A grasp–knapsack hybrid for a nurse-scheduling problem

Article *in* Journal of Heuristics · August 2009

DOI: 10.1007/s10732-007-9066-7 · Source: DBLP

3 authors, including:

Kathryn A. Dowsland
Gower Optimal Algorithms Ltd
62 PUBLICATIONS **3,136** CITATIONS

SEE PROFILE

Jonathan M. Thompson
Cardiff University
17 PUBLICATIONS **663** CITATIONS

SEE PROFILE

A GRASP-KNAPSACK HYBRID FOR A NURSE-SCHEDULING PROBLEM

MELISSA D. GOODMAN[1], KATHRYN A. DOWSLAND[1,2,3] AND JONATHAN M. THOMPSON[1*]

[1]*School of Mathematics, Cardiff University, Cardiff, UK*
[2]*Gower Optimal Algorithms Ltd., 5, Whitestone Lane, Newton, Swansea, UK*
[3]*School of Computer Science and IT, University of Nottingham, Nottingham, UK*

[*]*Correspondence to: Jonathan Thompson, School of Mathematics, Cardiff University, Senghennydd Road, Cardiff, CF24 4AG, UK. E-mail: thompsonjm1@cardiff.ac.uk. Tel: +44 (0) 29 2087 5524.*

ABSTRACT

This paper is concerned with the application of a GRASP approach to a nurse-scheduling problem in which the objective is to optimise a set of preferences subject to a set of binding constraints. The balance between feasibility and optimality is a key issue. This is addressed by using a knapsack model to ensure that the solutions produced by the construction heuristic are easy to repair. Several construction heuristics and neighbourhoods are compared empirically. The best combination is further enhanced by a diversification strategy and a dynamic evaluation criterion. Tests show that it outperforms previously published approaches and finds optimal solutions quickly and consistently.

KEY WORDS: Nurse-scheduling, rostering, GRASP, hybrid, knapsack.

1. INTRODUCTION

Construction heuristics, neighbourhood search techniques and evolutionary algorithms have all been used successfully in the solution of practical scheduling problems. However, there is often a conflict between feasibility and solution quality, and the difficulty in maintaining a suitable balance between these two objectives is well documented. For example Michalewicz and Fogel (2004) devote a whole chapter to the handling of constraints in evolutionary algorithms. Early examples from the fields of timetabling and scheduling include Wright (1991) and Abramson (1991) while observations made by Aickelin and Dowsland (2004), Zhu and Lim (2006) and Lim, Rodrigues and Zhang (2006) show that these issues are still relevant. They arise for a number of reasons. In some cases it may not be practical to limit the solution space to the set of feasible solutions, as simply finding a feasible solution may itself be a non-trivial problem. Even if this is not so, it may still be difficult to define operators, such as crossover in a genetic algorithm or neighbourhood moves in local search, that preserve feasibility. In other cases, restricting the search to feasible solutions may result in a solution space that is very sparsely connected and this may hamper the ability of the search to seek out high quality solutions. (See for example Thompson and Dowsland (1996) and Dowsland (1998)). There are a number of ways of overcoming this, for example the use of penalty functions, repair functions, wider neighbourhoods etc. However, none of these is able to provide a universal answer and when they do work their success is often the result of exploiting information about the problem structure, for example in setting appropriate penalty weights, or allowing the efficient exploration of large neighbourhoods.

One approach that has received considerable interest in the solution of scheduling problems is Greedy Randomised Adaptive Search Procedure (GRASP). A construction phase is followed by an improvement phase and these are repeated for several cycles. Examples include Drexl and Salewski (1997) who use a two phase greedy randomised algorithm to solve the school timetabling problem, Binato et al. (2001) who incorporate intensification strategies into their GRASP for Job Shop Scheduling, and Gupta and Smith (2006) who schedule a single machine to minimise tardiness using a variation of Variable Neighbourhood Search in the improvement phase of their GRASP.

This paper is concerned with a GRASP approach to a variant of the nurse-scheduling problem in which the objective is to find a schedule that optimises a set of preferences while meeting a series of binding constraints. The constraints are often tight i.e. the nurses available are just sufficient to meet the requirements without any slack. Thus simple heuristic rules often fail to produce feasible solutions and existing approaches capable of producing high quality solutions across a range of problem instances all use problem specific information to balance the focus between optimality and feasibility (Dowsland (1998), Dowsland and Thompson (2000), Aickelin and Dowsland (2000, 2004)). The objectives of our investigation are two-fold: firstly to produce a robust solution approach to the problem, which is typical of that arising in many UK hospitals; and secondly to examine the impact of the construction heuristic used within the GRASP algorithm on solution quality. In particular, we investigate the use of problem specific knowledge, in the form of a look-ahead procedure based on a knapsack model, to strengthen the construction heuristic. This allows us to use a

straightforward descent algorithm based on simple neighbourhood moves during the improvement phase.

In the next section we describe the problem and outline other solution approaches reported in the literature, highlighting the fact that the most successful of these have all included special procedures to deal with the quality/feasibility issue. This is followed by an introduction to GRASP in Section 3. Section 4 deals with the family of GRASP heuristics we have developed for the problem, while Section 5 describes an enhancement achieved by hybridising the construction phase with a knapsack algorithm. Section 6 is concerned with computational experiments. These are based on 52 data sets from a large UK hospital and are designed to compare the different variants with each other, and to compare the most successful with other approaches from the literature. Finally, conclusions and further work are discussed in Section 7.

2. THE NURSE-SCHEDULING PROBLEM

The problem of allocating nurses to duty rosters occurs in hospital wards all over the world. The essence of the problem remains the same in each case: every nurse in the ward must be assigned a suitable set of shifts for the given planning period. However, the precise details of the problem differ from hospital to hospital. Objectives range from minimising costs to maximising the satisfaction of the personnel involved (e.g. (Jaumard et al. (1998), Berrada et al. (1996)), while a wide range of working practices leads to an equally wide range of problem constraints. Some hospitals employ a cyclic schedule where for each planning period the same schedule is repeated, and each nurse cycles around a subset of the selected shift patterns, (e.g. Rosenbloom and Goertzen (1987)). Others use non-cyclic schedules in order to provide more flexibility in dealing with absence, fluctuations in patient numbers and staff requests (e.g. Cheang et al. (2003)). Other variations include the length of the planning period, the numbers and lengths of shifts on each day, whether or not different skill levels need to be taken into account, and constraints on working practices such as the number of consecutive work days, minimum rest period between shifts, rotation of night/weekend working etc. The papers by Dowsland and Thompson (2000), Isken (2004), Bellanti et al. (2004) and Burke et al. (2003a) illustrate many of these differences. A range of solution approaches have been used to tackle the different variants of the problem, including exact approaches, often based on mathematical programming (e.g. Isken (2004)), artificial intelligence (e.g. Meyer auf'm Hofe (2000), Petrovic et al. (2003)) and a variety of heuristics including genetic algorithms (e.g. Burke et al. (2001), Moz and Pato (2007)), ant colony optimisation (Gutjahr and Rauner (2007)), simulated annealing (e.g. Brusco and Jacobs (1995)), tabu search (e.g. Dowsland (1998), Bellanti et al. (2004)) and other forms of local search e.g. SAWing (Parr and Thompson (2007)). Of these, tabu search seems to be the most popular. Most of the approaches reported in the literature deal with a fixed problem definition, but others such as the algorithm PLANE described in Burke et al. (1999) allow many of these details to be user defined. Recent surveys can be found in Burke et al. (2004) and Ernst et al. (2004).

The variant of the problem considered here is that introduced in Dowsland (1998) and also tackled in Dowsland and Thompson (2000), Aickelin and Dowsland (2000, 2004), Aickelin and Li (2007) and Burke et al. (2003b). The objective is to produce weekly work rosters for individual hospital wards. These must satisfy individual work

contracts and ensure there are sufficient nurses with the desired skill mix on duty at all times, whilst also meeting the nurses' preferences and requests as far as possible. The day is divided into three shifts; an 'early' day shift, a 'late' day shift and a night shift. The majority of nurses work either days or nights in a given week, although a few may be contracted to work a combination of the two. Night shifts are longer than day shifts. Due to the different shift lengths, a nurse working day shifts will typically, but not always, work more shifts than a nurse working nights. For example, a full time nurse works either 5 day shifts or 4 night shifts. Part time nurses work other combinations e.g. 4 days or 3 nights and 3 days or 2 nights, while some part-timers are contracted to work equal numbers of days or nights. In total there are ten possible day/night combinations. At the start of the planning period the combination pertaining to each nurse is known. We will refer to this combination as the *type* of the nurse.

Each nurse is graded according to their level of expertise and experience, resulting in 3 grade bands, with band 1 consisting of senior nurses who are capable of being in charge of the ward, and band 3 those nurses who are trained only to the minimum level. For each shift there is a minimum requirement of nurses of each grade, and as nurses of higher grades can cover all the duties of those at lower grades these are expressed cumulatively. For example, requirements of 1, 2 and 5 nurses at grades 1, 2 and 3 respectively means that at least one grade 1 nurse, two grade 1 or 2 nurses, and five nurses in total are required. Where there are more nurses than required the additional cover should be spread uniformly over the planning period and if there are insufficient nurses additional 'bank' nurses must be used. The requirements that each nurse works the correct number of shifts, that the minimal covering requirements are met, and that shifts at the start of the week are compatible with those worked at the end of the previous week are binding, and must be met if the solution is to be feasible. In addition, there is a requirement to produce rosters that are user friendly. This involves attempting to meet requests not to work certain shifts/days, rotation of nights and weekend working, and avoiding sequences of shifts that are unpopular e.g. working alternate days.

Thompson and Dowsland (2000) show that the problem can be pre-processed so that any bank nurses required can be added beforehand, and 'dummy' nurses can be added to deal with the problem of spreading any excess cover. This results in a tight problem in which there is no slack in the overall cover requirements. They also show that the allocation of day nurses to 'earlies' and 'lates' can be dealt with in a post-processing phase. Thus the 'early' and 'late' requirements can be amalgamated into a single 'day' requirement. The remaining problem can be modelled as an integer program as follows.

For nurses working d day shifts or e night shifts, the set of feasible shift patterns for that type is defined by a set of binary vectors of length 14, with the first 7 values representing the day shifts and the remaining 7 values the night shifts. A '1' implies that the nurse is working and a '0' implies a day or night off. The number of possible shift patterns for each nurse is $\binom{7}{d}$ day patterns and $\binom{7}{e}$ night patterns. Patterns for those nurse-types working a combination of days and nights can be defined similarly. The feasible patterns for individual nurses are then formed by removing any patterns that would violate the constraints relating to working practices, annual leave etc. The quality of each pattern is reflected by the allocation of a penalty cost, referred to in the

following as the *preference cost*, in the range [0, 100] with '0' implying the shift pattern is ideal and '100' signifying it is infeasible. These values were obtained as the result of an iterative consultation process with a major UK hospital. The IP representation is then:

$$\text{Minimise} \sum_{i=1}^{r} \sum_{j \in S_i} p_{ij} x_{ij} , \tag{1}$$

$$st \sum_{j \in S_i} x_{ij} = 1 \quad \forall i , \tag{2}$$

$$\sum_{i \in R_g} \sum_{j \in S_i} a_{jk} x_{ij} \geq C_{gk} , \quad \forall g, k \tag{3}$$

where

$$x_{ij} = \begin{cases} 1 & \text{if nurse } i \text{ works shift pattern } j \\ 0 & \text{otherwise} \end{cases}$$

$$a_{jk} = \begin{cases} 1 & \text{if shift pattern } j \text{ covers shift } k \\ 0 & \text{otherwise} \end{cases}$$

p_{ij} is the preference cost of nurse i working shift pattern j, S_i is the set of shift patterns that nurse i can work, C_{gk} is the number of nurses of grade g or better required on shift k, R_g is the set of nurses at grade g or better and r is the total number of nurses. The objective function (1) is the total preference cost, constraints (2) ensure that each nurse is assigned to a valid shift pattern and constraints (3) ensure that the cover requirements are satisfied. Typical problem dimensions are between 21 and 28 nurses and a total of 411 shift patterns. For some instances solving this IP to optimality can take several hours using the professional version of the Xpress-MP integer programming software (Fuller 1998).

Several heuristic methods have already been applied to this problem. The tabu search approach of Dowsland (1998) uses a series of complex neighbourhoods, evaluation functions and candidate lists to increase the scope of the search as it moves through the space of feasible solutions. Nevertheless a strategic oscillation strategy in which the search is periodically allowed to visit solutions outside of the feasible region is necessary in order to guarantee quality solutions. Aickelin and Dowsland (2000, 2004) develop two genetic algorithm (GA) approaches. In the first of these, the individuals of the population represent complete solutions to the problem. Fitness is defined as a weighted sum of the preference cost and the total number of undercovered shifts at each grade. In order to deal with constraints (3) (the covering constraints), three problem-specific modifications to the classical GA implementation are required. These are: a complex co-evolutionary strategy using sub-populations that allow constraints at different grades to be relaxed and then recombined intelligently; a hill-climbing repair operator; and an additional penalty/reward scheme that rewards offspring that are likely to be repairable and penalises those that are not. The second, more successful, approach uses a GA to order the nurses and then applies a heuristic decoder to produce a solution from this ordering. Several different heuristics using a different balance between feasibility and quality were tested and it was shown that a bias towards seeking feasibility while including some measure of

quality produced the best solutions. The final implementation also used a modified crossover operator and information from a lower bound to improve the decoder.

The success of the above approaches was achieved at the expense of a series of modifications and enhancements based on aspects of the problem structure. Two more generic approaches are those of Burke et al. (2003b) and Aickelin and Li (2007). The former applies a tabu search hyperheuristic that makes use of several of the simpler neighbourhoods used in Dowsland (1998). For each move the choice as to which neighbourhood to use is made via a scoring system based on the success of that type of move to date. However, their objective was to produce a fast, generic approach to the problem requiring little problem specific information or fine-tuning. While they achieved this objective the results are not comparable with those of Dowsland in terms of solution quality. Aickelin and Li schedule the nurses sequentially using a Bayesian network to determine an appropriate scheduling rule for each nurse. The results show that this approach consistently produces feasible solutions but once again solution quality does not match that of Dowsland's tabu search implementation.

The above suggests that balancing feasibility and optimality is indeed a difficult task for both population based and sequential search approaches to the problem. In the two more successful approaches i.e. Dowsland's Tabu Search method (1998) and Aickelin and Dowsland's indirect GA (2004), this was achieved by the use of problem specific information to drive a number of intelligent add-ons. In the following sections we investigate the possibility of using this type of information in the construction phase of a GRASP approach to the problem.

3. GRASP

GRASP was first introduced by Feo et al. (1994). It can be viewed as a multi-start local search approach in which the starting solutions are generated by a probabilistic greedy construction approach. In its original form, GRASP consists of *nrep* independent repetitions of two phases: *Construct* and *Improve*. *Construct* is the greedy construction phase that builds a solution one element at a time. At each stage in the construction each available element, *i*, is given a score *f(i)* based on a local measure of the benefit of adding that element next. The element to be added is then chosen probabilistically according to the scores. There are obviously many strategies available for achieving this but the most common approach is to determine a candidate list of the best *n* elements for a given *n* and then to select randomly from this list. This is usually achieved using roulette wheel selection where each element is selected with a probability proportional to its relative score. *Improve* is a local search based improvement heuristic, in which the incumbent solution is repeatedly replaced by a neighbour of higher quality until no such neighbour exists i.e. a local optimum has been reached.

As with any metaheuristic search technique, in order to implement a GRASP approach to a given problem, a number of problem specific decisions must be made. These include the usual local search decisions i.e. the definition of the solution space, neighbourhood structure and evaluation function, together with the details of the construction heuristic including the definition of the score function $f(i)$.

Since the first GRASP implementations were published, researchers have suggested a number of ways of improving performance. Some of these focus on improving the solutions obtained from the construction phase. For example Laguna and Martí (2001) describe a GRASP implementation for the graph colouring problem. The construction phase builds up the colour classes one at a time, completing one colour class before moving on to the next. Rather than just allowing a single attempt at each colour class several attempts are made and the best, according to a suitable criterion, is selected. Others, for example Binato et al. (2001), Fleurent and Glover (1999), try to improve the partial solutions by periodically applying local search during the construction phase.

An alternative is to focus on improving the local search phase. Typical examples involve replacing the simple descent algorithm with a more powerful local search technique such as simulated annealing or tabu search (e.g. Laguna and González-Velarde (1991)). However, Laguna and Martí (2001) found that these additions did not improve the solutions significantly when given a limited amount of time for the improvement phase. Others have attempted to exploit information from earlier cycles to seek out better solutions. A popular approach is to use some sort of feedback mechanism to influence the greedy selection process in the construction phase. This is achieved by increasing the probability of selecting elements that correspond to features of good solutions and/or decreasing those that correspond to bad solutions, in much the same way as the trail mechanism influences the construction phase of an ant algorithm (Fleurent and Glover (1999)). Others use previously gleaned information in other ways. For example, Aiex et al. (2003) use the best existing solutions as the basis for a path relinking phase which is added after the local search phase in each iteration. For a summary of these, and other developments in GRASP and its applications, see Resende (2001).

Our main focus is on improving the construction phase, and instead of selecting the next element purely on a myopic basis we employ a look-ahead strategy based on a knapsack model to produce solutions that are more likely to be repaired by the local search phase. The variables in the knapsack model also form the basis of a feedback mechanism aimed at diversification. We also make minor modifications to the improvement phase, by relaxing the requirement that only improving moves are accepted, and using feedback information to place a threshold on the preference cost.

4. GRASP ALGORITHMS FOR THE NURSE-SCHEDULING PROBLEM

Due to the complexity of the problem and the tightness of the constraints, generating feasible schedules is itself a demanding task and, as such, our solution space must include both feasible and infeasible schedules. Thus our construction phase needs to produce a complete, but not necessarily feasible, schedule that will be used as the starting point for the local search improvement phase.

4.1. Construction

At each stage of the construction phase a nurse and an appropriate shift pattern for that nurse must be selected. Both selections could be made according to the score function or the nurse could be selected *a priori* according to a predefined ordering.

(Note that as each shift pattern may be used once, more than once, or not at all, pre-ordering the shift patterns and then selecting a nurse is not a sensible option.) Given a partially constructed schedule, the allocation of unscheduled nurse i of grade g_i to shift pattern j will add p_{ij} to the total preference cost and will improve the cover for those shifts included in pattern j that do not currently meet the staffing requirements for grades g_i and below. We therefore need a 2-part score function to capture both these aspects. Experiments with a number of different score definitions were carried out and these will be described in detail after the following outline of the construction process.

For any given score function, $f(i,j)$, the construction phase proceeds as follows:

Let R be the set of nurses and R^+ be the set of nurses already allocated.

Step 1: Set $R^+ = \varnothing$.
Step 2: Calculate the score $f(i,j)$ associated with allocating nurse i to shift pattern j, for all feasible pairs (i,j). (Note that the set of feasible allocations (i,j) may include all nurses not already allocated, or may relate to a single nurse i given by a predefined ordering.)
Step 3: Let L be the candidate list of the n highest scoring options.
Step 4: Select $(i,j) \in L$ using roulette wheel selection, i.e. each (i,j) is selected with a relative probability proportional to $f(i,j)$.
Step 5: Update the schedule with this allocation and set $R^+ = R^+ \cup \{i\}$
Step 6: If $R^+ \neq R$ go to Step 2.

Aickelin and Dowsland (2004) suggest three different score functions to guide the greedy construction heuristics used as decoders for their indirect GA implementation. The part of the score relating to preference cost is given by:

$$prefscore_{ij} = 100 - p_{ij}, \text{ where } p_{ij} \text{ is as defined in Section 2.}$$

Two different functions are used for the part of the score relating to cover. Our construction methods are based on these scores and we therefore present them in detail.

Both cover related scores are functions of the current total undercover of shift k at grade g, defined as:

$$d_{gk}(R^+) = \max\left\{0, C_{gk} - \sum_{q \in R^+ \cap R_g} \sum_{j \in S_q} a_{jk} x_{qj}\right\}$$

The first cover score was designed for use without a preference cost element in a situation where the nurse i to be allocated next is decided *a priori*. It is defined as follows.

Let

$$g' = \begin{cases} \min\left\{ g \geq g_i : \sum_{k=1}^{14} d_{gk}(R^+) > 0 \right\} & \text{if such a } g \text{ exists} \\ 3 & \text{otherwise} \end{cases}$$

and let k' = argmax$\{d_{g'k}\}$, then

$$covscore1_{ij} = \sum_{k \in K} a_{jk} d_{g'k}(R^+),$$

where $K = \begin{cases} \{1,...,7\} & \text{if } k' \leq 7 \text{ or the nurse must be on days} \\ \{8,...,14\} & \text{if } k' \geq 8 \text{ or the nurse must be on nights.} \end{cases}$

For a given nurse *i*, this score is a measure of the improvement in cover at grade g_i (or at the first uncovered grade below g_i if all cover at g_i is already satisfied), giving a greater score to those shifts with the most undercover. If the greatest shortfall in cover on a single shift is on nights then the score is based on night shifts only and vice versa. As the scores for different nurses are measured over different grades this method of scoring is not suited to situations in which the nurse has not already been determined. The priority given to nights or days also make it unsuited to combination with the preference score as it is common for there to be a large difference between the preference costs for days and nights for some nurses, and such differences will not be considered.

The second cover score is intended to overcome these problems. It is more flexible in that it includes a term for each grade where cover would be improved. These terms are weighted so that cover at higher grades, for which there are fewer nurses available, are given priority. This score is simply the weighted sum of the undercover and is given by:

$$covscore2_{ij} = \sum_{g=g_i}^{3} w_g \left(\sum_{k=1}^{14} a_{jk} d_{gk}(R^+) \right),$$

where w_g is the weight associated with grade g for $g = 1,...,3$.

We use these individual scores to form the basis of a family of construction heuristics by changing the way in which the scores are combined, using both dynamic and static orderings of the nurses, and by incorporating a simple look-ahead rule. Each of these will provide a different balance between the opposing goals of optimising preference costs and meeting the covering constraints, and can be expected to provide solutions with different attributes as starting points for the local search phase.

Our simplest construction heuristics define the score function *f(i,j)* to be a weighted sum of the preference score and a cover score and combine this with a fixed ordering of the nurses. We implemented two variants using this strategy, denoted *cover* and *combined* and given by:

cover: $f_1(i,j) = covscore1_{ij}$

combined: $f_2(i,j) = w_p.prefscore_{ij} + w_c.covscore2_{ij}$,

where w_p and w_c are two weight parameters.

In *cover* all the emphasis is on attempting to satisfy the cover constraints, while *combined* is a more balanced strategy. However, given that the ordering of the nurses is fixed we would not expect either to be as successful in producing high quality solutions for the start of the improvement phase as a more flexible approach in which both the nurse and the shift pattern are selected dynamically by the score functions. Thus for these options, if the full GRASP implementation is to be successful, the local search phase will be required to play a significant role. In particular, for *cover*, all the optimisation of the preference cost will be left to the local search phase.

We also consider two construction heuristics based on a dynamic ordering. The first uses the *combined* score function to select both the nurse and shift pattern. This option is referred to as *holistic* as it encompasses the whole space of options at each stage. However the number of available options with this approach will be large, especially in the early stages of the construction. This may result in longer solution times and a lack of focus during this phase. We therefore introduce our fourth variant, *last chance*, which strengthens the construction by incorporating an element of look-ahead, while reducing the number of shift patterns considered for each nurse to one. This is achieved by comparing the highest scoring option for each nurse with the second highest and defining a single score function for each nurse based on the difference. The selected nurse is then allocated to the shift pattern with the highest score. The score function for this option is calculated as follows:

Let $j^*(i) = \arg\max_{j \in S_i} \{ f_2(i, j) \}$. Then

last chance: $f_3(i) = \max_{j \in S_i} \{ f_2(i, j) \} - \max_{\substack{j \in S_j \\ j \neq j^*(i)}} \{ f_2(i, j) \}$.

This variant has fewer options at each stage as only the nurse has to be chosen. However it does not suffer from the limitations of the fixed ordering in the *cover* and *combined* variants in that the best shift pattern associated with each nurse also changes adaptively, depending on the allocations made in previous stages. In addition, the inclusion of some degree of look-ahead into the score should help to overcome the problem inherent in all greedy approaches, that of potentially being left with a set of expensive options in the last few stages.

4.2 The improvement phase.

None of the above construction heuristics is guaranteed to produce a feasible solution. Therefore, the solution space for the improvement phase is the set of solutions in which each nurse is allocated to a feasible shift pattern, regardless of whether or not the covering constraints are satisfied. Our basic search strategy is that of random descent i.e. at each iteration we sample the space of neighbours of the current solution without replacement and accept an improving solution as soon as it is encountered. Our evaluation function reflects the importance of meeting the covering constraints over that of optimising the preference cost, and we define a solution as being better than its neighbour if the cover is improved or if the cover is unchanged and the penalty is improved. More precisely our evaluation function is defined as:

$$\sum_{i \in R} \sum_{j \in S_i} p_{ij} x_{ij} + W \cdot \sum_{k=1}^{14} \sum_{g=1}^{3} d_{gk}(R),$$

where $W >> p_{ij}$, $\forall i, j$.

We also consider two minor modifications to the straightforward descent strategy. Firstly, a typical data set may have many x_{ij} variables with identical p_{ij} values. This suggests that there may be large plateau-like areas in the solution landscape and for this reason we also experiment with accepting equal cost solutions. Secondly, during the search for a feasible solution, the above objective function may allow the preference cost to increase significantly from that produced at the end of the construction phase. We therefore introduce a threshold on the preference cost using feedback information from previous cycles.

As our objective is to exploit the construction part of the GRASP algorithm, rather than to develop a complex local search algorithm in which the construction phase plays only a minor role, our neighbourhood structure needs to be simple, but flexible enough to allow moves in different areas of the search space. For this reason we consider combinations of three natural neighbourhoods: a 1-opt move neighbourhood, a 2-opt move neighbourhood, and a swap neighbourhood as defined below. We then combine the neighbourhoods by selecting each with a given probability that may change according to whether or not the current solution is feasible.

1. Change neighbourhood
$N_C(s)$ is the set of solutions obtained from any solution, *s*, by changing the shift pattern of a single nurse. This neighbourhood can change both the cover and the preference cost and in theory any solution is reachable from any other by a series of moves using this neighbourhood. However, it is not sufficiently flexible in the context of our descent strategies, as if the cover constraints are tight, any solution, *s*, satisfying the covering constraints will not have any improving moves within $N_C(s)$.

2. Swap neighbourhood
$N_S(s)$ is the set of solutions obtained from *s* by swapping the shift patterns of two nurses where their patterns are compatible i.e. where both patterns are feasible for both nurses involved in the swap. As swapping the patterns of two nurses on days (nights) will not affect the cumulative cover at grade 3, this neighbourhood is likely to allow the search to progress further once a feasible solution is found even on tight problems. Note that even for tight problems there is likely to be slack at grades 1 and 2 but moves are only accepted if the shortfall in cover does not increase at any grade. However, as swaps are limited to nurses of a single type some areas of the solution space are likely to be unreachable.

3. Extended neighbourhood
$N_E(s)$ is the set of solutions obtained from *s* by changing the shift patterns of two nurses. This gives more flexibility than the above neighbourhoods when *s* satisfies the covering constraints. However it is a larger neighbourhood and it may be computationally expensive to find improving moves close to local optima when they

may be sparsely distributed. We overcome this by restricting its use to situations where s is already feasible and sampling from a candidate list of neighbours that is a superset of all those that result in an improvement. This is achieved as follows. Given s, an element of $N_E(s)$ is defined by the quadruple (i_1,j_1,i_2,j_2) where i_1 and i_2 are the 2 nurses involved and j_1 and j_2 are the new patterns for these nurses. The four parameters are sampled in the above order with all the options for the remaining parameters being exhausted before parameters higher in the order are changed. i_1 is sampled uniformly from the full set of nurses, but the candidate sets for the remaining parameters can be considerably reduced using the following logic. The brackets in the following refer to the case where we accept moves of equal cost as well as improving moves.

Given that s satisfies the cover constraints an improving move must result in a reduction in the preference cost. For any two nurses involved in such a move at least one must be moved to a pattern with lower (or equal) preference cost. As we are sampling from the set of nurses uniformly we can assume without loss of generality that this is true for i_1. Thus, given i_1 we sample j_1 from the set of patterns with penalty cost less than (or equal to) the cost of i_1's current allocation. Further efficiency gains are possible if the patterns for each nurse are pre-sorted into p_{ij} order. If there is little or no slack in the covering constraints then moving nurse i_1 to pattern j_1 may result in some shifts being under-covered. The second half of the move must repair this. Thus nurse i_2 must be of a sufficiently high grade to cover the shortfall and must currently be off duty on all the under-covered shifts. Our candidate list for i_2 is therefore restricted to nurses who satisfy these conditions. Finally pattern j_2 must be chosen from those patterns for which the increase in preference cost for i_2 is less than (or equal to) the reduction obtained from moving nurse i_1 to pattern j_1.

5. HYBRIDISING THE CONSTRUCTION HEURISTIC WITH A KNAPSACK ROUTINE

The construction heuristics and local search strategies described above can be combined in different ways to form a family of traditional GRASP algorithms based on a relatively straightforward greedy construction phase and a descent based improvement phase. As outlined in Section 3, many of the more successful GRASP implementations include some additional ingredients aimed at improving either the construction or improvement phases. In this section we describe a hybridisation of the construction phase with a knapsack model. This allows us to incorporate a powerful look-ahead mechanism that will ensure that solutions produced from the construction are promising from the point of view of producing feasible solutions during the improvement phase. Our rationale for this is based on observations in Aickelin and Dowsland (2004) and Dowsland (1998). They suggest that the solution landscapes produced by our neighbourhood structures and evaluation function are likely to consist of subspaces separated by 'ridges' so that the improvement phase will be restricted to a single subspace, even if we were to allow some small uphill steps. As stated in Section 2, the nurses are partitioned into types according to the number of day and night shifts for which they are available and the ridges are caused by the changes in cover that result in moving nurses of different types from days to nights and vice versa. The subspaces correspond to the number of nurses of each type

allocated to days and nights, and for tight problems many of these subspaces will not contain any feasible solutions. This problem can be overcome by a construction heuristic that only produces solutions that lie in those subspaces that also contain some feasible solutions.

Dowsland and Thompson (2000) observe that the fact that nurses work a different number of night shifts and day shifts means that it is not possible to decide if a problem is feasible simply by counting the number of shifts available. They go on to show that this problem can be overcome using a knapsack model. We will use this model to ensure that the solutions produced by the construction phase lie in suitable subspaces. As we will show, this is equivalent to ensuring that the day/night allocations have sufficient numbers of nurses to meet the total covering requirements on both days and nights for all grades, *g*.

Essentially Dowsland and Thompson show that there are sufficient nurses to meet the cumulative cover constraints for grade *g* if the solution to the knapsack problem:

$$\text{Maximise } Z = \sum_{t=1}^{T} e_t y_{tg} \qquad (4_g)$$

$$st \quad \sum_{t=1}^{T} d_t y_{tg} \le \sum_{t=1}^{T} d_t n_{tg} - D_g \qquad (5_g)$$

$$y_{tg} \le n_{tg} \qquad \forall t \qquad (6_g)$$

is at least E_g, where

$t = 1,...,T$ is the nurse type index

d_t = number of day shifts worked by a nurse of type *t*

e_t = number of night shifts worked by a nurse of type *t*

n_{tg} = number of nurses of type *t* of grade *g* or better

$D_g = \sum_{k=1}^{7} C_{kg}$ (i.e. the total day shift requirement at grade *g* and better)

$E_g = \sum_{k=8}^{14} C_{kg}$ (i.e. the total night shift requirement at grade *g* and better)

and y_{tg} is the number of nurses of type *t* and grade *g* or better assigned to nights.

The expression in the objective function, (4_g) is the total number of night shifts covered, constraint (5_g) ensures that there are sufficient day shifts remaining and constraints (6_g) ensure that the number of nurses allocated does not exceed those available. For a given problem instance *P* (i.e. for given *T*, d_t, e_t, n_{tg}, D_g and E_g) we will denote the problem of finding a solution to (4_g), (5_g), (6_g) of value at least E_g by $KS_g(P)$. Clearly, if *P* is feasible there must be solutions satisfying $KS_g(P)$ for g = 1, 2 and 3. However, this is not sufficient, as we need to ensure that the three solutions are compatible with one another. In order to ensure a compatible set of solutions we need to add the constraint:

$$0 \le y_{tg+1} - y_{tg} \le n_{tg+1} - n_{tg} \quad \forall t, g = 1,2 \qquad (7)$$

The left hand inequality ensures that the number of night allocations at grade g+1 and better is at least as many as that at grade g and better, and the right hand inequality ensures that there are sufficient grade g+1 nurses to make up the difference. We will refer to the set of constraints given by $KS_1(P)$, $KS_2(P)$, $KS_3(P)$ and (7) by Cons(P). For a given problem instance, P, we ensure that our constructions always satisfy Cons(P) as follows.

Let $P(R')$ denote a partial solution to instance P in which the number of day and night shifts worked by those nurses in the set R' is known. At the start R' consists of the set of nurses who work both days and nights, and the set of nurses who must be allocated to days together with those who must be allocated to work nights. We define this set to be R^{fixed}. At any stage in the construction let R^+ be the set of nurses already allocated to their shift patterns. Then the remaining nurses can be allocated to days and nights in such a way as to satisfy Cons(P) if there is a feasible solution to Cons($P(R')$) where $R' = R^{\text{fixed}} \cup R^+$. Assume the allocation selected is nurse i to shift pattern j. If $i \in R^{\text{fixed}}$ R' will not change when i is added to R^+. Thus Cons($P(R' \cup \{i\})$) must still be feasible and the allocation can be made i.e. the construction moves onto step 5. If $i \notin R^{\text{fixed}}$, we need to check for feasibility by attempting to find a feasible solution to Cons($P(R^{\text{fixed}} \cup R^+ \cup \{i\})$). If there is a solution then the allocation is made. If not, then if j is a night shift, nurse i must be restricted to days and vice versa. We therefore restrict the set of feasible shift patterns for nurse i, add i to R^{fixed} and return to Step 2 without making the allocation. The restriction is enforced for the remainder of the construction phase only. It remains to be shown that we can determine whether or not there is a feasible solution to Cons($P(R')$) for any P and R' at minimal computational expense.

Our approach is based on the following observations:

Observation 1.
For any R' we can obtain Cons($P(R')$) from Cons(P) by reducing n_{tg} by one for each nurse of type t and grade g or better in R' and reducing D_g and E_g by the number of day and night shifts of grade g or better covered by the assignment of nurses in R'.

Observation 2.
Given a solution Y_g^* to $KS_g(P(R'))$ any solution to the knapsack problem:

$$\max Z = \sum_{t=1}^{T} e_t y_{tg-1} \qquad (4_{g-1})$$

$$st \quad \sum_{t=1}^{T} d_t y_{tg-1} \le \sum_{t=1}^{T} d_t n_{tg-1} - D_{g-1} \qquad (5_{g-1})$$

$$y_{tg-1} \le \min(n_{tg-1}, y^*_{tg}) \qquad (8_{g-1})$$

$$y_{tg-1} \ge \max(0, y^*_{tg} - (n_{tg} - n_{tg-1})) \qquad (9_{g-1})$$

satisfying $Z \ge E_{g-1}$ is a compatible solution to $KS_{g-1}(P(R'))$.

Note that the upper bounds given by (8_{g-1}) are obtained from (6_{g-1}) and the left-hand part of (7) and the lower bounds given by (9_{g-1}) are simply the right-hand part of (7). We denote this problem $KS_{g-1}(P(R')|Y_g^*)$

Observation 3.
The set of all feasible solutions to a bounded knapsack problem with an objective function value at least Z_{target} can be enumerated using a branch and bound tree search in which the branches at level i represent the set of feasible values for the ith variable. The problems represented by the nodes in the tree are smaller knapsack problems in which some of the variables have been fixed. Therefore local lower bounds can be obtained from the linear programming relaxations of these problems. For knapsack problems these linear programmes can be solved trivially without the use of an LP-solver (see Martello and Toth (1990) for details). The tree is pruned by backtracking whenever the bound is less than Z_{target}.

We can use the above observations to solve the appropriate Cons($P(R')$) at each stage of the construction using a hierarchical tree search. The tree at the top of the hierarchy represents a branch and bound search for all solutions of $KS_3(P(R'))$. Each terminal node representing such a solution, Y_3^*, spawns a new tree representing a branch and bound search for solutions to problem $KS_2(P(R')|Y_3^*)$. Similarly, the nodes representing these solutions spawn new trees representing a branch and bound search for solutions to $KS_1(P(R')|Y_2^*)$. The whole structure is searched using a depth first strategy. Once a feasible solution with respect to grade g is found the search proceeds to the next tree in the hierarchy i.e. grade g-1. If the search is not successful then control is returned to the tree relating to grade g until the next feasible solution is found. When a solution is reached at the third hierarchy i.e. grade 1, then a feasible solution to Cons($P(R')$) has been obtained and the allocation of nurse i to shift j can be made. If the search terminates without finding a solution for grade 1 then Cons($P(R')$) is infeasible and the current allocation cannot be accepted. Appendix One illustrates this procedure plus that in the following paragraph.

The above procedure is able to find feasible solutions or confirm that none exists in very short computational time for typical data sets. However, we can make the whole process more efficient if we note that we do not necessarily need to solve a new knapsack problem every time we consider a new allocation. At any point in the construction we have the y_{tg} values for a feasible solution to Cons($P(R')$). If we then consider allocating nurse i of type t and grade g to nights, then the existing solution will still be feasible as long as $y_{tg} \geq 1$ and (if $g \neq 1$) $y_{tg} > y_{tg-1}$. Similarly a day allocation will be feasible if $n_{tg} - y_{tg} \geq 1$ and $n_{tg} - y_{tg} > n_{tg-1} - y_{tg-1}$ If these conditions hold, we can allocate nurse i without solving a new knapsack problem. Having made the allocation we update the variables and constants to reflect the new solution.

The process is summarised in Figure 1.

(Insert Figure 1 around here)

We incorporate this look-ahead rule based on the knapsack calculations described in this section into each of our four construction heuristics to form four further variants. The best of these is then subject to further modifications. These are described in the following section.

6. FURTHER ENHANCEMENTS

In this section we consider three enhancements to the above implementations. The first is a simple extension of the local search phase, namely the acceptance of non-worsening moves within the neighbourhood N_E. The remaining two utilise the cyclic structure of the GRASP algorithm and use information from previous cycles. This use of feedback takes two forms. The first involves the use of a threshold on the preference cost during the search for a feasible solution, while the second is a diversification mechanism based on the values of the knapsack variables.

1. Plateau moves.
The introduction of the swap neighbourhood N_E increases the probability of generating moves that do not change the preference cost. It is possible that a sequence of such moves may lead to a point where further improvements can be made. On the other hand allowing all such moves may mean that there is not sufficient pressure for improvement. We therefore experimented with accepting such moves in N_E with a given probability.

2. Preference cost threshold.
According to the evaluation function given in Section 4, if the cover is improved, a move is accepted regardless of any increase in the preference cost. Thus the influence of the preference cost in the score functions in the construction phase may be destroyed. However, it is not sensible to reject all moves that increase the preference cost as these are often necessary in reaching feasibility. Instead we place a threshold on the increase. Such a threshold will be data dependent. For example, it must be greater than the preference cost of the best feasible solution. We use feedback from previous cycles to impose such a threshold in two ways. Firstly, we do not impose a threshold until we reach a cycle where a feasible solution has been obtained. For subsequent cycles we set a threshold of best_feasible_solution + Q, where Q is a dynamic threshold, also determined by the results of previous cycles. This means that we can iterate towards a value of Q that is large enough to allow the improvement phase to find feasible solutions, but is small enough to preserve the quality of the initial solution as far as possible. Q is initialised with a value q_0, and increased according to the function $Q = Q + \alpha Q$ at the end of each cycle, where α is a function of the cover_cost of the solution produced in that cycle. Thus Q increases whenever a cycle fails to find a feasible solution. A move is then accepted during the improvement phase if cover_cost improves and preference cost is less than best_feasible solution + Q.

3. Knapsack based diversification.
The final enhancement addresses the issue of diversification. It is designed to ensure that different cycles visit different areas of the solution landscape based on the numbers of nurses of each type and each grade allocated to days/nights. These values are precisely the sets Y_g of *y* variables returned by the knapsack calculations. To ensure diversity we need to ensure that no sets Y_g occur too frequently. This is achieved by generating initial solutions as before, and rejecting those relating to sets Y_g whose frequency is above a given threshold. This threshold is given by:

$Freq_{min} + V_Y/3$ where $Freq_{min}$ is the minimum frequency of all the Y_g vectors accepted so far and V_Y is the number of different vectors generated so far. As the construction process is very fast, this is feasible as long as the number of rejected solutions does not get excessive, as is the case if there are some vectors with a very small probability of being generated. This situation is addressed by modifying the above process so that if no pattern satisfying the acceptance conditions is sampled after 20 trials, then one of the solutions relating to the vector with the minimum frequency of these 20 is accepted.

7. EXPERIMENTS AND RESULTS

7.1 Initial results.

Our experiments are based on fifty-two datasets from a large UK hospital. These datasets have also been considered by Dowsland (1998), Dowsland and Thompson (2000), Aickelin and Dowsland (2000, 2004) and Burke et al. (2003b). In all cases, the optimal solutions are known.

Our initial experiments were designed to evaluate the effect of the knapsack over the full range of data sets and construction heuristics, using a range of values for the parameters *n* (size of the candidate list), w_c and w_p (the weights on the cover cost and the preference cost in the objective function), and to identify the most promising for further investigation. The role of *n* is to balance aggression and diversity within the construction with smaller values of *n* leading to more emphasis on the better scoring options and larger values of *n* leading to more diversity. The value of *n* will obviously be partially dependent on the number of options available at each stage of the construction process. This corresponds to around 60 shift patterns for the *cover* and *combined* heuristics, around 1,500 nurse-pattern pairs for the *holistic* heuristic and around 25 nurses for the *last chance* heuristic. Thus the range of *n* will vary according to the heuristic. For the *combined*, *holistic* and *last chance* heuristics we also need to define w_c and w_p. Table 1 shows the parameters used.

(Insert Table 1 around here)

Each construction method was run with and without the knapsack routine, giving 8 different GRASP implementations. For each one, every combination of parameters was run once on each of the 52 data sets, using $nrep = 100$. The local search phases used neighbourhoods N_C and N_S, selected with equal probability for each move until feasibility was achieved and using neighbourhood N_S only after that point. (N_E was not included at this stage as some of the variants frequently failed to find feasible solutions.) Since sampling is without replacement, the search was allowed to continue until the local optimum had been reached.

In order to be able to combine the results from different data sets, the results have been standardised by subtracting the optimal cost. Figure 2 shows plots of the percentage of feasible solutions against the average standardised preference cost for

these feasible solutions, over all 100 cycles, for each of the eight heuristics. Each point represents a single combination of n, w_c and w_p.

(Insert Figure 2 around here)

It is clear from Figure 2 that the percentage of feasible solutions increases significantly with the use of the knapsack routine. A closer examination of the data also revealed that of the 228 parameter combinations without the knapsack only 5 generate at least one feasible solution for all 52 datasets, and these have much poorer solutions in terms of the preference cost than those generated by heuristics utilising the knapsack algorithm. Thus there is clear benefit in incorporating the knapsack into the construction phase.

For all but the *last chance* constructions, there is evidence that smaller values of n outperform larger ones suggesting that the added diversity of larger n is less beneficial than a greedier approach. Of the knapsack heuristics, Figure 2 suggests that the *last chance* variant is the most promising. In order to confirm this, each variant was run 10 times for 100 cycles on each of the 52 data sets using a single set of parameter values. Unlike the above experiment where results were averaged over all cycles we now consider each block of 100 cycles as a complete GRASP run and record only the best solution over these cycles. As all four heuristics produce a high proportion of feasible solutions the parameter choices were based on the average preference score over the 52 data sets and are shown in Table 2, along with the mean standardised preference costs.

(Insert Table 2 around here)

The results confirm the superiority of the *last chance* heuristic. It has an average preference cost lower than all other heuristics and about 95% of the cycles create feasible schedules. These results are promising but only provide an overall view averaged over all datasets. In the next subsection we evaluate the performance in more detail on each dataset and compare the results with those obtained when the neighbourhood N_E and the enhancements suggested in Section 6 are included.

7.2. More detailed results.

Although the amalgamated results in Figure 2 suggest that our GRASP based on the *last chance* construction together with the knapsack performs reasonably well, the variability of the results is apparent when we consider each data set separately. Figure 3(a) shows the number of runs returning an optimal solution and the number returning a result within 3 of the optimum. It is clear that there are many data sets for which the optimum is not found in any of the 10 runs, and several where there are no solutions even close to the optimum.

(Insert Figure 3 around here)

Figure 3(b) shows the results of expanding the local search to include the extended neighbourhood N_E. This was implemented by replacing N_S with N_E once the first feasible solution in each cycle has been found. The results have clearly improved but still fail to match those of Dowsland and Thompson (2000).

Further improvements were obtained by the introduction of plateau moves. The probability of accepting moves that do not change the preference cost was varied from 0% to 100% in steps of 25%, with 75% giving the best results. These results are given in Figure 3(c) and improve the results to the point where all instances are solved to optimality at least once.

Finally Figure 3(d) shows the results of adding the two feedback related enhancements. For the preference cost threshold, parameter choices $q_0 = 10$ for the initial threshold value and $\alpha = 4$*cover_cost for the rate of increase were found to work well. The knapsack based diversification was added as stated in Section 6.

Comparing these results, with those obtained previously, we can see that the algorithm now finds the optimal solution in every case for all but two datasets and nine times out of 10 for the remaining two. This is a better average performance than that reported in Thompson and Dowsland and has been obtained without recourse to complex neighbourhoods, which may be data specific.

7.3 The role of the construction heuristic.

The results of Section 7.1 suggested that even after the inclusion of the knapsack routine there was a clear difference between the different construction heuristics in terms of solution quality. Given the dramatic improvements obtained by the use of the extended neighbourhood, plateau moves and feedback mechanisms it is no longer clear that this is still the case. All four heuristics were therefore rerun 10 times on each data set with all the enhancements included. Feasible solutions were produced in all runs, with optimal solutions being produced for all 10 runs using all four heuristics for all but 8 of the 52 data sets. The results for these 8 are shown in Table 3. Once again results have been standardised by subtracting the optimal cost.

(Insert Table 3 around here)

The performance of all four algorithms is much improved but the *last chance* heuristic remains the best and most consistent performer. This, combined with the vast improvements offered by the knapsack routine, show that it is worthwhile investing some thought, effort and experimentation into the choice of construction heuristic when developing a GRASP implementation. The results from this best algorithm are compared with those quoted by others in Table 4. This shows the best results obtained from GRASP, Tabu Search (Dowsland and Thompson (2000)), Genetic Algorithm (Aickelin and Dowsland (2004)) and Estimated Distribution Algorithm (Aickelin and Li (2007)). The rows marked IP are the optimal solutions obtained from the integer programming formulation (Fuller 1998), which required several hours of computational time for some datasets. GRASP outperforms the EDA in terms of preference cost and the GA in terms of both preference cost and feasibility. As stated above when we compare average performance, GRASP also outperforms the Tabu Search approach. The run-times required by GRASP are also impressive. All experiments were performed on a 1 MHz Pentium III PC with 256 MB of RAM, and required between 1 and 4 seconds of CPU time per cycle. Although this is slightly slower than the Tabu Search approach, it is significantly faster than an exact IP approach. The time required to check for knapsack feasibility is almost negligible,

while using plateau moves has a significant impact on the time taken by the improvement phase. When averaged over all 10 runs for all 52 datasets, using 75% plateau moves, construction takes approximately 2.5% of the solution time and checking the knapsack takes less than 0.005% of the construction time. Without plateau moves construction takes 85% of the solution time.

(Insert Table 4 around here)

8. CONCLUSIONS AND FURTHER WORK

This paper has considered a nurse-scheduling problem in which the conflict between feasibility and optimality is an issue that must be addressed by any successful solution approach. We have developed a GRASP approach that outperforms other heuristic approaches for this particular variant of the problem. The key to achieving this was the incorporation of a look-ahead facility based on a knapsack model of a relaxation of the problem, to ensure that the solutions produced by the construction phase were relatively easy to convert into feasible solutions during the improvement phase. The paper also adds to the body of evidence showing the importance of a suitable neighbourhood structure and evaluation/acceptance criteria during the improvement phase, as well as the benefit of a feedback mechanism to allow for a learning process from one cycle to the next. Feedback was used in two ways. Firstly, as the basis of a diversification strategy to ensure that the construction phase produced solutions from different areas of the search space, and secondly, to adjust the acceptance conditions for the improvement phase. Both played an important role in providing high quality solutions on the more difficult data sets.

The need to balance feasibility with optimality in all parts of the search was evidenced not only by the improvements resulting from the knapsack routine, but also by the introduction of the preference cost threshold with N_C and N_S. The results also show that the choice of construction heuristic is important, although it is not clear why the *last chance* decoder proved to be the best of the four. Three possible factors are its use of a look-ahead strategy, the more aggressive use of the best shift-pattern for each nurse, and the fact that the number of options at each stage in the construction is smaller than for the other three.

This work has shown that, for this particular problem, a GRASP approach can be very effective in solving highly constrained problems, as long as the feasibility of the solution is given sufficient consideration during the construction. It would be interesting to use a similar approach on other problems, in particular the use of relaxations or other problem specific information to guide the construction phase towards feasible, or easily repairable, solutions. We are currently working on these ideas with respect to a scheduling problem arising in the allocation of medical students to ‘firms’ of consultants in order to cover a given set of clinical disciplines.

REFERENCES

Abramson, D. (1991). "Constructing school timetables using simulated annealing: sequential and parallel algorithms", *Man. Sci*., **37**, 98-113.

Aickelin, U. and K. A. Dowsland (2000). "Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem," *J. Sched.*, **3**, 139-153.

Aickelin, U. and K. A. Dowsland (2004). "An Indirect Genetic Algorithm for a nurse-scheduling problem," *Comput. Oper. Res.* **31**, 761-778.

Aickelin, U. and J. Li (2007). "An Estimation of Distribution Algorithm for Nurse Scheduling," to appear in *Annals of Oper. Res.*

Aiex, R.M., S. Binato, and M.G.C. Resende (2003). "Parallel GRASP with path-relinking for job shop scheduling," *Parallel Computing* **29**, 393-430.

Bellanti, F., G. Carello, F. Della Croce, and R. Tadei (2004). "A greedy-based neighbourhood search approach to a nurse rostering problem," *Eur. J. Oper. Res.* **153,** 28-40.

Berrada, I., J. A. Ferland, and P. Michelon (1996). "A Multi-objective Approach to Nurse Scheduling with both Hard and Soft Constraints," *Socio-Economic Planning Sciences*, **30**, 183-193.

Binato, S., W. J. Hery, D. M. Loewenstern and M. G. C. Resende (2001). "A GRASP for job shop scheduling", *Essays and surveys in metaheuristics*, **15**, 81-100.

Brusco, M.J. and L. W. Jacobs (1995). "Cost Analysis of Alternative Formulations for Personnel Scheduling in Continuous Operating Organizations." *Eur. J. Oper. Res* **86**, 249-261.

Burke, E., P. Cowling, P. De Causmaecker and G.Vanden Berghe (2001). "A Memetic Approach to the Nurse Rostering Problem", *Applied Intelligence*, **15**, No. 3, 199-214.

Burke, E., P. De Causemaecker, S. Petrovic, and G. Vanden Berghe (2003a), "Variable Neighbourhood Search for Nurse Rostering Problems", In Mauricio G. C. Resende and Jorge Pinho de Sousa (eds.), *METAHEURISTICS: Computer Decision-Making*, Chapter 7, Kluwer, pp. 153-172.

Burke, E., P. De Causemaecker and G. Vanden Berghe (1999). "A Hybrid Tabu Search Algorithm for the Nurse Rostering Problem", in B. McKay et al. (eds.), *Simulated Evolution and Learning*, Springer, Lecture Notes in Artificial Intelligence, Vol. 1585, pp.187-194 .

Burke, E., P. De Causemaecker, G. Vanden Burghe and H. Van Landeghem (2004). "The State of the Art of Nurse Rostering," *J. Sched*., **7**, 441-499.

Burke, E., G. Kendall and E. Soubeiga (2003b). "A Tabu-Search Hyperheuristic for Timetabling and Rostering," *J. Heuristics* **9**, 451-470.

Cheang, B., H. Li, A. Lim and B. Rodrigues (2003). "Nurse rostering problems – a bibliographic survey," *Eur. J. Oper. Res.* **151**, 447-460.

Dowsland, K. A. (1998) "Nurse scheduling with tabu search and strategic oscillation." *Eur. J. Oper. Res.* **106**, 393-407.

Dowsland, K. A. and J. M. Thompson (2000). "Solving a nurse-scheduling problem with knapsacks, networks and tabu search," *J. Oper. Res. Soc.* **51**, 825-833.

Drexl, A. and F. Salewski (1997). "Distribution requirements and compactness constraints in school timetabling", *Eur. J. Oper. Res.* **102**, 193-214.

Ernst, T., H. Jiang, M. Krishnamoorthy, and D. Sier (2004). "Staff scheduling and rostering: A review of applications, methods and models," *Eur. J. Oper. Res.* **153**, 3-27.

Feo, T. A., M. G. C. Resende and S.H. Smith (1994). "A greedy randomised adaptive search procedure for maximum independent set," *Oper. Res.* **42**, 860-878.

Fleurent, C. and F. Glover (1999). "Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory," *INFORMS J. Computing*, **11**, 198-204.

Fuller E. (1998). "Tackling Scheduling Problems Using Integer Programming". Master Thesis, University of Wales Swansea, United Kingdom.

Gupta, S. R. and J. S. Smith (2006). "Algorithms for single machine total tardiness scheduling with sequence dependent setups", *Eur. J. Oper. Res.* **175**, 722-739.

Gutjahr, W. J. and M. S. Rauner (2007). "An ACO algorithm for a dynamic regional nurse-scheduling problem in Austria", *Comput. Oper. Res.* **34**(3), 642-666.

Isken, M. (2004). "An implicit tour scheduling model with applications in healthcare," *Annals Oper. Res.* **128**, 91-109.

Jaumard, B., Semet, F. and T. Vovor (1998). " A generalized linear programming model for nurse scheduling", *Eur. J. Oper. Res.* **107**, 1-18.

Laguna, M.and J. L. González-Velarde (1991). "A search heuristic for just-in-time scheduling in parallel machines," *J. Intelligent Manufacturing*, **2**, 253-260.

Laguna, M. and R. Martí (2001). "A GRASP for coloring sparse graphs", *Computational Optimization and Applications* **19**, 165-178.

Lim A., B. Rodrigues and X. Zhang (2006). "A simulated annealing and hill-climbing algorithm for the travelling tournament problem", *Eur. J. Oper. Res.* **174**, 1459-1478.

Martello, S. and P. Toth (1990). “Knapsack Problems,” Wiley, Chichester.

Meyer auf’m Hofe, H. (2000). “Nurse rostering as constraint satisfaction with fuzzy constraints and inferred control strategies”, *DIMACS Workshop on Constraints Programming and Large Scale Discrete Optimisation*, 67-100.

Michalewicz, Z. and D. B. Fogel. (2004). “How to Solve It: Modern Heuristics” Springer Verlag (Berlin and Heidelberg).

Moz, M. and M. V. Pato (2007). “A genetic algorithm approach to a nurse re-rostering problem”, *Comput. Oper. Res.*, **34**(3), 667-691.

Parr, D. and J. Thompson (2007). “Solving the multi-objective nurse scheduling problem with a weighted cost function,” to appear in *Annals of Oper. Res.*

Petrovic, S., G. Beddoe, and G. Vanden Berghe (2003). “Storing and adapting repair experiences in personnel rostering”, in E.K. Burke and P. De Causemaecker (eds), *Practice and Theory of Automated Timetabling*, Fourth International Conference, Gent, Springer, *Lecture Notes in Computer Science*, **2740**, 149-166.

Resende, M. G. C. (2001). “Greedy Randomized Adaptive Search Procedures (GRASP),” *Encyclopedia of Optimisation*, Kluwer Academic Press, **2**, 373-382.

Rosenbloom, E.S. and N. F. Goertzen (1987). “Cyclic nurse scheduling,” *Eur. J. Oper. Res.*, **31**, 19-23.

Thompson, J. M. and K. A. Dowsland (1996). “Variants of simulated annealing for the examination timetabling problem”, *Annals of Oper. Res.*, **63**, 637-648.

Wright, M. B. (1991). “Scheduling cricket umpires”, *J. Oper. Res. Soc.* **42**, 447-452.

Zhu, Y and A. Lim (2006). “Crane scheduling with non-crossing constraint”, *J. Oper. Res. Soc.* **57**, 1464-1471.

Tables

Table 1. The combinations of parameters investigated.

Heuristic	N	w_c	w_p	Total combinations
Cover	3,12,20	—	—	3
Combined	3,12,20	1,2,…,5	1,2,…,5	75
Holistic	3,10,60	1,2,…,5	1,2,…,5	75
Last chance	3,6,10	1,2,…,5	1,2,…,5	75

Table 2. Comparison of 4 knapsack heuristics.

Heuristic	n	w_c	w_p	Cost	% feasible cycles.
Cover + kn.	20	—	—	23	76
Combined + kn.	3	2	5	6	89
Holistic + kn.	3	1	3	4	87
Last chance + kn.	3	2	4	2	95

Table 3. Average cost of best solutions obtained from 10 runs of 100 GRASP cycles for each of the four heuristics using the extended neighbourhood, plateau moves, preference cost threshold and knapsack based diversification.

Data Set	*Cover*	*Combined*	*Holistic*	*Last chance*
2	0.2	0.1	0	0.1
20	0	13.5	9.7	0
25	0	1	0	0
31	3.6	1.2	3.2	0.4
33	0.8	0	0	0
42	0.1	0	0.6	0
44	0	10.8	5.1	0
46	0	0.6	0	0

Table 4. Comparison of best results for all 52 datasets.

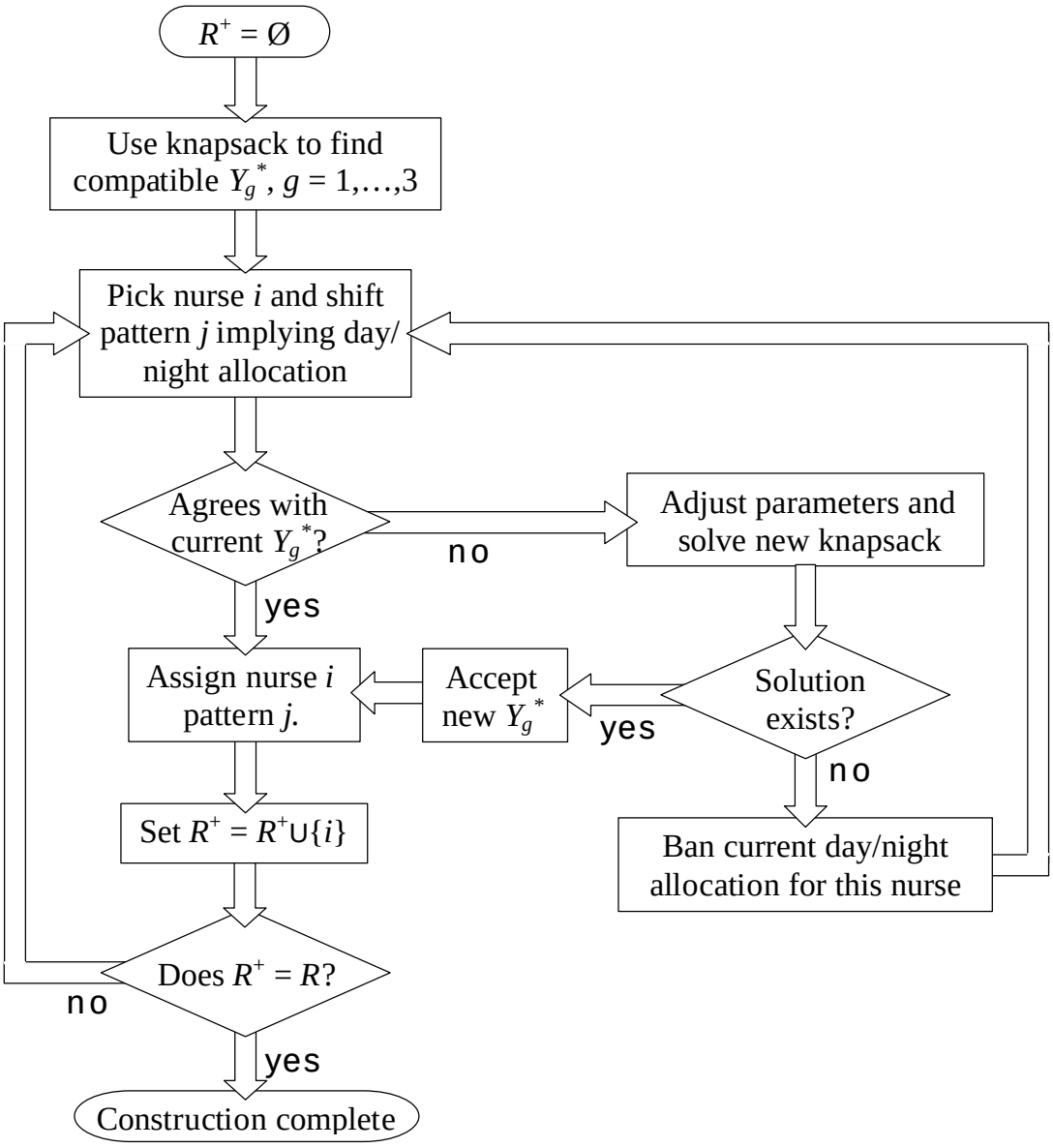
Dataset	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
IP	8	49	50	17	11	2	11	14	3	2	2	2	2	3	3	37	9	18	1	7	0	25	0	1	0	48
GRASP	8	49	50	17	11	2	11	14	3	2	2	2	2	3	3	37	9	18	1	7	0	25	0	1	0	48
Tabu	8	49	50	17	11	2	11	14	3	2	2	2	2	3	3	37	9	18	1	7	0	25	0	1	0	48
GA	8	50	50	17	11	2	11	15	3	4	2	2	2	3	3	38	9	19	1	8	0	26	0	1	0	48
EDA	8	56	50	17	11	2	14	15	14	2	2	3	3	4	4	38	9	19	10	7	1	26	1	1	0	52

Dataset	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52
IP	2	63	15	35	62	40	10	38	35	32	5	13	5	7	54	38	22	19	3	3	3	4	27	107	74	58
GRASP	2	63	15	35	62	40	10	38	35	32	5	13	5	7	54	38	22	19	3	3	3	4	27	107	74	58
Tabu	2	63	15	35	62	40	10	38	35	32	5	13	5	7	54	38	22	19	3	3	3	4	27	107	74	58
GA	2	63	141	42	166	99	10	48	35	41	5	14	5	8	54	38	39	19	3	3	4	6	30	211	-	-
EDA	28	65	109	38	159	43	11	41	46	45	7	25	8	8	55	41	23	24	6	7	3	5	30	109	171	67

N.B. The GA did not find any feasible solutions for datasets 51 and 52.

Figures

Figure 1. Flow chart showing implementation of the knapsack model.



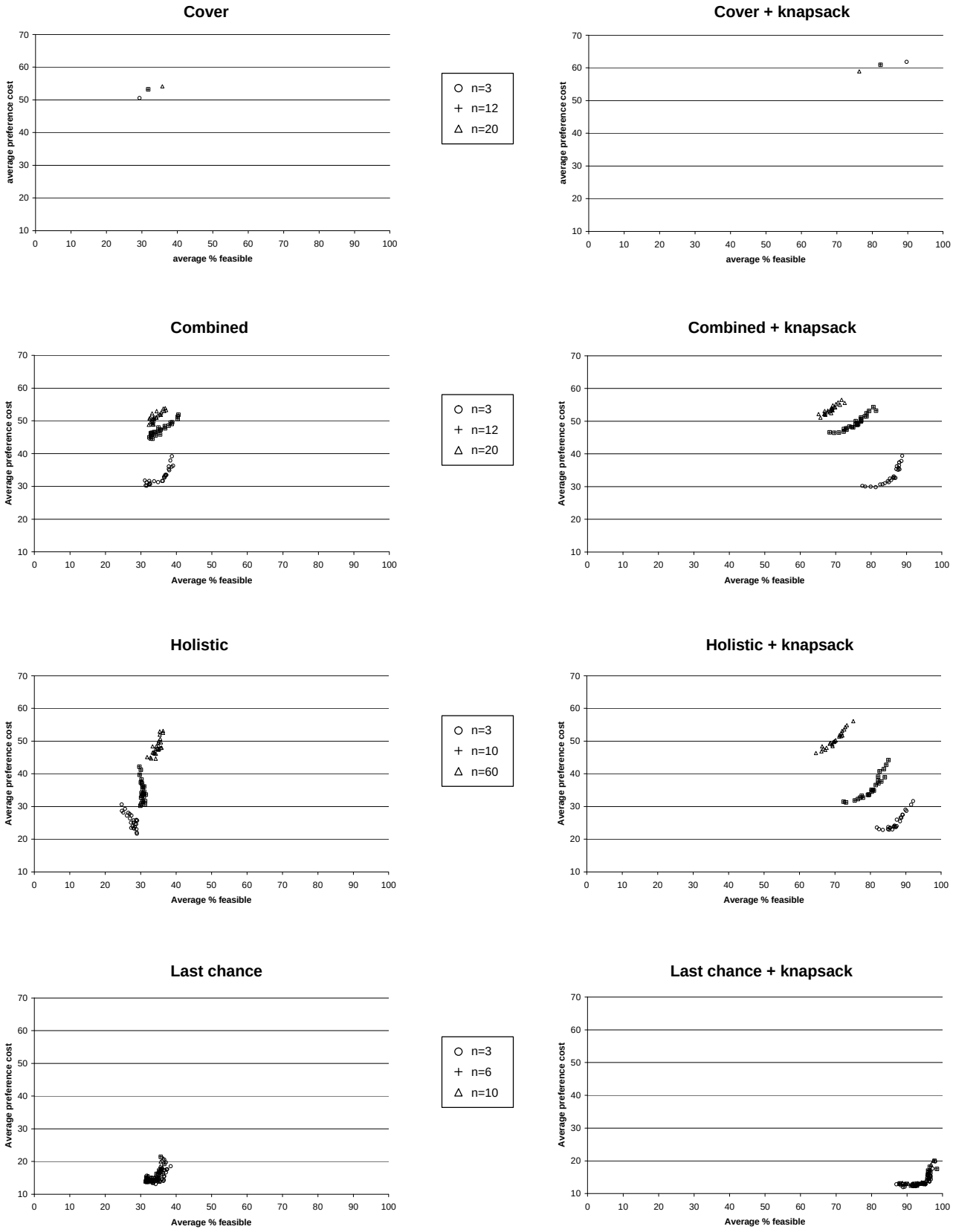


Figure 2. Plots of feasibility against average preference cost for the eight heuristics.

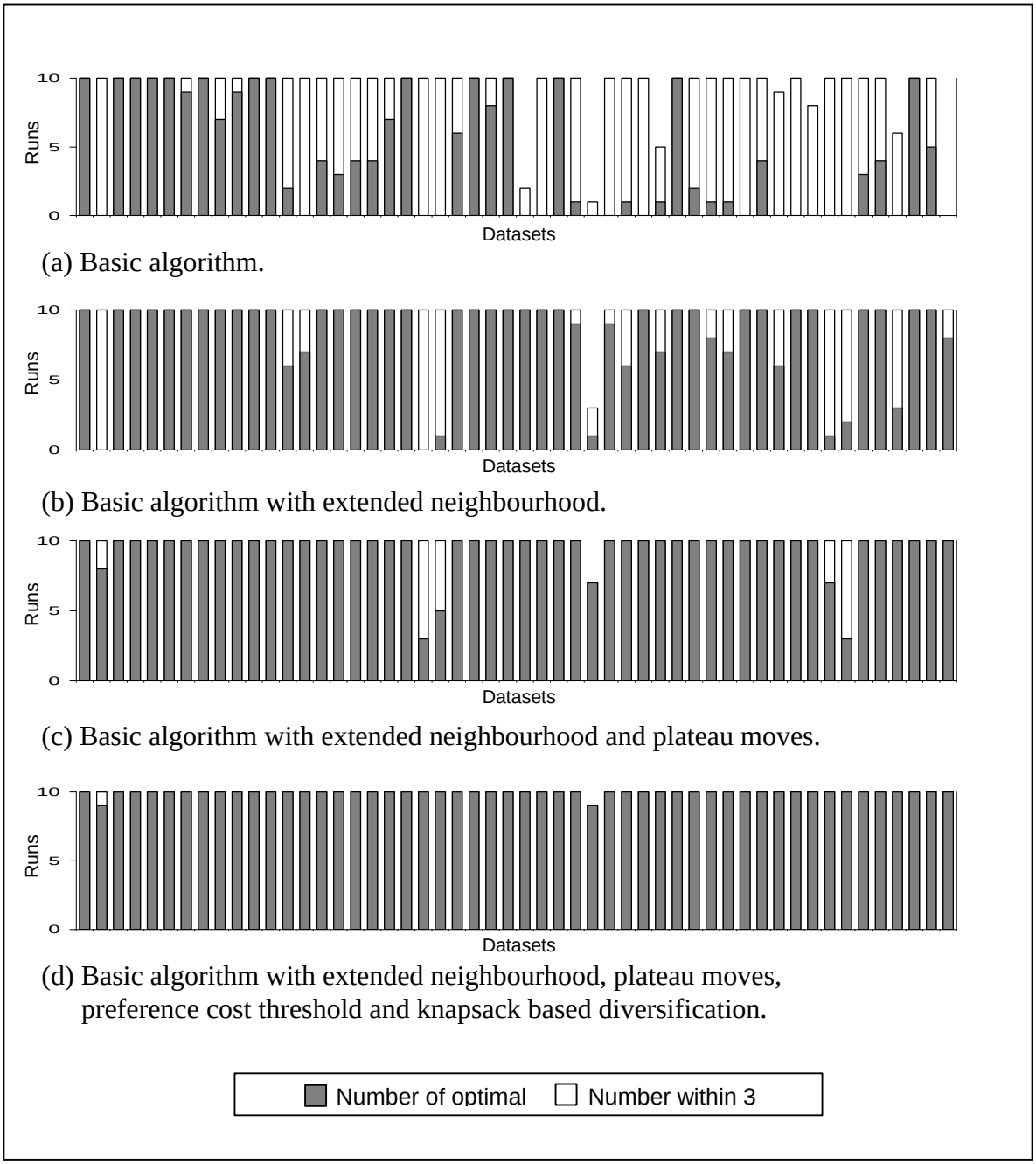


Figure 3. A breakdown of the results for all 52 datasets from 10 runs using different local search extensions.

Appendix One. Worked example of the knapsack look-ahead procedure.

Suppose we have a situation in which the set of nurses as yet unallocated who are free to work days or nights (i.e. the set R - R') are as given in Table 5a, the shifts that they must cover are as given in Table 5b, and the solution to the knapsack problems Cons($P(R')$) for the previous iteration is given by

$$Y^* = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 2 & 1 & 0 \end{pmatrix}$$

Table 5a. Details of unallocated nurses

Type index (t)	1	2	3	4
Days worked (d_t)	3	5	4	3
Nights worked (e_t)	3	4	3	2
Number grade 1	0	2	2	0
Number grade 2	1	3	2	2
Number grade 3	1	4	3	3

Table 5b. Cover Requirements

Grade (g)	Days required	Nights required
All (g=3)	26	14
1 and 2 (g=2)	19	10
1 (g=1)	6	7

Now assume that the greedy construction process selects nurse i to work pattern j, where i is a type 4, grade 2 nurse and j is a shift pattern covering nights. First we check the current Y^* matrix to see if our choice is compatible with the current knapsack solution. As the matrix represents a possible feasible allocation of nurses to nights, and $y_{24}^* = 0$, the current solution is not compatible. Thus it is necessary to resolve the knapsack problems to determine whether or not the suggested allocation is feasible i.e. we need to search for a solution to Cons($P(R' \cup \{i\})$) where i is assumed to be on nights. We therefore reduce the number of type 4 grade 2 nurses available by 1 and subtract e_4 (i.e. 2) from the night requirements for g = 2 and g = 3.

The cumulative number of nurses of each type available, n_{tg},is now given by the

$$\text{matrix } N = \begin{pmatrix} 0 & 2 & 2 & 0 \\ 1 & 3 & 2 & 1 \\ 1 & 4 & 3 & 2 \end{pmatrix}$$

The target values E_g are as given by the new night requirements (7,8,12) and the capacities of the knapsacks defined by the right-hand sides of constraints (5_g) are given by the vector (10,10,15). The tree relating to the search for a feasible solution is shown in Figure 4.

At the top level of the search, node A represents the knapsack problem KS_3 relating g = 3, where the target value E_3 = 12 , knapsack capacity = 15 and the upper bounds on

the variables are given by the bottom row of matrix *N*. The top segment of the Figure, i.e. the region denoted g=3, shows a tree search for all feasible solutions to this problem with objective value at least 12, using bounds based on the linear programming relaxation of the problem to prune the tree. Note that the types have been ordered in decreasing e_t/d_t order and the branches relating to each variable are ordered in decreasing value order. This improves the efficiency of the search.

The first solution is found at node B and is given by $Y_3^* = (1,1,1,1)$ i.e. 1 nurse of each type working nights and the remainder on days. Having reached the solution we spawn a new tree to search for solutions to the problem $(KS_2|Y_3^*)$ starting at node B′. The target value is now 8 and the knapsack capacity is 10. The vector of upper bounds (given by the right-hand side of constraint set (8_2)) = (1,1,1,1) and the vector of lower bounds (given by the right-hand side of (9_2)) = (1,0,0,0). Thus y_{12} is fixed at $y_{12} = 1$. Substituting the fixed value leaves us with a problem in the three remaining variables with an adjusted target of 8-3 = 5 and adjusted knapsack capacity of 10–3 = 7. The tree search corresponding to this problem starts at root node B′ and the first feasible solution is found at E corresponding to $Y_2^* = (1,1,0,0)$.

We now spawn a new tree to search for solutions to $(KS_1|Y_2^*)$. The target is 7 and the knapsack capacity is 10. The upper bounds are given by (0,1,0,0) and the lower bounds by (0,0,0,0). Thus all variables except y_{32} are fixed at 0 and as y_{32} has an upper bound of 1 the problem is clearly infeasible as we cannot meet the target value. We therefore return to node E and continue searching the tree corresponding to $(KS_2|Y_3^*)$. A second solution at this level in the hierarchy is found at F with $Y_2^* = (1,0,1,1)$. This spawns a new problem $(KS_1|Y_2^*)$ with upper bounds (0,0,1,0) and lower bounds (0,0,0,0). Once again we have a problem in just one variable which is clearly infeasible so control is returned to node F. Continuing the search at this level shows that there are no more feasible solutions to $(KS_2|Y_3^*)$ and control is returned to point B and the search for solutions to problem A continues.

The next solution is at C corresponding to $Y_3^* = (1,0,3,0)$. This spawns a new problem for $(KS_2|Y_3^*)$ at C′ with upper bounds (1,0,3,0) and lower bounds (1,0,2,0). This becomes a problem in y_{23} with a target of 5 and knapsack capacity = 7. This problem is clearly infeasible as the lower bound on y_{23} causes the knapsack capacity constraint (5_2) to be violated. Control therefore returns to C. The next solution occurs at D corresponding to $Y_3^* = (0,3,0,0)$ with upper bounds (0,3,0,0) and lower bounds (0,2,0,0). The tree for the resulting problem in y_{22} starts at D′ yielding a feasible solution at G corresponding to $Y_2^* = (0,2,0,0)$. This spawns problem $(KS_1|Y_2^*)$ at G′ with upper bounds (0,2,0,0) and lower bounds (0,1,0,0) and yielding a feasible solution at H. Thus there is a feasible solution that is compatible over all grades given by

$$Y^* = \begin{pmatrix} 0 & 2 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 3 & 0 & 0 \end{pmatrix}$$

We therefore make the allocation of nurse i to pattern j. Had the search failed to find a feasible solution nurse i would have been added to the set R^{fixed} (tagged as days only), the matrix Y^* restored to its previous set of values and n_{24} and n_{34} reduced by 1 and D_2 and reduced by 3.

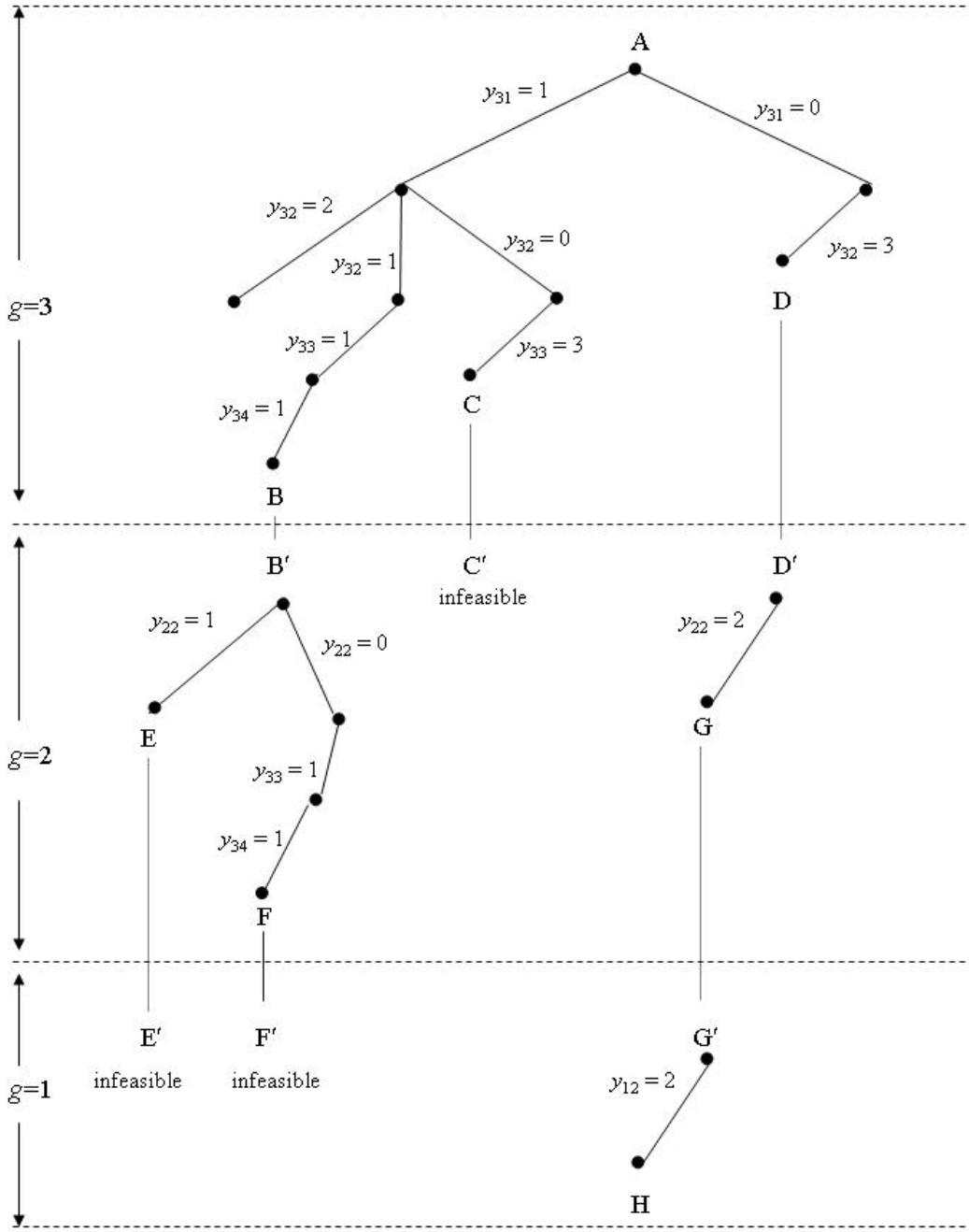


Figure 4. Search tree for finding a feasible solution