School of Computer Science and Information Technology
University of Nottingham
Jubilee Campus
NOTTINGHAM NG8 1BB, UK

Computer Science Technical Report No. NOTTCS-TR-2005-9

A Hybrid Heuristic Ordering and Variable Neighbourhood Search for the Nurse Rostering Problem

Edmund Burke, Timothy Curtois, Gerhard Post
Rong Qu and Bart Veltman

First released: November 2005

A Hybrid Heuristic Ordering and Variable Neighbourhood Search for the Nurse Rostering Problem

Edmund Burke[1], Timothy Curtois[1], Gerhard Post[2], Rong Qu[1], Bart Veltman[2]

[1] School of Computer Science & I.T. University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham. NG8 1BB. UK

[2] ORTEC bv, Groningenweg 6-33, 2803 PV Gouda. Holland.

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 backtracking. 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 algorithms.

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. (2004) identified 28 different categories of methods that have been used on personnel scheduling problems. These include constraint logic programming, constructive heuristic, 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 (Burke et al. 2004). 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 (Burke, De Causmaecker and Vanden-Berge, 1999). 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 forty Belgian hospitals and copes with many shift types, work regulations and skill categories. This work was hybridised with an evolutionary approach (Burke et al., 2001) to produce a methodology which could generate higher quality solutions but at the cost of increased computational time. Variable Neighbourhood Search (Mladenović and Hansen, 1997, 1999) has also been applied and tested on highly constrained real world nurse rostering data (Burke et al. 2003). 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".

Another investigation on real data explored a genetic algorithm approach (Aickelin and Dowsland, 2000), 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 (Dowsland, 1998). 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 (2004) 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. tackled a problem with hard constraints and objectives (or soft constraints) using various local search techniques (Bellanti

et el., 2004). 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 (Beddoe, Petrovic and Vanden-Berghe, 2002). 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 scheduling techniques and retrieving and performing these moves or repairs whenever the situation is encountered again. As an extension to their work the authors also suggest a method in which it could be combined with a meta-heuristic approach. Another relatively recent approach is a combination of constraint networks and knowledge-based rules (Meisels, Gudes and Solotorevski, 1995). Their approach was implemented in a commercial software package and has been successfully used in a number of hospitals.

Berrada, Ferland and Michelon (1996) 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 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 (Valouxis and Housos, 2000). Their method compared very favourably with a constrained programming approach. In 2004, Bard and Purnomo (2005a) employ a combination of heuristic and integer programming methods to solve nurse preference scheduling problems with up to one hundred nurses and approximately thirteen hard and soft constraints. Individual 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 schedule. Later they extended this work to further improve the quality of schedules by incorporating the use of a downgrading option (Bard and Purnomo 2005b).

By defining fuzzy constraints (i.e. constraints that may be partially satisfied and partially violated) Meyer auf'm Hofe (2000) solves real world nurse rostering problems as constraint optimisation rather than 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 (Meyer auf'm Hofe, 1997).

There are many more papers in the literature which are discussed in more detail in (Burke et al, 2004) and (Ernst et al, 2003). It is clear that relatively few papers in the literature have worked with real world data or been implemented in hospitals (Burke et al, 2004). 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. 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 on 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 2 and 3 detail the algorithm and results respectively. In section 4 we draw conclusions on the success of this approach and possible future extensions.

1. Problem Description

The data for this problem was provided by ORTEC, an international consultancy company who specialise in planning, optimisation and decision support solutions. They support hospitals and other organisations all over the world with automated workforce management solutions.

The schedules are planned in periods of one month. The 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. One of the full time nurses request no late shifts (hard constraint), another requests an early or a day shift on January 7th (soft constraint with weight 100). All the other constraints that need to be satisfied are presented in sections 1.2 and 1.3. The data is very typical of their clients' needs.

1.1 Shifts and Shift Demand

There are 4 different shift types in the problem. Table 1 shows the daily demand for these shifts. Each of the shift types cover 9 hours including one hour of resting time. So the actual working hour is 8 hours for each shift type.

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

Table 1 : Shift types and weekly demand.

1.2 Hard Constraints

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

- Cover needs to be fulfilled (i.e. no shifts must be left unassigned).
- For each day a nurse may start only one shift.
- Within a scheduling period a nurse is allowed to exceed the number of hours for which they are available for their department by at most 4 hours.
- The maximum labour time per week is on average 36 hours over a period of 13 consecutive weeks if this period does not include work during night shifts.
- The maximum number of night shifts is 3 per period of 5 consecutive weeks.

- A nurse must receive at least 2 weekends off duty per 5 week period. A weekend off duty lasts 60 hours including Saturday 00:00 to Monday 04:00.
- Following a series of at least 2 consecutive night shifts a 42 hours rest is required.
- During any period of 24 consecutive hours, at least 11 hours of rest is required. A night shift has to be followed by at least 14 hours rest. An exception is that once in a period of 21 days for 24 consecutive hours, the resting time may be reduced to 8 hours.
- The number of consecutive night shifts is at most 3.
- The number of consecutive shifts (workdays) is at most 6.

1.3 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.

1.4 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. also. The violation measurement factor is basically the degree to which the constraint is violated or the excess of the violation.

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 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 five 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 five week period (i.e. 2 more than preferred), then the penalty for this soft constraint violation would be 4000 (2^2 * 1000).

Constraint	**Weight**	**Penalty Function**	**Violation measurement factor**
For the period of 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 non-complete weekends
For any employees avoid stand-alone shifts. A stand alone shift is an isolated working day i.e. a shift on a day which is flanked by two days without shifts.	1000	Linear	Number of isolated shifts
For employees with availability of 30-48 hours per week, 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 range. 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.
For employees with availability of 0-30 hours per week, 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 range
The rest after a series of *day*, *early* or *late* shift is a minimum of 2 days.	100	Linear	Factor is one if only one day of rest otherwise zero
For employees with availability of 30-48 hours per week, within one week the number of shifts is within the range 4-5.	10	Quadratic	Difference between length of series and acceptable length range
For employees with availability of 0-30 hours per week, within one week the number of shifts is within the range 2-3.	10	Quadratic	Difference between length of series and acceptable length range
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 and acceptable length range
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 and acceptable length range
For all employees 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 and acceptable length range
For all employees 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 and acceptable length range
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

Table 2 : Soft Constraints

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 better than an infeasible schedule regardless of penalty values. Comparing feasible and infeasible schedules can be problematic for some algorithms such as a genetic algorithm as discussed by Aickelin and White. (2004). For our algorithm though these difficulties did not arise as we were always able to eventually find feasible solutions.

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 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 they have clearly described their requirements to the software administrator.

2. The Hybrid Variable Neighbourhood Search Algorithm

The algorithm that we present in this paper is 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 performed to further improve the quality of the schedules produced.

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

```
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. Backtrack one step)
    ENDIF
    Unassign shifts of a set of nurses
    Repair schedule (using heuristic ordering method)
UNTIL search terminated
```

Figure 1. Pseudo-code of the overall hybrid algorithm

2.1 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.

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

Table 3: Shift evaluation criteria

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.

2.2 Variable Neighbourhood Search

When the initial schedule has been created using the heuristic ordering method described above, a simple Variable Neighbourhood Search is applied. This makes use of two neighbourhoods. Both of these neighbourhoods are commonly used by meta-heuristic approaches and have been described before, see, for example, (Burke et al. 2002). The two neighbourhoods are defined by the following moves or changes to a schedule:

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 basically a Variable Neighbourhood Descent. Initially it was implemented in a greediest or steepest descent manner. However, after some

experimentation this was changed to a quickest descent algorithm as shown in the following pseudo-code.

```
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
              SET MoveMade to TRUE
          END IF
      END LOOP
      IF  MoveMade IS TRUE THEN
          go back to start of WHILE loop
      END IF
      FOR each move in neighbourhood two
          IF an improving move THEN
              make this move
              SET MoveMade to TRUE
          END IF
      END LOOP
  ENDWHILE
```

Figure 3. Pseudo-code of VNS

As can be seen from Figure 3 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 neighbourhood 1 and 2.

Initially the Variable Neighbourhood Search was implemented in a greediest or steepest descent manner. That is, for each of the moves in the neighbourhood, identify the move or swap that would bring the most improvement and then perform 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 on each move. This was especially noticeable in the second neighbourhood which is quite large.

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.

We will briefly explain why the available neighbourhoods are restricted to these two neighbourhoods. For example, in Burke et al, 2002 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 recent increases in hardware technology and computing power that we have witnessed over the past few years (although the paper was published in 2002, the experiments

were performed on an IBM RS6000 PowerPC as the scheduling system had already been in development for some years previously). Secondly, 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.

2.3 Making the Schedule Feasible

After the creation of the initial schedule described earlier, or the larger movements in the search space which are described later, the schedule is often still infeasible in that the shift cover may not yet have been fulfilled i.e. not all the required shifts have yet been assigned because there is no nurse to assign the shift to without violating a hard constraint. As hard constraints have no weighting and are not considered in the penalty function it is necessary to also keep track of the number of unassigned shifts or hard constraint violations for the current schedule. A schedule with less unassigned shifts than another schedule is considered to be better than the other, regardless of the difference in the penalty value. So if a move increases the penalty but decreases the number of unassigned shifts or hard constraints then it is accepted. 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. It is often the case that managing to assign an unassigned shift will increase the penalty significantly but a schedule with a high penalty and no hard constraint violations is preferable to any schedule with hard constraint violations.

2.4 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.

After the VNS, a local optimum will have been reached from which it is impossible to escape simply by examining the two neighbourhoods described. To improve upon the current schedule it is now necessary to move to a different area of the search space from which it will hopefully be possible to find a schedule with a penalty lower than the best found so far (by using the VNS again.)

The obvious way of making this 'jump' would be to use a tabu search or simulated annealing mechanism. However, preliminary experimentation revealed that it would be difficult to successfully hybridise one of these traditional meta-heuristics with this particular VNS and produce better schedules in reasonable computation times.

The reason for this is due to the large size of the second neighbourhood which increases the computation time required for the VNS. For a tabu search or simulated annealing approach to be successfully combined with this VNS, it was estimated that a search time of days would be necessary whereas (from the nature of the problem) it was preferable to keep the search time to a magnitude of hours. Although a computation time of days would not be totally unacceptable, a shorter time is preferred because often the head nurse or planner may be presented with last minute requests or situations which may require a new schedule to be created and in this scenario a very long execution time is unhelpful.

In an attempt to improve upon the schedules produced by the VNS in an acceptable computation time, an alternative method was developed. The idea is to heuristically select sections of the overall schedule which could possibly be improved and to then attempt to

improve them. To do this the nurses are ranked in order of the penalties for their own personal schedule. A certain number of nurses with the highest personal penalties are selected and all the shifts assigned to these nurses are unassigned. For these instances in which there are 16 nurses, it was discovered (after a series of experiments) that a good number of nurses to unassign the shifts allocated to was 3.

The unassigned shifts are then reassigned using the heuristic ordering method described earlier in section 2.1. VNS is then performed as before to see if a better schedule can be produced.

The process of un-assigning the shifts of a certain number of the most highly penalised nurses of the current schedule, repairing the schedule using heuristic ordering and then performing VNS could be carried out indefinitely until a satisfactory schedule is found. However, it was discovered that after the destruction, repair and VNS phases of the algorithm were performed the current schedule was often relatively poor and any subsequent schedules found from this point on were also relatively poor and only occasionally would the current schedule's penalty be reduced to an acceptable value again. The reason for this is believed to be due to the very high penalties associated with the night shifts for these particular instances. Once bad sequences of night shifts are introduced into the schedule, it is difficult to recover simply through the VNS with these neighbourhoods alone. Another weakness in this method is that, although better schedules were being found, it was too infrequently during the algorithm. It could be observed that although good schedules were being found they were often of all about the same penalty with no great improvements on the best penalty.

To overcome these problems we introduced the following process. If after the VNS, the penalty of the current schedule is not less than the best schedule found so far, then the current schedule is reset to the best schedule and the shifts for a different set of nurses are unassigned and reassigned. The set of nurses selected to have their shifts reassigned is the set with the next lowest collective penalty to the set selected previously if no overall improvement was found before.

2.5 Genetic Algorithm

Harmony uses a genetic algorithm to produce schedules and rosters. 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 nurse 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 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 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 GA phase a local search is performed to further optimise the best schedule found.

3. Results

To develop this algorithm the workforce management and planning software Harmony™ (Post and Veltman, 2004) developed by ORTEC 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 in optimising 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. Obviously 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 for comparing our work against.

Both algorithms were implemented using Delphi 5. The experiments were performed using a PC with a Pentium 1700MHz processor and Windows 2000 operating system.

3.2 Comparison of the Hybrid Variable Neighbourhood Approach with the Genetic Algorithm

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sept	Oct	Nov	Dec
GA (60 mins)	775	**1791**	2030	612	2296	9466	781	**4850**	**615**	736	2126	625
VNS (30 mins)	735	1950	2055	501	2285	9312	660	4975	761	665	2041	625
VNS (60 mins)	**735**	1866	**2010**	**457**	**2161**	**9291**	**481**	4880	647	**665**	**2030**	**520**

Table 4: Comparison of our Hybrid Approach and the Genetic Algorithm

Table 4 presents the results obtained after applying the algorithms to each of the problem instances. The best results among the approaches are highlighted in bold. As can been seen from the results, our hybrid VNS algorithm finds better solutions than the existing commercial genetic algorithm for nine out of the twelve problem instances. For seven of the twelve months the VNS after 30 minutes also outperforms the GA after 60 minutes.

It is interesting to see that although all the problem instances are quite similar, the penalties for each month are quite different (varying from 457-9466). These differences are due to the structure of the month itself. For example, if the period starts on a Sunday then many of the nurses will automatically gain an unavoidable penalty by not working a complete weekend as the previous scheduling period is empty. Due to the lack of scheduling history for each of these instances there is always an unavoidable penalty so a penalty of zero will not be found

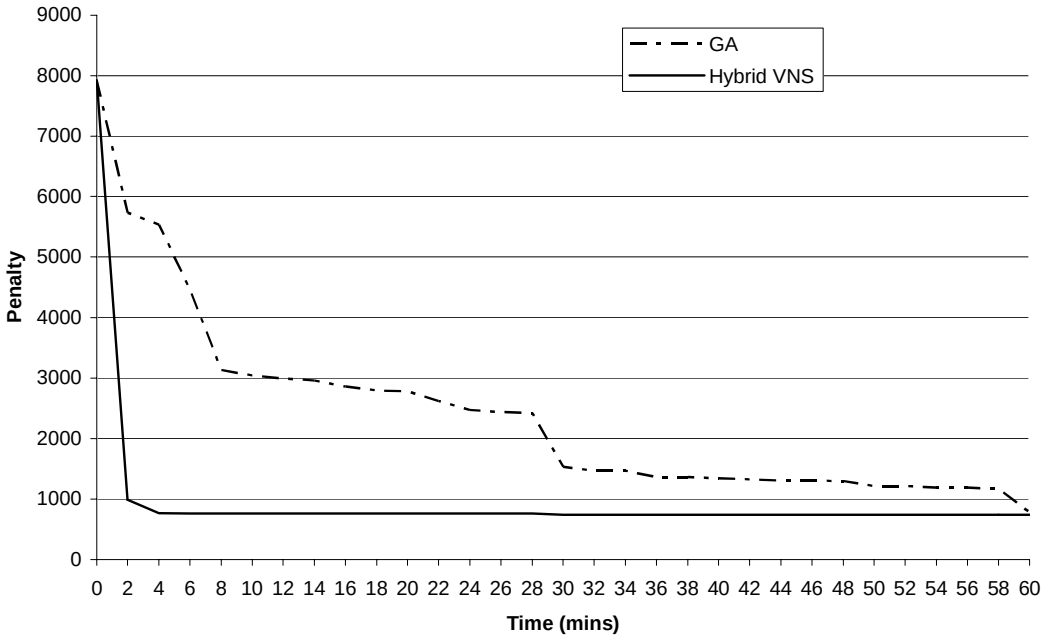

Figure 5. Comparison of the algorithms' progress for schedule January 2003

Figure 5. shows the progress of the two algorithms in finding schedules for the month of January. 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 sixty 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 two minutes for the GA is due to the final local optimisation 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 GA at the end of the sixty minutes. Between the fourth and sixtieth 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 sixty minutes there are usually three or four improvements in the best local optimum found between the fourth and sixtieth minute.

For each month, to ensure that the inputs are the same for both algorithms, the previous schedules are all emptied of assignments. Having an empty previous scheduling period already introduces penalties in the current scheduling period which cannot be removed by any possible shift assignments. The problem instances are similar in structure as the differences are due to the number of days in the month and the weekdays the month starts and finishes on.

3.2 Experimentation with longer computation times

The hybrid VNS algorithm is more likely to find a better solution the more time it is given. As can be seen, for eleven of the twelve months, the best schedule found after 60 minutes is better than the best schedule found after 30 minutes. 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 one hour.

The hybrid VNS was granted 12 hours of computation time for the scheduling period January 2003. For this scheduling period, 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 (which was produced manually) had a penalty of 587. 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 results are summarised in Table 4. As can be seen, if more computation time is given the schedule can be significantly improved. Moreover, in the context of this particular problem, it is quite appropriate to run the algorithm overnight (say 12 hours).

Algorithm	**Penalty**
Hybrid VNS after 30 minutes	736
Hybrid VNS after 60 minutes	706
Best ever G.A. (24 hours)	681
Previous best known (made by manual improvements)	587
Hybrid VNS after 12 hours	541

Table 4: Experimentation with longer computation times

4. Conclusions

The hybrid VNS algorithm described has been shown to be a relatively straightforward but highly effective approach for this problem and is a viable and more effective alternative to the existing genetic algorithm for the commercial workforce management and planning software HARMONY™. The VNS algorithm has been shown to regularly find superior schedules when compared against the genetic algorithm that is currently in use, and to have found best known schedules up to now for some of the scheduling periods (by running the algorithm for 12 hours). The VNS algorithm represents a significant improvement over a commercially successful methodology.

The shift un-assignment and repair using heuristic ordering method has been shown to be a 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.

Even though the results produced by this algorithm are good there are areas in which it could possibly be improved and need exploring, especially if it were being designed to be run over a longer time period than one hour. For example, after the VNS, when selecting the area of the schedule to un-assign shifts from, a simple method is used: Un-assign the shifts belonging to a fixed number of nurses with the highest personal 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 un-assign and reassign using the heuristic ordering. Also, eventually, the algorithm described here would run to completion. The time taken to do this would depend upon the machine executing it and the dimensions of the problem instance. It is possible to extend the search further if it were decided to give the search more time. For example, during the search the other schedules found could be saved and ranked. Then, once all the un-assignment, repair and back-track to best schedule found moves have been made, the algorithm could be restarted but back-tracking to the second best known schedule if necessary and if that does not find a better schedule try the third best and so on. This may prove to work as the un-assignment, repair, VNS sequence may be able to find an even better schedule if started from a different point in the search space, to just the best known schedule.

Acknowledgements

This work was supported by EPSRC grant GR/S31150/01. We would also like to thank the anonymous referees for their helpful comments and suggestions.

References

U. Aickelin, K. A. Dowsland (2000). “Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem.” *Journal of Scheduling*, Volume 3, Number 3, pages 139-153.

U. Aickelin and K. Dowsland (2003). “An Indirect Genetic Algorithm for a Nurse Scheduling Problem.” Computers and Operations Research, volume 31, pages 761-778.

U. Aickelin, J. Li (2004). “The Application of Bayesian Optimization and Classifier Systems In Nurse Scheduling.” *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature* (PPSN VIII), Lecture Notes in Computer Science 3242, pages 581-590, Springer, Birmingham, UK.

U. Aickelin, P. White (2004). “Building better nurse scheduling algorithms.” *Annals of Operations Research*, Volume 128, pages 159-177.

J. F. Bard, H. W. Purnomo (2005). “Preference scheduling for nurses using column generation.” *European Journal of Operational Research*, Volume 164, pages 510-534.

J. F. Bard, H. W. Purnomo (2005). “A column generation-based approach to solve the preference scheduling problem for nurses with downgrading.” *Socio-Economic Planning Sciences*, Volume 39, pages 193-213.

G. R. Beddoe, S. Petrovic, G. Vanden Berghe (2002). “Storing and Adapting Repair Experiences in Employee Rostering.” E.K. Burke, P. De Causmaecker (Eds.), *Selected Papers of the 4th International. Conference on Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science 2740 pages 148-165.

F. Bellanti, G. Carello, F. Della Croce, R. Tadei (2004). “A greedy-based neighborhood search approach to a nurse rostering problem.” *European Journal of Operational Research*, Volume 153, pages 28-40.

I. Berrada, J. Ferland, P. Michelon (1996). “A Multi-Objective Approach to Nurse Scheduling with both Hard and Soft Constraints.” *Socio-Economic Planning Science*, Volume 30, Number 3, pages 183-193.

E. K. Burke, P. De Causmaecker, G. Vanden Berghe, H. Van Landeghem (2004). “The State of the Art of Nurse Rostering.” *Journal of Scheduling*, Volume 7 Issue 6, pages 441-499.

E. K. Burke, P. De Causmaecker, S. Petrovic G. Vanden Berghe (2003). “Variable neighborhood search for nurse rostering problems.” *METAHEURISTICS: Computer Decision-Making* (edited by Mauricio G. C. Resende and Jorge Pinho de Sousa), Chapter 7, pages 153-172, Kluwer 2003 (Combinatorial Optimization Book Series).

E. K. Burke, P. Cowling, P. De Causmaecker, G. Vanden Berghe (2001). “A Memetic Approach to the Nurse Rostering Problem.” *Applied Intelligence*, Volume 15, Number 3, pages 199-214.

E. K. Burke, P. De Causmaecker, G. Vanden Berghe (1999). “A Hybrid Tabu Search Algorithm for the Nurse Rostering Problem.” *Simulated Evolution and learning, 1998, Lecture Notes in Artificial Intelligence*, Volume 1585, pages 187-194.

K. A. Dowsland (1998). “Nurse scheduling with tabu search and strategic oscillation.”*European Journal of Operational Research*, Volume 106, pages 393-407.

A. T. Ernst, H. Jiang, M. Krishnamoorthy, B. Owens and D. Sier (2004). “Annotated Bibliography of Personnel Scheduling and Rostering.” *Annals of Operations Research*, Volume 127, pages 21-144.

P. Hansen, N. Mladenović (1999). “An introduction to variable neighborhood search.” S. Voss et al. (Eds.): *Advances and Trends in Local Search Paradigms for Optimization*. Kluwer Academic Publishers, Boston, MA, pages 433–458.

A. Meisels, E. Gudes, G. Solotorevski (1995)**.** “Employee Timetabling, Constraint Networks and Knowledge-Based Rules: A Mixed Approach.” E.K. Burke, P. Ross (Eds.): *Practice and Theory of Automated Timetabling*, First International Conference Edinburgh, Springer Lecture Notes in Computer Science, pages 93-105.

H. Meyer auf’m Hofe (1997). “ConPlan/SIEDAplan: Personnel Assignment as a Problem of Hierarchical Constraint Satisfaction.” *Proceedings of the Third International Conference on the Practical Application of Constraint Technology*, (Practical Application Expo, London), pages 257-271.

H. Meyer auf'm Hofe (2000). "Solving Rostering Tasks as Constraint Optimization", E.K. Burke, W. Erben (Eds.): *Practice and Theory of Automated Timetabling, Third International Conference*, Konstanz, Springer, pages 191-212.

N. Mladenović, P. Hansen (1997). "Variable Neighborhood Search." *Computers & Operations Research.* Volume 24, pages 1097–1100.

G. Post, B. Veltman (2004). "Harmonious Personnel Scheduling." *Proceedings of the 5th International Conference on the Practice and Automated Timetabling* (PATAT 2004), pages 557-559.

C. Valouxis, E. Housos (2000). "Hybrid optimization techniques for the workshift and rest assignment of nursing personnel." *Artificial Intelligence in Medicine* Volume 20, pages 155-175.