Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 188 (2008) 330–341
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Discrete Optimization

A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem

Edmund K. Burke [a,*], Timothy Curtois [a], Gerhard Post [b], Rong Qu [a], Bart Veltman [b]

[a] School of Computer Science and Information Technology, University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, UK
[b] ORTEC, Groningenweg 6-33, 2803 PV Gouda, Holland, The Netherlands

Received 24 January 2006; accepted 24 April 2007
Available online 29 April 2007

Abstract

This paper is concerned with the development of intelligent decision support methodologies for nurse rostering problems in large modern hospital environments. We present an approach which hybridises heuristic ordering with variable neighbourhood search. We show that the search can be extended and the solution quality can be significantly improved by the careful combination and repeated use of heuristic ordering, variable neighbourhood search and back-tracking. The amount of computational time that is allowed plays a significant role and we analyse and discuss this. The algorithms are evaluated against a commercial Genetic Algorithm on commercial data. We demonstrate that this methodology can significantly outperform the commercial algorithm. This paper is one of the few in the scientific nurse rostering literature which deal with commercial data and which compare against a commercially implemented algorithm.

Keywords: Variable neighbourhood search; Heuristics and metaheuristics; Nurse rostering; Hybrid methods

1. Introduction

It is clear that the efficient rostering of healthcare personnel can lead to the more effective utilisation of valuable resources. Healthcare institutions around the world are becoming increasingly interested in the deployment of decision support technology to aid the personnel scheduling process. A very general form of the nurse rostering problem could be described as follows: Given a set of shifts and a set of nurses over a certain time period, assign each shift to a nurse subject to a set of constraints. The constraints are usually defined by regulations, working practices and the preferences of the nurses.

The problem of nurse rostering is relatively easily described but like most real world search problems it is far from easy to automatically generate very high quality solutions. Indeed, there have been many papers over the years from across operational research and artificial intelligence that have tackled the problem in one form or another. A wide range of approaches and techniques have been investigated and used. Ernst et al. [19,20] identified 28 different

* Corresponding author.
E-mail address: ekb@cs.nott.ac.uk (E.K. Burke).

0377-2217/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2007.04.030

categories of methods that have been used on personnel scheduling problems. These include constraint logic programming, constructive heuristics, expert systems, genetic algorithms, integer programming, set partitioning, simple local search and simulated annealing. A recent review of automated nurse rostering approaches found that, although there has been a lot of research in the area, surprisingly few of the methods were tested on real world data [17]. The paper went on to conclude that even fewer have actually been implemented in real hospital wards.

Of those techniques that have been applied on real-world problems metaheuristic methods seem to dominate. One approach which has been applied in multiple real world hospitals is a hybrid tabu search [15]. The tabu search is integrated with techniques which are usually observed in manual scheduling approaches. The algorithm has been incorporated into software that has been used to create nurse rosters in over 40 Belgian hospitals and copes with many shift types, work regulations and skill categories. This work was hybridised with an evolutionary approach [12] to produce a methodology which could generate higher quality solutions but at the cost of increased computational time. Variable Neighbourhood Search [21,28] has also been applied and tested on highly constrained real world nurse rostering data [14]. The authors found that VNS could be effectively used to escape from the local optima found using single neighbourhood heuristics. They also commented ''After reaching a local optimum, we recommend the exploration of wider environments''. Evaluation methods for challenging real world problems are presented and discussed in [13]. A methodology which can handle a more flexible approach to real world nurse rostering than the traditional fixed period based approach is presented in [11]. An overview of the work carried out by these authors in Belgian hospitals is presented in [16].

Another investigation on real data explored a genetic algorithm approach [1], which successfully exploits problem specific knowledge in tackling the problem. Although the method is tailored for that particular problem instance, the underlying concepts could be applied to other nurse rostering problems. In 1998, Dowsland was also able to match the quality of schedules produced by an expert human scheduler using a highly developed tabu search [18]. The algorithm 'oscillates' between trying to improve the cover and improving the preference costs. As well as using tabu lists, candidate lists and diversification strategies, the search also uses a large neighbourhood created by looking for chains of overall improving swaps. Aickelin and Li [2,3] have since experimented with the application of bayesian optimisation and classifier systems to similar nurse rostering problems. The results are close to those produced by an optimal integer programming method and the authors concluded that with further effort and experimentation the algorithms could well improve even more. Bellanti et al. [9] tackled a problem with hard constraints and objectives (or soft constraints) using various local search techniques. The authors presented good results for a tabu search and iterated local search which use neighbourhoods defined by changing the assignment of night shifts.

Another methodology that has been tested on complex real-world data from a UK hospital is case-based reasoning [30]. This approach avoids the use of evaluation functions but instead aims to imitate how an expert human scheduler would produce a good schedule. This is done by storing observed methods for repairing violations in schedules and retrieving, adapting and performing these repairs or moves whenever a similar violation is encountered again. As an extension to their work, the authors also suggest methods in which it could be combined with a meta-heuristic approach [7,8]. Another relatively recent methodology is a combination of constraint networks and knowledge-based rules [24]. The approach was implemented in a commercial software package and has been successfully used in a number of hospitals.

Berrada et al. [10] developed a multi-objective mathematical programming model to represent a real world problem containing both hard and soft constraints in a Canadian hospital. The schedules produced met the standards required by the head nurses. The authors also experimented with a tabu search and found that although it required greater computational time it was useful in some circumstances. Valouxis and Housos [32] approach a nurse rostering problem using an approximate integer linear programming model to produce initial solutions. The initial solutions are then further optimised using a local search with a '2-opt' neighbourhood and a tabu search. Their method compared very favourably with a constrained programming approach. In 2004, Bard and Purnomo [5] employed a combination of heuristic and integer programming methods to solve nurse preference scheduling problems with up to one hundred nurses and approximately 13 hard and soft constraints. Individ-

ual nurse schedules are created by modifying a base schedule using swaps. These columns are then used to form a set covering type problem which, when solved, creates the overall roster. Later they extended this work to further improve the quality of schedules by incorporating the use of a downgrading option [4] and in [6] present a model which combines cyclic and preference scheduling.

By defining fuzzy constraints (i.e. constraints that may be partially satisfied and partially violated) Meyer auf'm Hofe [27] solved real world nurse rostering problems as constraint optimisation rather than as constraint satisfaction problems. Branch and bound and iterative improvement are used to quickly produce good rosters. The approach was developed using experience gained developing a software rostering system that is used in approximately 60 German hospitals [26].

There are many more papers in the literature which are discussed in more detail in [17,19]. It is clear that relatively few papers in the literature have worked with real world data or been implemented in hospitals [17]. One of the main goals in this paper is to develop an effective and efficient search approach to improve upon the genetic algorithm based approach that is currently employed within ORTEC's Harmony software.[1] As such, the methodology has to be able to handle all the requirements and constraints that are inherent in nurse rostering problems from the modern complex environments that are represented by today's hospitals.

This paper presents our investigation into combining a variable neighbourhood search with a method of heuristically unassigning shifts and repairing schedules using heuristic ordering. The next section describes the nurse rostering problem we were dealing with. Sections 3 and 4 discuss the algorithm and results respectively. In Section 5 we draw conclusions on the success of this approach and present some possible future extensions in Section 6.

2. Problem description

The data for this problem was provided by ORTEC, a major supplier of software products and consulting in the field of advanced planning and scheduling. They support hospitals and other organisations all over the world with automated workforce management solutions.

[1] The results of this research are incorporated in the latest product versions of Harmony.

The number of nurses in the problem instances tested ranges from 30 to 12, the ratio of full to part time nurses also varies between wards. For example, one ward consists of 16 nurses, 12 of the nurses are full time and work 36 hours per week. One nurse works 32 hours per week and the other 3 are also part time and work 20 hours per week. Each instance also has a number of specific personal requests such as particular shifts and/or days requested off or on. All the other constraints that need to be satisfied are presented in Sections 2.2 and 2.3. The scheduling period for each instance is exactly one month.

The data was provided by ORTEC as a challenging real world problem and is very typical of their clients' needs. An approach which is successful in dealing with a problem as complex as this will provide direct benefits in a number of real world personnel scheduling scenarios.

2.1. Shifts and shift demand

There are four different shift types in the problem: day, early, late and night shifts. All the shifts except night shifts cover 9 hours including 1 hour of rest time. So the actual number of working hours for each shift type is 8. Night shifts last 8 hours but include no rest time and so are counted as 8 working hours. The total cover requirements for each shift for each day vary between instances. Generally, larger wards require more nurses on duty during each shift but similar sized wards can also have different cover requirements.

Table 1 shows the daily demand for these shifts in the instance described earlier with 16 nurses.

2.2. Hard constraints

The following rules must be met at all times otherwise the schedule is considered to be infeasible and unacceptable.

- Shift cover requirements need to be satisfied. Over coverage is not permitted.
- A nurse may start only one shift per day.
- The maximum overtime assigned to each nurse per month is 4 hours.
- The maximum hours worked per week is on average 36 hours over a period of 13 consecutive weeks which do not include night shift assignments.
- The maximum number of night shifts in any period of 5 consecutive weeks is 3.

Table 1
Shift types and an example weekly demand

Shift	Start time	End time	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Day (D)	08:00	17:00	3	3	3	3	3	2	2
Early (E)	07:00	16:00	3	3	3	3	3	2	2
Late (L)	14:00	23:00	3	3	3	3	3	2	2
Night (N)	23:00	07:00	1	1	1	1	1	1	1

- A nurse must receive at least 2 weekends off in any 5 week period. A weekend off lasts 60 hours including Saturday 00:00 to Monday 04:00.
- Following 2 or more consecutive night shifts, a 42 hour rest is required.
- During any period of 24 hours, at least 11 hours rest is required. A night shift has to be followed by at least 14 hours rest. Once in a period of 21 days, however, the rest period may be reduced to 8 hours.
- The maximum number of consecutive night shifts is 3.
- The maximum number of consecutive days worked is 6.

2.3. Soft constraints

Ideally these requirements should be fulfilled. However, to obtain a schedule that meets all the hard constraints it is often necessary to break some of the soft rules. A weight is assigned to each soft constraint to reflect its importance (especially in comparison to other soft constraints). A weighting is simply a number. The higher the number, the more strongly desired the constraint or request is. The weights are set either by the head nurses or through feedback from the nurses about what qualities they desire in their schedules. As a rough guide, the weights could be described as follows:

Weight 1000: The constraint should not be violated unless absolutely necessary.
Weight 100: The constraint is strongly desired.
Weight 10: The constraint is preferred but not critical.
Weight 1: Try and obey this constraint if possible but it is not essential.

In practice, exponentially scaled weights like these are most commonly used. However, the users do have the option of setting and changing the weight for each constraint to any positive integer value.

2.4. Evaluation function

The evaluation function is the sum of all the penalties incurred due to soft constraint violations. The penalty for each soft constraint is calculated either linearly or quadratically using the violation measurement factors listed in Table 2. The violation measurement factor is the degree to which the constraint is violated or the excess of the violation. The use of either quadratic or linear evaluation functions arises from practices in Harmony which were developed based on customer preferences and feedback.

A soft constraint with a linear penalty function is simply calculated as: The violation measurement factor multiplied by the weight. For example, it is preferable to have at most zero stand-alone or isolated shifts. This is a soft constraint with weight 1000. However, to produce a feasible schedule (i.e. one in which all the hard constraints are fulfilled) it may be necessary to allocate a nurse to an isolated shift. This is one more than is preferred so a penalty of 1000 is incurred. If the nurse had two isolated shifts, they would have a penalty of 2000 ($2 * 1000$).

A quadratic penalty function is calculated as: The violation measurement factor squared and multiplied by the weight. For example, it is preferable that, during a period of 5 weeks, a nurse performs no more than three night shifts. This is a soft constraint with a weight of 1000. However, it may be necessary to assign five night shifts in the 5 week period (i.e. 2 more than preferred), then the penalty for this soft constraint violation would be 4000 ($2^2 * 1000$).

It is now possible to define the objective of the problem: To find a feasible schedule with the lowest possible penalty caused by soft constraint violations. From the perspective of the head nurse, of course, the actual penalty hides a lot of information about the solution but it is not totally meaningless. By examining the penalty for each schedule it is possible to gain some idea of the schedule quality. For example, if the penalty is less than 1000 then we

Table 2
Soft constraints

Constraint	Weight	Penalty function	Violation measurement factor
From Friday 22:00 to Monday 0:00 a nurse should have either no shifts or at least 2 shifts ('complete weekend')	1000	Linear	Number of incomplete weekends
No stand-alone shifts, i.e. a day off, day on, day off sequence	1000	Linear	Number of isolated shifts
The length of a series of *night* shifts should be within the range 2–3. It could be before another series	1000	Quadratic	Difference between length of series and acceptable length. e.g. if 1 night shift, factor = 1, if 2 or 3 night shifts, factor = 0, if 4 night shifts, factor = 1, if 5 factor = 2 etc.
A minimum of 2 days rest after a series of *day*, *early* or *late* shifts	100	Linear	Factor is one if only one day of rest otherwise zero
Employees with availability of 30–48 hours per week, should receive a minimum of 4 shifts and a maximum of 5 shifts per week	10	Quadratic	Difference between number of shifts received and acceptable number per week
Employees with availability of 0–30 hours per week, should receive a minimum of 2 shifts and a maximum of 3 shifts per week	10	Quadratic	Difference between number of shifts received and acceptable number per week
For employees with availability of 30–48 hours per week, the length of a series of shifts should be within the range of 4–6	10	Quadratic	Difference between length of series received and acceptable series length
For employees with availability of 0–30 hours per week, the length of a series of shifts should be within the range 2–3	10	Quadratic	Difference between length of series received and acceptable series length
The length of a series of *early* shifts should be within the range 2–3. It could be within another series	10	Quadratic	Difference between length of series received and acceptable series length
The length of a series of *late* shifts should be within the range of 2–3. It could be within another series	10	Quadratic	Difference between length of series received and acceptable series length
An *early* shift after a *day* shift should be avoided	5	Linear	Number of early shifts after days shifts
A *night* shift after an *early* shift should be avoided	1	Linear	Number of night shifts after early shifts

know that all the constraints with weight 1000 have been satisfied. However, the key to producing satisfactory schedules is obviously setting the correct weights and ensuring that all the required constraints are defined. Therefore it is essential that the end user either has a good understanding of how to set the weights and define constraints or has clearly described the requirements to the software administrator.

As mentioned previously, a feasible schedule is a schedule that satisfies all the hard constraints. A penalty for an infeasible schedule can still be calculated but in our system a feasible schedule is always considered to be better than an infeasible schedule regardless of penalty values. The only infeasible schedules that may be introduced during the search or returned afterwards are those that provide insufficient cover. This is ensured by never assigning a shift to a nurse if it will violate a hard constraint. For example, at certain points in the algorithm, shifts may be unassigned in a schedule and so the coverage constraint will be violated. These shifts will then only be reassigned if no hard constraint violations occur in doing so. If the quality of infeasible schedules need to be compared, the schedules with the lowest number of unassigned shifts (i.e. minimum shift coverage violation) are ranked higher regardless of their penalties. If infeasible schedules have the same number of shifts unassigned, then the penalty function is used.

For all the instances we tested we were able to produce feasible schedules. It is possible though that there may be an instance for which a feasible schedule does not exist. In practice, if a feasible schedule cannot be found (either because one does not exist or it is too difficult to find) then the head nurse or manager decides whether to work with the best infeasible schedule or relax some of the constraints or hire extra personnel and/or to assign some extra nurses to the ward (usually agency or float nurses) and then restart the search.

3. The hybrid variable neighbourhood search algorithm

The algorithm that we present in this paper represents an iterative process in which variable neighbourhood search is followed by a schedule disruption and repair strategy. The repairing of the schedule is performed using a heuristic ordering technique. Back-tracking is also carried out to further improve the quality of the schedules produced.

The overall process is illustrated by the pseudo-code in Fig. 1.

3.1. Initialisation

A heuristic ordering is used to create the initial schedule. In the experimentation section, we will be comparing our approach against a commercial genetic algorithm developed by ORTEC and in use in real hospital environments. The commercial genetic algorithm this hybrid variable neighbourhood search is evaluated against uses a similar heuristic ordering method to create its initial population of schedules.

The aim of the heuristic ordering process is to sort all the shifts in order of the estimated difficulty of assigning them or how likely they are to cause high penalties (by using the criteria shown in Table 3). Using the weighted sum to identify them, the more troublesome shifts are then assigned earlier on in the schedule construction process.

Once the shifts have been sorted in the order in which to try and assign them, they are in turn assigned to each nurse to calculate the penalty that would be incurred if the shift was assigned to that nurse. The shift is then assigned to the nurse that gains the least penalty in receiving that shift.

The attributes of a shift that are examined when ranking the shifts in the order of *possible difficulty to assign* are described in Table 3 along with the functions used to assign its total weight for ranking.

The first two criteria in Table 3 are obvious to examine as there are high penalties associated with night shift and weekend shift constraints. The third criterion used is to deduce how many nurses are able to fulfil this shift. If there are many nurses able to undertake it then it can be scheduled later but if there are very few then it is a good idea to assign it early on in the process. The shift date criteria is used to try and ensure that shifts in the early days in the scheduling period are assigned earlier on in the process. This is useful as these shifts are more likely to conflict with the previous schedule's assignments. The shift date evaluation function is in units of days.

3.2. Variable neighbourhood search

When the initial schedule has been created using the heuristic ordering method described above, a variable neighbourhood search is applied. This makes use of two neighbourhoods. Both of these neighbourhoods are commonly used by meta-heuristics and other approaches and have been described before, see, for example, [22,23,25,29]. The two neighbourhoods are defined by the following moves or changes to a schedule:

```
Create Initial Schedule

REPEAT

    Variable Neighbourhood Search

    IF current penalty < best penalty  THEN

        SET best schedule to current schedule
        SET best penalty  to current penalty

    ELSE

        SET Current Schedule to Best Schedule (i.e. Back-track one step)

    ENDIF

    Unassign shifts of a set of nurses

    Repair schedule (using heuristic ordering method)

UNTIL search terminated
```

Fig. 1. Pseudo-code of the overall hybrid algorithm.

Table 3
Shift evaluation criteria

Shift criteria	Evaluation function	Weight
Night shift	Weight	100
Weekend shift	Weight	50
Number of valid nurses	(NumValidNurses/TotalNumNurses) * Weight	70
Shift date	Weight * (Schedule.EndDate − Shift.BeginDate)	20

1. Assigning a shift to a different nurse.
2. Swapping the nurses assigned to each of a pair of shifts.

The first neighbourhood is a lot smaller than the second neighbourhood. However, it is observed that moves in the second neighbourhood can improve the quality of the schedule quite significantly.

Our variable neighbourhood approach is a variable neighbourhood descent. As can be seen from Fig. 2, the smaller neighbourhood (neighbourhood 1) is repeatedly examined for an improving move and the move is executed if found. When there are no improving moves left in neighbourhood 1, then the much larger neighbourhood 2 is examined. If a move in neighbourhood 2 is used then neighbourhood 1 is examined again. This is repeated until there are no improving moves left in neighbourhoods 1 and 2.

Initially, the Variable Neighbourhood Search was implemented in a steepest descent manner. That is, for each of the moves in the neighbourhood, we identified the move or swap that would bring the most improvement and then performed that move or swap. The disadvantage in steepest descent is the extra time required to examine every move and swap, especially in a highly constrained problem like this in which there are many constraints to check and penalties to calculate at each move. This was especially noticeable in the second neighbourhood, which is quite large.

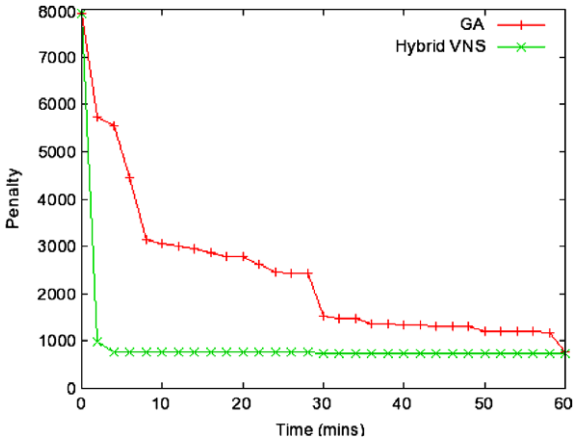

Fig. 2. Pseudo-code of VNS.

In an attempt to decrease the running time of the algorithm, a quickest descent form of VNS was tested. That is, until no more improving moves are found, examine each move and swap and execute the move or swap if it decreases the schedule's overall penalty at all.

It was interesting to discover that, for this problem, using these neighbourhoods, the quickest descent method was not only faster than steepest descent but it was usually at least as good and sometimes better in comparison. This was an interesting observation that was initially difficult to understand. On closer investigation, though, a possible explanation became apparent. The heuristic ordering is very effective at satisfying the constraints with the highest penalties. This means that the soft constraint violations that the VNS needs to repair are often ones with smaller similar sized penalties. If there is a high probability that all the possible improving moves will yield a similar sized improvement, it is not efficient to examine all of them to find the absolute best if it will be only slightly larger than the average of all available improving moves.

We will briefly explain why the available neighbourhoods are restricted to these two neighbourhoods. For example, in [14] a VNS for a nurse rostering problem is introduced which uses a larger set of neighbourhoods. If these neighbourhoods are examined more closely, however, it can be observed that many of them are already included in our larger two. Merging many of these neighbourhoods and searching them exhaustively is now possible due to the recent increases in hardware technology and computing power that we have witnessed over the past few years. Note that the VNS experiments in [14] were carried out on an IBM RS6000 PowerPC. Also, some of the other neighbourhoods are used to add moves which diversify the search and are used

regardless of the effect on the schedule's penalty. So they are not appropriate for use in a VNS descent.

3.3. Schedule feasibility

After the creation of the initial schedule described earlier, or the larger movements in the search space which are described later, the schedule may be infeasible in that the shift cover may not yet have been fulfilled. Therefore, during the VNS, if there are still unassigned shifts, then after a successful move or swap an attempt is made to see if it is now possible to assign any of the unassigned shifts without creating hard constraint violations.

3.4. Schedule disruption and repair

Generally, at the end of the VNS phase the schedule not only has a lower penalty than before but the schedule is also usually now feasible by satisfying the cover requirements if it was not before.

The heuristic ordering and VNS is capable of producing high quality schedules in a number of minutes. However, for most instances it is more likely that a good local optimum rather than the global optimum has been found. Some users may wish to continue the search for a longer time period to try and produce an even higher quality schedule e.g. running the search during a lunch break or over night. Also, as computers get faster and more powerful it is practical to have an approach which can scale with these increases. A 1 hour search today may only last 1 minute in 5 years or so.

To extend the search, a heuristic restart mechanism was developed. The idea is to select sections of the overall schedule which could possibly be improved and to then attempt to improve them.

This is done by selecting a fixed number of nurses who have the worst individual schedules (the penalty is calculated just for their individual schedule) and then unassigning all shifts assigned to this set of nurses. Using the heuristic ordering method, these shifts are then reassigned (over all available nurses) and then the VNS is performed to try and produce a better schedule. This schedule disruption and repair cycle is used repeatedly until the user terminates the search.

The algorithm was initially implemented to unassign shifts from the current schedule after the VNS. However, on some occasions, it was observed that the current schedule could be significantly worse than the best found so far and it could take a number of iterations to get the current schedule penalty back close to the best found. To reduce this effect it was found to be more efficient to return to the best found (if the current schedule is worse than the best found) before the disruption phase.

As stated, the shifts selected for unassigning are those belonging to a fixed number of nurses with the worst individual schedules i.e. those with the highest individual penalties. To prevent cycling though, one of these nurses is selected randomly and replaced with another randomly selected nurse not belonging to this set.

To identify the best number of nurses from which to unassign shifts, a number of experiments were conducted on each instance in which this number ranged between 1 and 14. The results are provided in Section 4.

3.5. Genetic algorithm

Harmony uses a genetic algorithm to produce schedules. This existing algorithm provides a benchmark upon which to compare the performance of the algorithm described here.

The genetic algorithm of Harmony is designed to be robust and effective for a wide variety of rostering problems. To achieve this, like our algorithm, it does not heavily rely on problem specific knowledge or use detailed knowledge of the problems' structures. An algorithm designed for a specific problem which heavily exploits its particular structure is likely to be more effective but less useful when other problems are considered. The genetic algorithm has, however, already performed in a more than satisfactory manner for a number of ORTEC's clients with varying requirements.

The algorithm has a number of phases. Firstly, the initial population of schedules is created using a similar heuristic ordering method to the one described in this paper but ensuring that each individual (or schedule) is different enough to introduce sufficient diversity in the population. Successive generations are created using roulette wheel parent selection, two types of crossover and three types of mutation. The particular crossover and/or mutations used are determined statistically by measuring their success in previous use between generations. The genetic algorithm terminates when a minimum threshold of improvement between generations is reached. After the genetic algorithm phase, a local search is performed to further improve the best schedule found.

4. Results

To develop this algorithm, the workforce management and planning software ORTEC Harmony [31] was used. Employing Harmony provided a number of advantages from a research point of view. The software has a highly developed user interface with which a large number and wide variety of nurse rostering problems can be defined and created. All data structures and methods for manipulating the problem instances themselves already exist with many hours of work already performed to increase their access and use. This meant we were able to concentrate on creating, testing and improving an efficient algorithm for a wide variety of nurse rostering problems. The software also provides a clear visual display of the schedules and with precise breakdowns of why each employee receives the penalty they have. It was also particularly useful to have an existing commercial strength algorithm with which to compare against our work.

The experiments were performed using a PC with a P4 2.4 GHz processor.

4.1. Effects of varying the number of nurses to unassign shifts from

Table 4 presents the results of varying the number of nurses from which to unassign shifts in the disruption and repair phase. The 'penalty after first VNS' column is the penalty of the schedule after the VNS is first applied to the initial schedule. The columns '1–14' show the penalty of the best schedule found after the search has been applied for 1 hour when that number of nurses were selected for shift unassignment during the disruption.

The results show the best number of nurses to use is between three and five. Using these settings, the final schedule is, on average, 14% lower than the schedule found after the first VNS. Using two nurses can also generate some improvement but using one nurse alone is generally ineffective and does not provide sufficient diversification in the search. Using six and seven nurses can also provide some good results but, above seven, the performance deteriorates with eleven to fourteen providing little improvement and suggesting too much diversification.

There does not seem to be any correlation between the size of the instance in terms of the number of nurses and the optimal number of nurses to

Table 4
Hybrid VNS results

				Number of nurses selected for unassignment													
Instance	Nurses	Genetic algorithm	Penalty after first VNS	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	30	3626	3766	3766	3721	3746	3751	3766	3751	3766	3766	3766	3766	3766	3766	3766	3766
2	30	2381	2390	2375	2330	2285	2285	2295	2390	2390	2390	2390	2390	2390	2390	2390	2390
3	28	4325	4687	4557	4476	4687	4687	4687	4687	4301	4687	4687	4687	4687	4687	4687	4687
4	26	1301	1366	1311	1246	1231	1216	1151	1081	1186	1035	1191	1176	1366	1366	1366	1366
5	24	5230	6575	6450	5340	5291	6545	5465	6575	6575	6350	6575	6575	6575	6297	6575	6575
6	24	25,406	32,171	31,986	31,896	31,971	27,038	29,931	28,826	28,826	28,821	29,857	28,941	32,141	32,171	29,925	29,874
7	22	15,661	22,602	22,602	21,476	15,550	16,601	21,756	15,276	18,325	21,055	16,348	15,437	16,455	19,480	18,551	22,353
8	22	22,877	25,829	25,824	23,694	24,808	23,678	24,799	24,843	22,991	22,822	23,926	23,851	23,716	23,926	24,717	24,663
9	20	22,478	24,297	24,277	24,174	24,228	24,163	23,298	24,284	23,394	23,334	24,297	24,297	24,297	24,297	24,297	24,297
10	18	Infeasible	Infeasible	Infeasible	Infeasible	15,706	15,696	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible
11	18	4525	4546	4546	4530	4460	4506	4546	4546	4546	4546	4546	4546	4546	4546	4546	4546
12	16	775	996	905	831	760	690	805	830	862	996	951	996	996	996	996	996
13	14	1757	5026	4951	2741	1596	1591	1597	1740	1770	5026	3951	5026	5026	5026	5026	5026
14	14	760	800	755	591	556	645	621	700	650	790	800	800	800	800	800	800
15	13	1500	1626	1345	1275	1365	1401	1341	1550	1610	1626	1626	1626	1626	1626	1626	
16	12	18,202	18,873	18,873	14,746	18,822	15,867	11,850	13,000	16,121	16,991	18,873	16,052	18,873	18,873		
Average improvement in penalty on schedule found after first VNS (%)				2.7	11.4	13.6	13.9	13.7	12.4	11.1	4.5	5.1	4.9	2.2	1.6	1.9	0.9

use for unassignment. Three to five is the best range for instances with varying sizes.

The success of the disruption and repair also varies between instances. For example, on instance 13, using three, four or five nurses provides almost 70% improvement on the schedule after one VNS whereas, on instance one, the final improvement is less than 1%.

4.2. Comparison of the hybrid VNS with the genetic algorithm

If the number of nurses selected in the disruption phase is three or four then the hybrid VNS outperforms the genetic algorithm on 9 of the 16 instances. Interestingly, the hybrid VNS is more effective on the instances with less than 20 nurses. For example, in the experiments in which four nurses are selected, all the schedules found for instances with less than 20 nurses have lower penalties than the genetic algorithm. If three are selected, the hybrid VNS outperforms on all but one.

It can also be seen that using the VNS phase alone is not sufficient to outperform the genetic algorithm. For all instances, after the first VNS iteration the schedule is worse than the final schedule produced by the genetic algorithm. The disruption and repair phases are required to further improve the schedule.

Fig. 3 shows the progress of the two algorithms in finding schedules for instance 12. The graph shows the penalty for the best schedule found so far for each algorithm after x minutes. For the genetic algorithm, a steady decrease in penalty can be seen over the 60 minutes as, after each generation, a new best schedule is often found as a result of the crossover and repair operations. A drop of over 1000 in penalty in under a couple of minutes is most likely due to one of the constraints with a weight of 1000 being satisfied as well as other small improvements being made. The relatively steep (as all the soft constraints with weight 1000 have now been satisfied) decrease in penalty in the last 2 minutes for the genetic algorithm is due to the final local search phase.

For the hybrid VNS, it can be seen that within four minutes (after a couple of iterations of the algorithm) the best schedule already has a penalty close to that produced finally by the genetic algorithm at the end of the 60 minutes. Between the 4th and 60th minute, an additional better schedule is found as a result of the schedule disruption, repair and VNS. From observing the algorithm when applied to the other scheduling periods, within the first 60 minutes there are usually three or four improvements in the best solution found between the 4th and 60th minute.

```
SET MoveMade to TRUE

WHILE MoveMade is TRUE

      SET MoveMade to FALSE

      FOR each move in neighbourhood one

          IF an improving move THEN
              make this move
          END IF

      END LOOP

      FOR each move in neighbourhood two

          IF an improving move THEN
              make this move
              SET MoveMade to TRUE
          END IF

       END LOOP

  ENDWHILE
```

Fig. 3. Comparison of the algorithms' progress.

4.3. Experimentation with longer computation times

As can be seen, the hybrid VNS algorithm is more likely to find a better solution the more time it is given. However, in most hospitals, schedules can be produced a long time in advance of when they are required. This observation motivated our experiments with granting the algorithm more computation time than just 1 hour.

The hybrid VNS was granted 12 hours of computation time for one of the instances (instance 12) on which a lot of testing using the genetic algorithm had been previously performed by ORTEC. For this instance, the best schedule ever found by an extended run of the genetic algorithm (for a period of about 24 hours) had a penalty of 681. The best schedule previously known for this period had a penalty of 587. This was produced over a long time period through an iterative process of using the genetic algorithm and then making some manual changes to a solution before reapplying the genetic algorithm and so on.

After 12 hours, the hybrid VNS had found a schedule with penalty 541. It is important to note that our approach is producing the best known solution (produced either automatically or manually) on this real world problem instance. Moreover, it is producing it within a period (overnight) which is quite appropriate for this kind of problem. The

Table 5
Experimentation with longer computation times

Algorithm	Penalty
Hybrid VNS after 30 minutes	736
Hybrid VNS after 60 minutes	706
Best ever genetic algorithm (24 hours)	681
Previous best known (made using manual improvements)	587
Hybrid VNS after 12 hours	541

results are summarised in Table 5. As can be seen, if more computation time is given, the schedule can be significantly improved.

5. Conclusions

The hybrid VNS algorithm described has been shown to be a relatively straightforward but highly effective approach for this problem. It is particularly effective on medium and small sized instances with less than twenty nurses. It is a viable alternative to the existing genetic algorithm for the commercial workforce management and planning software Harmony and has been added alongside the genetic algorithm in the latest versions.

For instances with less than twenty nurses, the VNS algorithm has been shown to regularly find superior schedules when compared against the genetic algorithm that is currently in use. For these sized instances, the VNS algorithm represents a significant improvement over a commercially successful methodology. It has also found best known schedules for some of the scheduling periods (by running the algorithm for 12 hours).

On instances with more than twenty instances, the VNS algorithm is competitive with the genetic algorithm and outperforms it on some instances. However, on average, the genetic algorithm is more successful on these larger instances.

The shift unassignment and repair using heuristic ordering method has been shown to be an efficient and effective method of exploring the search space and when it is combined with the VNS, schedules of high quality can be found. It was also discovered that back-tracking was very useful in finding better solutions more quickly by reducing the exploration of paths which only led to poor quality solutions.

6. Further research

Even though the results produced by this algorithm are strong, there are areas in which it could possibly be improved and which need exploring, especially if it were being designed to be run over a longer time period than 1 hour. For example, after the VNS, when selecting the area of the schedule to un-assign shifts from, a simple method is used: Unassign the shifts belonging to a fixed number of nurses with the worst individual schedules. This is an obvious heuristic and has been shown to work well. However, it is possible that there is a more effective method of selecting which, and how many, shifts to unassign and reassign using the heuristic ordering.

It may also be interesting to try replacing the VNS phase with tabu search or simulated annealing. A preliminary investigation revealed that a tabu search over a 1 hour period was not as effective as the genetic algorithm but if a longer time period is used it may be possible to achieve similar results using tabu search, especially if combined with the schedule disruption and repair method.

Acknowledgements

This work was supported by EPSRC grant GR/S31 150/01. We thank the anonymous referees for their helpful comments and suggestions.

References

[1] U. Aickelin, K.A. Dowsland, Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem, Journal of Scheduling 3 (3) (2000) 139–153.
[2] U. Aickelin, J. Li, A Bayesian optimization algorithm for the nurse scheduling problem, in: Proceedings of 2003 Congress on Evolutionary Computation (CEC2003), IEEE Press, Canberra, Australia, 2003, pp. 2149–2156.
[3] U. Aickelin, J. Li, The application of Bayesian optimization and classifier systems in nurse scheduling, in: Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)Springer Lecture Notes in Computer Science, vol. 3242, Springer, Birmingham, UK, 2004, pp. 581–590.
[4] J.F. Bard, H.W. Purnomo, A column generation-based approach to solve the preference scheduling problem for nurses with downgrading, Socio-Economic Planning Sciences 39 (3) (2005) 193–213.
[5] J.F. Bard, H.W. Purnomo, Preference scheduling for nurses using column generation, European Journal of Operational Research 164 (2) (2005) 510–534.
[6] J.F. Bard, H.W. Purnomo, Cyclic preference scheduling of nurses using a Lagrangian-based heuristic, Journal of Scheduling 10 (1) (2007) 5–23.
[7] G.R. Beddoe, S. Petrovic, Combining case-based reasoning with tabu search for personnel rostering problems, Technical Report, Automated Scheduling Optimisation and Planning Research Group, School of Computer Science and Information Technology, University of Nottingham, 2004.

[8] G.R. Beddoe, S. Petrovic, Selecting and weighting features using a genetic algorithm in a case-based reasoning approach to personnel rostering, European Journal of Operational Research 175 (2) (2006) 649–671.
[9] F. Bellanti, G. Carello, F.D. Croce, R. Tadei, A greedy-based neighborhood search approach to a nurse rostering problem, European Journal of Operational Research 153 (2004) 28–40.
[10] I. Berrada, J.A. Ferland, P. Michelon, A multi-objective approach to nurse scheduling with both hard and soft constraints, Socio-Economic Planning Sciences 30 (3) (1996) 183–193.
[11] E.K. Burke, P. De Causmaecker, S. Petrovic, G. Vanden Berghe, Metaheuristics for handling time interval coverage constraints in nurse scheduling, Applied Artificial Intelligence 20 (3) (2006).
[12] E.K. Burke, P. Cowling, P. De Causmaecker, G. Vanden Berghe, A memetic approach to the nurse rostering problem, Applied Intelligence 15 (3) (2001) 199–214.
[13] E.K. Burke, P. De Causmaecker, S. Petrovic, G. Vanden Berghe, Fitness evaluation for nurse scheduling problems, in: Proceedings of the Congress on Evolutionary Computation (CEC2001), IEEE Press, Seoul, Korea, 2001, pp. 1139–1146.
[14] E.K. Burke, P. De Causmaecker, S. Petrovic, G. Vanden Berghe, Variable neighborhood search for nurse rostering problems, in: M.G.C. Resende, J.P. de Sousa (Eds.), Metaheuristics: Computer Decision-Making, Kluwer, 2004, pp. 153–172.
[15] E.K. Burke, P. De Causmaecker, G. Vanden Berghe, A hybrid tabu search algorithm for the nurse rostering problem, in: B. McKay et al. (Eds.), Simulated Evolution and Learning, Selected Papers from the 2nd Asia-Pacific Conference on Simulated Evolution and Learning, SEAL 98, Springer Lecture Notes in Artificial Intelligence, vol. 1585, Springer, 1999, pp. 187–194.
[16] E.K. Burke, P. De Causmaecker, G. Vanden Berghe, Novel Meta-heuristic Approaches to Nurse Rostering Problems in Belgian Hospitals, in: J. Leung (Ed.), Handbook of Scheduling: Algorithms, Models and Performance Analysis, CRC Press, 2004.
[17] E.K. Burke, P. De Causmaecker, G. Vanden Berghe, H. Van Landeghem, The State of the Art of Nurse Rostering, Journal of Scheduling 7 (6) (2004) 441–499.
[18] K.A. Dowsland, Nurse scheduling with tabu search and strategic oscillation, European Journal of Operational Research 106 (2) (1998) 393–407.
[19] A.T. Ernst, H. Jiang, M. Krishnamoorthy, B. Owens, D. Sier, An Annotated Bibliography of Personnel Scheduling and Rostering, Annals of Operations Research 127 (2004) 21–144.
[20] A.T. Ernst, H. Jiang, M. Krishnamoorthy, D. Sier, Staff scheduling and rostering: A review of applications, methods and models, European Journal of Operational Research 153 (1) (2004) 3–27.
[21] P. Hansen, N. Mladenović, An introduction to variable neighborhood search, in: S. Voss et al. (Eds.), Meta-heuristics: Advances and trends in local searchs paradigms for optimization, Kluwer Academic Publishers, 1999, pp. 433–458.
[22] A. Jaszkiewicz, A metaheuristic approach to multiple objective nurse scheduling, Foundations of Computing and Decision Sciences 22 (3) (1997) 169–183.
[23] H. Li, A. Lim, B. Rodrigues, A hybrid AI approach for nurse rostering problem, in: Proceedings of the 2003 ACM symposium on Applied computing, 2003, pp. 730–735.
[24] A. Meisels, E. Gudes, G. Solotorevsky, Employee timetabling, constraint networks and knowledge-based rules: a mixed approach, in: E. Burke, P. Ross (Eds.), Selected papers from the First International Conference on Practice and Theory of Automated Timetabling, Springer Lecture Notes in Computer Science, vol. 1154, Springer, 1995, pp. 93–105.
[25] A. Meisels, A. Schaerf, Modelling and solving employee timetabling problems, Annals of Mathematics and Artificial Intelligence 39 (2003) 41–59.
[26] H. Meyer auf'm Hofe, ConPlan/SIEDAplan: Personnel assignment as a problem of hierarchical constraint satisfaction, in: PACT-97: Proceedings of the Third International Conference on the Practical Application of Constraint Technology, 1997, pp. 257–272.
[27] H. Meyer auf'm Hofe, Solving rostering tasks as constraint optimization, in: E. Burke, W. Erben (Eds.), Selected papers from the Third International Conference on Practice and Theory of Automated Timetabling, Springer Lecture Notes in Computer Science, vol. 2079, Springer, 2000, pp. 191–212.
[28] N. Mladenović, P. Hansen, Variable neighborhood search, Computers and Operations Research 24 (11) (1997) 1097–1100.
[29] S. Petrovic, G.R. Beddoe, G. Vanden Berghe, Case-based reasoning in employee rostering: learning repair strategies from domain experts, Technical Report, Automated Scheduling Optimisation and Planning Research Group, School of Computer Science and Information Technology, University of Nottingham, 2002.
[30] S. Petrovic, G.R. Beddoe, G. Vanden Berghe, Storing and adapting repair experiences in employee rostering, in: E.K. Burke, P. De (Eds.), Selected Papers from the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2002), Springer Lecture Notes in Computer Science, Springer, 2003, pp. 149–166.
[31] G. Post, B. Veltman, Harmonious Personnel Scheduling, in: E.K. Burke, M. Trick (Eds.), Proceedings of the 5th International Conference on the Practice and Automated Timetabling (PATAT 2004), Pittsburgh, PA, USA, 2004, pp. 557–559.
[32] C. Valouxis, E. Housos, Hybrid optimization techniques for the workshift and rest assignment of nursing personnel, Artificial Intelligence in Medicine 20 (2000) 155–175.