programming approach, Operations Research 24 (1976) 842–856.