

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

Computer Science Technical Report No. NOTTCS-TR-2007-7

## **A Scatter Search for the Nurse Rostering Problem**

*Edmund K. Burke, Timothy Curtois  
Rong Qu and Greet Vanden Berghe*

*First released: 2007*

© Copyright 2007 Edmund K. Burke, Timothy Curtois, Rong Qu and Greet Vanden Berghe

In an attempt to ensure good-quality printouts of our technical reports, from the supplied PDF files, we process to PDF using Acrobat Distiller. We encourage our authors to use outline fonts coupled with embedding of the used subset of all fonts (in either Truetype or Type 1 formats) except for the standard Acrobat typeface families of Times, Helvetica (Arial), Courier and Symbol. In the case of papers prepared using TEX or LATEX we endeavour to use subsetted Type 1 fonts, supplied by Y&Y Inc., for the Computer Modern, Lucida Bright and Mathtime families, rather than the public-domain Computer Modern bitmapped fonts. Note that the Y&Y font subsets are embedded under a site license issued by Y&Y Inc.

For further details of site licensing and purchase of these fonts visit <http://www.yandy.com>

# A Scatter Search for the Nurse Rostering Problem

Edmund K. Burke<sup>1</sup>, Timothy Curtois<sup>1</sup>, Rong Qu<sup>1</sup>, Greet Vanden Berghe<sup>2</sup>

<sup>1</sup>School of Computer Science & I.T., University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham. NG8 1BB. UK

<sup>2</sup>Information Technology Engineering Department, KaHo St. Lieven, Gebr. Desmetstraat 1, 9000 Gent. Belgium

## Abstract

The benefits of automating the rostering process include reducing the planner's workload and associated costs and being able to create higher quality and more flexible schedules. This has become more important recently in order to retain nurses and attract more people into the profession. Better quality rosters also reduce fatigue and stress due to overwork and poor scheduling and help to maximise the use of leisure time by satisfying more requests. A more contented workforce will lead to higher productivity, increased quality of patient service and a better level of healthcare.

This paper presents a scatter search approach for the problem of automatically creating nurse rosters. Scatter search is an evolutionary algorithm which has been successfully applied across a number of problem domains. To adapt and apply scatter search to nurse rostering, it was necessary to develop novel implementations of some of scatter search's subroutines. The algorithm was then tested on publicly available benchmark instances and compared against previously published approaches. The results show the proposed algorithm is a robust and effective method on a wide variety of real world instances.

## 1 Introduction

This paper presents a scatter search for the nurse rostering problem. Like genetic algorithms [40, 47, 50, 78], memetic algorithms [59, 60, 73] particle swarm optimisation [57, 69] and ant colony optimisation [31, 32], a key feature of scatter search is the maintenance of a population of solutions. This is in contrast to many other metaheuristics which generally work with one solution, for example simulated annealing [1, 58], tabu search [41, 43], GRASP [39], variable neighbourhood search [49, 72] etc. In genetic algorithms, these sets of solutions are often referred to as populations. To continue the metaphor, the individuals within a population may be labelled as parents and new solutions are usually called offspring. New solutions are generally created from two parents in the population through crossover and mutation operations. Although, for different problems, the details of the crossover and mutation functions can vary, there is typically some stochastic element to their operation. This contrasts with scatter search in which the method for forming new solutions is designed to minimize (if not eliminate) decisions being allocated to random (or more usually pseudo-random) chance. The idea is to try and replace calls to the random number function with "*systematic and strategically designed rules*" [44].

Another difference is found in the way that new solutions are added to the population or reference set. In many genetic algorithms, new solutions are allowed to enter the current population if their quality (usually determined by an objective function) is greater than the worst member of the current population. In scatter search, a method for comparing the similarity of two solutions is used to measure the reference set's overall diversity. Whether or not a new solution enters the reference set may then be decided by not only its quality, but also its contribution to the reference set's diversity. The goal is to try and maintain the highest quality, yet diverse, reference set.

Some genetic algorithms also use a local search or other optimisation method on each of the new solutions between generations in order to improve their quality. These methods may also act to repair the solutions if they were incomplete or infeasible after the crossover stage.

These genetic algorithms plus local search are often labelled as memetic algorithms but may also be referred to as hybrid genetic algorithms and genetic local search. The idea of using a heuristic improvement process on new solutions is also common to scatter search. These improvers/repairers can be a noticeable bottleneck in the algorithm though. Also, as new solutions can be created from more than one reference solution (in contrast to genetic algorithms), even with a small reference set, many new solutions can be created at each iteration. Therefore, the reference set is typically a lot smaller than the corresponding population in a genetic algorithm.

However, as the boundaries between metaheuristic algorithm classification sometimes overlap and as different metaheuristic approaches are often hybridised, so also, features of scatter search may appear in genetic algorithms and vice versa. The comparisons between genetic algorithms and scatter search described here are just a basic introduction. For further information on scatter search and a more in depth analysis see [42, 44, 45, 61].

Population based optimisation methods have previously and successfully been applied to employee timetabling problems in various forms. Some of these approaches are briefly discussed.

Tanomaru [79] uses a genetic algorithm to solve a staff scheduling problem which has no predefined shift types. Instead of standard shifts with fixed start and end times (e.g. early, late, night etc), the planning horizon is split into uniform time intervals and a minimum cover requirement for each time interval is specified. Employees are categorized into groups with each group having different working constraints. Cover requirements are further broken down by giving a minimum number of employees required from each group. Wages vary between employees and the objective is to minimise the combination of employment costs and constraint violations. At each generation after a standard reproduction and crossover stage, a number of heuristic procedures are applied to further increase the quality of the schedules. An optimal solution is found for a smaller instance and good solutions are produced within a few minutes for medium sized instances.

Jan et al. [53] evaluate the use of a genetic algorithm to solve a nurse scheduling problem. Although the authors note that the problem is simplified slightly as they are conducting preliminary tests, some common hard (coverage and personal requests) and soft constraints (mostly related to night shifts) are still present. The objectives are to minimise the total penalty incurred due to soft constraint violations and to minimise the variance in the individual nurse schedule penalties so ensuring fairness. A population of non-dominated solutions according to these two objectives is maintained. New solutions are not produced via crossover between solutions in this population however, but rather by mutating a single schedule. This aids the maintenance of only feasible solutions. The authors also suggest a method for allowing a decision maker to adjust the schedule and guide the search during its execution.

Cai and Li [24] present a genetic algorithm for solving a staff scheduling problem with three objectives of decreasing importance:

1. Minimize total costs in satisfying cover demands (each feasible schedule has a different cost).
2. Maximise the staff surplus (in the case of underestimated cover requirements).
3. Minimise the variation in surplus over the planning periods.

Feasible weekly schedules with varying costs are predefined and are assigned when the best number of workers for each schedule is found. The employment of fitness values for selecting parents is replaced with a ranking scheme according to the objectives. When comparing the

objectives, an extra parameter is included to allow slightly different objective function values to be treated as equal if preferred. Crossover is performed using carefully constructed masks aiming to maintain diversity but not to create overly infeasible offspring. Repairs are performed heuristically by repeatedly identifying the most violated constraint and assigning an additional schedule to satisfy it subject to other constraints. If there is more than one possible schedule for performing the repair, the best one according to the objectives is used. The results were of sufficient quality for the approach to be included in an existing scheduling system.

Aickelin and Dowsland [3, 5] developed and tested a genetic algorithm in place of the tabu search method in [34]. They were able to achieve a similar performance with the genetic algorithm and felt it was more robust when applied to a greater variety of instances. To achieve this success, the genetic algorithm required a lot of adapting to the problem (as did the tabu search also) to exploit its structure. This was done by splitting the population into a hierarchy of sub-populations based on nurse grades and then strategically using crossover on these better building blocks. Also, if a solution exhibits an under/over coverage structure, which is difficult for the genetic algorithm to repair, it is more severely penalised. To improve solution quality further, a basic hill climber involving testing a different work pattern for each nurse and accepting it if it increases fitness is also applied to some solutions.

In [6] they also tested an indirect genetic algorithm on the same problem. This time, the genetic algorithm is used to identify permutations of nurses which are then passed to a decoder which applies heuristic rules to this permutation to assign work patterns and to construct the roster. In effect, the decoder acts as a fitness function for the genetic algorithm. Using this method removes, from the genetic algorithm, the complications of dealing with the problem specific constraints and infeasibility. Three decoders are experimented with, each having a different bias between producing feasible solutions and solutions which maximise nurse satisfaction. After fine tuning the heuristics and penalty weights, the algorithm was capable of slightly better results than the direct genetic algorithm. Aickelin and White [8] later presented an improved method for fine tuning algorithms which is superior to their manual attempts in selecting the best parameters. They discuss the difficulties in comparing results which may contain feasible and infeasible solutions and suggest a method for handling them.

Burke et al. [17] present a number of memetic algorithms for nurse scheduling. Experiments are conducted combining different crossover operators and local improvements methods. The best approach is a hybridisation of a tabu search [21] and a crossover operator based on selecting the 'best' events from each parent. Although the best memetic algorithm required a greater computation time than the tabu search, the solutions produced are nearly always of a higher quality. This approach is discussed further in section 4.

Dias et al. [30] developed a tabu search and a genetic algorithm for solving rostering problems in Brazilian hospitals. Generally, each employee only works one type of shift (morning, evening or night) which helps simplify the problem. However, there are still a number of soft constraints such as minimum and maximum working days, a required number of days and weekends off and minimum and maximum cover for each day. These constraints are weighted and used to form the objective function. Tests showed the genetic algorithm slightly outperformed the tabu search but, in practice, both approaches were welcomed by the hospital users without preference as they were both significantly superior to manual efforts. A user interface which easily allowed small changes to the schedule by hand was also appreciated by the staff.

Inoue et al. [52] argue that evaluation functions for nurse rostering problems are not always accurate enough as it is difficult for head nurses to describe all the qualities they wish to see in a high quality schedule. To overcome this, they describe an interactive scheduling

approach. A fitness function based on a measure of the violation of easily defined constraints such as cover requirements (including skill mixes), forbidden shift patterns and personal requests is used in an evolutionary algorithm. However, at each generation, the user may modify and fix parts of the schedules in the population to increase their quality based on the user's perception. The results of using various combinations of crossover, mutation and heuristics for repairing the crossover in the genetic algorithm are presented.

Özcan [76] presents a number of mutation, crossover and hill climbing operators in a memetic approach to solving a nurse rostering problem. Two different shift types must be assigned over a planning period of a fortnight. There are a relatively small number of hard and soft constraints. The hill climbing method examines one constraint at a time and tries to repair violations by changing shift assignments. Experiments are performed on randomly generated instances to determine the best types of mutation and crossover.

Other examples of successful evolutionary approaches applied to employee timetabling include [35, 56, 74]. Although genetic and memetic algorithms have proven popular methodologies for personnel scheduling, a wide variety of other Operations Research and Artificial Intelligence methods have also been used to produce high quality rosters. These include mathematical programming [9, 12, 71, 80] and in particular and more recently column generation methods [10, 11, 38, 55, 66, 77]. Constraint programming [2, 27, 29, 67, 68, 70, 82, 83] and local search and metaheuristics [15, 18, 20, 33, 34, 51, 68, 75, 81] are also successful. More novel approaches include Bayesian optimisation algorithms [4, 7], case-based reasoning [13, 14], hyperheuristics [23] and Pareto optimisation [54, 62]. Overviews of many of these approaches and others can be found in the survey papers [22, 36, 37].

At the time the work presented here was undertaken, there had been very little research into investigating scatter search for personnel scheduling and no known applications of it to nurse rostering. This made it an appealing method to test, especially considering how successful evolutionary approaches for nurse rostering have been previously. This is a conclusion which was simultaneously (but independently) made by Maenhout and Vanhoucke. They have also implemented and tested a scatter search for the nurse rostering problem [65]. Surprisingly, their implementation of Glover's template is almost entirely different to ours. However, they also were able to achieve successful results albeit on a variation of the nurse rostering problem. Scatter search and path relinking strategies have been applied to a considerable variety of problems other than nurse scheduling though. For example, arc routing [48], linear ordering [25], quadratic assignment [28], mixed integer programming [46] and exam proctor assignment [64]. All of these studies have demonstrated promising results.

In the next section we introduce the nurse rostering problem. Section 3 describes the scatter search implementation and section 4 contains the results of testing this algorithm on a variety of benchmark nurse rostering problems. To help draw conclusions the scatter search has been compared against Brucker et al.'s constructive method [16] and the memetic algorithm of Burke et al. [17]. The variable depth search of [19] is used for comparisons also. Finally we conclude in section 5 with some views on the success of this research.

## 2 Benchmark Instances

Basically stated, the nurse rostering problem requires the assignment of shifts to personnel to ensure that sufficient employees are present to perform the duties required. There are usually a number of constraints such as working regulations and legal requirements and a number of objectives such as maximising the nurses working preferences and/or minimising wage costs.

The instances we are solving are benchmark data sets taken from real world scenarios. An xml data format has been developed for the purpose of presenting and sharing complex personnel scheduling instances and solutions. All the instances along with best known solutions are publicly available at the research website (<http://www.cs.nott.ac.uk/~tec/NRP>). Source code for the objective functions, parsers and software for viewing and manually altering rosters is also provided. Providing previously implemented objective functions helps ensure the accuracy of results and verify new solutions. It also possible to create and view an html version of the solution rosters which highlights violations and explains the penalties for each violation. Examples of these for best known solutions are available at the website. The source code for the algorithms presented in this paper and other approaches are also available.

The instances used here for the experiments and are listed in Table 1.

Instance	Nurses	Shift types	Skill levels	Planning horizon
BCV-1.8.1	8	4	2	28 days
BCV-2.46.1	46	4	1	28 days
BCV-3.46.1	46	3	1	26 days
BCV-4.13.1	13	4	2	29 days
BCV-5.4.1	4	4	1	28 days
BCV-6.13.1	13	4	2	30 days
BCV-7.10.1	10	6	1	28 days
BCV-8.13.1	13	4	2	28 days
BCV-A.12.1	12	4	2	31 days
ORTEC01	16	4	1	31 days

Table 1 Test Instances

For all these instances the cover requirements or nurse demand are specified per shift type for each day in the planning period. Over cover and under cover is not permitted. Another hard constraint is that a nurse may not work more than one of the same shift type on the same day (this ensures solution structure). The only other hard constraint is that a shift which requires a certain skill can only be assigned to a nurse that has that skill.

Every other constraint is soft. This means it may be violated but a penalty will be added to this solution which is proportional to the severity (size) of the violation and the relative importance of the constraint (set using weights). The objective is to minimize the sum of all penalties due to soft constraint violations.

Each employee can have a unique set of constraints (and priorities) for their schedule (work pattern). This considerably increases the flexibility of the model but also the complexity. Examples of constraints that can be set for each employee include:

- Maximum number of shifts worked during the scheduling period.
- Maximum and minimum number of hours worked during the scheduling period or per week.
- Maximum and minimum number of consecutive working days.
- Maximum and minimum number of consecutive non-working days.
- Maximum number of a specific shift type worked. For example, maximum zero night shifts for the planning period or a maximum of seven early shifts. This constraint can also be specified for each week. For example, a nurse may request no late shifts for a certain week.

- Maximum number of weekends worked in four weeks (a weekend definition is also a user definable parameter i.e. Friday and/or Monday may be considered as part of the weekend).
- Maximum number of consecutive weekends worked.
- No night shifts before a weekend off.
- No split weekends, i.e. shifts on all days of the weekend or no shifts over the weekend.
- Identical shift types over a weekend. For example, if a nurse has a day shift on Saturday then he/she may prefer to have a day shift on Sunday also.
- Minimum number of days off after night shifts.
- Valid numbers of consecutive shift types. For example, three or four consecutive early shifts may be valid but two or five consecutive early shifts may not.
- Shift type successions. For example, if shift rotation is allowed, is shift type A allowed to follow B the next day?
- Maximum total number of assignments for all Mondays, Tuesdays, Wednesdays... For example, a nurse may request not to work on Wednesdays or may require to work a maximum of two Tuesdays during the scheduling period.
- Avoid a secondary skill being used by a nurse. Sometimes a nurse may be able to cover a shift which requires a specific skill but they may be reluctant to do so as it is not their preferred duty. An example would be a head nurse not wanting to stand in for a regular nurse.
- Day on/off or specific shift on/off requests with associated priorities.

### 3 The Scatter Search for Nurse Rostering Problems

In Glover's template for scatter search [42], five component subroutines for the overall process are outlined. The following sections describe an implementation of these subroutines for the nurse rostering problem. The overall scatter search is outlined in Figure 1.

0. Create initial set of diverse solutions
1. Improve each solution in diverse set
2. Create initial reference set (*RefSet*)
3. Make a copy of the reference set (*RefSetCopy*)
4. FOR each untried subset of solutions in *RefSet*
  5. Combine solutions in subset to produce a new solution (*NewSolution*)
  6. Improve *NewSolution*
  7. Replace a solution in *RefSetCopy* with *NewSolution* subject to certain criteria
8. ENDFOR
9. IF *RefSet* and *RefSetCopy* are not identical
  10. SET *RefSet* := *RefSetCopy*
  11. GOTO 3.
12. ENDIF
13. Return the best solution in *RefSet* or GOTO 0.

Figure 1 Scatter search overview

#### 3.1 The Diversification Generation Method

A diversification generation method is required to create a diverse set of solutions. These solutions are then improved (according to the objective function) and added to the initial reference set subject to certain criteria. When creating the diverse set of solutions, the objective value for each solution is not relevant, only its similarity to other solutions in the set is of interest. A number of methods were tested for creating diverse solution sets with varying degrees of success. Of these methods, the one outlined in Figure 2 consistently produced the most diverse set of solutions.

1. Create an empty set (*set*) of size *n* for the diverse solutions
2. UNTIL *set* is full
3. Create a roster (*roster*) with no assignments made
4. FOR each day (*day*) in *roster*
5. FOR each shift type (*shift*) to be covered on *day*
6. UNTIL the cover is satisfied
7. Assign *shift* to a nurse who has been assigned *shift* on *day* the least number of times in all other rosters in *set* (subject to no hard constraint violations)
8. If more than one nurse has received *shift* on *day* the least number of times then randomly select one of them
9. ENDUNTIL
10. ENDFOR
11. ENDFOR
12. Add *roster* to *set*
13. ENDUNTIL

Figure 2 Pseudocode for the scatter search initial set creation

To measure the similarity of two schedules, a simple but effective method is used: counting the number of *nurse to shift* assignments in common. An example of this is given in Figure 3 which shows the individual schedules of three nurses (labelled D, E, and F) from two different schedules. Identical assignments are highlighted, for example nurse E has a late shift (L) on Monday 4<sup>th</sup> in both schedules. In this example, just looking at these three nurses' schedules, there are seventeen identical assignments in the two rosters.

	1					2					3					4														
	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S		
D	V	V	V	L	L			L	L	L			V	V	D	D	D	V	V							N	N	N	N	
E	L	L	L			D	D	V	V		D	D			V	V	V	D	DH					L	V	V		DH	DH	
F			N	N		V	V	D	N	N					N	N		L	L	V	V	V	V			L	L	L	D	D

  

D	V	V			D	D			L	L	L	N	N	N	N		D	D		V	V	V			N		L	L		
E	L	L	V	V	V			D	D	D			D	D	V	V	L	L	DH							N	N		DH	DH
F			N	N		L	L		V	V	V	V			L	L			D	D	D	L				D	D	N	N	N

Figure 3 Example of the roster similarity measure

To demonstrate the success of the method, we compare it to a purely random approach which simply assigns randomly selected shifts to randomly chosen nurses. The assignments are made subject to no hard constraint violations (e.g. skills needed to perform certain shifts) until total cover is satisfied. A set of ten solutions was generated using both methods and the total number of shift assignments in common between all solutions in the set counted. Five repeats were performed on each instance and the average numbers of common assignments for each instance are given in Table 2. The results show that, on nine out of the ten instances, the random method made approximately twice as many common assignments. On the other instance, the random method made roughly 30% more common assignments.

Instance	Diversification method	Random Assignment
----------	------------------------	-------------------

ORTEC01	1064	2101
BCV-1.8.1	286	786
BCV-2.46.1	1705	3752
BCV-3.46.1	1770	3675
BCV-4.13.1	609	1190
BCV-5.4.1	614	842
BCV-6.13.1	986	1859
BCV-7.10.1	374	1102
BCV-8.13.1	901	1631
BCV-A.12.1	434	1192

Table 2 Comparison of common assignments by different generation methods

### 3.2 Improvement Method

The goal is to try to improve any solutions according to their objective function. If necessary, it may also repair solutions. The solutions which it works upon may be those produced by the diversification generation method (see section 3.1) or the solution combination method (see section 3.5).

The improvement method used is the time predefined variable depth search presented in [19]. Experiments were performed using a number of different maximum run times to investigate how it affected the performance of the scatter search. The results are provided in section 4.

The variable depth search belongs to the family of very large-scale neighbourhood searches. It works by chaining together moves and swaps of shifts between nurses. These single moves and swaps define specialised neighbourhoods that have been shown very effective for nurse rostering. The success of the variable depth search is due to a number of heuristics used to select the moves to chain together and whether to continue a current chain. For the experiments in this paper the heuristics PG (the positive gain criterion) with TR (the time restriction heuristic) and VF (only selecting moves on days that have violations) were used.

### 3.3 Reference Set Update Method

The reference set update method is used at two separate stages in the algorithm. It is used to create the initial reference set from the solutions produced by the diversification method. Afterwards, it is used to maintain the reference set. It decides whether to add to the reference set new solutions that are produced by the combination and improvement methods.

The reference set is initialised in a similar manner to that used by Glover et al. [45]. After all the solutions produced by the diversification method are improved by the variable depth search, they are ranked according to the objective function. The best  $b_1$  of these solutions are then added to the reference set. From the remaining solutions,  $b_2$  are selected and added, based on their contribution to the diversity of the reference set. Figure 4 outlines the process.  $b_1$ ,  $b_2$  and the number of diverse solutions to initially generate are all parameters which may affect the running time of the algorithm and the quality of schedules produced. In section 4, the results of varying these parameters are presented.

$P$  is the set of solutions created using the diversification generation method.

RefSet is the reference set and is initially empty.

$b_1$  and  $b_2$  are algorithm specific parameters (integers  $\geq 0$ ).

1. FOR 1 to  $b_1$
2. Select from  $P$  the best solution according to the objective function
3. Remove the solution from  $P$  and add it to *RefSet*

4. ENDFOR
5. FOR 1 to  $b_2$
6. For each solution in  $P$  calculate its total similarity to all the solutions currently in  $RefSet$  (using the similarity function)
7. Select the least similar solution (the schedule with least assignments in common with other rosters in  $RefSet$ )
8. Remove the solution from  $P$  and add to  $RefSet$
9. ENDFOR

Figure 4 Scatter search reference set initialisation

After the solution combination method, new solutions are added to the reference set if their objective function value is better than the reference set's current worst solution and the set does not already contain an identical solution. If a new solution is added, the current worst solution is removed.

### 3.4 Subset Generation Method

The subset generation method is used to identify the subsets of solutions in the reference set that will be used by the combination method to create new solutions. A commonly used subset generation method in scatter search is that suggested by Glover [42]. This approach is also adopted here. Using this method, four different types of subsets of increasing size are identified. They are:

1. All unique subsets of the reference set containing 2 elements.
2. Subsets of size 3 identified by adding to each 2-element subset (above) the best solution not already in this subset.
3. Subsets of size 4 identified by adding to each 3-element subset (above) the best solution not already in this subset.
4. Subsets containing the best  $i$  solutions, for  $i = 5$  to  $|RefSet|$

The best solutions here refer only to the objective function values.

At each iteration, it is also necessary to keep a record of which solutions in the reference set are new. This avoids combining sets of old solutions which were already combined in the previous iteration.

### 3.5 The Solution Combination Method

The solution combination method uses two or more solutions (selected by the subset generation method) for reference and produces one or more new solutions, often using a path relinking mechanism. These new solutions are then improved by the improvement method and then either added to the reference set or discarded by the reference set update method.

Although the subset generation method can be easily adapted to a wide range of problems, the solution combination method is often more specifically designed for each problem. Glover et al. [44] discuss a number of forms that the solution combinations or path relinking could take. The solution combination method developed here is categorized in their paper as a constructive neighbourhood approach “where the guiding solutions vote for attributes to be included in the initiating solution” [44]. In our case, the attributes are *shift to nurse* allocations within the rosters.

A solution can be regarded as simply a number of *shift to nurse* assignments. In the solution combination method, each *shift to nurse* allocation in each solution to be combined is regarded as a *vote* for a *candidate*. The candidates available for selection are all the possible *shift to nurse* assignments for the problem instance. All these votes are then analysed and used

to construct a new solution. The shift assignments (candidates) for the new solution are made according to the number of votes they received from the guiding solutions. The pseudocode in Figure 5 outlines the process.

In Figure 5, a candidate is a *shift to nurse* assignment on a specific day. The voters are the guiding solutions, each solution is a voter and each assignment within that solution is a vote for a specific candidate. The first step in the process is to create a new solution which initially has no assignments.

1. Identify *candidates* as the set of all possible shift to nurse assignments for this instance
2. Collect all the candidates' votes from each solution in the guiding set
3. Remove from *candidates* any candidate with zero votes
4. IF *candidates* is empty  
GOTO 10.
5. Sort *candidates* by:
  - a) decreasing total number of votes
  - b) increasing total number of votes successful for voters selecting this candidate
  - c) increasing sum of objective function values for voters selecting this candidate
6. Select and remove the first candidate in *candidates*
7. Make the assignment represented by this candidate in the new solution unless it exceeds cover requirements or breaks any hard constraints
8. IF the assignment was made  
GOTO 4.
9. IF *candidates* is not empty  
GOTO 6.
10. Return new solution

Figure 5 An outline of the scatter search solution combination method

At step 5 of Figure 5, the list of candidates (that is, *shift to nurse* assignments) is sorted by the number of votes they received. As the guiding sets of solutions are small (see the subset definitions), the candidates are often tied. If this is the case, two tie-breakers are used. If two candidates receive the same number of votes, the candidate whose voters have had the least total number of successful votes is ordered first. If this does not differentiate between the two candidates, then the candidate whose voters have the lowest sum of objective function values comes first (lowest as it is a minimisation problem).

The requirements at step 7 ensure that no hard constraints will be violated by the new solution except possibly not providing full cover (but not exceeding cover). If the shift coverage constraint is not satisfied, it is repaired by the variable depth search.

## 4 Results

The algorithm was tested using the benchmark data sets introduced in section 2.

Two preliminary investigations were conducted with the scatter search. Firstly, we varied the amount of computation time given to the improvement method (variable depth search) each time it is used. Secondly, we evaluated the effect of using a larger reference set. The main aim in these experiments was to examine the difference in solution quality with respect to the variation in computation time. It was expected that the larger the reference set and/or the more time given to the variable depth search, the better the solutions. The purpose though was to examine the trade off between computation time and solution quality in order to investigate the balance.

To conduct these experiments, a reference set of size 5 ( $b_1=3$ ,  $b_2=2$ , initial number of solutions=8) was used and the variable depth search was given 5 seconds, 30 seconds and 5 minutes. An additional experiment was conducted where the variable depth search was

replaced with a hill climbing algorithm which uses a single shift move neighbourhood commonly used in nurse rostering approaches e.g. [26, 54, 63, 68]. This local search typically has a very short execution time ( $< 1$  second on smaller instances).

Secondly, a reference set of size 10 ( $b_1=6$ ,  $b_2=4$ , initial number of solutions=20) was used. As a larger reference set would require the improvement method to be called many more times, the improvement method was restricted this time to the hill climber and then tried with the variable depth search with a limit of 5 seconds per execution.

The results of these experiments are shown in Table 3. As well as the computation time, the number of solutions evaluated is also given. The experiments were performed on desktop PC with an Intel Pentium 4 2.4GHz processor.

Data set	Penalty	Evaluations	Time	Penalty	Evaluations	Time	Penalty	Evaluations	Time
		RefSet=5 (b1=3, b2=2), Initial solutions=8, Improvement method=HillClimber			RefSet=5 (b1=3, b2=2), Initial solutions=8, Improvement method=VDS (5secs)			RefSet=5 (b1=3, b2=2), Initial solutions=8, Improvement method=VDS (30secs)	
BCV-A.12.1	1813	2,765,040	6 mins, 42 sec	1378	43,256,411	1 hr, 23 mins, 43 sec	1518	44,925,889	1 hr, 31 mins, 7 sec
ORTEC01	2551	4,195,911	3 mins, 51 sec	470	39,643,699	22 mins, 19 sec	341	174,202,494	1 hr, 32 mins, 35 sec
BCV-1.8.1	275	918,415	1 mins, 1 sec	253	33,082,154	23 mins, 18 sec	252	91,816,046	1 hr, 3 mins, 37 sec
BCV-2.46.1	1574	18,097,242	16 mins, 27 sec	1577	16,711,732	13 mins, 13 sec	1572	102,694,355	1 hr, 16 mins, 43 sec
BCV-3.46.1	3427	31,587,573	24 mins, 6 sec	3301	250,015,392	2 hrs, 22 mins, 32 sec	3312	298,790,524	3 hrs, 21 mins, 7 sec
BCV-4.13.1	12	1,786,825	1 mins, 43 sec	10	15,009,273	9 mins, 18 sec	10	46,820,989	28 mins, 36 sec
BCV-5.4.1	48	88,930	0 mins, 13 sec	48	4,414,062	3 mins, 5 sec	48	18,586,063	13 mins, 1 sec
BCV-6.13.1	915	2,582,780	2 mins, 54 sec	768	20,872,954	15 mins, 8 sec	768	67,159,785	54 mins, 54 sec
BCV-7.10.1	382	808,381	1 mins, 7 sec	381	10,184,684	7 mins, 39 sec	381	52,674,941	46 mins, 26 sec
BCV-8.13.1	149	1,558,705	1 mins, 45 sec	148	10,698,537	7 mins, 26 sec	148	29,749,689	24 mins, 43 sec
		RefSet=5 (b1=3, b2=2), Initial solutions=8, Improvement method=VDS (300secs)			RefSet=10 (b1=6, b2=4), Initial solutions=20, Improvement method=HillClimber			RefSet=10 (b1=6, b2=4), Initial solutions=20, Improvement method=VDS (5secs)	
BCV-A.12.1	1440	491,902,115	16 hrs, 41 mins, 19 sec	1640	18,257,546	43 mins, 52 sec	1490	154,689,835	5 hrs, 7 mins, 7 sec
ORTEC01	325	306,473,260	2 hrs, 40 mins, 45 sec	1881	25,548,930	25 mins, 59 sec	500	221,390,330	1 hr, 56 mins, 58 sec
BCV-1.8.1	252	475,692,443	5 hrs, 29 mins, 6 sec	267	6,412,094	7 mins, 6 sec	253	112,208,101	1 hr, 11 mins, 3 sec
BCV-2.46.1	1572	1,679,197,437	1 day, 21 mins, 22 sec	1573	75,473,986	1 hr, 15 mins, 15 sec	1573	61,658,977	50 mins, 14 sec
BCV-3.46.1	3294	2,747,238,252	1 day, 8 hrs, 47 mins, 21 sec	3414	199,490,595	2 hrs, 40 mins, 37 sec	3293	4,031,912,911	1 day, 1 hr, 35 mins, 24 sec
BCV-4.13.1	10	313,030,852	3 hrs, 10 mins, 47 sec	10	11,588,880	11 mins, 13 sec	10	87,667,326	51 mins, 45 sec
BCV-5.4.1	48	32,520,202	22 mins, 35 sec	48	408,408	0 mins, 52 sec	48	19,016,777	11 mins, 8 sec
BCV-6.13.1	768	319,751,554	4 hrs, 37 mins, 14 sec	886	10,784,081	12 mins, 54 sec	768	142,817,348	1 hr, 51 mins, 42 sec
BCV-7.10.1	381	332,434,426	5 hrs, 6 mins, 16 sec	381	5,478,574	7 mins, 32 sec	381	121,430,659	1 hr, 21 mins, 26 sec
BCV-8.13.1	148	191,706,580	2 hrs, 49 mins, 44 sec	148	6,299,783	7 mins, 54 sec	148	79,125,315	49 mins, 38 sec

Table 3 Results of varying the reference set size and the improvement method

As expected, increasing the size of the reference set and increasing the maximum execution time of the variable depth search, increased the total execution time. A benefit in terms of increased solution quality is not obvious however. Using a reference set of size 5 (RefSet=5) and with the variable depth search given a maximum time of 5 minutes (VDS=5mins) produced very long execution times. The results were not significantly better than RefSet=5, VDS=30secs though which had a shorter execution time. Again, using a larger reference set did not result in much better solutions in relation to the required execution time. The results suggest that the best trade off between solution quality and execution time is with RefSet=5 and VDS=5secs. This was able to produce very good solutions on all instances without unfeasibly long run times. The result for BCV-A.12.1 using these parameters is particularly encouraging as it is a new best result.

Using these settings, further tests were performed to compare this approach to the memetic algorithm, MEH, of Burke et al [17]. MEH is a hybrid approach which performs a tabu search on individuals in the population between generations and a greedy shuffling step on the best solution at the end. It was shown to be a robust approach and the best method on more difficult instances. The same settings as described in the original paper were used (underlying memetic algorithm ME4, population size of 12 and stop criterion of no improvement during two generations). Five repeats of both the scatter search and MEH were executed. The best and average solutions and average computation times are shown in Table 4. The scatter search with RefSet=5 and using the hill climbing improver was also tested for comparative purposes.

Instance	MEH			Scatter search using hill climber (SS1)			Scatter search using VDS at 5 secs. (SS2)		
	Best	Average	Time (secs.)	Best	Average	Time (secs.)	Best	Average	Time (secs.)
ORTEC01	1580	2904	3351	601	1707	713	405	445	1381
BCV-1.8.1	275	285	99	263	268	66	253	253	815
BCV-2.46.1	1574	1589	2560	1573	1588	1665	1575	1594	1076
BCV-3.46.1	3439	3471	10714	3379	3396	5226	3344	3380	3814
BCV-4.13.1	12	19	93	11	12	114	10	10	374
BCV-5.4.1	48	48	27	48	135	9	48	48	126
BCV-6.13.1	815	959	385	806	904	207	768	768	592
BCV-7.10.1	381	390	66	381	385	76	381	381	361
BCV-8.13.1	148	166	219	148	148	123	148	148	226
BCV-A.12.1	1990	2349	929	1685	1813	518	1434	1522	1786

Table 4 Scatter search compared to MEH

The results show that the scatter search using the hill climber as the improver (SS1) produces average solutions which are found in less time and are better than MEH on 7 out of the 10 instances. For the other 3 instances, the solutions are better for 2 of them but used slightly more time and worse on 1 but using less time. The scatter search using the variable depth search with a maximum run time of 5 seconds as the improver (SS2) outperforms SS1, but with a longer run times for all instances except BCV-3.46.1. SS2's average results compared to SS1 are better for 8 of the instances, equal on 1 and were worse on the other.

For further comparisons, Table 5 contains the average results of SS1, the best results of Brucker et al's heuristic constructive approach [16] and Burke et al's best result on the ORTEC01 data set [18]. Brucker et al.'s experiments were all performed on the same machine as SS1.

As can be seen, SS1 outperforms the constructive approach on all but two instances (for one of which they are equal). It can also be noted that SS1 uses less computation time on all instances but one. SS1 does not outperform the result of Burke et al. on instance ORTEC01 although it does use considerably less computation time. SS2 does produce a better result in less time for ORTEC01 though.

	SS1		Brucker et al. [16]		Burke et al. [18]	
	Penalty	Time (s)	Penalty	Time (s)	Penalty	Time
ORTEC01	1707	713	-	-	541	12 hours
BCV-1.8.1	268	66	323	136	-	-
BCV-2.46.1	1588	1665	1594	3424	-	-
BCV-3.46.1	3396	5226	3601	2888	-	-
BCV-4.13.1	12	114	18	208	-	-
BCV-5.4.1	135	9	200	16	-	-
BCV-6.13.1	904	207	890	304	-	-
BCV-7.10.1	385	76	396	216	-	-
BCV-8.13.1	148	123	148	224	-	-
BCV-A.12.1	1813	518	3335	944	-	-

Table 5 Comparisons of scatter search with other algorithms

For a final comparison, the variable depth search (VDS) was tested on its own but with a pre-defined maximum run time equal to the average time used by MEH and then SS1. Five repeats were also performed and the best and average results are shown in Table 6.

Instance	VDS with max run time same as used by MEH			VDS with max run time same as used by SS1		
	Best	Average	Time (secs)	Best	Average	Time (secs)
ORTEC01	355	377	3351	360	420	713
BCV-1.8.1	252	260	99	253	258	66
BCV-2.46.1	1572	1592	2560	1574	1588	1665
BCV-3.46.1	3290	3313	10714	3312	3337	5226
BCV-4.13.1	10	11	93	10	10	114
BCV-5.4.1	48	48	27	48	48	9
BCV-6.13.1	768	768	385	768	777	207
BCV-7.10.1	381	438	66	381	412	76
BCV-8.13.1	148	148	219	148	148	123
BCV-A.12.1	1495	1694	929	1734	1865	518

Table 6 VDS with the same max. run times as MEH and SS1

Looking at average results, the variable depth search outperforms MEH on 7 of the 10 instances, is equal on 1 and worse on the other 2. Using best results, VDS is better on 7 out of 10 instances and equal on 3. When compared to SS1 and using identical run times, for the average results, VDS is better on 6 out of 10 instances, equal for 2 and worse on 2. Comparing best results, VDS is better than SS1 on 5 instances, equal on 3 and worse on 2. This indicates that although the scatter search is a successful method, the variable depth search has a slight edge over it on most instances.

## 5 Conclusions

A scatter search has been presented for the nurse rostering problem and tested using benchmark instances. Scatter searches are similar to memetic algorithms except that the random decisions are replaced with intelligently designed rules and solutions may be created

from more than one parent. Applying the approach to the nurse rostering problem required the development of new methods for measuring similarity between solutions, creating solutions from multiple parents and maintaining diversity in the population. For measuring similarity, a simple but effective method was used : counting the number of common shift assignments in rosters. To create new solutions for each generation a ‘democratic’ approach was adopted in which each guiding solution ‘votes’ for the assignments in the new solutions. Experiments were also conducted to examine the trade offs between solution quality and computation time, when the size of the reference sets and the run time of the improvement method (variable depth search or a simple hill climber) were varied. It was discovered that the best trade off was a relatively small reference set combined with using the variable depth search with a maximum execution time of 5 seconds. These settings produced high quality solutions in acceptable computation times on all instances and a new record for a particularly complex instance.

When compared against the heuristic constructive method of Brucker et al. the scatter search found better solutions on all but two instances and using less computational time on all but one. On instance ORTEC01, the scatter search using the variable depth search as the improvement method found a better solution than the hybrid approach of Burke et al., also in less time.

When compared with Burke et al.’s memetic algorithm, the scatter search (with hill climber improvement method) found better solutions for 7 out of 10 instances in less time. This shows that it is an efficient and successful approach. When scatter search was compared to the variable depth search on its own with a maximum run time identical to that used by the scatter search, it produced better results on two instances and equal results on two more.

Although this suggests variable depth search on its own is still the superior method, there is one instance on which the scatter search outperforms and is consistently strong; instance BCV-A.12.1. This is highlighted by the solution for BCV-A.12.1 with penalty 1378 which was found in the first set of experiments. 1378 is a new record and was particularly impressive considering the days of computation that had been used on it previously by other methods; for example the variable depth search during its development and testing. It is not clear why the scatter search should be so suited to this instance. BCV-A.12.1 does contain constraint types which do not appear in the other instances (for example the tutorship constraint), which may be a possible explanation.

Compared to the variable depth search, the scatter search with the hill climber as the improvement method is easier to implement with less potential for introducing bugs. The solution similarity comparison method is simple and intuitive and the solution combination method is also easily understandable. When these subroutines are combined into the overall scatter search, a relatively straightforward yet demonstrably robust and effective approach is produced.

All the test instances, best solutions and source code for the algorithms are all publicly available at the research website (<http://www.cs.nott.ac.uk/~tec/NRP/>).

## **Acknowledgements**

This work was supported by EPSRC grant GR/S31150/01.

## **References**

1. Aarts, E., J. Korst, and W. Michiels, *Simulated Annealing*, in *Search Methodologies. Introductory Tutorials in Optimization and Decision Support Techniques*, E.K. Burke and G. Kendall, Editors. 2005, Springer. pp. 187-210.
2. Abdennadher, S. and H. Schlenker. *Nurse scheduling using constraint logic programming*, in *Proceedings of the Eleventh Conference on Innovative Applications of Artificial Intelligence*. 1999: AAAI Press. pp. 838-843.
3. Aickelin, U. *Genetic Algorithms for Multiple-Choice Problems*. PhD thesis, University of Wales Swansea, 1999.
4. Aickelin, U., E.K. Burke, and J. Li, *An Estimation of Distribution Algorithm with Intelligent Local Search for Rule-based Nurse Rostering*. Journal of the Operational Research Society, 2007(in print).
5. Aickelin, U. and K.A. Dowsland, *Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem*. Journal of Scheduling, 2000. **3**(3): pp. 139-153.
6. Aickelin, U. and K.A. Dowsland, *An Indirect Genetic Algorithm for a Nurse Scheduling Problem*. Computers and Operations Research, 2003. **31**(5): pp. 761-778.
7. Aickelin, U. and J. Li, *An Estimation of Distribution Algorithm for Nurse Scheduling*. Annals of Operations Research, 2007(in print).
8. Aickelin, U. and P. White, *Building Better Nurse Scheduling Algorithms*. Annals of Operations Research, 2004. **128**: pp. 159-177.
9. Bailey, J. and J. Field, *Personnel scheduling with flexshift models*. Journal of operations management 1985. **5**(3): pp. 327-338.
10. Bard, J.F. and H.W. Purnomo, *A column generation-based approach to solve the preference scheduling problem for nurses with downgrading*. Socio-Economic Planning Sciences, 2005. **39**(3): pp. 193-213.
11. Bard, J.F. and H.W. Purnomo, *Preference scheduling for nurses using column generation*. European Journal of Operational Research, 2005. **164**(2): pp. 510-534.
12. Bard, J.F. and H.W. Purnomo, *Cyclic Preference Scheduling of Nurses Using A Lagrangian-Based Heuristic*. Journal of Scheduling, 2007. **10**(1): pp. 5-23.
13. Beddoe, G.R. and S. Petrovic, *Selecting and Weighting Features Using a Genetic Algorithm in a Case-Based Reasoning Approach to Personnel Rostering*. European Journal of Operational Research, 2006. **175**(2): pp. 649-671.
14. Beddoe, G.R. and S. Petrovic, *Enhancing Case-Based Reasoning for Personnel Rostering with Selected Tabu Search Concepts*. Journal of the Operational Research Society, To appear.
15. Bellanti, F., G. Carello, F.D. Croce, and R. Tadei, *A greedy-based neighborhood search approach to a nurse rostering problem*. European Journal of Operational Research, 2004. **153**: pp. 28-40.
16. Brucker, P., E.K. Burke, T. Curtois, R. Qu, and G. Vanden Berghe, *Adaptive Construction of Nurse Schedules: A Shift Sequence Based Approach and New Benchmarks*. Under review, 2006.
17. Burke, E.K., P. Cowling, P. De Causmaecker, and G. Vanden Berghe, *A Memetic Approach to the Nurse Rostering Problem*. Applied Intelligence, 2001. **15**(3): pp. 199-214.
18. Burke, E.K., T. Curtois, G. Post, R. Qu, and B. Veltman, *A Hybrid Heuristic Ordering and Variable Neighbourhood Search for the Nurse Rostering Problem*. European Journal of Operational Research, Accepted for publication, to Appear 2007.
19. Burke, E.K., T. Curtois, R. Qu, and G. Vanden Berge, *A Time-Predefined Variable Depth Search for Nurse Rostering*. 2007, Under review.

20. Burke, E.K., P. De Causmaecker, S. Petrovic, and G. Vanden Berghe, *Variable Neighborhood Search for Nurse Rostering Problems*, in *Metaheuristics: Computer Decision-Making*, M.G.C. Resende and J.P. de Sousa, Editors. 2004, Kluwer. pp. 153-172.
21. Burke, E.K., P. De Causmaecker, and G. Vanden Berghe. *A Hybrid Tabu Search Algorithm for the Nurse Rostering Problem*, in *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 Volume 1585*. B. McKay, et al., Editors. 1999: Springer. pp. 187-194.
22. Burke, E.K., P. De Causmaecker, G. Vanden Berghe, and H. Van Landeghem, *The State of the Art of Nurse Rostering*. *Journal of Scheduling*, 2004. **7**(6): pp. 441 - 499.
23. Burke, E.K., G. Kendall, and E. Soubeiga, *A Tabu-Search Hyperheuristic for Timetabling and Rostering*. *Journal of Heuristics*, 2003. **9**(6): pp. 451 - 470.
24. Cai, X. and K.N. Li, *A genetic algorithm for scheduling staff of mixed skills under multi-criteria*. *European Journal of Operational Research*, 2000. **125**(2): pp. 359-369.
25. Campos, V., F. Glover, M. Laguna, and R. Martí, *An Experimental Evaluation of a Scatter Search for the Linear Ordering Problem*. *Journal of Global Optimization*, 2001. **21**(4): pp. 397-414.
26. Chiarandini, M., A. Schaerf, and F. Tiozzo. *Solving Employee Timetabling Problems with Flexible Workload using Tabu Search*, in *Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT-2000)*. E.K. Burke and W. Erben, Editors. 2000. Konstanz, Germany. pp. 298-302.
27. Chun, A.H.W., S.H.C. Chan, G.P.S. Lam, F.M.F. Tsang, J. Wong, and D.W.M. Yeung. *Nurse Rostering at the Hospital Authority of Hong Kong*, in *Proceedings of the Twelfth Conference on Innovative Applications of Artificial Intelligence*. 2000. pp. 951 - 956.
28. Cung, V.-D., T. Mautor, P. Michelon, and A. Tavares. *A Scatter Search Based Approach for the Quadratic Assignment Problem*, in *Proceedings of IEEE International Conference on Evolutionary Computation*. 1997. pp. 165-169.
29. Darmoni, S.J., A. Fajner, N. Mahé, A. Leforestier, M. Vondracek, O. Stelian, and M. Baldenweck, *Horoplan: computer-assisted nurse scheduling using constraint-based programming* *Journal of the Society for Health Systems*, 1995. **5**: pp. 41-54.
30. Dias, T.M., D.F. Ferber, C.C. de Souza, and A.V. Moura, *Constructing nurse schedules at large hospitals*. *International Transactions in Operational Research*, 2003. **10**(3): pp. 245-265.
31. Dorigo, M., V. Maniezzo, and A. Colorni, *Ant system: optimization by a colony of cooperating agents*. *IEEE Transactions on Systems, Man, and Cybernetics. Part B: Cybernetics*, 1996. **26**(1): pp. 29-41.
32. Dorigo, M. and T. Stützle, *Ant Colony Optimization*. 2004: The MIT Press.
33. Dowsland, K.A., *Nurse scheduling with tabu search and strategic oscillation*. *European Journal of Operational Research*, 1998. **106**(2): pp. 393-407.
34. Dowsland, K.A. and J.M. Thompson, *Solving a nurse scheduling problem with knapsacks, networks and tabu search*. *Journal of the Operational Research Society*, 2000. **51**(7): pp. 825-833.
35. Easton, F.F. and N. Mansour. *A Distributed Genetic Algorithm for Employee Staffing and Scheduling Problems*, in *Proceedings of the 5th International Conference on Genetic Algorithms*. 1993. San Mateo. pp. 360-367.
36. Ernst, A.T., H. Jiang, M. Krishnamoorthy, B. Owens, and D. Sier, *An Annotated Bibliography of Personnel Scheduling and Rostering*. *Annals of Operations Research*, 2004. **127**: pp. 21-144.

37. Ernst, A.T., H. Jiang, M. Krishnamoorthy, and D. Sier, *Staff scheduling and rostering: A review of applications, methods and models*. European Journal of Operational Research, 2004. **153**(1): pp. 3-27.
38. Eveborn, P. and M. Rönnqvist, *Scheduler – A System for Staff Planning*. Annals of Operations Research, 2004. **128**: pp. 21–45.
39. Feo, T.A. and M.G.C. Resende, *Greedy Randomized Adaptive Search Procedures*. Journal of Global Optimization, 1995. **6**(2): pp. 109-133.
40. Fogel, D.B., *Evolutionary Computation: The Fossil Record* 1998: Wiley-IEEE Press.
41. Gendreau, M. and J.-Y. Potvin, *Tabu Search*, in *Search Methodologies. Introductory Tutorials in Optimization and Decision Support Techniques*, E.K. Burke and G. Kendall, Editors. 2005, Springer. pp. 165-186.
42. Glover, F., *A Template for Scatter Search and Path Relinking*, in *Lecture Notes in Computer Science, 1363*, J.K. Hao, et al., Editors. 1998, Springer. pp. 13-54.
43. Glover, F. and M. Laguna, *Tabu Search*. 1997: Kluwer Academic Publishers.
44. Glover, F., M. Laguna, and R. Martí, *Fundamentals of Scatter Search and Path Relinking*. Control and Cybernetics, 2000. **29**(3): pp. 653-684.
45. Glover, F., M. Laguna, and R. Martí, *Scatter Search*, in *Advances in Evolutionary Computing*, A. Ghosh and S. Tsutsui, Editors. 2003, Springer. pp. 519-539.
46. Glover, F., A. Løkketangen, and D.L. Woodruff, *Scatter Search to Generate Diverse MIP Solutions*, in *OR Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*, M. Laguna and J.L. González-Velarde, Editors. 2000, Kluwer. pp. 299-317.
47. Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*. 1989: Addison-Wesley.
48. Greistorfer, P., *A Tabu Scatter Search Metaheuristic for the Arc Routing Problem*. Computers & Industrial Engineering, 2003. **44**: pp. 249–266.
49. Hansen, P. and N. Mladenovic, *Variable Neighborhood Search*, in *Search Methodologies. Introductory Tutorials in Optimization and Decision Support Techniques*, E.K. Burke and G. Kendall, Editors. 2005, Springer. pp. 211-238.
50. Holland, J.H., *Adaptation in Natural and Artificial Systems*. 1992: The MIT Press.
51. Ikegami, A. and A. Niwa, *A Subproblem-centric Model and Approach to the Nurse Scheduling Problem*. Mathematical Programming, 2003. **97**(3): pp. 517-541.
52. Inoue, T., T. Furuhashi, H. Maeda, and M. Takaba, *A Proposal of Combined Method of Evolutionary Algorithm and Heuristics for Nurse Scheduling Support System*. IEEE Transactions on Industrial Electronics, 2003. **50**(5): pp. 833- 838.
53. Jan, A., M. Yamamoto, and A. Ohuchi. *Evolutionary Algorithms for Nurse Scheduling Problem*, in *Proceedings of the 2000 Congress on Evolutionary Computation*. 2000. California, USA: IEEE Press. pp. 196-203.
54. Jaszkiwicz, A., *A metaheuristic approach to multiple objective nurse scheduling*. Foundations of Computing and Decision Sciences, 1997. **22**(3): pp. 169-183.
55. Jaumard, B., F. Semet, and T. Vovor, *A Generalized Linear Programming Model for Nurse Scheduling*. European Journal of Operational Research 1998. **107**(1): pp. 1-18.
56. Kawanaka, H., K. Yamamoto, T. Yoshikawa, T. Shinogi, and S. Tsuruoka. *Genetic algorithm with the constraints for nurse scheduling problem*, in *Proceedings of the 2001 Congress on Evolutionary Computation*. 2001. Seoul, South Korea: IEEE Press. pp. 1123-1130.
57. Kennedy, J. and R. Eberhart. *Particle swarm optimization*, in *Proceedings of the 1995 IEEE International Conference on Neural Networks*. 1995. Perth, Australia. pp. 1942-1948.

58. Kirkpatrick, S., C.D. Gelatt, and M.P. Vecchi, *Optimization by Simulated Annealing*. Science, 1983. **220**(4598): pp. 671-680.
59. Krasnogor, N., W. Hart, and J. Smith, Editors. *Recent Advances in Memetic Algorithms, Studies in Fuzziness and Soft Computing*. 2004, Springer.
60. Krasnogor, N. and J. Smith, *A Tutorial for Competent Memetic Algorithms: Model, Taxonomy and Design Issues*. IEEE Transactions on Evolutionary Computation, 2005. **9**(5): pp. 474-488.
61. Laguna, M. and R. Martí, *Scatter Search. Methodology and Implementation in C*. 2003: Kluwer Academic Publishers.
62. Le, K.N. and D. Landa-Silva, *Multi-objective Nurse Scheduling to Increase Staff Satisfaction*. Under review, 2006.
63. Li, H., A. Lim, and B. Rodrigues. *A Hybrid AI Approach for Nurse Rostering Problem*, in *Proceedings of the 2003 ACM symposium on Applied computing*. 2003. pp. 730-735.
64. Lourenço, H., M. Laguna, and R. Martí, *Assigning Proctors to Exams with Scatter Search*, in *Computing Tools for Modeling, Optimization and Simulation*, M. Laguna and J.L. González-Velarde, Editors. 2000, Springer. pp. 215-228.
65. Maenhout, B. and M. Vanhoucke, *New computational results for the nurse scheduling problem: a scatter search algorithm*, in *Lecture notes in Computer Science, 3906*. 2006, Springer. pp. 159 -170.
66. Mason, A.J. and M.C. Smith. *A Nested Column Generator for solving Rostering Problems with Integer Programming*, in *International Conference on Optimisation: Techniques and Applications*. L. Caccetta, et al., Editors. 1998. Perth, Australia. pp. 827-834.
67. Meisels, A., E. Gudes, and G. Solotorevsky, *Employee Timetabling, Constraint Networks and Knowledge-Based Rules: A Mixed Approach*, in *Selected papers from the First International Conference on Practice and Theory of Automated Timetabling, Springer Lecture Notes in Computer Science Volume 1154*, E. Burke and P. Ross, Editors. 1995. pp. 93-105.
68. Meisels, A. and A. Schaerf, *Modelling and solving employee timetabling problems*. Annals of Mathematics and Artificial Intelligence, 2003. **39**: pp. 41-59.
69. Merkle, D. and M. Middendorf, *Swarm Intelligence*, in *Search Methodologies. Introductory Tutorials in Optimization and Decision Support Techniques*, E.K. Burke and G. Kendall, Editors. 2005, Springer. pp. 401-436.
70. Meyer auf'm Hofe, H., *Solving Rostering Tasks as Constraint Optimization*, in *Selected papers from the Third International Conference on Practice and Theory of Automated Timetabling, Springer Lecture Notes in Computer Science Volume 2079* E. Burke and W. Erben, Editors. 2000. pp. 191-212.
71. Millar, H.H. and M. Kiragu, *Cyclic and non-cyclic scheduling of 12 h shift nurses by network programming* European Journal of Operational Research, 1998. **104**(3): pp. 582-592.
72. Mladenović, N. and P. Hansen, *Variable neighborhood search*. Computers and Operations Research, 1997. **24**(11): pp. 1097-1100.
73. Moscato, P., *On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms*. 1989, Caltech Concurrent Computation Program Report 826, California Institute of Technology.
74. Moz, M. and M.V. Pato, *A genetic algorithm approach to a nurse rerostering problem*. Computers & Operations Research, 2007. **34**: pp. 667–691.

75. Nonobe, K. and T. Ibaraki, *A tabu search approach to the constraint satisfaction problem as a general problem solver*. European Journal of Operational Research, 1998. **106**(2-3): pp. 599-623.
76. Özcan, E. *Memetic Algorithms for Nurse Rostering*, in *The 20th International Symposium on Computer and Information Sciences*. 2005: Springer-Verlag. pp. 482-492.
77. Purnomo, H.W. and J.F. Bard, *Cyclic preference scheduling for nurses using branch and price*. Naval Research Logistics, 2007. **54**(2): pp. 200-220.
78. Sastry, K., D. Goldberg, and G. Kendall, *Genetic Algorithms*, in *Search Methodologies. Introductory Tutorials in Optimization and Decision Support Techniques* E.K. Burke and G. Kendall, Editors. 2005, Springer. pp. 97-126.
79. Tanomaru, J. *Staff scheduling by a genetic algorithm with heuristic operators*, in *Proceedings of the IEEE Conference on Evolutionary Computation*. 1995. pp. 456-461.
80. Thornton, J. and A. Sattar. *Nurse Rostering and Integer Programming Revisited*, in *International Conference on Computational Intelligence and Multimedia Applications*. B. Verma and X. Yao, Editors. 1997. pp. 49-58.
81. Valouxis, C. and E. Housos, *Hybrid optimization techniques for the workshift and rest assignment of nursing personnel*. Artificial Intelligence in Medicine, 2000. **20**: pp. 155-175.
82. Weil, G., K. Heus, P. Francois, and M. Poujade, *Constraint programming for nurse scheduling*. IEEE Engineering in Medicine and Biology Magazine, 1995. **14**(4): pp. 417-422.
83. Wong, G.Y.C. and H.W. Chun. *Nurse Rostering Using Constraint Programming and Meta-level Reasoning*, in *Developments in Applied Artificial Intelligence: 16th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*. 2003. Loughborough, UK: Springer. pp. 712-721.