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

Computer Science Technical Report No. NOTTCS-TR-2007-4

omponent Based Heuristic Search method with Adaptive Perturbations for Hospital Personnel Schedu

Jingpeng Li, Uwe Aickelin and Edmund K. Burke

First released: Nov 2006

A Component Based Heuristic Search method with Adaptive Perturbations for Hospital Personnel Scheduling

Jingpeng Li, Uwe Aickelin and Edmund K. Burke
{jpl, uxa, ekb}@cs.nott.ac.uk
School of Computer Science and Information Technology
The University of Nottingham
Nottingham, NG8 1BB
United Kingdom

Abstract- Nurse rostering is a complex scheduling problem that affects hospital personnel on a daily basis all over the world. This paper presents a new component-based approach with adaptive perturbations, for a nurse scheduling problem arising at a major UK hospital. The main idea behind this technique is to decompose a schedule into its components (i.e. the allocated shift pattern of each nurse), and then mimic a natural evolutionary process on these components to iteratively deliver better schedules. The worthiness of all components in the schedule has to be continuously demonstrated in order for them to remain there. This demonstration employs a dynamic evaluation function which evaluates how well each component contributes towards the final objective. Two perturbation steps are then applied: the first perturbation eliminates a number of components that are deemed not worthy to stay in the current schedule; the second perturbation may also throw out, with a low level of probability, some worthy components. The eliminated components are replenished with new ones using a set of constructive heuristics using local optimality criteria. Computational results using 52 data instances demonstrate the applicability of the proposed approach in solving real-world problems.

Keywords: Nurse Rostering, Constructive Heuristic, Local Search, Adaptive Perturbation

1 Introduction

Employee scheduling has been widely studied for more than 40 years. The following survey papers give an overview of the area: Bradley and Martin, 1990; Ernst et al., 2004a and 2004b. Employee scheduling can be thought of as the problem of assigning employees to shifts or duties over a scheduling period so that certain organizational and personal constraints are satisfied. It involves the construction of a schedule for each employee within an organization in order for a set of tasks to be fulfilled. In the domain of healthcare, this is particularly challenging because of the presence of a range of different staff requirements on different days and shifts. Unlike many other organizations, healthcare institutions work twenty-four hours a day for every single day of the year. Irregular shift work has an effect on the nurses' well being and job satisfaction (Mueller and McCloskey, 1990). The extent to which the staff roster satisfies the staff can impact significantly upon the working environment.

Automatic approaches have significant benefits in saving administrative staff time and also generally improve the quality of the schedules produced. However, until recently, most personnel scheduling problems in hospitals were solved manually (Silvestro and Silvestro, 2000). Scheduling by hand is usually a very time consuming task. Without an automatic tool to generate schedules and to test the quality of a constructed schedule, planners often have to use very straightforward constraints on working time and idle time in the recurring process. Even when hospitals have computerized systems, testing and graphical features are often used but automatic schedule generation features are still not common. Moreover, there is a growing realisation that the automated generation of personnel schedules within healthcare can provide significant benefits and savings. In this paper, we focus on the development of new techniques for automatic nurse rostering systems. A general overview of various approaches for nurse rostering can be found in Sitompul and Randhawa (1990), Cheang et al. (2003) and Burke et al. (2004).

Most real world nurse rostering problems are extremely complex and difficult. Tien and Kamiyama (1982), for example, say nurse rostering is more complex than the travelling salesman problem due to the additional constraint of total number of working days within the scheduling period. Since the 1960's, many papers have been published on various aspects of nurse rostering. Early papers (Warner and Prawda, 1972; Miller, Pierskalla and Rath, 1976) attempted to solve the problem by using mathematical programming models. However, computational difficulties exist with these approaches due to the enormous size of the search space. In addition, for most real problems, the goal of finding the 'optimal' solution is not only completely infeasible, but also largely meaningless. Hospital administrators normally want to quickly create a high quality schedule that satisfies all hard constraints and as many soft constraints as possible.

The above observations have led to a number of other attempts to solve real world nurse rostering problems. Several heuristic methods have been developed (e.g., Blau, 1985; Anzai and Miura, 1987). In the 1980's and later, artificial intelligence methods for nurse rostering, such as constraint programming (Meyer auf'm Hofe, 2001), expert systems (Chen and Yeung, 1993) and knowledge based systems (Beddoe and Petrovic, 2006) were investigated with some success. In the 1990's and later, many of the papers tackle the problem with meta-heuristic methods, which include simulated annealing (Brusco and Jacobs, 1995), variable neighbourhood search (Burke et al., 2004), tabu search (Dowsland 1998; Burke, De Causmaecker and Vanden Berghe, 1999) and evolutionary methods (Burke et al., 2001; Kawanaka et al., 2001). In very recent years, there have been increasing interests in the study of mathematical programming based heuristics (Bard and Purnomo, 2006 and 2007; Beliën and Demeulemeester, 2006) and the study of hyper-heuristics (Burke et al., 2003; Ross, 2005) for the problem (Burke, Kendall and Soubeiga, 2003; Özcan 2005).

This paper tackles a nurse rostering problem arising at a major UK hospital (Aickelin and Dowsland, 2000; Dowsland and Thompson, 2000). Its target is to create weekly schedules for wards of nurses by assigning each nurse one of a number of predefined shift patterns in the most efficient way. Besides the traditional approach of Integer Linear Programming (Dowsland and Thompson, 2000), a number of meta-heuristic approaches have been explored for this problem. For example, in (Aickelin and Dowsland, 2000 and 2003; Aickelin and White, 2004) various approaches based on genetic algorithms are presented. In (Li and Aickelin, 2004) an approach based on a learning classifier system is investigated. In (Burke, Kendall and Soubeiga, 2003) a tabu search hyperheuristic is introduced, and in (Aickelin and Li, 2006) an estimation of distribution algorithm is described. In this paper we will report a new component-based heuristic search approach with adaptive perturbations, which implements optimization on the components within single schedules. This approach combines the features of iterative improvement and constructive perturbation with the ability to avoid getting stuck at local minima.

The framework of our new algorithm is an iterative improvement heuristic, in which the steps of Evaluation, Perturbation-I, Perturbation-II and Reconstruction are executed in a loop until a stopping condition is reached. In the Evaluation step, a current complete schedule is first decomposed into assignments for individual nurses, and then the assignment for each nurse is evaluated by a function based upon both hard constraints and soft constraints. In the Perturbation-I step, some nurses are marked as 'rescheduled' and their assignments are removed from the schedule according to the evaluating values of their assignments. In the Perturbation-II step, each remaining nurse still has a small chance to be rescheduled, disregarding the evaluating value of his/her assignment. Finally, in the Reconstruction step, a refined greedy heuristic is designed to repair a broken solution and the obtained complete solution is fed into the Evaluation step again to repeat the loop.

Our proposed method belongs to the general class of local search. In particular, it is somewhat similar to the Iterated Local Search algorithm (Lourenco, Martin and Stutzle, 2002): they include a solution perturbation phase and an improvement phase. However, they differ in the way in which these two phases are implemented: The purpose of perturbation in Iterated Local Search is to transform one complete solution into another complete solution. This serves as the starting point for the local heuristics which follow. However, the aim of the perturbation in our method is to transform one complete solution into a partial solution which is then fed into the reconstruction heuristics for repair.

The rest of this paper is organized as follows. Section 2 gives an overview of the nurse rostering problem, and introduces the general framework of our methodology. Section 3 presents our algorithm for nurse rostering. Benchmark results using real-world data sets collected from a major UK hospital are presented in section 4. Concluding remarks are in section 5.

2 Preliminaries

2.1 The Nurse Rostering Problem

The nurse rostering problem tackled in this paper is to create weekly schedules for wards of up to 30 nurses at a large UK hospital. These schedules have to meet the demand for a minimum number of nurses of different grades on each shift, whilst being seen to be fair by the staff concerned and satisfying working contracts. The fairness objective is achieved by meeting as many of the nurses' requests as possible and considering historical information (e.g. previous weekends) to ensure that unsatisfied requests and unpopular shifts are evenly distributed. In our model, the day is partitioned into three shifts: two types of day shift known as 'earlies' and 'lates', and a longer night shift. Due to hospital policy, a nurse would normally work either days or nights in a given week (but not both), and because of the difference in shift length, a full week's work would normally include more days than nights. However, some special nurses work other mixtures and the problem can hence not simply be decomposed into days and nights.

However, as described in Dowsland and Thompson (2000), the problem can be split into three independent stages. The first uses a knapsack model to ensure that there are sufficient nurses to meet the covering constraints. If not, additional nurses (agency staff) are allocated to the ward, so that the problem tackled in the second phase is always feasible. The second stage is the most difficult and involves allocating the actual days or nights a nurse works. Once this has been decided, a third phase uses a network flow model (Ahuja et al., 1993) to allocate those on days to 'earlies' and 'lates'. Since stages 1 and 3 can be solved quickly, this paper is only concerned with the highly constrained second step.

The days or nights that a nurse could work in one week define the set of feasible weekly work patterns (i.e. shift patterns) for that nurse. Each shift pattern can be represented as a 0-1 vector with 14 elements, where the first 7 elements represent the 7 days of the week and the last 7 elements the corresponding 7 nights of the week. A '1' or '0' in the vector denotes a scheduled day/night "worked" or "not worked". For example, (1111100 0000000) would be a pattern where the nurse works the first 5 days and no nights. In total, the hospital allows just under 500 such shift patterns. A specific nurse's contract usually allows 50 to 100 of these. Depending on the nurses' preferences, the recent history of patterns worked, and the overall attractiveness of the pattern, a preference cost is allocated to each nurse-shift pattern pair. These values were set in close consultation with the hospital and range from 0 (perfect) to 100 (unacceptable), with a bias to lower values. Due to the introduction of these preference costs which takes into account historic information (e.g. weekends worked in previous weeks), we are able to reduce the planning horizon from the original five weeks to the current one week without affecting solution quality. Further details about the problem can be found in Dowsland (1998).

The problem can be formulated as follows.

Decision variables:
x_{ij} = 1 if nurse i works shift pattern j, 0 otherwise.

Parameters:
m = Number of possible shift patterns;
n = Number of nurses;
g = Number of grades;
a_{jk} = 1 if shift pattern j covers period k, 0 otherwise;
q_{is} = 1 if nurse i is of grade s or higher, 0 otherwise;
p_{ij} = Preference cost of nurse i working shift pattern j;
R_{ks} = Demand for nurses with grade s on period k;
A(i) = Set of feasible shift patterns for nurse i.

Target function:

$$\text{Min} \sum_{i=1}^{n} \sum_{j \in A(i)} p_{ij} x_{ij} \,. \qquad (1)$$

Subject to:

$$\sum_{j \in A(i)} x_{ij} = 1, \ \forall i \in \{1,\ldots,n\}, \qquad (2)$$

$$\sum_{j \in A(i)} \sum_{i=1}^{n} q_{is} a_{jk} x_{ij} \geq R_{ks}, \ \forall k \in \{1,\ldots,14\}, s \in \{1,\ldots,g\}. \qquad (3)$$

The constraints outlined in (2) ensure that every nurse works exactly one shift pattern from his/her feasible set. The constraints represented by (3) ensure that the demand for nurses is fulfilled for every grade on every day and night and in line with hospital policy more nurses than necessary may work during any given period. In practise, there is an acute shortage of nurses and actual overstaffing is very rare. Note that the definition of q_{is} allows that higher graded nurses can substitute those at lower grades if necessary. This problem can be regarded as a multiple-choice set-covering problem. The sets are given by the shift pattern vectors and the objective is to minimize the cost of the sets needed to provide sufficient cover for each shift at each grade. The constraints described in (2) enforce the choice of exactly one pattern (set) from the alternatives available for each nurse.

2.2 General Description of the Component Based Heuristic Method with Adaptive Perturbation (CHAP)

The basic methodology iteratively operates the steps of Evaluation, Perturbation-I, Perturbation-II and Reconstruction in a loop on one solution (see the pseudo code presented in Figure 1). At the beginning of the loop, an Initialization step is used to obtain a starting solution and initialize some input parameters (e.g. stopping conditions). In the Evaluation step, the fitness (i.e. the degree of suitability) of each component in the current solution is evaluated under an evaluation function. Then, the fitness measure is used probabilistically to select components to be eliminated in the Perturbation-I step. Components with high fitness have a lower probability of being eliminated. Furthermore, to escape local minima in the solution space, capabilities for uphill moves must be incorporated. This is carried out in the Perturbation-II step by probabilistically eliminating even some superior components of the solution in a totally random manner.

The resulting partial solutions are then fed into the Reconstruction step, which implements application specific heuristics to derive a new and complete solution from partial solutions. Throughout these iterations, the best solution is retained and finally returned as the final solution. This algorithm uses a greedy search strategy to achieve improvement through iterative perturbation and reconstruction.

```
CHAP ( )
{
    t=0;
    Create an initial solution S(0) with an associate cost C(0);
    C_best= C(0);
    While (stopping conditions not reached) {
        /* Decompose the solution into its component (i.e. shift
           Patterns of individual nurses) */
        S(t)={s_1, s_2,..., s_n};
        /* The Evaluation step
        Use an evaluation function to assign each component a score;
        /* The Perturbation-I step
        Eliminate some well-arranged components from S(t);
        Obtain an incomplete solution S'(t);
        /* The Perturbation-II step
        Randomly eliminate some components from S'(t);
        /* The Reconstruction step
        Add new components into S'(t) to make it complete;
        S(t)=S'(t);
        If (C(t) is better than C_best) C_best=C(t);
        t = t+1;
    }
    Return the best solution with the cost C_best;
}
```

Figure 1: The pseudo code of the basic algorithm.

In summary, our methodology differs from some other local search methods such as simulated annealing (Kirkpatrick, Gelatt and Vecchi, 1983) and tabu search (Glover, 1989) in the way that it does not follow one trajectory in the search space. By systematically eliminating components of a solution and then replenishing with new components, this algorithm essentially employs a long sequence of moves between iterations, thus permitting more complex and more distant changes between successive solutions. This feature means that our method has the ability to jump quite easily out of local minima. Furthermore, unlike population-based evolutionary algorithms which need to maintain a number of solutions as parents for offspring propagation in each generation, this method operates on a single solution at a time. Thus, it eliminates the extra CPU-time needed to maintain a set of solutions.

3 A Component Based Heuristic procedure with Adaptive Perturbation for Nurse Rostering

The basic idea behind the method is to determine, for each current schedule, the fitness of shift patterns assigned to individual nurses. The process keeps the shift patterns of some nurses that are well chosen (having high fitness values) in the current schedule and tries to replace the shift patterns of other nurses that have low fitness values. To enable the algorithm to execute iteratively, at each iteration, a randomly-produced threshold (in the range [0,1]) is generated, and all shift patterns whose fitness values exceed the threshold are labelled as "good patterns" and survive in the current schedule. The remaining shift patterns are labelled as "bad patterns" and do not survive (become extinct). The fitness value therefore corresponds to the survival chance of a shift pattern assigned to a specific nurse. The "bad" shift patterns are removed from the current schedule and the corresponding nurses are released, waiting for their new assignments by a constructive heuristic. Following this, the above steps are iterated. Thus the global scheduling procedure is based on iterative improvement, while an iterative constructive process is performed within.

3.1 Initialization

In this step, an initial solution is generated to serve as a seed for its iterative improvement. It is well known that for most meta-heuristic algorithms, the initialization strategy can have a significant influence on performance. Thus, normally, a significant effort will be made to generate a starting point that is as good as possible. For nurse rostering, there are a number of heuristic techniques that can be applied to produce good starting solutions.

For our methodology, due to the fact that the replacement rate in its first iteration is relatively high, the performance is generally independent of the quality of the initial solution. However, if the seed is already a relatively good solution, the overall computation time will decrease. Since the major purpose of this paper is to demonstrate the performance and general applicability of the proposed methodology, we deliberately generate an extremely poor initial solution by randomly assigning a shift pattern to each nurse. The steps described in section 3.2 to 3.5 are executed in sequence in a loop until a stopping condition (i.e. solution quality or the maximum number of iteration) is reached.

3.2 Evaluation

In this step, the fitness of individual nurses' assignments, based on complete schedules, is evaluated. The evaluation function should be normalized and hence can be formulated as

$$F(E_i) = \sum_{k=1}^{2} w_k f_k(E_i), \quad \forall i \in \{1,\dots,n\}, \tag{4}$$

subject to

$$\sum_{k=1}^{2} w_k = 1. \tag{5}$$

Where E_i are the shift pattern assigned to the i-th nurse, n is the number of nurses, $f_1(E_i)$ and $f_2(E_i)$ is the contribution of E_i towards the preference and the feasibility aspect of the solution respectively.

$f_1(E_i)$ evaluates the shift pattern assigned to a nurse in terms of the degree to which it satisfies the soft constraints (i.e. this nurse's preference on his/her assigned shift pattern). It can be formulated as

$$f_1(E_i) = \frac{p_{max} - p_{ij}}{p_{max} - p_{min}}, \quad \forall i \in \{1,\dots,n\}, \tag{6}$$

where p_{ij} is the preference cost of nurse i working shift pattern j and p_{max} and p_{min} are the maximum and minimum cost values among the shift patterns of all nurses on the current schedule, respectively.

$f_2(E_i)$ evaluates how far the shift pattern assigned to a nurse satisfies the hard constraints (i.e. coverage requirement and grade demands). This can be formulated as

$$f_2(E_i) = \frac{c_{ij} - c_{min}}{c_{max} - c_{min}}, \quad \forall i \in \{1,\dots,n\}, \tag{7}$$

where c_{ij} is the coverage contribution of nurse i working shift pattern j and c_{max} and c_{min} are the maximum and minimum coverage contribution values among the shift patterns of all nurses on the current schedule, respectively.

In a current schedule, the coverage contribution of each nurse's shift pattern is its contribution to the cover of all three grades, which can be calculated as the sum of grade one, two and three covered shifts that would become uncovered if the nurse does not work on this shift pattern. Therefore, we formulate c_{ij} as

$$c_{ij} = \sum_{s=1}^{3} q_{is} \left(\sum_{k=1}^{14} a_{jk} d_{ks}\right), \tag{8}$$

Where $q_{is} = 1$ if nurse i is of grade s or higher, 0 otherwise;

$a_{jk} = 1$ if shift pattern j covers period k, 0 otherwise;

$d_{ks} = 1$ if there is a shortage of nurses during period k of grade s (i.e. the coverage value without considering shift pattern j is smaller than demand R_{ks}), 0 otherwise.

3.3 Perturbation-I

This step is to determine whether the i-th nurses' assignment (denoted as E_i, $\forall i \in \{1,\dots,n\}$) should be retained for the next iteration or whether it should be eliminated and the nurse placed in the queue waiting for the next rescheduling. This is done by comparing his/her assignment fitness $F(E_i)$ to a random number r_s generated for each iteration in the range [0,1]. If $F(E_i) \leq r_s$, then E_i will be removed from the current schedule; otherwise E_i will survive in its present position. The days and nights that the nurses' shift pattern covers are then released and updated for the next Reconstruction step (see below). By using this step, an assignment E_i with a larger fitness value $F(E_i)$ has a proportionally higher probability of survival in the current schedule. This mechanism performs in a similar way to roulette wheel selection in genetic algorithms.

3.4 Perturbation-II

Following the Perturbation-I step, the shift pattern of each remaining nurse still has a chance to be eliminated from the partial schedule at a given rate of r_m. The days and nights that an eliminated shift pattern covers are then released for the next Reconstruction step. As usual for mutation operators, compared with the elimination rate in the Perturbation-I step, the rate here should be relatively smaller to facilitate convergence. Otherwise, there will be no bias in the sampling, leading to a random restart type algorithm. From a series of experiments we found that $r_m \leq 5.0\%$ yields good results and hence is the value adopted by us for our experiments. This process is analogous to the mutation operator in a genetic algorithm. Note that our method uses its Perturbation-II step to eliminate some fitter components and thus generate a new diversified solution indirectly.

3.5 Reconstruction

The Reconstruction step takes a partial schedule as the input, and produces a complete schedule as the output. Since the new schedule is based on iterative improvement from the previous schedule, all shift assignments in the partial schedule should remain unchanged. Therefore, the Reconstruction task is reduced to assigning shift patterns to all unscheduled nurses to complete a partial solution.

Based on the domain knowledge of nurse rostering, there are many rules that can be used to build schedules. For example, Aickelin and Dowsland (2003) introduce three building rules: a 'Cover' rule, a 'Contribution' rule and a 'Combined' rule. Since the last two rules are quite similar, in this paper we only apply the 'Cover rule and the 'Combined' rule to fulfil the Reconstruction task.

The 'Cover' rule is designed to achieve the feasibility of the schedule by assigning each unscheduled nurse the shift pattern that covers the most number of uncovered shifts. For instance, assume that a shift pattern covers Monday to Friday night shifts. Further assume that the current requirements for the night shifts from Monday to Sunday are as follows: (-4, 0, +1, -3, -1, -2, 0), where negative symbol means undercover and positive means over-cover. The given shift pattern hence has a cover value of 3 as it covers the night shifts of Monday, Thursday and Friday. Note that for nurses of grade s, this rule only counts the shifts requiring grade s nurses as long as there is a single uncovered shift for this grade. If all shifts of grade s are covered, shifts of grade (s-1) are counted. This operation is necessary as otherwise higher graded nurses might fill lower graded demand first, leaving the higher graded demand might unmet at all.

The 'Combined' rule is designed to achieve a balance between solution quality and feasibility by going through the entire set of feasible shift patterns for a nurse and assigning each one a score. The one with the highest (i.e. best) score is chosen. If there is more than one shift pattern with the best score, the first such shift pattern is chosen. The score of a shift pattern is calculated as the weighted sum of the nurse's preference cost p_{ij} for that particular shift pattern and its contribution to the cover of all three grades. The latter is measured as a weighted sum of grade one, two and three uncovered shifts that would be covered if the nurse worked this shift pattern, i.e. the reduction in shortfall. More precisely and using the same notation as before, the score S_{ij} of shift pattern j for nurse i is calculated as

$$S_{ij} = w_p(100 - p_{ij}) + \sum_{s=1}^{3} w_s q_{is} \left(\sum_{k=1}^{14} a_{jk} e_{ks}\right), \qquad (9)$$

where w_p is the weight of the nurse's preference cost p_{ij} for the shift pattern and w_s is the weight of covering an uncovered shift of grade s. q_{is} is 1 if nurse i is of grade s or higher, 0 otherwise. a_{jk} is 1 if shift pattern j covers day k, 0 otherwise. e_{ks} is the number of nurses needed to at least satisfy the demand R_{ks} if there are still nurses in shortage during period k of grade s, 0 otherwise. $(100-p_{ij})$ must be used in the score, as higher p_{ij} values are worse and the maximum for p_{ij} is 100.

Using the above two rules at the rates of p_1 and p_2 respectively, the Reconstruction step assigns shift patterns to all unscheduled nurses until the broken solution is complete. In addition, to avoid stagnation at local optima, randomness needs to be introduced into the Reconstruction steps. This is achieved by allowing each unscheduled nurse to have an additional small rate p_3 to be scheduled by a randomly-selected shift pattern. Note that the sum of p_1, p_2 and p_3 should be 1. Also note that because we solve the problem without relying on any prior knowledge about which nurses should be scheduled earlier and which nurses later, the indexing order of nurses given in the original data set will be applied throughout the Reconstruction step.

After a broken solution is repaired, the fitness of this complete solution has to be calculated. Unfortunately, due to the highly-constrained nature of the problem, feasibility cannot be guaranteed. Hence, the following penalty function approach is used to evaluate the solutions obtained

$$\mathrm{Min} \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}x_{ij} + w_{demand}\sum_{k=1}^{14}\sum_{s=1}^{g} \max\left[R_{ks} - \sum_{i=1}^{n}\sum_{j=1}^{m} q_{is}a_{jk}x_{ij};0\right], \qquad (10)$$

where constant w_{demand} is the penalty per uncovered shifts in the solution, and a "max" function is used due to the penalization of undercovering.

4 Computational Results

This section describes the computational experiments used to test our proposed algorithm. For all experiments, 52 real data sets (as provided by the hospital) are available. Each data set consists of one week's requirements (i.e. 14 time periods) for all shift and grade combinations and a list of nurses available together with their preference costs p_{ij} and qualifications. Typically, there will be between 20 and 30 nurses per ward, 3 grade-bands and 411 different shift patterns. They are moderately sized problems compared to other problems reported in the literature (Burke et al., 2004). The data was collected from three wards over a period of several months and covers a range of scheduling situations, e.g. some data instances have very few feasible solutions whilst others have multiple optima. A zip file containing all these 52 instances is available to download at http://www.cs.nott.ac.uk/~jpl/Nurse_Data/NurseData.zip.

4.1 Algorithm Details

Table 1 lists detailed computational results of various approaches over 52 instances. The results listed in Table 1 are based on 20 runs with different random seeds. The second last row (headed 'Av.') contains the mean values of all columns, and the last row (headed '%') shows the relative percentage deviation values of the above mean values to the optimal solution values. When computing the mean, a censored cost value of 255 has been used if an algorithm fails to find a feasible solution (denoted as N/A). The following notations are employed in the table:

- IP: optimal or best-known solutions found by XPRESS MP, a commercial integer programming solver (Dowsland and Thompson, 2000);

- GA-1: best result out of 20 runs of a basic genetic algorithm (Aickelin and White, 2004).
- GA-2: best result out of 20 runs of an adaptive GA, which is the same as the basic genetic algorithm revision, but it also tries to self-learn good parameters during the runtime starting from the values given below (Aickelin and White, 2004).
- GA-3: best result out of 20 runs of a multi-population genetic algorithm, which is the same as the adaptive one, but also features competing sub-populations (Aickelin and White, 2004).
- GA-4: best result out of 20 runs of the hill-climbing genetic algorithm, which is the same as the multi-population genetic algorithm, but it also includes a local search in the form of a hill-climber around the current best solution (Aickelin and White, 2004).
- GA-5: best result out of 20 runs of an indirect genetic algorithm, which maps the constraint solution space into an unconstrained space, then searches within that new space and eventually translates solutions back into the original space (Aickelin and Dowsland, 2003). Up to four different rules and a hill-climber are used in this algorithm.
- EDA: best result out of 20 runs of an estimation of distribution algorithm (Aickelin and Li, 2006);
- LCS: best result out of 20 runs of a Learning Classifier System (Li and Aickelin, 2004);
- Con-heu: best result out of 20 runs of our method without the two steps of perturbation;
- CHAP: our full Component based Heuristic method with both Adaptive Perturbation steps;
- Best: best result out of 20 runs of CHAP;
- Mean: average result of 20 runs of CHAP;
- Inf: number of runs terminating with the best solution being infeasible;
- #: number of runs terminating with the best solution being optimal;
- ≤3: number of runs terminating with the best solution being within three cost units of the optimum. The value of three units was chosen as it corresponds to the penalty cost of violating the least important level of requests in the original formulation. Thus, these solutions are still acceptable to the hospital.

Set	IP	GA	GA	GA	GA	GA	EDA	LCS	Con	CHAP (20 runs)				
		-1	-2	-3	-4	-5			-heu	Best	Mean	Inf	#	≤3
01	8	9	9	8	8	8	8	9	31	8	8.0	0	20	20
02	49	57	57	50	50	51	56	60	100	49	54.9	0	2	3
03	50	51	51	50	50	51	50	68	94	50	51.9	0	12	17
04	17	17	17	17	17	17	17	17	20	17	17.0	0	20	20
05	11	12	11	11	11	11	11	15	22	11	11.5	0	19	19
06	2	7	7	2	2	2	2	2	20	2	2.1	0	18	20
07	11	N/A	N/A	11	13	12	14	31	45	11	11.5	0	12	20
08	14	18	18	15	14	15	15	43	41	14	16.0	0	10	15
09	3	N/A	N/A	3	3	4	14	17	N/A	3	8.5	0	12	12
10	2	6	6	4	2	3	2	5	13	3	3.6	0	0	20
11	2	4	4	2	2	2	2	2	N/A	2	2.0	0	20	20
12	2	14	14	2	2	2	3	4	N/A	2	2.4	0	15	19
13	2	3	3	2	2	2	3	5	103	2	2.3	0	14	20
14	3	4	4	3	3	3	4	17	21	3	19.2	0	3	5
15	3	6	6	3	3	3	4	5	5	3	3.0	0	20	20
16	37	40	40	38	38	39	38	38	159	37	37.2	0	16	20
17	9	12	12	9	9	10	9	22	N/A	9	9.2	0	18	20
18	18	19	19	19	19	18	19	33	125	18	18.1	0	19	20

19	1	5	5	1	1	1	10	32	N/A	1	1.6	0	11	20
20	7	10	10	8	8	7	7	7	36	7	14.2	0	8	8
21	0	7	7	0	0	0	1	6	23	0	0.1	0	18	20
22	25	43	35	26	25	25	26	38	150	25	26.9	0	6	16
23	0	8	8	0	0	0	1	3	N/A	0	0.1	0	19	20
24	1	4	3	1	1	1	1	1	N/A	1	1.0	0	20	20
25	0	6	5	0	0	0	0	0	4	0	1.1	0	15	20
26	48	N/A	N/A	48	48	48	52	93	148	48	68.6	0	8	16
27	2	17	17	2	2	4	28	19	N/A	3	17.7	0	0	2
28	63	66	66	63	63	64	65	67	N/A	63	63.3	0	11	20
29	15	20	20	141	17	15	109	56	N/A	15	62.4	1	9	11
30	35	44	44	42	35	38	38	41	97	35	43.3	0	5	5
31	62	N/A	284	166	95	65	159	123	N/A	66	69.5	0	0	0
32	40	51	51	99	41	42	43	42	N/A	40	45.7	0	8	15
33	10	N/A	N/A	10	12	12	11	15	N/A	11	12.0	0	0	18
34	38	42	42	48	40	39	41	70	N/A	38	42.7	0	5	14
35	35	36	36	35	35	36	46	64	N/A	36	43.5	0	0	2
36	32	N/A	36	41	33	32	45	54	198	32	41.7	0	4	5
37	5	8	8	5	5	5	7	12	62	6	7.0	0	0	16
38	13	N/A	N/A	14	16	15	25	30	121	14	46.5	0	0	10
39	5	9	8	5	5	5	8	13	118	5	5.9	0	5	20
40	7	14	10	8	8	7	8	15	26	7	8.2	0	18	18
41	54	N/A	65	54	54	55	55	57	121	54	54.2	0	18	20
42	38	41	41	38	38	39	41	80	51	40	41.1	0	0	16
43	22	24	24	39	24	23	23	58	N/A	22	23.6	0	16	17
44	19	36	36	19	48	25	24	34	N/A	19	28.7	0	1	4
45	3	N/A	9	3	3	3	6	15	111	3	4.5	0	4	19
46	3	17	10	3	6	6	7	28	N/A	3	5.8	0	2	13
47	3	N/A	5	4	3	3	3	3	N/A	3	3.0	0	20	20
48	4	9	9	6	4	4	5	18	N/A	5	12.9	0	0	5
49	27	36	36	30	29	30	30	37	N/A	27	38.3	0	1	2
50	107	N/A	N/A	211	110	110	109	110	N/A	107	107.5	0	12	20
51	74	N/A	N/A	N/A	75	74	171	125	N/A	89	180.9	3	0	0
52	58	N/A	N/A	N/A	75	58	67	85	N/A	58	85.7	1	3	4
Av.	21.1	79.8	65.0	37.1	23.2	22.0	29.7	35.5	157.4	21.7	28.6	0.1	9.6	14.4
%	0	278	208	76	10	4	41	68	646	2.7	35.5			

Table 1: Comparison of results by various approaches over 52 instances.

For all data instances, we used the following set of fixed parameters in our experiments:

- Stopping criterion: a maximum iteration of 50,000, or an optimal/best-known solution has been found;
- Rate of Perturbation-II in Section 3.4: $r_m = 0.05$.
- Rates of Reconstruction in Section 3.5: $p_1 = 0.80$, $p_2 = 0.18$, $p_3 = 0.02$;
- Weight set in formula (9): $w_p = 1$, $w_1 = 8$, $w_2 = 2$ and $w_3 = 1$;
- Penalty weight in fitness function (10): $w_{demand} = 200$;

Note that some parameter values (i.e. the maximum number of iterations, r_m, p_1, p_2 and p_3) are based on our experience and intuition and thus we cannot prove they are the best for each instance. The rest of the values (i.e. w_p, w_1, w_2, w_3 and w_{demand}) are the same as those used in previous papers solving the same 52 instances, and we are continuing to use them for consistency.

Our method was coded in Java 2, and all experiments were undertaken on a Pentium 4 2.1GHz machine under Windows XP. To test the robustness of the proposed algorithm, each data instance was run twenty times by fixing the above parameters and varying the pseudo random number seed at the beginning. The execution time per run and per data instance varies from several milliseconds to 20 seconds depending on the difficulty of the individual data instance. Table 2 lists the average runtimes of various approaches over the same 52 instances: the first six (i.e. IP, GA-1, GA-2, GA-3, GA-4 and GA-5) were run on a different Pentium III PC, while the following two (i.e. EDA and LCS) on a similar Pentium 4 2.0GHz PC. Obviously, the IP is much slower than any of the above meta-heuristics. Among these meta-heuristic methods, our algorithm takes no more time although an accurate comparison in terms of runtime is difficult due to the different environments (i.e. machines, compilers and programming languages) in use. For example, the genetic algorithms are coded in C and the EDA is coded in C++. The comparison in terms of the number of evaluations is also difficult because the other algorithms evaluate each candidate solution as a whole, while our algorithm evaluates partial solutions as well.

	IP	GA-1	GA-2	GA-3	GA-4	GA-5	EDA	LCS	CHAP
Time (sec)	>24hours	19	23	13	15	12	23	45	12

Table 2: Comparison of the average runtime of various approaches.

4.2 Analysis of Results

The results of all the approaches in Table 1 are obtained by using the same 52 benchmark test instances, with the bold figure representing the optimal solution found by a commercial software package. Compared with the results of the mathematical programming approach which can take up to 24 hours runtime (shown in the 'IP' column), our results (shown in the 'Best' column) are only 2.7% more expensive on average but they are all achieved within 20 seconds. Compared with the best results of various meta-heuristic approaches, in general the CHAP results are slightly better than those of the best-performing indirect genetic algorithm (with a relative percentage deviation value of 4%) and are much better than the others (with deviation values from 10% to 278%).

Since our proposed methodology uses a 'Cover' rule and a 'Combined' rule in its Reconstruction step for schedule repairing, it may be interesting to know if the good performance of our algorithm is mainly due to these two delicate building rules. To clarify this, we performed an additional set of experiments by skipping the two perturbation steps, i.e. only implementing the Reconstruction step to build a schedule from an empty solution. This method does not yield a single feasible solution for 24 instances, as the 'Con-heu' column shows. This underlines the difficulty of this problem, and most importantly it underlines the key roles played by the two elimination steps in our full methodology, as the Reconstruction step alone is not capable of solving the problem.

Figures 2 and 3 show the results of our method and the best indirect genetic algorithm graphically in more detail. The bars above the y-axis represent solution quality out of 20 runs: the black bars show the number of optimal solutions found (i.e. the value of '#' in Table 1), and the dotted bars represent the number of good feasible solutions which are within 3 cost units of their optimal solutions (i.e. the value of '≤3' in Table 1). The bars below the y-axis represent the number of times the algorithm failed to find a feasible solution in these 20 runs (i.e. the value of 'Inf' in Table 1). Hence, the less the area below the y-axis and the more above, the better the algorithm's performance. Note that 'missing' bars mean that, in 20 runs, feasible solutions are obtained at least once, but none of them are optimal or of good quality (within 3 units of optimal values).

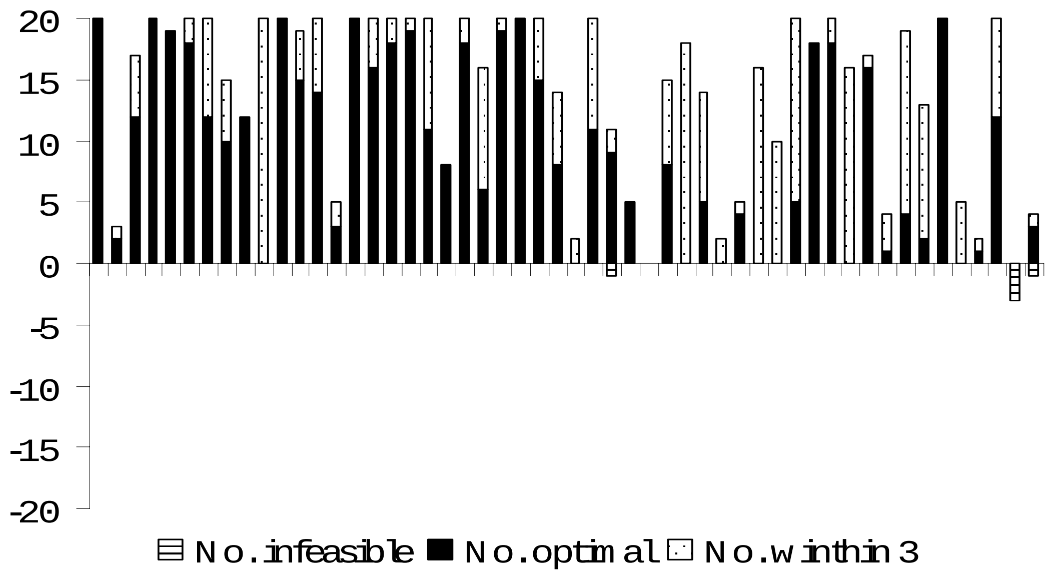

Figure 2: Results from CHAP.

Figure 2 shows that for CHAP, 21 out of 52 data instances are solved well (i.e. with 100% solutions being within 3 units of optimal values), 42 instances are solved optimally at least once, and overall there are 5 infeasible solutions for 3 instances. For the best indirect genetic algorithm (shown in figure 3), the results are slightly worse: 15 data instances are solved well, 28 are solved to optimality at least once, and in total there are 56 infeasible solutions for 6 data instances.

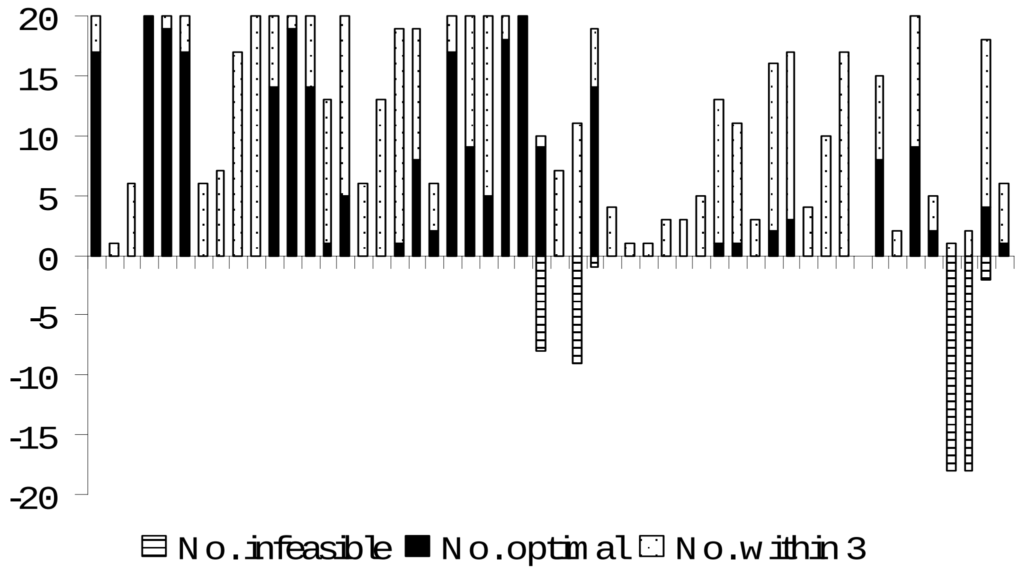

Figure 3: Results of the best indirect genetic algorithm (i.e. GA-5).

Figure 4 shows a summary of Table 1 in graphical format and gives an overall comparison of performance of the other approaches with our proposed methodology. The best results for these instances are obtained by the IP software, and in general, our approach performs better than the previous best-performing approach. The basic genetic algorithm (i.e. GA-1), the adaptive genetic algorithm (i.e. GA-2), the multi-population genetic algorithm (i.e. GA-3) and even the hill-climbing genetic algorithm (i.e. GA-4) which includes multiple populations and an elaborate local search are all significantly outperformed in terms of feasibility, best and average results.

The other three approaches (i.e. the GA-5, the EDA and the LCS) belong to the class of indirect approaches, in which a set of heuristic rules, including the 'Cover' rule and the 'Combined' rule used in our approach, is used for schedule building. Compared with the EDA and the LCS, our new approach performs much better in terms of the best and average results, and slightly worse in terms of feasibility. Compared with the GA-5 which performs best among all the heuristic algorithms, our approach performs better in all aspects of feasibility (99% vs. 95%), best results (21.7 versus 22.0) and average results (28.6 vs. 35.6). In addition, it is worth mentioning that the GA-5 uses the best possible order of the nurses (which, of course, has to be found) for the greedy heuristic to build a schedule, while our algorithm only uses a fixed indexing ordering given in the original data sets.

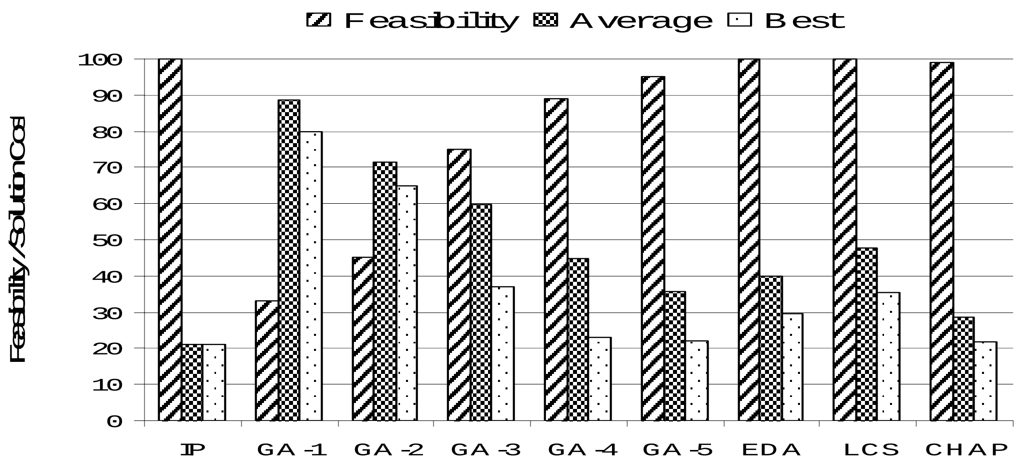

Figure 4: Summary results of various search algorithms.

5 Conclusions

This paper presents a new approach to address the hospital personnel scheduling problem. The major idea behind this method is to decompose a solution into components, and then to mimic a natural evolutionary process on these components to make iterative improvements in each single schedule. In each iteration, an unfit portion of the solution is removed. Any broken solution is repaired by a refined greedy building process.

Taken as a whole, the proposed approach has a number of distinct advantages. Firstly, it is simple and easy to implement because it uses greedy algorithms and local heuristics. Secondly, due to its features of maintaining only a single solution at each iteration and eliminating inferior parts from this solution, it can quickly converge to local optima. Thirdly, the technique has the ability to jump out of local optima in an effective manner. Finally, this approach can be easily combined with other meta-heuristics to achieve its peak performance on solution quality if CPU-time is not the major concern. For example, tabu search can be used in the Reconstruction step to explore the neighbouring solutions in an aggressive way and avoid cycles by declaring attributes of visited solutions as tabu. In addition, simulated annealing could be used as the acceptance criteria for the resulting solutions after Reconstruction to accept not only improved solutions as in the current form, but also worse ones with a certain level of probability.

Acknowledgements

The work was funded by the UK Government's major funding agency, the Engineering and Physical Sciences Research Council (EPSRC), under grants GR/R92899/02 and GR/S70197/1.

References

Ahuja, R.K., Magnanti, T.L. and Orlin, J.B. (1993). Network Flows: Theory, Algorithms, and Applications. Prentice Hall, NJ.

Aickelin, U. and Dowsland, K. (2000) "Exploiting Problem Structure in a Genetic Algorithm Approach to a Nurse Rostering Problem," Journal of Scheduling 3: 139-153.

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

Aickelin, U. and Li, J. (2006) "An Estimation of Distribution Algorithm for Nurse Scheduling," (to appear) Annals of Operations Research.

Aickelin, U. and White, P (2004). "Building Better Nurse Scheduling Algorithms," Annals of Operations Research 128: 159-177.

Anzai, M. and Miura, Y. (1987) "Computer Program for Quick Work Scheduling of Nursing Staff," Medical Informatics 12: 43–52.

Bard, J. and Purnomo, H.W. (2006) "Preference Scheduling for Nurses Using Column Generation," European Journal of Operational Research 164: 510-534.

Bard, J. and Purnomo, H.W. (2007) "A Cyclic Preference Scheduling of Nurses Using a Lagrangian-based Heuristic," Journal of Scheduling 10 (to appear).

Beddoe, G. and Petrovic, S. (2006) "Selecting and Weighting Features using a Genetic Algorithm in a Case-Based Reasoning Approach to Personnel Rostering," (to appear) European Journal of Operational Research.

Beliën, J. and Demeulemeester, E.L. (2006) "Building Cyclic Master Surgery Schedules with Leveled Resulting Bed Occupancy," European Journal of Operational Research 176: 1185-1204.

Blau, R. (1985) "Multishift Personnel Scheduling with a Microcomputer," Personnel Administrator 20: 43–58.

Bradley, D. and Martin, J. (1990) "Continuous Personnel Scheduling Algorithms: A Literature Review," Journal of the Society for Health Systems 2:8-23.

Brusco, M.J. and Jacobs, L.W. (1995) "Cost Analysis of Alternative Formulations for Personnel Scheduling in Continuously Operating Organisations," European Journal of Operational Research 86:249-261.

Burke, E.K., Cowling, P., De Causmaecker, P. and Vanden Berghe, G. (2001) "A Memetic Approach to the Nurse Rostering Problem," Applied Intelligence 15:199–214.

Burke, E.K., De Causmaecker, P. and Vanden Berghe, G. (1999) "A Hybrid Tabu Search Algorithm for the Nurse Rostering Problem," In McKay B. et al. (eds.), Simulated Evolution and Learning, Springer LNAI 1585:187–194.

Burke, E.K., De Causmaecker, P., Petrovic, S. and Vanden Berghe, G. (2004) "Variable Neighbourhood Search for Nurse Rostering Problems," In Resende M.G.C. and De Sousa J.P. (eds.), Metaheuristics: Computer Decision-Making (Combinatorial Optimization Book Series), Kluwer, Chapter 7, pp. 153-172.

Burke, E.K., De Causmaecker, P., Vanden Berghe, G. and Van Landeghem, H. (2004) "The State of the Art of Nurse Rostering," Journal of Scheduling 7:441-499.

Burke, E.K., Kendall, G., Newall, J., Hart, E., Ross, P., and Schulenburg, S. (2003) "Hyper-heuristics: an Emerging Direction in Modern Search Technology," In Glover F. and Kochenberger G. (eds), Handbook of Meta-Heuristics, Kluwer, Chapter 16, pp. 451-470.

Burke, E.K., Kendall. G. and Soubeiga, E. (2003) "A Tabu-Search Hyperheuristic for Timetabling and Rostering," Journal of Heuristics 9:451-470.

Cheang, B., Li, H., Lim, A. and Rodrigues B. (2003) "Nurse Rostering Problems--a Bibliographic Survey," European Journal of Operational Research 151:447-460.

Chen, J.G. and Yeung, T. (1993) "Hybrid expert system approach to Nurse scheduling," Computers in Nursing:183–192.

Dowsland, K. (1998) "Nurse Scheduling with Tabu Search and Strategic Oscillation," European Journal of Operational Research 106:393-407.

Dowsland, K. and Thompson, J. (2000) "Nurse Scheduling with Knapsacks, Networks and Tabu Search," Journal of the Operational Research Society 51:825-833.

Ernst, A.T., Jiang, H., Krishnamoorthy, M., Owens, B. and Sier, D (2004a) "An Annotated Bibliography of Personnel Scheduling and Rostering," Annals of Operations Research 127: 21–144.

Ernst, A.T., Jiang, H., Krishnamoorthy, M. and Sier, D. (2004b) "Staff Scheduling and Rostering: a Review of Applications, Methods and Models," European Journal of Operational Research 153: 3-27.

Glover, F. (1989) "Tabu Search – Part I,". ORSA Journal on Computing 1:190–206.

Kawanaka, H., Yamamoto, K., Yoshikawa, T., Shinigi, T. and Tsuruoka, S. (2001) "Genetic Algorithm with the Constraints for Nurse Scheduling Problem," in Proceedings of Congress on Evolutionary Computation (CEC), pp. 1123-1130.

Kirkpatrick, S. Gelatt C.D. and Vecchi, M.P. (1983) "Optimization by Simulated Annealing," Science 220: 671–680.

Li, J. and Aickelin, U. (2004) "The Application of Bayesian Optimization and Classifier Systems in Nurse Scheduling," In Yao, X., et al. (eds.): Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII). Springer Lecture Notes in Computer Science Volume 3242, pp. 581-590.

Lourenco, H.R., Martin, O.C. and Stutzle, T. (2002) "Iterated Local Search," In Glover, F. and Kochenberger, G. (eds.): Handbook of Metaheuristics, pp. 321-353, ISORMS 57, Kluwer.

Meyer auf'm Hofe, H. (2001) "Solving Rostering Tasks as Constraint Optimization," In: Burke, E.K. and Erben, W. (eds). Practice and Theory of Automated Timetabling, 3rd International Conference. Springer Lecture Notes in Computer Science Volume 2079, pp. 191-212.

Miller, H.E., Pierskalla, W. and Rath, G. (1976) "Nurse Scheduling using Mathematical Programming," Operations Research 24: 857–870.

Mueller, C.W. and McCloskey, J.C. (1990) "Nurses' Job Satisfaction: A Proposed Measure," Nursing Research 39: 113-117.

Özcan, E. (2005) "Memetic Algorithms for Nurse Rostering," In Proceedings of the 20th International Symposium on Computer and Information Sciences, Springer Lecture Notes in Computer Science Volume 3733, pp. 482-492.

Ross, P. (2005) "Hyper-heurisrics," In Burke, E.K. and Kendall, G. (eds.): Chapter 16 in Search Methodologies: Introductory Tutorials in Optimization and Decision Support Methodologies. Springer.

Silvestro, R. and Silvestro, C. (2000) "An Evaluation of Nurse Rostering Practices in National Health Service", Journal of Advanced Nursing 32: 525-535.

Sitompul, D. and Randhawa, S. (1990) "Nurse Scheduling Models: A State-of-the-Art Review," Journal of the Society of Health Systems 2: 62-72.

Tien, J.M. and Kamiyama, A. (1982) "On Manpower Scheduling Algorithms," Society for Industrial and Applied Mathematics 24: 275–287.

Warner, M. and Prawda, J. (1972) "A Mathematical Programming Model for Scheduling Nursing Personnel in a Hospital," Management Science 19: 411–422.