Curtois, Timothy (2008) Novel heuristic and metaheuristic approaches to the automated scheduling of healthcare personnel. PhD thesis, University of Nottingham.

Novel Heuristic and Metaheuristic Approaches to the Automated Scheduling of Healthcare Personnel

Timothy Curtois

A thesis submitted to The University of Nottingham for the degree of Doctor of Philosophy

July 2007

Contents

List of Figures

List of Tables

Abstract

This thesis is concerned with automated personnel scheduling in healthcare organisations; in particular, nurse rostering. Over the past forty years the nurse rostering problem has received a large amount of research. This can be mostly attributed to its practical applications and the scientific challenges of solving such a complex problem. The benefits of automating the rostering process include reducing the planner's workload and associated costs and being able to create higher quality and more flexible schedules. This has become more important recently in order to retain nurses and attract more people into the profession. Better quality rosters also reduce fatigue and stress due to overwork and poor scheduling and help to maximise the use of leisure time by satisfying more requests. A more contented workforce will lead to higher productivity, increased quality of patient service and a better level of healthcare.

Basically stated, the nurse rostering problem requires the assignment of shifts to personnel to ensure that sufficient employees are present to perform the duties required. There are usually a number of constraints such as working regulations and legal requirements and a number of objectives such as maximising the nurses working preferences. When formulated mathematically this problem can be shown to belong to a class of problems which are considered intractable. The work presented in this thesis expands upon the research that has already been conducted to try and provide higher quality solutions to these challenging problems in shorter computation times.

The thesis is broadly structured into three sections.

1) An investigation into a nurse rostering problem provided by an industrial collaborator (ORTEC): A hybrid heuristic ordering and variable neighbourhood

search is developed and tested using commercial nurse rostering data. The efficiency and strength of the approach is demonstrated through experimental comparisons with an existing commercially implemented genetic algorithm. The genetic algorithm is part of ORTEC's Harmony software package and it operates successfully in a number of real world scenarios. The results of the research presented in this thesis are now incorporated in the latest product versions of Harmony.

2) A framework to aid research in nurse rostering: A number of research tools have been created and are presented in this thesis. They have been made publicly available (including source code) in order to facilitate the establishing of benchmark nurse rostering instances and results. Practically oriented benchmark instances have been requested by the nurse rostering research community for some time. This work fills the void and provides a solid foundation for future research.

3) The development of a number of advanced algorithms for solving highly complex, real world problems. A number of search neighbourhoods previously used in local search and metaheuristic approaches to nurse rostering are examined and tested using the benchmark data sets. The results of this investigation are then used to create a variable depth search which effectively chains together moves and swaps from the most successful neighbourhoods. A variety of heuristics were developed to efficiently find improving chains. The algorithm also accepts a predefined computational time limit and dynamically adjusts in order to use its time more effectively. When compared against previously published algorithms (even when dynamically adjusting to their run times), the variable depth search is shown to be very successful.

In an attempt to produce even higher quality solutions, the variable depth search was incorporated as the improvement method into an evolutionary algorithm: scatter search. The scatter search was found to be competitive with other evolutionary approaches and particularly strong on some problem instances.

Acknowledgments

This work presented in this thesis was conducted as part of a larger project involving several researchers. The group also included Edmund Burke, Rong Qu, Jingpeng Li, Greet Vanden Berghe, Gerhard Post and Bart Veltman. Each member had some involvement with the research presented here. Edmund Burke's contributions were made in capacity as supervisor. Gerhard Post and Bart Veltman's main contributions were in the research that was conducted in collaboration with ORTEC (Chapter 3 and [50]). Their support was not only in practical matters such as relating to using Harmony but also in more theoretical and experimental aspects. This was mostly provided through in depth discussions and feedback on presentations and draft papers. Laurens Fijn van Draat (another ORTEC researcher) also provided significant support with Harmony. Rong Qu was involved to some degree in all the research presented here. Again her contributions were particularly through improving the experimental design and presentation of the research. Greet Vanden Berghe collaborated on the work presented in Chapters 4, 5 and 6. It is also important to acknowledge that a lot of this research builds on Vanden Berghe's earlier work [238]. Jinpeng Li joined the project midway and conducted a significant amount of work but had less involvement with the research presented in this thesis. His integer programming model of the ORTEC problem is included in Chapter 3. The project was supported by EPSRC grant GR/S31150/01.

1 Introduction

Personnel scheduling belongs to a wide class of timetabling problems which includes educational timetabling [67, 68, 212], sports timetabling [95] and transport timetabling [154]. Personnel timetabling problems are found in a wide variety of industries and environments. The employees can range from airline crew [19] to ambulance officers [97], from factory [38] to fast food restaurant workers [121], from police [230] to call centre staff [108] and many more. This thesis is concerned with personnel scheduling in healthcare, in particular, nurse rostering. The nurse rostering problem is not only one of the more commonly occurring problems (the UK's NHS alone currently employs approximately 400,000 nurses [236]) but it is also one of the most complex. This high complexity is due to a number of factors, some of which (but rarely all) may be found in other employee scheduling problems. These factors include:

- Hospitals operate for twenty four hours a day, seven days a week. This introduces a number of legal constraints and working preferences relating to night shifts, minimum rest times, working on weekends and national holidays and so on.

- The workforces consist of nurses with varying skills and grades which need to be considered when constructing rosters.

- A variety of shifts. Even the more basic problems usually have a minimum of three shift types (e.g. early, late and night). More frequently, there are a

number of other shift types to assign, each with varying durations and associated constraints.

- Large numbers of employees.

- Cover requirements may not be uniform but vary from day to day.

- Long planning horizons. They can range up to twelve weeks or even a year for some instances.

- Many, often conflicting constraints and objectives. For example, constraints or objectives relating to:
  - Cover requirements.
  - Day on/off and shift on/off requests.
  - Minimum and maximum length stretches of days on, off, or specific shifts.
  - Minimum and maximum hours and/or shifts worked during certain periods.
  - Shift rotations.
  - Desirable and undesirable work patterns.
  - Minimum and maximum numbers of specific shift types (possibly during certain periods).
  - Minimum and maximum ratios of shift types worked.
  - Tutorship or oppositely ensuring certain employees do not work at the same time.

These features make the problem not only hard to solve but also to model. The effort required is worthwhile though when high quality rosters are produced, as these provide a number of significant benefits which can be outlined as follows:

- **Reduced hospital expenditure**. Nurse salaries are a significant proportion of a hospital's running costs. Better scheduling can reduce this expense in a number of ways:

  - Through minimising over coverage (not assigning more nurses than are required for a shift).
  - Via cutting the reliance on expensive agency nurses to fill gaps in schedules when it may appear to be the only solution.
  - Through increased work performance due to reduced fatigue and stress amongst nurses caused by poor scheduling (e.g. overwork, insufficient rest, bad shift combinations etc).

- **Higher staff retention and a recruiting aid**. A number of countries have experienced a reduction in the number of people training to become nurses and/or an increase in the number of nurses leaving the profession. As the populations of these countries age, the demand for healthcare will increase and these nurse shortage problems will become more acute. In order to encourage more people to become nurses and to reduce the number of people leaving the nursing profession, various initiatives have been proposed. One of these is to allow more part time contracts and to provide the nurses with more

flexibility and input on when they work. This allows, for example, more parents with young children to remain in nursing.

- **Reduction in absenteeism and tardiness**. Many organisations incur a reduction in productivity due to staff absenteeism and tardiness. Hospitals are no exception. The reasons for nurses arriving late or taking days off are various. This can partly be attributed, though, to dissatisfaction with their schedules or fatigue due to bad scheduling. This can be reduced through better rostering and giving the nurses more say in their work patterns. For example, a nurse is less likely to be absent for a shift which they actually requested.

- **Personal preferences**: Increasing the nurses' satisfaction with their schedules by providing them with more choice and allowing them to better plan and use their leisure time can also increase general morale levels. This, in turn, can lead to benefits such as higher productivity and lower staff turnover with its associated costs.

- **Increased patient safety and quality of service**. Nurses are able to spend more time with patients if they are not overworked or the ward is not understaffed as a result of poor scheduling. In the worst case, fatigue and stress can result in medical error endangering the patient's health and safety and damaging the hospital's reputation.

Constructing high quality rosters, however, is a challenging process which is made more difficult by providing increased flexibility and a variety of work contracts. In many hospitals though the schedules are still produced by hand. This

unwelcome and time consuming assignment typically falls to senior nurses and can distract them from their primary duty of looking after patients. Often they end up reluctantly taking the work home. The task can also be stressful and frustrating. The planner is presented with a number of requests and scheduling requirements which can rarely be fully satisfied. They are required to ensure that all legal and binding rules are obeyed whilst trying to grant as many requests as possible. Often, unfavourable shifts must be assigned and requests denied whilst trying to maintain fairness and impartiality.

By using a computer to automatically create schedules, it is possible not only to remove this chore and the associated costs but also to create much higher quality rosters. The scheduling is performed with a fraction of the effort and the schedules are usually better than expert human planners can achieve. Legal requirements can be checked, which a planner may miss, and more requests and working preferences are satisfied. The nurse that was previously assigned this work now has more time to care for patients. This is especially noticeable when regular rescheduling is required due to staff sickness and unpredicted absences. As the schedules are computer generated, the nurses also feel less victimised if they believe that their schedule is worse than a colleague’s.

Using a computer in the scheduling process provides a number of additional benefits other than reduced labour and better rosters. These include:

- Collecting management statistics and report generation (e.g. average hours worked per week, the number of sick days etc).
- Linking the schedules to payroll and accounting systems.

- Publishing the schedules on the web or sending them to the nurses via email.
- Allowing the nurses to make their requests via web interfaces.

Nurse scheduling software is also often used as a decision support tool, allowing planners to test different scenarios. For example, how would the quality of the schedules change if nurses were assigned to or removed from wards?

1.1 Research Objectives

The research presented in this thesis focuses on algorithms used to automatically construct nurse rosters. Most published models of the nurse rostering problem are NP-hard. For the interested, most nurse rostering problems can be shown to be NP-hard through a polynomial-time reduction of the set covering problem or 3SAT [157]. It is likely (although not proven) that P and NP are not the same. This actually means that there are instances of the problem for which no algorithm can guarantee to produce the *very best* (whatever definition of *best* we might employ) solution within a practical time limit. For example, it is possible to produce an instance of the nurse rostering problem which, even if the best known algorithm for solving it used the computing power of every computer on Earth, it may still take many millions of years to produce the *best* solution. In practice, we need to heuristically produce solutions to the problem. This means that optimality cannot be guaranteed. However, for many real world instances, very high quality solutions can be produced within feasible time limits. This thesis is concerned with analysing existing approaches and developing improved methods of achieving this. The main objectives of this PhD programme can be summarised as follows:

- Review and critically examine the development of nurse rostering problem solvers from initial related work in the 1950's to the state of the art in 2007.

- Collaborate with our commercial partner, ORTEC, and focus on solving real world problems. ORTEC is a supplier of software products and consulting in the field of advanced planning and scheduling. They have direct experience of working with (and have provided) practical and challenging nurse rostering problems.

- Develop improved, powerful and robust search methodologies to address real world problems.

- Thoroughly and fairly analyse these new methods and evaluate against the state of the art and commercial strength algorithms through structured and well designed experimentation.

- Create a variety of benchmark nurse rostering problems based on real world scenarios. This includes developing a format for describing complex nurse rostering problems in order to share and make publicly available the benchmark data sets.

- Provide research tools and source code for working with the nurse rostering problems. For example, parsers, data structures, user interfaces, visualisations. This will provide a foundation for future research and encourage more research into highly practical nurse rostering problems.

- Investigate and test a variety of heuristics and search neighbourhoods, with a focus on the balance between intensification and diversification during computational search.

- Examine trade offs between computation time/solution quality and develop methods which adapt to predefined, maximum run time parameters. These algorithms will more accurately meet user requirements and scale with future increases in technology. For example, algorithms which, as recently as a few years ago, took minutes can now be performed in seconds on today's average PCs. However, there is no easily accessible mechanism provided for extending these searches when a longer computation time is acceptable in order to produce better solutions.

These objectives were defined in order to answer the main research question of the thesis: To what extent can the state of the art metaheuristic approaches to nurse rostering be improved upon, particularly to meet today's real world needs in complex operating environments? In order to answer this question the following hypotheses will be tested:

Hypothesis 1: Based on recent advances in metaheuristic approaches to nurse rostering, improvements can be made on the genetic algorithm in ORTEC's software Harmony. The genetic algorithm is a commercial strength algorithm that has operated successfully for a number of years. However, since its development a large amount of research on metaheuristics has been conducted. It may be

possible to use some of this knowledge to develop a better algorithm. The new algorithm will be compared against the genetic algorithm using statistically evaluated tests.

Hypothesis 2: The research community will significantly benefit from the development of a collection of real world benchmark data sets. It will not be possible to completely test this hypothesis as it would have to be tested over a long time frame. The data sets and related software will be developed though in order to initiate this test.

Hypothesis 3: Very large scale neighbourhood search techniques can be successfully applied to nurse rostering. These methods have been very effective in other problem domains (see [9] for a recent survey) and although their application to nurse rostering has been limited, there is no obvious reason why they should not succeed here. The approach will be compared against previously published algorithms and the results statistically analysed.

Hypothesis 4: A successful time predefined algorithm can be developed for the nurse rostering problem. Again, these algorithms have been useful and effective on other problems [47, 48]. An investigation will be conducted to see if a similar approach be developed for nurse rostering.

Hypothesis 5: A class of search neighbourhoods that are known to be very effective for the nurse rostering problem but are computationally intensive to use, can now be applied equally successfully but with much shorter computation

times. A number of search neighbourhoods have been identified recently which can be used to produce very high quality rosters. The disadvantage of these neighbourhood operators is the large computation times they require. An investigation will be conducted to discover whether recent increases in computing power have made it possible to use these neighbourhoods more intensively within practical execution times. The efficacy of these neighbourhoods will also be analysed.

Hypothesis 6: If a successful very large scale neighbourhood search algorithm can be developed, it will be possible to incorporate it in a novel evolutionary algorithm for increased robustness. A number of evolutionary algorithms have previously been developed for nurse rostering. Scatter search is an evolutionary algorithm which has worked well on other problems but had little application to nurse rostering. A scatter search which uses a very large scale neighbourhood search technique as the improvement method between generations could be very effective. The scatter search will be compared against other metaheuristic and evolutionary approaches to test this hypothesis.

1.2 Thesis Structure

Chapter 2 examines different categories of nurse scheduling problems. Terms and expressions that are frequently used in nurse scheduling contexts are also defined. The majority of the publications concerning automated nurse rostering that have appeared over the past forty years or so are then reviewed. The papers are grouped according to the methodologies used.

Chapter 3 presents a nurse rostering algorithm developed to solve a commercial problem provided by ORTEC. The approach hybridises heuristic ordering with variable neighbourhood search. When compared with the genetic algorithm of ORTEC's Harmony software, the method proved to be much more effective. The results of this research are incorporated in the latest product versions of Harmony.

Chapter 4 describes a software framework developed to provide benchmark nurse rostering problems and reduce the gap between research and practice in nurse rostering. A number of tools have been created to help, encourage and strengthen research in automated nurse rostering. This work provides a solid platform for future research.

Chapter 5 begins by testing and analysing the efficiency of a variety of neighbourhood operators that have been used in local search and metaheuristic approaches to solving nurse rostering problems. The analysis is performed using the real world, benchmark problems introduced in chapter 4. A variable depth search is then developed based on the results of the investigation. The algorithm heuristically chains together moves and swaps which define the more effective search neighbourhoods. A number of heuristics for creating these chains were developed and the results of experiments (conducted to identify the best ones) are presented. As end users vary in how long they are willing to wait for solutions, a particular goal of this research was to create an algorithm that accepts a user specified computational time limit and uses it effectively. When compared against previously published approaches the results show that the algorithm is very successful.

Chapter 6 presents an investigation into combining the variable depth search into an evolutionary algorithm in order to provide higher quality solutions and a more

robust approach. Scatter search is investigated due to its success in other problem domains and its potential promise despite the scarcity of any previous applications to personnel scheduling. A number of different parameters and settings are tested and the best setup compared to the variable depth search on its own and the successful, hybrid memetic algorithm of Burke et al. [49].

Chapter 7 concludes the thesis. The contributions of the research are summarised and possible future directions discussed.

2 Literature Review

Personnel scheduling has received a large amount of research coverage. A recent annotated bibliography of employee scheduling compiled by Ernst et al. [98] contains roughly seven hundred references dating from as early as 1954 [83, 96]. The size of the bibliography gives an indication of the breadth of study that has been conducted in this area, especially when noting that it is far from exhaustive. A large proportion of the studied personnel scheduling problems come from healthcare organisations such as hospitals and clinics and require the scheduling of nurses. The significant presence of nurse scheduling problems is due to their importance, scientific challenge and complexity (as discussed in chapter 1). The most commonly researched nurse scheduling problems can be broadly placed into one of two categories: staffing and rostering. In the first section of this chapter these two general problems are introduced and discussed. The second section of this chapter reviews the models and methods that have been used to solve nurse rostering problems of varying complexity, in a number of environments around the world. As this thesis is primarily concerned with the nurse rostering problem, this is where we will focus most of our attention in this chapter.

The publications reviewed and discussed in the second section of the chapter are categorised according to solution methodology in a similar manner to the literature review of Burke et al. [60] although with the addition of two new categories (hyperheuristics and case-based reasoning). Many of the papers reviewed here are also reviewed in [60] and [98]. To provide a new contribution to the research community though, an effort has been made to highlight key and interesting points which have not been previously mentioned. Although this was

not always possible, on the whole there is a large number amount of additional information. In the years following the publication of [60] and [98] a number of related publications have appeared. These have also been reviewed here. The conclusions drawn in [60] are also shared by the author here. In fact much of the research presented in this thesis was directed by and based upon these conclusions. For example, Burke et al. conclude that parameter-less or time-predefined algorithms would have significant benefits as would the development of benchmark data sets.

2.1 Vocabulary

Before continuing it is necessary to provide a very short glossary of some of the key terms and expressions that are frequently used in the nurse scheduling literature (but which are often assumed to be known). Many specialist subjects develop a vocabulary or jargon which can make them unintelligible to outsiders and nurse scheduling is no exception. There is one distinction though which is non-standard and which will be used throughout this thesis. That is, the difference between schedule and roster. In practice, the two words are often used interchangeably. However, a schedule is sometimes also used to describe an individual work pattern for a single employee as well as to describe an entire roster. To avoid confusion, in this thesis, the term *schedule* will only be used to represent a single employee's work pattern whereas *roster* will be used to represent a set of employees' work patterns.

Agency and **bank nurses** are temporary staff that may be employed to cover gaps due to absences in the permanent employees.

Cover requirements represent the number of nurses required at work each day or at specific times (i.e. during shifts) each day. This may also be called **shift or coverage demand**.

Float nurses move between units and departments to cover gaps in staff cover due to absences e.g. sick, vacation leave etc.

Hard constraints are rules which must be satisfied for the roster to be feasible. They may also be called **binding constraints** or **imperative planning rules**.

Scheduling horizon is the time period over which the roster is provided. It may also be called the **planning period**.

Shift rotation is the situation when an employee works a different shift to the one they worked previously. Depending on whether the start time is earlier or later than before, it is called backward or **forward rotation**.

Soft constraints are rules which should ideally be satisfied but in order to provide a feasible solution may be broken. They may also be called **non binding constraints, floppy constraints, preference planning rules** or **aversion costs**. Soft constraints are often given priorities which are relative to each other. If the priorities are assigned using weights then a higher priority constraint may be violated if it means a number of lower priority constraints will be satisfied.

Split weekend is the situation where an employee works on only one day of the weekend (i.e. Saturday or Sunday). A **complete weekend** is the opposite (i.e. the employee works on neither or both days of the weekend).

A **stand alone shift** or stand alone day is an *off-on-off* work pattern. It may also be called an **isolated work day**.

A **work pattern** is an individual's schedule over a planning period. That is, the days they have on and off and possibly also the shifts they have on the days on. Predefined patterns may also be called **stints**.

2.2 Staffing, Demand Modelling and Workforce Scheduling

One of the first steps in the entire employee scheduling process is to determine the required workforce size and structure. That is, to identify how many employees are required and which skills are needed over a specific period in order to achieve certain goals. This is generally known as the staffing, demand modelling or workforce scheduling problem. In most scenarios the goals tend to be to minimise costs (wages) and to maximise service levels. In the healthcare organisation context, for example, the standard of service may be measured by nurse to patient ratios or whether certain requirements can be satisfied. To model and solve these problems, a number of predictions may have to be made (possibly based on past data). For example, the expected number of patients and the severity of their illnesses, the availability and cost of bank/agency nurses, predicted absenteeism, sick and annual leave and the available budget. Employee productivity may also need to be estimated for various workforce sizes (if there is

understaffing, then individual productivity may increase but may be unsustainable).

Although workforce scheduling problems do occur in hospitals they tend to be solved less frequently due to the preference for longer term contracts and the adverse effects of a constantly changing workforce. They are more common in environments with rapidly changing service requirements such as call centres (see [98, 99]).

2.3 Rostering

In comparison to the staffing problem, rostering is required in hospitals more frequently. Like the staffing problem though, many different methods have been used to solve it. There are two general approaches to nurse rostering: cyclical and non cyclical scheduling. Each method has its advantages and disadvantages and is suitable for different situations.

2.3.1 Cyclical Scheduling

In cyclical scheduling (sometimes called **rotational scheduling**) a single schedule for a fixed planning period is created that can be assigned to all employees. The schedule is designed so that it restarts once the end of the planning period is reached (hence, the term cyclical). The schedule is offset (e.g. by a week) before assigning to each employee. This ensures that the cover requirements, which need to be considered when creating the schedule, are satisfied. Cyclical scheduling has a number of advantages. As everyone has the same schedule, nurses cannot feel their schedule is worse than anyone else's. Secondly, once a good cyclical schedule is produced, it can be reused until the scheduling requirements change. In theory, this means the nurses can know their

schedule a long time in advance and therefore plan holidays and other events for days off. Cyclical scheduling does have disadvantages though. For example, it becomes a more challenging problem when cover requirements are less uniform or fluctuate from week to week. The largest drawback with this approach, however, is the fact that individual requests and preferences are very difficult to take into consideration and satisfy. As such, cyclical scheduling is less popular with nurses and planners. Examples of cyclical scheduling approaches include [8, 168, 192, 216].

2.3.2 Non Cyclical Scheduling

Non cyclical scheduling (sometimes called **preference scheduling**), as the name suggests, is the opposite to cyclical scheduling. In theory, each nurse can have a unique schedule which satisfies as many of their personal preferences and requests as possible. Due to its flexibility, preference scheduling is more popular with nurses. However, it is generally a much more complex problem to solve and needs to be addressed each new scheduling period (which is not necessarily the case with cyclical scheduling). The problems examined in this thesis are all non cyclical. Therefore, the publications reviewed in this chapter are all related to non cyclical nurse rostering.

Rostering problems can be further categorised by whether they require the assignment of just days on and off for a given planning period or whether they also require the decision of which shift to assign for the days on. The latter are generally more difficult. If the shifts are not defined beforehand, that is, the days on/off for each employee and the start, end and rest times for the days on need to be determined, this is commonly called the (labour) **tour scheduling** problem.

The term *tour scheduling* is also sometimes applied even if the shift types are fixed and known beforehand. This definition is less frequently used however and therefore appears less in the nurse rostering literature as the shift types are normally predefined. A review of publications post 1990 which specifically address tour scheduling can be found in [17]. Tour scheduling is not necessarily cyclical and is more often used in situations where service demand fluctuates and minimising employee costs is important.

2.4 Personnel Rostering Methods

This section collects and reviews the majority of publications that have addressed non-cyclic nurse rostering problems. The methods used to solve the problems can be placed in one of two broad categories: exact and heuristic optimisation methods. **Exact methods** (e.g. mathematical programming [43] and constraint programming [106]) have the advantage that they will guarantee to produce optimal solutions. The disadvantage, however, is that for many real world nurse rostering problems the time required to produce these solutions is unfeasible. As a result, most exact optimisation methods applied to real world nurse rostering problems do one or more of the following:

1. Solve a relaxation of the problem.
2. Use a number of heuristics (such as how to branch in the search trees).
3. Terminate before the optimal solution is found.

Exact optimisation techniques are still very successful, however, and have other advantages. For example, often it is only necessary to model the problem and then use a highly developed system such as CPLEX [136] to solve it. On the other

hand, designing the model or formulation is not always trivial and can have a critical effect on the success of the algorithms. Also, CPLEX licences can be expensive and possibly beyond the budget of a hospital's IT department.

Approaches which do not guarantee to produce optimal solutions are broadly categorised as **heuristic methods**. Included in this category are some methods which under certain conditions will promise optimal solutions e.g. simulated annealing [147] but obviously these too have exponential worst case time complexity. Heuristic approaches are commonly applied to and are particularly suited to nurse rostering problems for a number of reasons. Firstly, it is actually very difficult to define what would be an optimal solution. The problem formulations are often based on subjective decisions and vague preferences so that an "optimal" solution may not actually be the best or most preferable solution. For example, an employee might say "I would quite like a day off on…" or "I don't really want a night shift on…". These sort of statements are difficult to translate into exact mathematical expressions. Secondly, users are often impatient and want short waiting times for solutions. As such they are willing to trade "high quality" solutions for lower quality solutions in order to reduce running times. This is particularly the case for nurse rostering which may need new solutions at short notice due to absences and sick leave. The other advantage of heuristics is that they are successful at exploiting problem specific information or structure to obtain higher quality solutions (the importance of this is highlighted by the no free lunch theorem [246]). One criticism of heuristic approaches, however, is that they can be inconsistent between problem instances and may require more programming. Another major criticism is, of course, that you cannot, in general,

guarantee optimality. On the other hand, complicated objective functions may be easier to handle with higher level programming languages.

2.4.1 Mathematical Programming

Included in this section are publications which use linear programming and integer linear programming methods [43]. The most well known algorithm for linear programming problems is Dantzig's simplex method [84]. If the linear program has too many variables to define explicitly then column generation may be used. If all the variables are required to be integers (integrality constraint) then the problem is known as an integer programming problem. If only some of the variables are required to be integer then the term mixed integer programming is used. If they must be 0 or 1 then the term 0-1 integer programming is employed. Integer programming problems are usually solved using branch and bound or branch and cut approaches. The design of the model or formulation and how to branch in the tree are often critical to success. For further reading on mathematical programming, one of the most frequently cited references is [194].

One of the first nurse rostering problems to be approached with an exact optimisation method was that presented by Warner and Prawda [243]. The authors formulate a staffing problem as a mixed integer quadratic programming problem. A solution to the problem represents a staffing pattern which specifies the number of nurses with specific skills to cover the shifts for six wards. The goal is to minimize shortage costs while satisfying constraints which cover the total number of skilled nurses in employment and shift coverage. The model allows for some substitution of tasks among skill classes.

Warner [242] later develops a mathematical programming approach for solving shift allocation problems at other U.S. hospitals. Solutions to this problem allocate preferable working patterns to wards of up to 47 nurses over planning periods of 4 or 6 weeks (the problem is decomposed into 14 day periods during solving though). The nurses are first asked to allocate all, or a portion, of a fixed number of points to a small set of schedule properties to describe their preferences for these different properties (the number of points a nurse has to allocate is related to the number of hours they work and points not used may be carried over to the next planning period). For example, a nurse may specify a stronger preference for non-isolated working days to a 7 day work stretch by allocating 10 points to the former and 5 points to the latter. These point allocations (and also day off requests) are then used to allocate a score for different working patterns for each nurse. The objective is to maximise the sum of the scores for each assigned pattern whilst meeting coverage demands and so increase the quality of the overall schedule. The number of possible working patterns is reduced by using fixed shift rotations. The overall system was welcomed by the nursing administration, especially the head nurses. The software later evolved into a system called ANSOS (Automated Nurse Scheduling Office System) which provided additional features such as staffing, management reporting and short term scheduling [244].

Miller et al. [182] use mathematical programming to define a nurse rostering problem but actually solve it using a cyclic descent (local search) algorithm. The solutions, although not guaranteed to be optimal, are found quickly and are close to the optimal solutions produced by a branch and bound method. The system

only takes one shift type into consideration and does not consider part time employees. However, feasible extensions are suggested which could allow the system to include shift rotations and other work contracts.

Hard constraints such as the maximum number of working days and the maximum and minimum numbers of consecutive working days are used to reduce the number of 14 day work patterns to examine during the search. However, these hard constraints can be overruled if they conflict with a requested day off. Soft constraints with weights or *aversion coefficients* are used along with desired staffing levels to formulate the objective function. The quality of a nurse's previous schedules is also considered to try and maintain a level of fairness over longer time periods. The soft constraints that are employed include maximum weekends worked, split weekends and maximum consecutive free days.

Bailey and Field [23] propose an alternative to the traditional, fixed start time, 8-hour shifts for meeting personnel demands in any 24-hour work environment (not just hospitals). 6, 8 and 10-hour shifts are used with variable start times to define a problem which is then relaxed and solved using linear programming. If the solutions are non-integer then another algorithm is used to convert them into optimal integer solutions to the general problem. The authors found that their 'flexshift' model outperformed the fixed 8 hour shift model with a reduction in staff size, overstaffing and idle time. They also suggest that this model provides more choice to employees in their work patterns. A self-scheduling method for assigning the shifts is suggested but not implemented or tested.

In another paper, Bailey [22] presents an approach which combines the problem of shift planning (where hourly demand fluctuates) and the assignment of those

shifts to employees whilst considering some basic work pattern constraints. The objective is to minimise understaffing subject to a fixed workforce size and overtime restrictions. Linear programming is sufficient to identify the optimal shifts and on-off patterns. The shifts are then matched to the patterns heuristically, aiming to minimise the difference in a nurse's shift start times over the period. Ozkarahan and Bailey [203] later extended this model using goal programming. The new approach allows users to set their own priorities for goals related to understaffing, overstaffing and total workforce utilisation.

Thornton and Sattar [232] use branch and bound integer programming to solve a nurse rostering problem in an Australian hospital. The model requires *nurse to feasible schedule* assignments for full time employees and *nurse to shift* assignments for part time employees. As part time employees have fewer and simpler constraints, they have too many feasible schedules to enumerate them all. The problem can be decomposed by not differentiating between late and early shift assignments until a final phase which is solved separately. The objective is to minimise undesirable consecutive day on/off stretches and optionally under/over coverage also.

Mason and Smith [169] describe column generation methods for efficiently solving a nurse rostering problem using linear and integer programming techniques. Columns are generated by solving dynamic programming shortest path problems concerning the nurse's preferences for different shifts, consecutive on/off patterns and the transition between different work start times each consecutive day. The cover demands are fixed and the approach was able to

satisfactorily solve problems with 86 nurses, 7 skill types and 5 shift types over planning periods of 28 days.

Jaumard et al. [142] solve a nurse rostering problem with the objective of reducing salary costs, improving nurse preference satisfaction and improving the ratio of experienced to less experienced staff in teams. Again, column generation techniques are used with the columns corresponding to individual schedules for each nurse but this time they are generated by solving a resource constrained shortest path problem. The constraints for this auxillary problem are related to the individual nurse's requirements. For example, the maximum and minimum hours worked per week, the number of consecutive weekends on and then off, shift rotation constraints, the minimum and maximum number of consecutive days worked and the ratios of shift types worked. Nodes in the branch and bound tree are linear relaxations of the master problem which are solved using the column generation (i.e. branch and price). Branching in the tree is performed by progressively fixing or not allowing shift assignments to nurses. Preliminary tests showed that good solutions could be found within acceptable time limits after partially completing the branch and bound.

Millar and Kiragu [181] model a cyclic and non-cyclic nurse rostering problem using networks. Instead of using single shift assignments for each day as nodes in the network, the nodes are actually short patterns of consecutive shifts or days off (called stints). The problem is decomposed by then allowing each node to be one of only seventeen unique stints. The number of stints can be reduced as there are only two, 12 hour shifts and the stints can be no longer than four days. Arcs

between these nodes are then assigned penalty costs to model certain soft constraints or the arcs may be removed if they violate hard constraints, e.g. a nurse requesting not to work on a specific day. The objective of the non-cyclic problem is to minimise the sum of the soft constraint related penalties and the imbalance of day and night shifts worked by each nurse. The network based formulation of the problem is solved using the CPLEX mixed integer solver (branch and bound). Although optimal solutions could not be found in a feasible time period, acceptable solutions could be produced quickly.

Eveborn and Rönnqvist [100] combine integer programming techniques (in the form of branch and price) with heuristics to solve non-cyclical tour scheduling problems. The algorithms are part of a commercial staff scheduling software package. The problem objective is to minimise schedule costs (a combination of total schedule hours and the violation of staff preferences) and the deficit or excess of staff covering each task. Particularly bad individual schedules can optionally be minimised too, to increase perceived overall fairness. The results of using the system to solve the scheduling needs of call centres and a zoo are provided.

Bard and Purnomo [26] combine heuristic and integer programming methods to solve a nurse rostering problem with up to 100 nurses and approximately 13 hard and soft constraints. The objective of the problem is to minimise the costs incurred through employing outside nurses and to maximise the satisfaction of nurses' working preferences. High quality individual nurse schedules are created using a single or double shift swapping heuristic on a base schedule. These

columns are then used to form a set covering-type problem which is solved using branch and bound. The authors found that, for most of the instances the algorithm was tested on, the majority of the computation time was being spent on generating the columns (executing swaps, checking for constraint violations and duplicate schedules) rather than the branch and bound. The overall rosters produced were of high enough quality for the initiation of system deployment in a number of U.S. hospitals. The authors elaborate with methods and results for including downgrading in this model in [25]. More recently, Bard and Purnomo proposed a nurse rostering model which combines cyclic and preference scheduling. They solved it in [27] using (amongst other mathematical programming techniques) Lagrangian relaxation and branch and price in [210].

As can be seen from the publications discussed above, column generation is often and increasingly being used in mathematical programming approaches to nurse rostering. The columns in nurse rostering problems represent possible work patterns for individual nurses. Due to computational limitations, in the earlier publications, a restricted set of columns is predefined for assignment e.g. [242]. More recently, for example in [26], the columns are generated heuristically by modifying other columns via swapping assignments. The alternative approach is to generate columns using an exact approach, such as a shortest path algorithm, and incorporate the column generation in a more sophisticated method such as branch and price e.g. [142, 169, 210].

2.4.2 Constraint Programming

The word ‘programming’ in linear programming is related to the planning and scheduling problems it was originally used on (i.e. to create a program or

schedule for a specific task). In contrast, ‘programming’ in constraint programming is from the more familiar definition relating to computational methods and computer languages. Constraint programming problems are usually defined in terms of variables, domains (possible values) for these variables and constraints which restrict the simultaneous values that the variables can take. A solution is an assignment to each variable from its domain such that all the constraints are satisfied. Most of the searches used to solve these problems make use of constraint propagation and domain reduction. That is, assigning values to the variables and then using the constraints to reduce the domains for unassigned variables. For the more difficult problems, it is often necessary to develop efficient heuristics for navigating the search tree, e.g. which variables to assign first. [28, 106] provide nice introductions to constraint programming and list references for further reading.

Darmoni et al. [85] use constraint programming to solve scheduling problems in a French hospital. The system allows a wide range of constraints and rules to be imposed. The search strategy (that is, how to branch) is based on trying to ensure equally fair schedules among nurses. For example, they each work similar numbers of Sunday mornings and have similar numbers of requests satisfied. Branching is also guided by trying to ensure complete weekends and avoiding too frequent a shift rotation. The approach was able to produce satisfactory schedules over planning horizons of up to 6 weeks. The authors also found that using an automated interactive system was able to save significant labour time for the head nurses who previously had the burden of producing the schedules.

Weil et al. [245] use ILOG's constraint programming engine [137] to solve a nurse scheduling problem with a number of typical constraints such as minimum rest time/days off and shift type successions. Although employees with different skills and part time employees are not considered, example instances of the problem were solved very quickly. The search uses the smallest domain heuristic.

Meisels et al. [172] present a number of rules and heuristics for processing constraint networks in order to solve nurse scheduling problems. The algorithm is part of a commercial employee timetabling software package and is tested on data from an Israeli hospital. The heuristics and knowledge-based rules are mostly either inspired by manual scheduling methods, designed to produce more fair and balanced schedules or interactively specified by head nurses to reflect scheduling preferences and priorites. More information on the design of the software from the user's perspective is provided in [173].

Using the same model, Meisels and Lusternik [174] developed a random test bed of problem instances to investigate how various parameters affect the difficulty of solving them using constraint processing methods. They found that it is not only the size of the instance that appears to affect its difficulty but also the structure, especially if it is close to the border between the solvable/insolvable instances. The results are similar to the *phase transition* investigations for other combinatorial optimisation problems [74, 131, 183].

Cheng et al. [76, 77] present a constraint programming method for solving a week long nurse scheduling problem in a Hong Kong hospital. A *redundant modelling* idea is described which involves formulating the same problem in two distinct

ways (*shift to nurse* and *nurse to shift-type* assignments). As the search progresses, both formulations are simultaneously updated and fedback into each other. The authors found that this approach, although slightly increasing memory overheads, can provide significant computation time improvements in finding solutions for some instances. The problem contains some common, but also a few unusual, constraints. For example, the nurses prefer frequent shift rotation, that is, alternating consecutive A.M. and P.M. shifts (in most other problems, nurses prefer minimal shift rotation). Branching decisions are made in the search tree based on the relative priorities of the soft constraints.

Meyer auf'm Hofe [178] combines heuristic local search ideas with constraint programming techniques to create an automated nurse rostering system tested in a German hospital. Hard constraints in the model are based on legal regulations and working time restrictions. Soft constraints are organised into hierarchies of different priorities to reflect their importance. For example, providing a minimal coverage is a higher priority than providing a preferred number of staff which in turn is a higher priority than guaranteeing a nurse's requests. A higher priority constraint may not be violated even if its violation allowed *all* constraints of lower priority to be satisfied. This is the key difference between weights and priorities. A constraint with a high weight may be violated if it permits a number of constraints with smaller weights to be satisfied. Weights are assigned to each constraint within the hierarchies.

The constraints are used to define a constraint satisfaction optimisation problem which is addressed using branch and bound. However, problem instances of any magnitude cannot be solved solely by using this exact approach. So, heuristics are

added to help produce high quality schedules in acceptable time limits. The heuristics are based on identifying and repairing violations. For example, they assign more or less shifts to improve coverage constraint satisfaction.

In [179], Meyer auf'm Hofe extends the previous work to provide a more flexible, powerful and robust system. A new constraint hierarchy is defined which allows better interaction with the end-user and fuzzy constraints (i.e. constraints which may be partially satisfied and partially violated) are introduced. The paper also highlights the disadvantages of some of the basic local search algorithms (which change one variable or assignment at a time) and how this hybrid heuristic approach overcomes this problem, effectively by simultaneously making multiple changes.

Abdennadher and Schlenker [3, 4] present a system which is used interactively for the semi-automatic creation of nurse rosters. A partial constraint satisfaction problem is formulated and a multi step method combined with standard constraint programming techniques is used to solve it. At each phase or step only certain shift types (e.g. night, morning, day, free shifts) are assigned, mimicking a manual approach. The problem is further decomposed by using the assignment of good sequences of shifts rather than single assignments. The software is designed to allow the user to interrupt the search and make any modifications to the current partial schedule and then allow it to continue again with these manual assignments in place. This sometimes helps create more satisfactory rosters and/or reduce computation times.

Chun et al.[79] developed a nurse rostering system which was deployed in 250 wards in multiple hospitals in Hong Kong. Although the authors do not provide full details on the algorithms developed, heuristics are combined with constraint programming. The software is flexible enough to accommodate a variety of hard and soft constraint and the number of wards in which the system is used suggests a successful approach. Wong and Chun [247] solve a simplified week long nurse rostering problem using constraint programming methods. Tsang et al. [235] model a nurse rostering problem using constraint programming and solve it using guided local search. The approach is able to solve tightly constrained instances of the test problem used.

2.4.3 Goal Programming and Multiobjective Optimisation

Most nurse rostering problems have a number of objectives. However, as many of these objectives conflict with each other, a feasible solution which simultaneously satisfies all of them rarely exists. Instead, the objectives are often treated as goals or soft constraints with user specified priorities or weights. The objectives are then often combined into a single (often weighted) sum.

An alternative approach is Pareto optimisation, which aims to return the Pareto optimal front for a multiobjective problem. The Pareto optimal front consists of all the non-dominated solutions (a solution is non-dominated if there is no other solution which is better than it for all objectives). The user can then select a solution from this front which best represents their trade-off preference for the objectives. A good introduction to Pareto optimisation approaches to scheduling and timetabling can be found in [87, 156].

Arthur and Ravindran [20] solve a nurse rostering problem using goal programming followed by heuristic assignment. In the first phase, a goal programming model is used to assign days on/off to nurses over weekly periods. One of three shift types is then assigned to each nurse for the on days using heuristics based on minimising under-cover. Although the approach does not permit the substitution of different grade/class nurses for each other or consider part time workers, feasible extensions are suggested to accommodate these requirements.

Musa and Saxena [191] use a zero-one integer goal programming method to solve a very basic nurse rostering problem. Although the problem includes full and part time nurses of different grades, the only instance tested had eleven nurses, a two week scheduling horizon and one shift type. Seven goals are defined relating to cover requirements, weekends and consecutive days off and minimum/maximum number of days worked.

Franz et al. [105] use integer goal programming to solve a slightly different health personnel scheduling problem. In this scenario, nurses can be assigned to a number of different clinics with different geographical locations. This complicates the problem as travel costs and nurse preferences for working in different locations have to be considered. The problem is simpler in another respect though as cover only has to be provided at each clinic Monday-Friday, 8:00am-9:00pm. Each clinic has a varying skill mix and staff number requirements to provide a satisfactory service for the predicted patient numbers. The objectives or goals of the problem are to maximise staff to patient ratios in

order to reduce waiting times, minimise travel costs for the staff and maximise staff preferences for working in specific clinics at certain times. Although integer solutions to the problem could not be produced in acceptable computation times, a number of modifications to the problem are considered, one of which enables the fast production of solutions which are comparable to the manually created ones.

Berrada et al. [39] test three techniques for solving a nurse scheduling problem with multiple objectives. Although the problem is simplified by not considering shift rotations, a number of common soft constraints or objectives are included. For example, no isolated working days, a maximum length of consecutive working days, grouping days off together and personal shift and day off requests. The objectives are assigned a priority ordering to reflect scheduling preferences. Two mathematical programming techniques are tested to produce (loosely) non-dominated solutions with respect to the objectives used. A tabu search with a neighbourhood based on swapping a working and non-working day for an individual nurse is also applied (this swap is possible as cover requirements for a specific day do not represent a strict hard constraint). All three techniques produced schedules of a similar satisfactory quality although the tabu search required more computation time. Further experiments with tabu search on a very similar problem formulation can be found in [102].

Jaszkiewicz [141] uses a metaheuristic approach to solve a multiple objective nurse scheduling problem for a surgery unit in a Polish hospital. The five objectives defined are similar to those discussed in other nurse rostering problems

in the form of constraints or objectives. For example, preferred lengths of consecutive working days, non-working days, shift rotations, balance of shift types worked and equal assignments of surplus nurses over the week. A population of initial solutions is created using a constraint programming method and then a simulated annealing approach is employed to identify the Pareto optimal front, or at least a good approximation to it. The algorithm uses dynamically altered weights for each objective to guide the search over the trade off surface. A randomly selected move (from three) may be applied to a solution and the move accepted probabilistically. The approach was able to fairly quickly produce solutions that dominated those produced manually.

Gascon et al. [109] developed a goal programming model to solve a problem requiring the scheduling of flying squad nurses. Rather than always working in the same care unit, a flying squad nurse can be assigned to one of a number of units in order to meet cover demand. Working in different care units helps the nurses maintain the skills required to operate in that unit but frequent movement between units is undesirable as it lessens the quality of service provided. Solutions to the problem must allocate days on and off to the nurses as well as specifying which unit they are stationed at on their working days. Although the model assumes that the nurses work the same shift type, there are a number of objectives and constraints to satisfy. A combined priority ordering and weighted method for the objectives is used in solving the problem.

Through surveys, feedback from head nurses, hospital regulations and analysing published recommended work practices for nurses, Azaiez and Al Sharif [21]

formulate a goal programming model for a nurse scheduling problem in a Saudi Arabian hospital. Initially, the problem was too large to solve, so a heuristic was introduced to decompose the problem. The nurses were split into groups (ensuring a balance of skills), schedules for these groups were found separately and then the overall schedule was formed by recombining the individual group schedules. The authors found that for the majority of instances, this method was able to produce optimal schedules (all goals completely satisfied). After the system was tried and the diverse workforce typical in Saudi Arabian hospitals was surveyed for a second time, an improvement in the rosters was generally noticed. Cost savings through reduced overtime (one of the goals) was also introduced but a few of the nurses did not appreciate this achievement as it prevented them from having the opportunity of earning more money.

2.4.4 Decision Support and Expert Systems

Smith and Wiggins [226, 227] created an interactive system to simplify the rostering process and to reduce its burden. The devised methodology consists of three phases. In the first phase staff deficiencies due to holidays and vacations are highlighted to encourage reassignment of nurses to understaffed units. Next, preliminary schedules are generated using some simple heuristics based on reducing assignment conflicts and nurse dissatisfaction. Finally, the schedules are improved manually with the aid of the system revealing violations, under coverage and over coverage. The authors found that the system reduced the time spent producing schedules and the staffing clerks had little trouble using it.

Bell et al. [36] produced a decision support system that constructed basic cyclical schedules and which enabled the head nurse to modify the rosters as required

through a user interface which was quite sophisticated at the time. They selected this approach as they felt that the real world problem they were examining contained several "*complexities which would be difficult to incorporate into a formal algorithm*" such as varying cover requirements and work contracts.

Ozkarahan [202] describes a support system for formulating nurse scheduling problems which can then be solved using goal programming. The system allows a number of factors to be incorporated into the problems including fluctuating cover demand, nurse preferences, skill substitution and the movement of nurses between units.

Chen and Yeung [75] created a nurse scheduling system which uses zero-one goal programming and an expert system to aid the creation of rosters. Firstly a working pattern (i.e. days on and off) is generated using the goal programming by only considering constraints which are relevant to the working pattern e.g. minimizing overtime, preventing stand-alone shifts, satisfying vacation and day off requests and providing sufficient daily coverage. Fixed 8 hour shifts are then assigned to these work patterns using an expert system consisting of 37 rules. The rules are designed to improve schedule quality and are based on constraints and preferences such as limiting the lengths of consecutive night shifts, forward rotation and providing supervision for inexperienced nurses. By comparing the rosters created using this approach to the ones produced manually by the head nurse, the authors found that the computerised system was able to significantly reduce the time needed to create the rosters as well as being able to satisfy more constraints, requests and working preferences.

Begur et al. [34] successfully developed a spatial decision support system to help schedule and route home-health-care nurses (who visit a number of different patients at their homes each day). The nurse to patient assignments and routes are improved by using heuristic methods.

2.4.5 Case-based Reasoning

The basic idea behind case-based reasoning is to use past experiences in solving new problems. Previous solutions to problems and related information are stored as cases. When a new problem is encountered, relevant cases are retrieved and used to provide solutions to the problem by making any required adaptations. When the new solution is found, a new case describing the problem and solution is created for future use.

Case-based reasoning has been applied to other scheduling problems such as university exam and course timetabling [65, 69, 206] but its application to personnel rostering is quite novel and so there are fewer relevant publications.

Scott and Simpson [222] use a case-based reasoning method for the automated construction of nurse rosters. The case-base contains good, week long, patterns of shift assignments. The number and type of employees and personal requests in the current instance are used to heuristically select the patterns and to assign them. Under or over coverage is then repaired by removing and/or making extra assignments and this final solution may then be used to update the case-base for later use.

Petrovic et al. [204, 205] have written a number of papers which explore the use of case-based reasoning for solving nurse rostering problems in a UK hospital. The approach is novel in that no objective function, search space or search mechanism is employed. Instead, the aim is to capture the knowledge and experience of expert schedulers in a case-base and reuse it to repair, improve and solve new problem instances. Initially, a partially complete and infeasible schedule is created by the nurses after a self rostering process (see [224] for more information on self rostering). Violations in this schedule are identified and the case-base consulted to find the best way of repairing the violation by analysing similar violations which were previously identified and repaired. Once the best repair is selected, adapted and executed this is recorded as a new case and the case-base is updated for future reference. The case-base is initially populated by analysing past partial rosters and the corresponding final roster produced by the human expert.

Repairing one violation may generate one or more new violations and so it is possible to enter a non-terminating loop or cycle of violation repairing. To avoid this, Beddoe and Petrovic [31, 33] experimented with combining tabu list mechanisms with the case-based reasoning methodology and found that it made the approach more robust. They also experimented with adding an objective function (the number of violations) to the case-based reasoning approach and found that it further improved the quality of the rosters produced. In [32], they also successfully developed a genetic algorithm to identify the best subset of violation features and their relative importance to use in classifying violations. This led to an improvement in the accuracy of the case retrieval and subsequently an increase in the quality of the schedules produced.

More recently, Le and Landa Silva [158] also solved a multiobjective formulation of this problem using an evolutionary algorithm.

2.4.6 Heuristic and Local Search

The largest proportion of approaches in the literature can be categorised as heuristic, local search or metaheuristic methods. Heuristics may be constructive and build solutions from scratch or be improving, for example, repairing violations in rosters. Local search and metaheuristics draw upon the idea of neighbourhood searches. Identifying efficient neighbourhood operators can often have a significant impact on the performance of these algorithms. This is discussed further in the context of nurse rostering in chapter 5. For more information on local search and metaheuristics see [2, 62, 123, 199, 241].

Blau and Sear [41] use Miller et al's [182] local search approach to solve a similar problem which also considers only one shift type and a scheduling period of 14 days. The objective is to minimise a weighted sum of under coverage, over coverage, requested days off not granted, excessively long work stretches, on-off-on and off-on-off work patterns.

Anzai and Miura [18] use heuristic methods which involve swapping shifts between nurses to repair violations in rosters. The model used takes into consideration constraints such as maximum working days per month, individual requests for specific shifts and days off, minimum cover requirements and maximum consecutive night shifts. A constraint which prevents inexperienced nurses from being unsupervised during night shifts by more experienced nurses is also included. The system is limited, however, by assuming all the nurses have

the same work contracts and constraints other than their shift on and day off requests.

Okada and Okada [198] developed a system using Prolog and heuristic ordering which automatically follows general rules to manually construct rosters. As some of the rules are quite complex, the system relieved the manual scheduler's workload considerably. One of the more complicated objectives is to try and ensure each nurse works with as many different colleagues as possible. Okada [197] later extended this approach to give users flexibility in defining and modifying some of the rules/constraints so the system could be used in a variety of departments with different requirements.

Kostreva and Jennings [148] describe a two phase algorithm for solving a nurse scheduling problem. In the first step, a set of feasible schedules that ensure minimum shift coverage is satisfied, is constructed by assigning blocks of consecutive shifts. In the next step, each nurse's aversion to working each of these fortnight long schedules is calculated as a cost based on the nurse's previously specified preferences for different schedule characteristics. The schedules are then optimally assigned to minimise total aversion costs using the Hungarian method. These two steps can be repeated as many times as required by generating a different set of schedules at the first step each time. The type of schedule characteristics considered when calculating aversion costs include types of shifts assigned before days off, lengths of consecutive work stretches, single days on/off and types of shifts assigned at weekends.

Randhawa and Sitompul [213] present a two step algorithm for solving a nurse rostering problem. In the first phase day on/off working patterns are generated for the scheduling period. Each pattern is assigned a score corresponding to soft constraint violations such as exceeding maximum lengths of consecutive days on and off. Shifts are then assigned to a set of the best working patterns trying to match cover requirements as closely as possible. The shift assignments are such that every nurse works the same type of shift in a seven day period The system also provides a user interface for entering parameters such as the soft constraint penalties and cover requirements.

Khoong et al. [146] use heuristic and exact search techniques to provide an automated rostering system which is suitable for a variety of workforces. The software (called ROMAN) can produce both cyclic and non-cyclic schedules and allows for the consideration of many of the common constraints (e.g. minimum/maximum length work stretches for on and off days, shift rotation etc) and objectives (e.g. reducing costs, improving cover provided).

Liao and Kao [162] present a heuristic constructive approach for solving a month long nurse rostering problem in a Taiwanese hospital. The problem requires the consideration of full and part-time employees, two skill levels and 8 hour and 12 hour shifts. Soft constraints include satisfying cover, preferred lengths of consecutive work days, sufficient rest time between shifts, avoiding stand alone shifts and granting requested days, holidays and weekends off. Some decisions in the schedule construction phase may be made randomly. This allows the algorithm to be executed a number of times and the best schedule from all the

executions is the one returned. The method is able to quickly produce higher quality schedules than those which took approximately 4 hours to create manually.

Burke et al. [54, 56] present a variable neighbourhood search which uses a small neighbourhood defined by reassigning single shifts between nurses and larger neighbourhoods (called shuffling) defined by swapping blocks of adjacent shifts between nurses (See chapter 5 for more details). A number of heuristics and methods for introducing diversification into the search are also used. Best results were all produced when the ‘shuffle’ neighbourhoods were included. However, it was more efficient to use these larger neighbourhoods towards the end of the algorithm on higher quality solutions. This was because they are time consuming to search and produce less improving moves on better solutions. The problem addressed and the methods described in this paper are discussed further in chapters 4 and 5.

Schaerf and Meisels [176, 221] performed a number of experiments using local search and constraint programming methods on employee timetabling problems including nurse rostering. A basic hill climber which uses a neighbourhood defined by reassigning single shifts is compared to a more advanced method. In addition to including moves which add or remove single shifts at a time (and so allowing partial solutions), the more advanced method allows certain constraints to, in effect, be relaxed by dynamically adjusting associated violation weights. The search is further guided by a heuristic which estimates how difficult it will be to complete a partial solution. The constraint programming method adequately

solves easier instances but the local searches (particularly the simpler method) are more effective on the larger and more difficult instances.

Ikegami and Niwa [135] use a heuristic decomposition method to solve nurse rostering problems in Japanese hospitals. The instances feature a number of typical constraints such as minimum/maximum numbers of shift types, day off requests, minimum/maximum length work and non-work stretches etc. Atypically though, frequent shift rotation during a week appears to be acceptable and common in practice. The approach is successfully applied to a 2-shift and a 3-shift problem. For the more complex 3-shift problem it was necessary to extend the search with a branch and bound procedure. Experiments were performed to compare the algorithm against Millar and Kiragu's [181] and Nonobe and Ibaraki's [195] solvers. Although it is difficult to make exact comparisons due to differences in machines used for the tests, the results suggest their approach is competitive.

Aickelin and Li [14, 160, 161] test a Bayesian optimisation approach on the nurse rostering problem examined in [12, 91]. Working patterns for assigning to nurses are selected according to one of four rules. The algorithm aims to find the best rule to use for each different nurse (a rule string) to construct the highest quality overall schedule. A population of rule strings is used to construct a Bayesian network which reflects the frequency of the application of each rule to each nurse. A new set of rule strings is created using this network and the current population of rule strings is updated with these new 'solutions' according to the quality of the schedules they create. The population is used to create a new Bayesian

network and so on until a termination condition is satisfied such as a maximum number of generations. The rules for selecting a pattern to assign are designed to either avoid local optima, increase the quality of the schedule in terms of nurse preferences and/or increase cover satisfaction and feasibility. This is the first application of Bayesian networks to personnel scheduling. Aickelin et al. [11] later improved this approach by adding two more rules and using an ant colony optimisation algorithm to improve rule strings between generations.

2.4.7 Tabu Search

Tabu search is often used as a label for any algorithm which escapes local optima in a neighbourhood search by moving to a worse solution and then uses a list of "tabu" moves to reduce the chance of returning to the local optimum. Tabu search as proposed by Glover [112, 113, 116] however includes a number of other features which may also be used:

- Intensification (concentrating on good areas of a search space).
- Diversification (ensuring a wide cover of the search space).
- Aspiration criteria (accepting moves to solutions currently in the tabu list if the solution displays a particular quality).
- Candidate lists (not examining all solutions in the current neighbourhood).
- Strategic oscillation (moving between feasible and infeasible solutions).
- Path relinking (exploring paths between two or more solutions in a search space).
- Compound moves (chaining moves for a particular neighbourhood together to escape local optima with respect to that neighbourhood).

Nonobe and Ibaraki [195] use tabu search to solve a number of problems modelled as a constraint satisfaction problem. The problems tested range from graph colouring to set covering to timetabling and also include a nurse scheduling problem. The length of the tabu list is dynamically altered during the search to try to improve its performance. Although it is designed as a general problem solver, the results across the variety of problems are competitive. Feasible solutions are also produced for the Japanese nurse rostering problem tested. The problem features a number of common constraints such as maximum lengths of consecutive shift type assignments and no isolated shifts. Two skill levels or grades are also considered.

Dowsland [91] developed a tabu search method for solving a nurse rostering problem in a major UK hospital. The problem required the production of weekly schedules consisting of only day or night shifts. The complexity of the problem is increased by the inclusion of different qualification levels for the nurses and the cover requirements and part time nurses. Although the approach is based on the tabu search framework, tabu lists play a minor role in the algorithm. Instead, a key feature of the algorithm is strategic oscillations between two phases of trying to minimise cover violation and then minimising penalty costs corresponding to the quality of the schedules from the nurses' perspectives. In both phases, ejection chains of moves are used. The chains consist of either single on/off day swaps or the exchange of whole weekly patterns between nurses. The schedules produced by this method were able to match the quality of those produced by a human expert.

Dowsland and Thompson [92] later improved the model to provide a number of extra rostering requirements. Two major changes needed the addition of new pre- and post-processing phases to the tabu search. The pre-processing phase examines whether there are sufficient nurses to satisfy the cover requirements and, if not, outputs the number of bank nurses that are needed. The problem is modelled as a knapsack problem which can be efficiently solved using a branch and bound method. The post-processing phase optimally allocates early and late shifts on the day shift assignments according to nurse working preferences and cover requirements. It is possible to perform this after the tabu search as its overall effect on the solution quality is relatively low. Network flow models are used to represent this problem which again can be solved quickly. The vast majority of the solutions produced by the tabu search were also proven to be optimal after modelling and solving the problem using integer programming.

Burke et al. [58] present a tabu search which uses a neighbourhood of reassigning single shifts. This is combined with a diversification step based on repairing complete weekend constraint violations and schedule improvements using the large shuffle neighbourhoods (see chapter 5 for a discussion of this neighbourhood). The basic tabu search outperforms a steepest descent method and when extended using *shuffle* neighbourhoods, the results improve further although at the expense of extra computation time. This approach is later extended in [57] to allow coverage constraints to be specified in time intervals rather than fixed shift types. Although this increases the size of the search space, the quality of the nurses' schedules can be improved. This approach has been

implemented in over 40 hospitals in Belgium. An overview of this research is provided in [59].

Valouxis and Housos [237] modelled a nurse rostering problem using integer programming but found the problem instances too large and intractable. The model was therefore simplified to make it easily solvable using integer programming techniques. These solutions for the approximate model are then used as initial solutions to the complete problem for subsequent improvement using local search and tabu search techniques. The searches are based on swapping blocks of shifts and/or days off between nurses. The approach successfully solved instances with 16 nurses, 3 shift types and a planning horizon of 28 days and also compared favourably to a constraint programming method.

Li et al. [159] present a hybrid approach for solving a nurse rostering problem with a smaller number of hard and soft constraints. In the first phase of the algorithm, constraint programming techniques are used to find a solution to a relaxation of the problem. The problem is relaxed by removing personal requests that would cause coverage (a hard constraint) to be violated if they were granted. This ensures that it is not over-constrained and a feasible initial solution can be found. In the second phase the solution is improved with regard to the satisfaction of soft constraints by a tabu search using a neighbourhood based on swapping shifts between nurses on single days. The method was successful when tested on a case study.

Bellanti et al. [37] use a heuristic local search approach to solve a rostering problem in an Italian hospital. The authors found that the integer programming model of the problem contained too many variables and constraints even for a small instance and the linear relaxation could not provide satisfactory bounds either. The problem contains a number of constraints and objectives which include the minimisation of cover shortage. The nature of the hard constraints makes it difficult to define a search neighbourhood which operates over feasible solutions only. Therefore, a neighbourhood involving the movement of night shifts between nurses in partial solutions is used. The partial solutions contain only the assignment of holidays, requested days off and night shifts. Partial solutions are then completed by heuristically assigning morning and afternoon shifts. This neighbourhood is incorporated into a tabu search and an iterated local search which both produce better solutions than those found manually.

Louw et al. [166] use tabu search to solve a nurse rostering problem in a South African hospital. The objective is to minimise a weighted combination of total wage costs and nurse dissatisfaction from not receiving preferred day on/off requests. A minimum cover for each nurse rank (or skill) per shift is a hard constraint along with a minimum and maximum number of shifts worked in the planning period and a maximum number of consecutive shifts. These are user definable parameters for each nurse rank. Every three months, the nurses alternate between working only day or night shifts so that the problem can be decomposed into only assigning one shift type to each nurse. The search uses compound moves (ejection chains) which identify strictly feasible neighbour solutions. The chains of moves are formed by swapping on/off days in individual schedules such

that the next link in the chain corrects the excess coverage caused by the previous move. The final move in the chain corrects the under coverage caused by the first move in the chain. Results showed that these compound moves were more effective than single moves and swaps. A user interface was also developed to allow manual changes and short notice repair to the schedules.

2.4.8 Simulated Annealing

The idea of simulated annealing is to always accept improving moves and to accept un-improving moves with a probability which decreases during the algorithm's execution and is proportional to the change in solution quality produced by the move. In effect, many moves which cause large decreases in the current solution quality may be accepted at the beginning of the algorithm and at the end, few un-improving moves are accepted and those that are, produce very small decreases in quality. Parameter selection can have a large influence on performance. The algorithm was introduced by Kirkpatrick et al. [147] and is so called because the framework was provided by an analogy to annealing in solids.

Isken and Hancock [139] define an integer programming model for a tour scheduling problem from a hospital care unit. The coverage demands for each half hour of the day are known and the objective is to determine a set of shifts for assignment each day such that over and under coverage costs will be minimised. Each shift can vary in start time and has one of three durations (8h, 10h and 12h). Additional constraints are added to ensure that a suitable work pattern over a ten day period can be easily created for each employee in a separate phase. The problem is solved by using a rounding heuristic on a solution to the linear relaxation. This initial solution is then improved by applying simulated annealing

which uses a simple swap neighbourhood. In the example instance described, the final solution was within 19% of the linear program lower bound.

2.4.9 Evolutionary Algorithms

Like most metaheuristics, there is no strict and universally accepted definition for evolutionary algorithms and so a large number of publications could arguably fall into this category. However, evolutionary algorithms are sometimes loosely inspired by natural or biological processes and often use one or more of the following:

- Generations (algorithmic iterations).
- Populations (multiple concurrent solutions).
- Crossover (combining features from two solutions to produce new solutions).
- Mutation (random changes to solutions).
- Population management (e.g. elitism, survival of the fittest, diversification strategies).

These features are not exclusive to evolutionary algorithms though and may appear in other metaheuristics in one form or another too. For example, ‘shuffles’ or ‘kicks’ in variable neighbourhood search and iterated local search could be described as mutations. Examples of evolutionary algorithms include genetic algorithms [104, 124, 132], memetic algorithms (genetic algorithms plus local search) [187] and genetic programming [149].

Tanomaru [229] uses a genetic algorithm to solve a staff scheduling problem which has no predefined shift types. Instead of standard shifts with fixed start and end times (e.g. early, late, night etc), the planning horizon is split into uniform time intervals and a minimum cover requirement for each time interval is specified. Employees are categorized into groups with each group having different working constraints. Cover requirements are further broken down by giving a minimum number of employees required from each group. Wages vary between employees and the objective is to minimise the combination of employment costs and constraint violations. At each generation after a standard reproduction and crossover stage, a number of heuristic procedures are applied to further increase the quality of the schedules. An optimal solution is found for a smaller instance and good solutions are produced within a few minutes for moderate sized instances.

Jan et al. [140] evaluate the use of a genetic algorithm to solve a nurse scheduling problem. Although the authors note that the problem is simplified slightly as they are conducting preliminary tests, some common hard (cover and personal requests) and soft constraints (mostly related to night shifts) are still present. The objectives are to minimise the total penalty incurred due to soft constraint violations and to minimise the variance in the individual nurse schedule penalties so ensuring fairness. A population of non-dominated solutions according to these two objectives is maintained. New solutions are not produced via crossover between solutions in this population however, but rather by mutating a single schedule. This aids the maintenance of only feasible solutions. The authors also

suggest a method for allowing a decision maker to adjust the schedule and guide the search during its execution.

Cai and Li [70] present a genetic algorithm for solving a staff scheduling problem with three objectives of decreasing importance:

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

Feasible weekly schedules with varying costs are predefined and are assigned when the best number of workers for each schedule is found. The employment of fitness values for selecting parents is replaced with a ranking scheme according to the objectives. When comparing the objectives, an extra parameter is included to allow slightly different objective function values to be treated as equal if preferred. Crossover is performed using carefully constructed masks aiming to maintain diversity but not to create overly infeasible offspring. Repairs are performed heuristically by repeatedly identifying the most violated constraint and assigning an additional schedule to satisfy it subject to other constraints. If there is more than one possible schedule for performing the repair, the best one according to the objectives is used. The results were of sufficient quality for the approach to be included in an existing scheduling system.

Aickelin and Dowsland [10, 12] developed and tested a genetic algorithm in place of the tabu search method in [92]. They were able to achieve a similar performance with the genetic algorithm and felt it was more robust when applied to a greater variety of instances. To achieve this success, the genetic algorithm required a lot of adapting to the problem (as did the tabu search also) to exploit its structure. This was done by splitting the population into a hierarchy of sub-populations based on nurse grades and then strategically using crossover on these better building blocks. Also, if a solution exhibits an under/over coverage structure, which is difficult for the genetic algorithm to repair, it is more severely penalised. To improve solution quality further, a basic hill climber involving testing a different work pattern for each nurse and accepting it if it increases fitness is also applied to some solutions.

In [13] they also tested an indirect genetic algorithm on the same problem. This time, the genetic algorithm is used to identify permutations of nurses which are then passed to a decoder which applies heuristic rules to this permutation to assign work patterns and to construct the roster. In effect, the decoder acts as a fitness function for the genetic algorithm. Using this method removes, from the genetic algorithm, the complications of dealing with the problem specific constraints and infeasibility. Three decoders are experimented with, each having a different bias between producing feasible solutions and solutions which maximise nurse satisfaction. After fine tuning the heuristics and penalty weights, the algorithm was capable of slightly better results than the direct genetic algorithm. Aickelin and White [15] later presented an improved method for fine tuning algorithms which is superior to their manual attempts in selecting the best

parameters. They discuss the difficulties in comparing results which may contain feasible and infeasible solutions and suggest a method for handling them.

Burke et al. [49] present a number of memetic algorithms for nurse scheduling. Experiments are conducted combining different crossover operators and local improvements methods. The best approach is a hybridisation of a tabu search [58] and a crossover operator based on selecting the ‘best’ events from each parent. Although the best memetic algorithm required a greater computation time than the tabu search, the solutions produced are nearly always of a higher quality. This research is discussed further in chapter 6.

Dias et al. [88] developed a tabu search and a genetic algorithm for solving rostering problems in Brazilian hospitals. Generally, each employee only works one type of shift (morning, evening or night) which helps simplify the problem. However, there are still a number of soft constraints such as minimum and maximum working days, a required number of days and weekends off and minimum and maximum cover for each day. These constraints are weighted and used to form the objective function. Tests showed the genetic algorithm slightly outperformed the tabu search but, in practice, both approaches were welcomed by the hospital users without preference as they were both significantly superior to manual efforts. A user interface which easily allowed small changes to the schedule by hand was also appreciated by the staff.

Inoue et al. [138] argue that evaluation functions for nurse rostering problems are not always accurate enough as it is difficult for head nurses to describe all the

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

Özcan [201] presents a number of mutation, crossover and hill climbing operators in a memetic approach to solving a nurse rostering problem. Two different shift types must be assigned over a planning period of a fortnight. There are a relatively small number of hard and soft constraints. The hill climbing method examines one constraint at a time and tries to repair violations by changing shift assignments. Experiments are performed on randomly generated instances to determine the best types of mutation and crossover.

2.4.10 Hyperheuristics

Hyperheuristics are designed for problems where good quality solutions are required within reasonable time limits but finding optimal or very close to optimal solutions is not critical. Instead it is more important to implement the algorithm quickly and cost effectively without spending large amounts of time fine tuning and adapting the algorithm to the problem. A key motivation of hyperheuristics is the ability to operate on different problems i.e. raising the level of generality. To achieve this, hyperheuristics generally operate by intelligently

selecting a heuristic or algorithm for the next step in improving or constructing a solution to a given problem. See [61, 217] for an introduction to hyperheuristics.

In [63], Burke et al. successfully applied a hyperheuristic to the nurse rostering problem studied in [12, 91]. The main aim of the research was to show the algorithm could successfully operate on two different problems, as the same hyperheuristic was also effective on a university course timetabling problem. The algorithm manages a competition of the heuristics against each other by rewarding points to a successful heuristic and penalising an ineffective one. The heuristic with most points is the one applied at each iteration and it is then penalised or rewarded according to its affect on the current solution quality. The results of this hyperheuristic compared favourably to a choice function hyperheuristic [80] applied to the same problem. The choice function hyperheuristic uses a statistical record of the individual and joint performance of the heuristics.

2.5 Surveys, Overviews and Bibliographies

A number of literature reviews related to personnel scheduling and nurse rostering have also been published.

In 1976, Fries [107] compiled an early bibliography of applications of operations research methods in health care systems. It includes some nurse staffing and scheduling approaches.

Tien and Kamiyama [233] consider an entire manpower scheduling process and decompose it into five stages or sub problems. General models for each problem

are defined and a review is conducted of algorithms which have been developed to solve one or a combination of these problems. The five stages for a fixed planning period are: 1) determine the number of staff required for each shift, 2) determine the total number of staff required, 3) identify day-off blocks for assignment, 4) assign the days off, 5) assign shifts for the days on.

In 1990, Sitompul and Randhawa [225] produced a state of the art review of nurse scheduling models. The problems are categorized into cyclical and non-cyclical scheduling and the methods for solving them are classified as optimising or heuristic techniques. According to the classification scheme, the optimising approaches use an objective function whereas the heuristic methods do not. The non-cyclical problems reviewed are mostly solved using an optimising approach and the majority of the cyclical problems have a heuristic technique applied to them. Decision support systems are also examined. The authors suggest future nurse scheduling systems may be more effective if they provide increased flexibility and adaptability. A decision support system is proposed as a possible solution.

In 1991, Siferd and Benton [223] reviewed the literature relevant to a wide range of nurse staffing and scheduling issues. Cost reducing and containment pressures on American hospitals and their managers at the time were examined and the effect of these forces on nurse staffing and scheduling were discussed with reference to the relevant literature at the time. A few papers describing solutions to specific nurse staffing and rostering problems were reviewed. The authors also provided the results from a survey they conducted of 348 nursing managers which

included questions related to nurse scheduling. For example, at the time, 59% of the managers performed scheduling without computer help and 37% rated the task as "*highly complex*".

Bradley and Martin [44] reviewed a number of personnel scheduling algorithms published before 1991. The problems examined are classified as either staffing, rostering or short term scheduling. The latter, for example, includes assigning nurses to other wards to reduce under cover caused by short notice events such as staff illness or an unexpected increase in the number of patients. The algorithms are identified as heuristic or mathematical programming approaches and further grouped according to whether they produce cyclic or non-cyclic rosters. The authors highlight the dependencies between the staffing and rostering phases and suggest that it may be beneficial to consider these links in future models.

In 1995, Hung [134] produced a bibliography of publications mostly related to experiments with different nurse scheduling policies.

In 2003, Cheang et al. [73] wrote a bibliographic survey of nurse rostering problems and methodologies used to solve them. The paper details the common models used to define nurse rostering problems and lists frequently used constraints. A number of previously applied exact and heuristic approaches for solving the problems are reviewed. Other relevant issues such as evaluating the performance of different algorithms is also discussed. Concluding the paper, the authors argue in favour of the benefits that the existence of benchmark nurse rostering problems would have for future research.

In 2004, Burke et al. [60] conducted a review of the state of the art of nurse rostering. The paper contains a thorough examination of the most successful nurse rostering systems over the previous 40 years and compares them based on the objectives used and constraints considered. Staffing and cyclical scheduling is briefly observed but the paper concentrates on non-cyclical rostering problems. The approaches are categorised according to the solution methods used e.g. mathematical programming, expert systems, metaheuristics etc. The paper concludes with recommendations on future research directions in nurse rostering. These include:

- The development of benchmark nurse rostering problems with an emphasis on representing real world scenarios.
- Algorithms which produce more robust and flexible schedules with regard to scheduling uncertainty and changes at short notice.
- More user friendly systems to increase user uptake and provide better human interaction with the scheduling process.

The authors also suggest that algorithms which are less dependent on parameter tuning, decompose problems and/or exploit problem specific knowledge may be promising research ideas.

Ernst et al. [99] review staff scheduling models and algorithms over a very wide range of industries. A complete scheduling process is broken down into six steps or modules. The majority of the literature can then be classified according to

which of these steps it examines. Commonly used models and algorithms are reviewed along with their advantages and disadvantages. The authors conclude that near future research directions in personnel scheduling may include the development of more generalised and flexible models which simultaneously consider more than one step in the overall scheduling process. For example, if possible, combining the staffing (or demand modelling) and shift assignment may yield benefits. It is also suggested that future rostering solutions may give more consideration to the requests and preferences of individual employees rather than treating them anonymously.

Using the same classification schemes, Ernst et al. [98] also compiled an impressive bibliography of 700 references from the field or personnel scheduling and rostering. Each reference was briefly summarised and whenever possible checked with the publication's author.

2.6 Related Research

The papers reviewed in this chapter have focussed on nurse rostering problems and the methods used to solve them. There are, however, many other publications that examine some of the other Operations Research problems related to nurse scheduling. For example, Kwak and Lee [153] use goal programming to solve a staffing problem involving physicians, nurses and technicians in three different hospital departments. Nooriafshar [196] developed a decision support system for solving a trainee nurse staffing problem. Staffing requirements must be met in different wards while giving different trainee nurses the required experience and training from working in the various departments. Wright et al. [248] investigate how different policies such as nurse to patient ratios affect total wage costs and work schedule quality. Punnakitikashem et al. [209] developed a stochastic

integer programming model for the problem of assigning nurses to patients before a shift starts so as to minimise excess workloads for nurses. Al-Zubaidi and Christer [16] use simulations to predict the effect of different management and operational policies on maintenance manpower requirements in a UK hospital. Moz and Vaz Pato [188-190] developed a number of methods for solving the nurse rerostering problem. Re-rostering involves adjusting rosters to cope with unexpected absences when there is no reserve pool of nurses available. Gutjahr and Rauner [126] use an ant colony optimisation approach to assign a pool of nurses to a number of hospitals within the Vienna region, Austria. The hospitals make requests for extra nurses on certain days due to excess demand. The nurses are then allocated to the hospitals, taking into consideration the nurses' working preferences, the hospitals' requirements and cost constraints. Beliën and Demeulemeester [35] combine a nurse rostering and surgery scheduling problem and successfully solve it using column generation.

Blake and Carter [40] used goal programming to determine patient level targets at a hospital that was about to experience a significant reduction in funding. Patient requirements, operating costs and doctors' preferred incomes and level and type of workload all needed to be considered.

Abernathy et al. [7] present a stochastic programming model for solving staffing problems in hospitals which have highly variable personnel demands. The three planning stages that the process can be decomposed into are: policy decisions such as the number of nurses allocated to each ward over a scheduling period, staff planning such as the number of nurses needed in employment and short term scheduling using the decisions from the previous stages.

Trivedi and Warner [234] describe a method for optimally allocating float nurses among nursing units at the start of shifts. A model for representing the head nurse's perception on the need for additional nurses at specific units is used to form the objective of a problem which is then solved using a branch and bound algorithm. Factors considered for describing the shortage severity include patient load, patient classifications and staff absences.

Musliu et al. [193] handle weekly shift scheduling problems using local search techniques. Thompson [231] solves a general employee shift scheduling problem using a simulated annealing based approach. Glover et al. [122] outline local search ideas for solving a week long employee scheduling problem in which employee availability and cover requirements fluctuate. Baker et al. [24] present an optimal constructive approach to a very simplified rostering problem.

Easton et al. [94] observe that in order to retain higher numbers of nurses and reduce nurse turnover and the associated costs, some hospitals are providing more attractive work schedules. Some of these scheduling policies include preferable shift rotations, less weekend assignments and higher wages for undesirable shifts. To examine the effects of some of these scheduling rules on the changes to total nurse wage costs and workforce sizes required, the authors conduct simulations and solve nurse scheduling problems using these new scheduling rules. The results suggest that although more restrictive scheduling policies may be more attractive to the nurses, the costs incurred to the hospital through higher total wages and larger workforces should perhaps be considered too.

There are also a number of papers which do not present a specific method for nurse rostering but discuss different ways of modelling problems or ideas which

may be applicable to a number of approaches. For example in [55], Burke et al. suggest an alternative approach to the evaluation function which reduces the knowledge required by the user of the system when setting weights for soft constraints. In [53], Burke et al. describe a fast implementation of the evaluation function for a nurse rostering problem. Due to the large number of complex constraints which are typical in nurse rostering problems, the evaluation function is often a bottleneck. Therefore any gain in the speed of evaluation functions will increase the efficiency of the algorithms. Instead of writing individual evaluation functions for each soft constraint, they developed a single evaluation function. This function accepts certain data structures (called numberings and counters) which are cleverly initialised for each constraint. Performance gains occur due to the fact that certain soft constraints are able to share these data structures, hence reducing the number of evaluation function calls. For instances in which these data structures are significantly shared, this method is very efficient. De Causmaecker and Vanden Berghe [86] present algorithms for improving roster quality by manipulating coverage constraints. The algorithms mimic the way expert human planners sometimes alter coverage constraints in order to increase the quality of the nurses' work patterns.

Blöchliger [42] provides a tutorial on modelling a nurse scheduling problem with three objectives: minimize employment costs, maximize fairness by evenly distributing unpopular shifts and minimize soft constraint violations. No method for solving the problem is given but potential general approaches are suggested.

It is worth mentioning that there is also a very large body of research around the effects of different nurse scheduling policies. Examples include:

- The health risks of shift work, such as the disruption of circadian rhythms and sleep disorders [207].
- The benefit of providing flexible working in order to encourage commitment to nursing [45].
- The effect of longer shifts on performance [103].
- Stress due to shift working [171] and so on.

Finally, the publications reviewed in this chapter have primarily focussed on nurse rostering. It is worth noting though that the physician scheduling problem can be very similar and methods which are effective for this problem may also have applications in nurse rostering. See for example [29, 72, 110, 150, 218].

2.7 Conclusion

A number of approaches have been used to address the problem of automated nurse rostering. In this chapter, these methods have been placed in ten different categories. Arguably, these categories could have been further divided and in the future, novel methods for solving this problem are likely to appear. Most of the approaches have been shown to produce high quality rosters. There is no way of knowing which the ‘best’ method is. Implementing and comparing all the different algorithms over all the published problems would be an impractical task. In the author’s experience, many researchers have a favourite method and will argue its strengths (occasionally even dismissing all other techniques). In reality, it is likely that different methods will dominate on different instances and variations of the problem. Also, it is well known that any algorithm is only as good as its implementation. It can also be significantly improved by ‘tailoring’ it

to the problem at hand with heuristics and other operators that can exploit some structure or knowledge of the problem (or even problem instance).

It is also interesting to see how the approaches used are highly dependent on the technology available at the time. Early systems were severely constrained in terms of the problem complexity that was examinable by computational limitations and also by user interface restrictions. For example, in some of the early approaches punch cards were used to input data and paper forms were needed for data collection. As computing power has increased, the scheduling approaches have become more flexible and take into account more working preferences. For example, in 1983, Blau and Sear [41] suggest that the 1-2 hours run time could be reduced through use of "*a hard disk integrated with this microcomputer*" and a "*microprocessor with a clock speed of 4MHz instead of 2Mhz*". Some of the current state of the art approaches to automated nurse rostering require similar run times on personal computers with 3000MHz processors and numerous other improvements. This highlights either a serious lack of progress over the past 25 years, or more likely, the limitations on the size and complexity of the problems that could be solved in the past, and the increase in complexity of the problems that are solved now. This increase in computing power is also set to continue in the future so we should expect even better solutions produced more quickly on even harder real world problems.

All the publications reviewed in this chapter make a contribution of some degree to the collective body of personnel scheduling research. However a number of papers are particularly noteworthy either for their originality, quality and/or practical impact. These include:

- [242] : Warner provides not only an early nurse rostering paper but also one of the first mathematical programming models. A number of subsequent researchers build on this work.
- [92] : Dowsland and Thompson provide an excellent example of combining Operations Research methodologies to produce high quality rosters. Later research on the same problem confirmed the strength of this approach.
- [178, 179] : Meyer auf'm Hofe's publications are notable for a very effective constraint programming approach and for the rostering software's commercial success.
- [59] : Burke et al. contributed a number of novel metaheuristic approaches for a very flexible and practically oriented rostering model. The software and algorithms were successfully used in a large number of hospitals.
- [30, 32] : Petrovic and Beddoe's work is significant as being entirely novel and the first real application of case based reasoning to personnel scheduling. Even though the approach removes many of the familiar features of search and optimisation algorithms (such as objective functions, branching methods, search neighbourhoods etc), it is still able to produce high quality rosters.

Although automated rostering systems are becoming more common in hospitals, in the author's opinion, they will not become pervasive until a number of goals are achieved. Firstly, user interfaces need to be significantly improved. They should be more attractive and more user friendly. Entering new data, changing

constraints and modifying schedules should be as intuitive, fast and easy as possible. Developing software which allows this is not a simple task, but there appear to be no reasons to suggest that it is impossible. It will also become easier as users become more technologically capable and aware. Secondly, the algorithms need to become more powerful to solve a wider range of problems more quickly. This goal is being made easier by increases in hardware performance but there is also a significant level of potential for improvement in the algorithms themselves. Benchmark problems from real world environments would be particularly useful as a means for improving and validating the algorithms. Creating useful real world benchmark instances is not easy, however, as they are nearly always very complicated problems. Chapter 4 of this thesis discusses this issue further and presents a viable way forward. The rest of the thesis is concerned with developing improved nurse rostering algorithms, in particular, with harnessing the computing power currently available.

3 A Hybrid Heuristic Ordering and Variable Neighbourhood Search

This chapter describes a new approach for solving a complex and challenging nurse rostering problem. The algorithm was developed in collaboration with ORTEC, a major supplier of software products and consultancy in the field of advanced planning and scheduling. One of the main goals of the research described in this chapter was to develop an effective and efficient search approach to improve upon the genetic algorithm based technique that is currently employed within ORTEC's commercially available Harmony software[1]. As such, the methodology has to be able to handle all the requirements and constraints that are inherent in the nurse rostering problems that appear in the modern complex environments that are represented by today's hospitals.

The developed algorithm combines a variable neighbourhood search with a method of heuristically unassigning shifts and then repairing rosters using heuristic ordering. Variable neighbourhood search (VNS) is a metaheuristic first proposed by Mladenović in 1995 [184]. It has proved to be very popular with successful applications to a number of problems. It is described in detail in [127-129]. Its main feature, however, is a simple but effective idea: that of changing neighbourhood operators during a search. If a local optimum is reached using one neighbourhood, it is possible that this optimum can be *escaped* from by examining solutions in another neighbourhood. This basic principle is exploited by the approach presented here. The search is extended, though, by a heuristically guided solution disruption and repair

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

phase. The approach is evaluated using commercial data against Harmony's commercial strength genetic algorithm.

3.1 ORTEC Problem Description

The data for this problem was provided by ORTEC. They support hospitals and other organisations all over the world with automated workforce management solutions.

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

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

3.1.1 Shifts and Shift Demand

There are 4 different shift types in the problem: day, early, late and night shifts. All the shifts except night shifts cover 9 hours including one hour of rest time. So the actual number of working hours for each shift type is 8. Night shifts last 8 hours but include no rest time and so are counted as 8 working hours. The total cover requirements for each shift for each day varies between instances. Generally, larger

wards require more nurses on duty during each shift but similar sized wards can also have different cover requirements.

Table 3.1 shows the daily demand for these shifts in the instance described earlier with 16 nurses.

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

Table 3.1 Shift types and example weekly demand

3.1.2 Hard Constraints

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

- Shift cover requirements need to be satisfied. Over coverage is not permitted.
- A nurse may start only one shift per day.
- The maximum overtime assigned to each nurse per month is 4 hours.
- The maximum hours worked per week is on average 36 hours over a period of 13 consecutive weeks which do not include night shift assignments.
- The maximum number of night shifts in any period of 5 consecutive weeks is 3.
- A nurse must receive at least 2 weekends off in any 5 week period. A weekend off lasts 60 hours including Saturday 00:00 to Monday 04:00.
- Following 2 or more consecutive night shifts, a 42 hour rest is required.
- During any period of 24 hours, at least 11 hours rest is required. A night shift has to be followed by at least 14 hours rest. Once in a period of 21 days, however, the rest period may be reduced to 8 hours.

- The maximum number of consecutive night shifts is 3.
- The maximum number of consecutive days worked is 6.

3.1.3 Soft Constraints

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

Weight 1000 : The constraint should not be violated unless absolutely necessary.

Weight 100 : The constraint is strongly desired.

Weight 10 : The constraint is preferred but not critical.

Weight 1 : Try and obey this constraint if possible but it is certainly not essential.

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

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

Table 3.2 Soft Constraints

3.1.4 Evaluation Function

The evaluation function is the sum of all the penalties incurred due to soft constraint violations. The penalty for each soft constraint is calculated either linearly or quadratically using the violation measurement factors listed in Table 3.2. The

violation measurement factor is the degree to which the constraint is violated or the excess of the violation. The use of either quadratic or linear evaluation functions arises from practices in Harmony which were developed based on customer preferences and feedback.

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

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

It is now possible to define the objective of the problem: To find a feasible roster with the lowest possible penalty caused by soft constraint violations. From the perspective of the head nurse, of course, the actual penalty hides a lot of information about the solution but it is not totally meaningless. By examining the penalty for each schedule it is possible to gain some idea of the schedule quality. For example, if the penalty is less than 1000 then we know that all the constraints with weight 1000 have been satisfied. However, the key to producing satisfactory schedules is obviously setting the correct weights and ensuring that all the required constraints are defined.

Therefore, it is essential that the end user either has a good understanding of how to set the weights and define constraints or has clearly described the requirements to the software administrator.

To further clarify the problem and provide a more formal description it will now be presented as an integer programming model. The model was originally produced by Jingpeng Li and can be found in [64].

Parameters:

I = Set of nurses available.

$I_t \mid t \in \{1,2,3\} =$ Subset of nurses that work 20, 32, 36 hours per week respectively, $I = I_1 + I_2 + I_3$.

J = Set of indices of the last day of each week within the scheduling period = {7, 14, 21, 28, 35}.

K = Set of shift types = {1(*early*), 2(*day*), 3(*late*), 4(*night*)}.

K' = Set of undesirable shift type successions = {(2,1), (3,1), (3,2), (1,4)}.

d_{jk} = Coverage requirement of shift type k on day j, $j \in \{1,...,7|J|\}$.

m_i = Maximum number of working days for nurse i within the scheduling period.

n_1 = Maximum number of consecutive *night* shifts within the scheduling period.

n_2 = Maximum number of consecutive working days within the scheduling period.

c_k = Desirable upper bound of consecutive assignments of shift type k.

g_t = Desirable upper bound of weekly working days for the t-th subset of nurses.

h_t = Desirable lower bound of weekly working days for the t-th subset of nurses.

Decision variables:

x_{ijk} = 1 if nurse i is assigned shift type k for day j, 0 otherwise

The soft constraints are formulated as goals. The overall objective function is:

$$\text{Min}\,\overline{G}(x) = \sum_{i=1}^{8} w_i \overline{g}_i(x),$$

Where the goals are:

1. Complete weekends (i.e. Saturday and Sunday are both working days or both off).

$$\overline{g}_1(x) = \sum_{i\in I}\sum_{j\in J}\left|\sum_{k\in K}[x_{i(j-1)k} - x_{ijk}]\right|$$

2. Avoid isolated shifts (i.e. a working day preceded and followed by a day off).

$$\overline{g}_2(x) = \sum_{i\in I}\sum_{j=2}^{7|J|-1}\max\left\{0, \sum_{k\in K}[-x_{i(j-1)k} + x_{ijk} - x_{i(j+1)k}]\right\}$$

3. A minimum number of days off after a series of shifts.

$$\overline{g}_3(x) = \sum_{i\in I}\sum_{j=2}^{7|J|-1}\max\left\{0, \sum_{k\in K}[x_{i(j-1)k} - x_{ijk} + x_{i(j+1)k}] - 1\right\}$$

4. A maximum number of consecutive shifts of a specific type.

$$\overline{g}_4(x) = \sum_{i\in I}\sum_{r=1}^{7|J|-3}\sum_{k\in\{1,3\}}\max\left\{0, \sum_{j=r}^{r+3} x_{ijk} - c_k\right\}$$

5. A minimum number of consecutive shifts of a specific type.

$$\overline{g}_5(x) = \sum_{i\in I}\sum_{j=2}^{7|J|-1}\sum_{k'\in\{1,3\}}\max\left\{0, -x_{i(j-1)k} + x_{ijk} - x_{i(j+1)k}\right\}$$

6. A minimum and maximum number of working days per week.

$$\overline{g}_6(x) = \sum_{t=1}^{3}\sum_{i\in I_t}\sum_{w=1}^{|J|}\left[\max\left\{0, \sum_{j=7w-6}^{7w}\sum_{k\in K} x_{ijk} - g_t\right\} + \max\left\{0, h_t - \sum_{j=7w-6}^{7w}\sum_{k\in K} x_{ijk}\right\}\right]$$

7. A maximum number of consecutive working days for part time nurses.

$$\overline{g}_7(x) = \sum_{i\in I_1}\sum_{r=1}^{7|J|-3}\max\left\{0, \sum_{j=r}^{r+3}\sum_{k\in K} x_{ijk} - 3\right\}$$

8. Avoiding certain shift rotation (e.g an *early* shift after a *day* shift).

$$\bar{g}_8(x) = \sum_{i\in I}\sum_{j=1}^{7|J|-1}\sum_{k'\in K'} \max\left\{0, x_{ijk} + x_{i(j+1)k_2} - 2\right\}$$

The constraints are:

1. Shift cover requirements.

$$\sum_{i\in I} x_{ijk} = d_{jk}, \quad \forall j \in \{1,...,7|J|\}, k \in K$$

2. A nurse may not start more than one shift each day.

$$\sum_{k\in K} x_{ijk} \le 1, \quad \forall i \in I, j \in \{1,...,7|J|\}$$

3. Maximum number of working days.

$$\sum_{j=1}^{7|J|}\sum_{k\in K} x_{ijk} \le m_i, \quad \forall i \in I$$

4. Maximum of three working weekends.

$$\sum_{j\in J}\sum_{k\in K} x_{ijk} \le 3, \quad \forall i \in I$$

5. Maximum of three night shifts.

$$\sum_{j=1}^{7|J|} x_{ij4} \le 3, \quad \forall i \in I$$

6. A minimum of two consecutive night shifts.

$$x_{i(j-1)4} - x_{ij4} + x_{i(j+1)4} \ge 0, \quad \forall i \in I, j \in \{2,...,7|J|-1\}$$

7. A minimum of two days off after a series of consecutive night shifts. This constraint is equivalent to the following three sub-constraints which rule out the sequences of ‘N01’, ‘N10’ and ‘N11’ respectively, where ‘N’ denotes a *night* shift, ‘0’ an off-duty day and ‘1’ an on-duty day:

$$x_{i(j-1)4} - \sum_{k\in K} x_{ijk} + \sum_{k\in K} x_{i(j+1)k} \le 0, \quad \forall i \in I, j \in \{2,...,7|J|-1\}$$

$$x_{i(j-1)4} + \sum_{k\in K} x_{ijk} - \sum_{k\in K} x_{i(j+1)k} \le 0, \quad \forall i \in I, j \in \{2,...,7|J|-1\}$$

$$x_{i(j-1)4} + \sum_{k \in K} x_{ijk} + \sum_{k \in K} x_{i(j+1)k} \leq 2, \quad \forall i \in I, j \in \{2,...,7|J|-1\}$$

8. Maximum number of consecutive night shifts.

$$\sum_{j=r}^{r+n_1} x_{ij4} \leq n_1, \quad \forall i \in I, r \in \{1,...,7|J|-n_1\}$$

9. Maximum number of consecutive working days.

$$\sum_{j=r}^{r+n_2} \sum_{k \in K} x_{ijk} \leq n_2, \quad \forall i \in I, r \in \{1,...,7|J|-n_2\}$$

As discussed earlier and the weights w_i (given in Table 3.2) are set based on user preferences.

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

For all the instances we tested, we were able to produce feasible rosters. It is possible though that there may be an instance for which a feasible roster does not exist. In

practice, if a feasible roster cannot be found (either because one does not exist or it is too difficult to find) then the head nurse or manager decides whether to work with the best infeasible roster or relax some of the constraints or hire extra personnel and/or to assign some extra nurses to the ward (usually agency or float nurses) and then restart the search.

3.2 The Hybrid Variable Neighbourhood Search Algorithm

The algorithm presented in this section represents an iterative process in which variable neighbourhood search is followed by a roster disruption and repair strategy. The repairing of the roster is performed using a heuristic ordering technique. Backtracking is also carried out to further improve the quality of the produced schedules.

The overall process is illustrated by the pseudocode in Figure 3.1.

```
1.  Create Initial Roster
2.  REPEAT
3.     Variable Neighbourhood Search
4.     IF current penalty < best penalty  THEN
5.        SET best roster to current roster
6.        SET best penalty  to current penalty
7.     ELSE
8.        SET current roster to best roster (i.e. backtrack one step)
9.     ENDIF
10.    Unassign shifts of a set of nurses
11.    Repair roster (using heuristic ordering method)
12. UNTIL search terminated
```

Figure 3.1 Pseudocode of the overall hybrid VNS algorithm

3.2.1 Initialisation

A heuristic ordering is used to create the initial roster. In the experimentation section, the approach is compared against a commercial genetic algorithm developed by ORTEC and in use in real hospital environments. The commercial genetic algorithm that this hybrid variable neighbourhood search is evaluated against uses a similar heuristic ordering method to create its initial population of rosters.

The aim of the heuristic ordering process is to sort all the shifts in order of the estimated difficulty of assigning them or by how likely they are to cause high penalties (by using the criteria shown in Table 3.3). Using the weighted sum to identify them, the more troublesome shifts are then assigned earlier on in the roster construction process.

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

The attributes of a shift that are examined when ranking the shifts in the order of *possible difficulty to assign* are described in Table 3.3 along with the functions used to assign its total weight for ranking.

Shift Criteria	**Evaluation Function**	**Weight**
Night Shift	Weight	100
Weekend Shift	Weight	50
Number of valid nurses	(NumValidNurses / TotalNumNurses) * Weight	70
Shift Date	Weight * (Roster.EndDate – Shift.BeginDate)	20

Table 3.3 Heuristic ordering shift evaluation criteria

The first two criteria in Table 3.3 are obvious to examine as there are high penalties associated with night shift and weekend shift constraints. The third criterion used is to deduce how many nurses are able to fulfil this shift. If there are many nurses able to undertake it, then it can be scheduled later but if there are very few then it is a good idea to assign it early on in the process. The shift date criteria is used to try and ensure that the shifts in the early days in the scheduling period are assigned earlier on in the process. This is useful as these shifts are more likely to conflict with the previous schedule's assignments. The shift date evaluation function is in units of days.

3.2.2 Variable Neighbourhood Search

When the initial roster has been created using the heuristic ordering method described above, a variable neighbourhood search is applied. This makes use of two neighbourhoods. Both of these neighbourhoods are commonly used by metaheuristic and other approaches and have been described before, see, for example, [141, 159, 176, 204]. The two neighbourhoods are defined by the following moves or changes to a roster:

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

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

Our variable neighbourhood approach is a variable neighbourhood descent. As can be seen from Figure 3.2, the smaller neighbourhood (neighbourhood 1) is repeatedly examined for an improving move and the move is executed if found. When there are no improving moves left in neighbourhood 1, then the much larger neighbourhood 2 is examined. If a move in neighbourhood 2 is used then neighbourhood 1 is examined again. This is repeated until there are no improving moves left in neighbourhood 1 and 2.

```
1.  SET MoveMade to TRUE
2.  WHILE MoveMade is TRUE
3.      SET MoveMade to FALSE
4.      FOR each move in neighbourhood one
5.          IF an improving move THEN
6.              make this move
7.          END IF
8.      END LOOP
9.      FOR each move in neighbourhood two
10.         IF an improving move THEN
11.             make this move
12.             SET MoveMade to TRUE
13.         END IF
14.     END LOOP
15. ENDWHILE
```

Figure 3.2 Pseudocode of VNS

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

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

It was interesting to discover that, for this problem, using these neighbourhoods, the quickest descent method was not only faster than steepest descent but it was usually at least as good and sometimes better in comparison. This was an interesting observation that was initially difficult to understand. On closer investigation, though, a possible explanation became apparent. The heuristic ordering is very effective at satisfying the constraints with the highest penalties. This means that the soft constraint violations that the VNS needs to repair are often ones with smaller similar

sized penalties. If there is a high probability that all the possible improving moves will yield a similar sized improvement, it is not efficient to be examining all of them to find the absolute best if it will only be slightly larger than the average of all the available improving moves.

3.2.2 Roster Feasibility

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

3.2.3 Roster Disruption and Repair

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

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

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

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

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

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

To identify the best number of nurses from which to unassign shifts, a number of experiments were conducted on each instance in which this number ranged between 1 and 14. The results are provided in section 3.3.1.

3.2.4 Genetic Algorithm

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

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

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

3.3 Results

To develop this algorithm, the workforce management and planning software ORTEC Harmony [208] was used. Using Harmony provided a number of advantages

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

The experiments were performed using a PC with a Pentium 4, 2.4GHz processor.

3.3.1 Varying the Number of Nurses to Unassign Shifts From

Table 3.4 presents the results of varying the number of nurses from which to unassign shifts in the disruption and repair phase. The ‘penalty after first VNS’ column is the penalty of the roster after the VNS is first applied to the initial roster. The columns ‘1’ to ‘14’ show the penalty of the best roster found after the search has been applied for one hour when that number of nurses were selected for shift unassignment during the disruption.

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

to fourteen providing little improvement suggesting that there is too much diversification.

There does not seem to be any correlation between the size of the instance, in terms of the number of nurses, and the optimal number of nurses to use for unassignment. Three to five appears to be the best range for instances with varying sizes.

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

				Number of nurses selected for unassignment													
Instance	Nurses	Genetic algorithm	Penalty after first VNS	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	30	3626	3766	3766	3721	3746	3751	3766	3751	3766	3766	3766	3766	3766	3766	3766	3766
2	30	2381	2390	2375	2330	2285	2285	2295	2390	2390	2390	2390	2390	2390	2390	2390	2390
3	28	4325	4687	4557	4476	4687	4687	4687	4687	4301	4687	4687	4687	4687	4687	4687	4687
4	26	1301	1366	1311	1246	1231	1216	1151	1081	1186	1035	1191	1176	1366	1366	1366	1366
5	24	5230	6575	6450	5340	5291	6545	5465	6575	6575	6350	6575	6575	6575	6297	6575	6575
6	24	25406	32171	31986	31896	31971	27038	29931	28826	28826	28821	29857	28941	32141	32171	29925	29874
7	22	15661	22602	22602	21476	15550	16601	21756	15276	18325	21055	16348	15437	16455	19480	18551	22353
8	22	22877	25829	25824	23694	24808	23678	24799	24843	22991	22822	23926	23851	23716	23926	24717	24663
9	20	22478	24297	24277	24174	24228	24163	23298	24284	23394	23334	24297	24297	24297	24297	24297	24297
10	18	Infeasible	Infeasible	Infeasible	Infeasible	15706	15696	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible
11	18	4525	4546	4546	4530	4460	4506	4546	4546	4546	4546	4546	4546	4546	4546	4546	4546
12	16	775	996	905	831	760	690	805	830	862	996	951	996	996	996	996	996
13	14	1757	5026	4951	2741	1596	1591	1597	1740	1770	5026	3951	5026	5026	5026	5026	5026
14	14	760	800	755	591	556	645	621	700	650	790	800	800	800	800	800	800
15	13	1500	1626	1345	1275	1365	1401	1341	1550	1610	1626	1626	1626	1626	1626	1626	
16	12	18202	18873	18873	14746	18822	15867	11850	13000	16121	16991	18873	16052	18873	18873		
Average improvement in penalty on roster found after first VNS (%)				2.7	11.4	13.6	13.9	13.7	12.4	11.1	4.5	5.1	4.9	2.2	1.6	1.9	0.9

Table 3.4 Results for the Hybrid VNS

3.3.2 Comparison of the Hybrid VNS with the Genetic Algorithm

If the number of nurses selected in the disruption phase is three or four, then the hybrid VNS outperforms the genetic algorithm on nine of the sixteen instances. Using the Sign test and the Wilcoxon signed rank test (both are non-parametric) this is not however statistically significant at the 0.05 level. (The null hypothesis (H_0) tested was that the difference in objective values of the solutions produced by the two algorithms are symmetrically distributed around the central point of zero. Note this is a two tailed test. For the ranked test, the difference between an infeasible and feasible solution is ranked higher than the difference between two feasible solutions.)

Interestingly, the hybrid VNS appears to be more effective on the instances with less than twenty nurses. For example, in the experiments in which four nurses are selected, all the rosters found for instances with less than twenty nurses have lower penalties than the genetic algorithm (this was not significant at the 0.05 level using the Sign test and the Wilcoxon signed rank test though). If three are selected, the hybrid VNS outperforms on all but one. This was significant at the 0.05 level using both the Sign test and the Wilcoxon signed rank test.

It can also be seen that using the VNS phase alone is not sufficient to outperform the genetic algorithm. For all instances, after the first VNS iteration, the roster is worse than the final roster produced by the genetic algorithm (significant at the 0.05 level using both tests). The disruption and repair phases are required to further improve the roster.

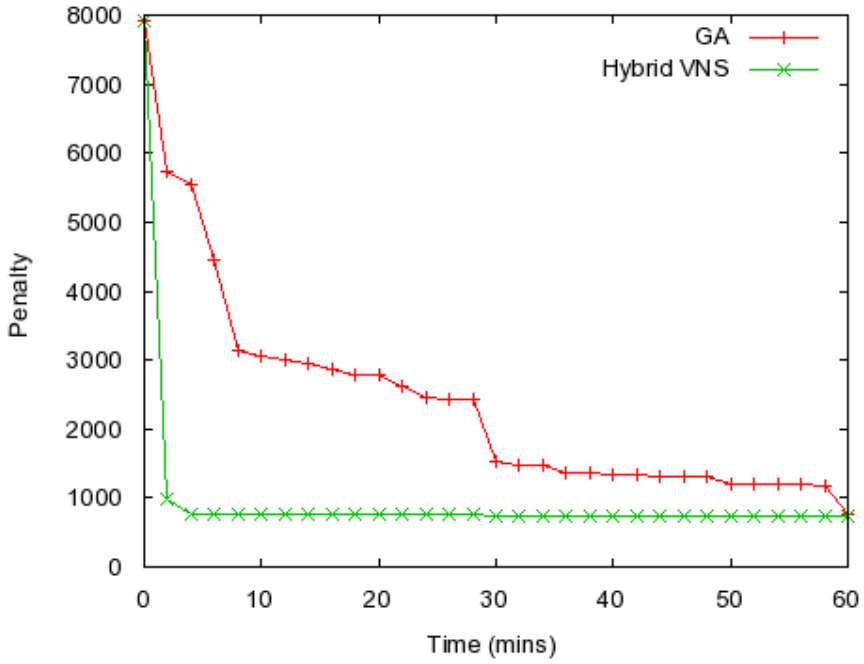

Figure 3.3 Penalty vs. time for the GA and Hybrid VNS

Figure 3.3 shows the progress of the two algorithms in finding rosters for instance 12. The graph shows the penalty for the best roster found so far for each algorithm after *x* minutes. For the genetic algorithm, a steady decrease in penalty can be seen over the sixty minutes as, after each generation, a new best roster is often found as a result of the crossover and repair operations. A drop of over 1000 in penalty in under a couple of minutes is most likely due to one of the constraints with a weight of 1000 being satisfied as well as other small improvements being made. The relatively steep (as all the soft constraints with weight 1000 have now been satisfied) decrease in penalty in the last two minutes for the genetic algorithm is due to the final local search phase.

For the hybrid VNS, it can be seen that within four minutes (after a couple of iterations of the algorithm) the best roster already has a penalty close to that produced finally by the genetic algorithm at the end of the sixty minutes. Between the fourth and sixtieth minute, an additional better roster is found as a result of the

roster disruption, repair and VNS. From observing the algorithm when applied to the other scheduling periods, within the first sixty minutes there are usually three or four improvements in the best solution found between the fourth and sixtieth minute.

3.3.3 Longer Computation Times

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

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

After 12 hours, the hybrid VNS had found a roster with penalty 541. It is important to note that this approach is producing the best known solution (produced either automatically or manually) on this real world problem instance. Moreover, it is producing it within a period (overnight) which is quite appropriate for this kind of problem. The results are summarised in Table 3.5. As can be seen, if more computation time is given, the roster can be significantly improved.

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

Table 3.5 Experimentation with longer computation times

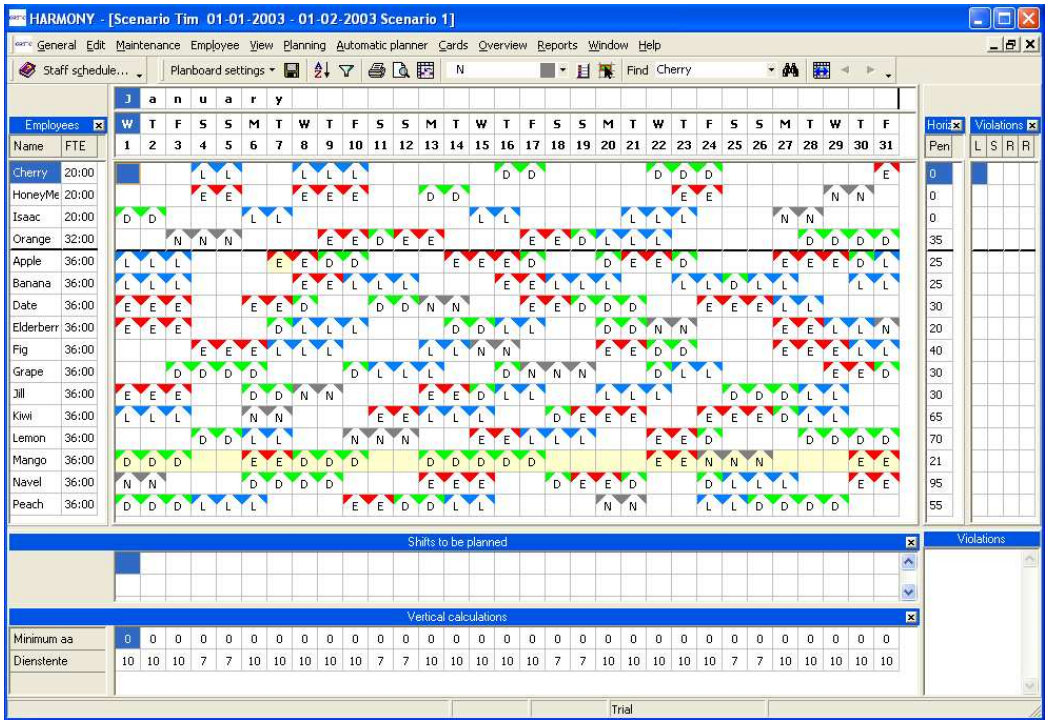

Figure 3.4 Screenshot of Harmony displaying a roster with penalty 541

3.4 Conclusion

The hybrid VNS algorithm described has been shown to be a relatively straightforward but highly effective approach for this problem. It is particularly effective on medium and small sized instances with less than twenty nurses. It is a viable alternative to the existing genetic algorithm for the commercial workforce management and planning software Harmony and has been added alongside the genetic algorithm in the latest versions. This work has also been accepted for publication in The European Journal of Operational Research [50].

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

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

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

4 A Rostering Engine and Benchmarks

The previous chapter presented a very successful and relatively simple algorithm for the nurse rostering problem. In order to validate the approach it was compared against a commercial strength, tried and tested, genetic algorithm. Comparing the method against an independently commercially implemented, successful algorithm was an analysis that other nurse rostering researchers have been unable to perform. There are a number of reasons for this. This chapter discusses these problems and describes work that has been performed in order to overcome them. The result is a framework for describing and sharing nurse rostering problems in order to establish benchmarks and to underpin the research base in this area. A foundation has been created that will allow high quality, practically orientated nurse rostering research to be conducted. The results of which would have direct practical benefits. A number of other issues are also discussed such as guidelines for experimentation, research aids and possible avenues of research.

Benchmark instances have played an important role in the progress made on a wide range of problems. They are a useful tool in developing and validating increasingly powerful solvers and are a source of competition and collaboration which often drives progress. Problems ranging from satisfiability [133] to the travelling salesman problem [215] to examination timetabling [211] all have benchmark instances. It is somewhat surprising, however, that the nurse rostering problem has very few publicly available benchmark instances, especially when considering its common occurrence, its practical applications and the wealth of research literature in the area. In the author’s opinion, the reasons for this are

twofold. Firstly, there is no typical nurse rostering problem. Nearly all the published research tackles a slightly different version of the problem with varying constraints, objectives and associated priorities. Secondly, defining and maintaining a data format which could contain and describe all these variations of the problem is a considerable and challenging task. The most appealing solution to this lack of benchmarks would perhaps be to define an easily manageable, simplified nurse rostering problem which contains a reduced set of constraints and a simple objective function. That approach, however, has very definite disadvantages.

At the 2006 Practice and Theory of Automated Timetabling (PATAT) conference, a plenary presentation given by Barry McCollum was titled "University Timetabling: Bridging the Gap between Research and Practice" [170]. One of the key points made by the speaker was that the research in university timetabling does not always follow the direction of or keep up with the requirements and complexities of the 'real world'. The reason for this can be partly attributed to the benchmark data sets used by researchers to test and develop the latest algorithms. Unfortunately, the data sets are too often simplified and infrequently updated to reflect the dynamic and complex scenarios found in modern real world environments. This partly occurs due to the fact that it is a long, time consuming process for researchers to write the code to deal with these complex problems. Although this may or may not be a valid reason, the major difficulty, is that it will become harder to justify research when the practical benefits become less apparent (i.e. without updating, the real problems will move further from the researchers' data sets). The ideal solution to this gap between research and practice would be a system for describing and sharing real world

problems which could be used as benchmarks and which minimises the start up time required by researchers to examine and provide solutions to these complex real world scenarios.

To investigate whether this is possible, a data format has been developed (and is continuing to be extended) which can describe complex nurse rostering problems with multiple, complicated constraints and objectives. A key design goal was to produce a format which can evolve to handle new problem formulations as they appear. All software and source code developed to work with this data (including solvers) has also been made available. This is important not only to reduce the researcher's burden and to make these unwieldy problems more accessible but also to provide validation and verification of new solutions. A facility that is essential for complicated benchmark problems.

4.1 Other Nurse Rostering Benchmarks and Data Formats

This is not the first attempt at defining a format for employee timetabling problems in order to share instances and establish benchmarks. Meisels and Schaerf [175, 176] described a general model and format for employee timetabling and provided some real world instances [http://www.cs.bgu.ac.il/~am/ETP_Home/Main_Page.html]. Unfortunately though, the project appears to have received little recent work in order to develop it further and incorporate a wider variety of real world instances.

Özcan [200] proposed an XML format for timetabling problems called TTML. This format was flexible enough to describe a real world nurse rostering problem [201] which is available online [http://cse.yeditepe.edu.tr/~eozcan/research/TTML/]. Again though, it does not

appear to have received much further attention in order to expand the number and variety of instances which can be presented.

A recent benchmark project has been started by Vanhoucke and Maenhout [239, 240]. They have created a nurse scheduling problem library called NSPLib. This is a useful contribution but currently the number of constraints is limited. Therefore, the large number of computer generated instances may not be reflective of the real world. However, they are able to generate a wide variety of instances in terms of size and complexity which data sets based on real world problems can not always achieve. As such, our projects complement each other well.

4.2 About the Format

The data instances and solutions are presented using **Ex**tensible **M**arkup **L**anguage (XML). XML is a relatively new technology (work began in 1996) although it derives from SGML, the foundations of which began in the 1960's. One of its main design goals was to facilitate the sharing of data, particularly over the internet. It has proven to be very popular with wide and successful adoption in many industries. Like HTML, XML uses elements (tags) and attributes. Unlike HTML however, it allows new tags to be introduced (i.e. it is extensible).

Using XML provides a number of benefits when working with data that will be shared and transferred:

- Schemas and DTD's (document type definition) can be used to validate the data and ensure that all the necessary information has been provided. For example, ensuring parameters are legal values and in the correct

format and undefined element ID's are not referenced. Schemas also facilitate the writing of more robust parsers.

- It is human readable and can be edited in any text editor.

- Tags and attributes provide a degree of self description. For example, most people would have a rough idea of what information is represented by `<EmployeeName>John Smith</EmployeeName>`.

- It is platform independent. For example, differences in newline characters between systems do not cause problems. XML also supports Unicode, allowing different languages with unusual characters to be included.

- Application programming interfaces and libraries for working with XML are available in most programming languages.

- Parsers can be designed to continue working correctly even when the file format has been updated (e.g. after adding new elements). This allows backwards compatibility and removes the need to update old files when the format is updated.

4.3 Overview of the Latest Version

The first version was based on ANROM (Advanced Nurse ROstering Model) [238]. However, as already emphasised, this is an evolving format. As new nurse rostering problems are encountered with new constraints and objectives, so the

format will be updated to allow this new information to be included. In its short history it has already had a number of revisions.

In the latest version (1.1) the information for a single instance can be broadly split into seven sections. The majority of information in each section is optional. This is a design feature that simplifies describing problems which may be significantly different to each other. An overview of each section now follows.

4.3.1 Shift Types

These are the different types of shifts which need to be assigned to employees over the scheduling period. Relevant information, for example, includes, start times and end times, hours counted as work (if different from the difference between the end time and start time), the skills required to perform this shift, the minimum rest times before and after the shift. It is also possible to specify information used for displaying schedules such as shift labels, descriptions and colours. An example shift definition in XML is shown in Figure 4.1.

```
<Shift ID="D1">
  <GeneralType>D</GeneralType>
  <Label>D1</Label>
  <Colour>Red</Colour>
  <Description>Head Nurse Day Shift</Description>
  <StartTime>09:00:00</StartTime>
  <EndTime>17:00:00</EndTime>
  <HoursWorked>8.0</HoursWorked>
  <FreeTimeBefore>660</FreeTimeBefore>
  <FreeTimeAfter>480</FreeTimeAfter>
  <Skills>
    <Skill>HeadNurse</Skill>
  </Skills>
</Shift>
```

Figure 4.1 Example shift definition

4.3.2 Employees

It is necessary to define the employees available during the scheduling period. Each employee is linked to a contract which specifies their working regulations

(constraints). Each employee's skills, qualifications or training are also specified to ensure that they are not assigned shifts that they cannot perform. An example employee definition in XML is shown in Figure 4.2.

Example employee definition in XML:

```
<Employee ID="ExampleEmployee">
  <Name>John Smith</Name>
  <ContractID>ExampleContract</ContractID>
  <EmploymentStartDate>2007-05-01</EmploymentStartDate>
  <Skills>
    <Skill>HeadNurse</Skill>
  </Skills>
</Employee>
```

Figure 4.2 Example employee definition

4.3.3 Contracts

Each employee is linked to a contract. An employee can have a unique contract but, more commonly, a number of employees share the same contract. Allowing employees to have unique contracts, however, provides a higher degree of flexibility. Within the contract is all the information regarding an employee's working regulations and preferences. For example, the minimum and maximum number of hours worked during the scheduling period, the minimum and maximum number of consecutive working days or consecutive free days etc. In the current version there are approximately twenty constraints that can be specified in a contract. They are:

- Maximum number of shifts worked during the scheduling period.
- Maximum and minimum number of hours worked during the scheduling period or per week.
- Maximum and minimum number of consecutive working days.

- Maximum and minimum number of consecutive non-working days.
- Maximum number of a specific shift type worked. For example, maximum zero night shifts for the planning period or a maximum of seven early shifts. This constraint can also be specified for each week. For example, a nurse may request no late shifts for a certain week.
- Maximum number of weekends worked in four weeks (a weekend definition is also a user definable parameter i.e. Friday and/or Monday may be considered as part of the weekend).
- Maximum number of consecutive weekends worked.
- No night shifts before a weekend off.
- No split weekends, i.e. shifts on all days of the weekend or no shifts over the weekend.
- Identical shift types over a weekend. For example, if a nurse has a day shift on Saturday then he/she may prefer to have a day shift on Sunday also.
- Minimum number of days off after night shifts.
- Valid numbers of consecutive shift types. For example, three or four consecutive early shifts may be valid but two or five consecutive early shifts may not.
- Shift type successions. For example, if shift rotation is allowed, is shift type A allowed to follow B the next day?
- Maximum total number of assignments for all Mondays, Tuesdays, Wednesdays… For example, a nurse may request not to work on Wednesdays or may require to work a maximum of two Tuesdays during the scheduling period.

- Avoid a secondary skill being used by a nurse. Sometimes a nurse may be able to cover a shift which requires a specific skill but they may be reluctant to do so as it is not their preferred duty. An example would be a head nurse not wanting to stand in for a regular nurse.

The majority of these constraints are from ANROM and their implementation is based on their specification in [238]. As such, all these constraints are modelled as soft constraints. Weights are used to reflect their relative priority and can be specified within a contract for each constraint. If they are not provided in the contract they are specified in a different section and globally for all contracts. An example contract definition in XML is shown in Figure 4.3.

```
<Contract ID="Trainee">
 <MaxNumAssignments>15</MaxNumAssignments>
 <MinNumAssignments weight="10">8</MinNumAssignments>
 <MaxConsecutiveWorkingDays>4</MaxConsecutiveWorkingDays>
 <MinConsecutiveWorkingDays>2</MinConsecutiveWorkingDays>
 <MaxWorkingBankHolidays>2</MaxWorkingBankHolidays>
 <MaxConsecutiveFreeDays>4</MaxConsecutiveFreeDays>
 <MinConsecutiveFreeDays>2</MinConsecutiveFreeDays>
 <MaxConsecutiveWorkingWeekends>2</MaxConsecutiveWorkingWeekends>
 <MaxWorkingWeekendsInFourWeeks>2</MaxWorkingWeekendsInFourWeeks>
 <WeekendDefinition>FridaySaturdaySunday</WeekendDefinition>
 <CompleteWeekends>true</CompleteWeekends>
 <TwoFreeDaysAfterNightShifts>true</TwoFreeDaysAfterNightShifts>
 <AlternativeSkillCategory>true</AlternativeSkillCategory>
 <MaxAssignmentsForDayOfWeek>
   <MaxAssignments>
     <Day>Tuesday</Day><Value>1</Value>
   </MaxAssignments>
   <MaxAssignments>
     <Day>Wednesday</Day><Value>1</Value>
   </MaxAssignments>
 </MaxAssignmentsForDayOfWeek>
 <MaxShiftTypes>
   <MaxShiftType>
     <ShiftType>N</ShiftType><Value>0</Value>
   </MaxShiftType>
 </MaxShiftTypes>
 <MaxShiftTypesPerWeek>
   <MaxShiftTypePerWeek>
     <ShiftType>E</ShiftType><Week>1</Week><Value>2</Value>
   </MaxShiftTypePerWeek>
 </MaxShiftTypesPerWeek>
 <MaxHoursWorked>140.00</MaxHoursWorked>
```

```
 <MinHoursWorked>100.00</MinHoursWorked>
</Contract>
```

Figure 4.3 Example contract definition

4.3.4 Cover Requirements

These are the number of shifts needing to be assigned to employees for each day in the planning period. That is, the number of employees required at certain time periods over the planning horizon. They can be specified for specific dates and/or for a general day of the week (e.g. ‘Monday’). An example partial definition is shown in Figure 4.4.

```
  <CoverRequirements>
    <DayOfWeekCover>
      <Day>Sunday</Day>
      <Cover>
        <Shift>E</Shift>
        <Count>2</Count>
        <Type>Required</Type>
      </Cover>
      <Cover>
        <Shift>L</Shift>
        <Count>2</Count>
        <Type>Required</Type>
      </Cover>
    </DayOfWeekCover>
    <DayOfWeekCover>
      <Day>Monday</Day>
      <Cover>
        <Shift>E</Shift>
        <Count>3</Count>
        <Type>Required</Type>
      </Cover>
      <Cover>
        <Shift>L</Shift>
        <Count>3</Count>
        <Type>Required</Type>
      </Cover>
    </DayOfWeekCover>
  </CoverRequirements>
```

Figure 4.4 Example cover requirements definition

4.3.5 Day On/Off and Shift On/Off Requests

Employees may request particular days on or off with associated priorities (set using weights). A request to work or not work a specific shift is also possible. Example definitions are shown in Figure 4.5.

```
  <DayOffRequests>
    <DayOff weight="1000">
      <EmployeeID>A</EmployeeID>
      <Date>2007-01-03</Date>
    </DayOff>
    <DayOff weight="1000">
      <EmployeeID>A</EmployeeID>
      <Date>2007-01-04</Date>
    </DayOff>
    <DayOff weight="10" holiday="false">
      <EmployeeID>B</EmployeeID>
      <Date>2007-01-20</Date>
    </DayOff>
  </DayOffRequests>

  <ShiftOnRequests>
    <ShiftOn weight="1">
      <ShiftGroupID>Early</ShiftGroupID>
      <EmployeeID>E1</EmployeeID>
      <Date>2007-02-05</Date>
    </ShiftOn>
    <ShiftOn weight="1">
      <ShiftGroupID>Late</ShiftGroupID>
      <EmployeeID>E2</EmployeeID>
      <Date>2007-02-06</Date>
    </ShiftOn>
    <ShiftOn weight="1">
      <ShiftTypeID>L</ShiftTypeID>
      <EmployeeID>E3</EmployeeID>
      <Date>2007-02-07</Date>
    </ShiftOn>
  </ShiftOnRequests>
```

Figure 4.5 Example day on/off and shift on/off request definitions

4.3.6 History

Some of the constraints are dependent on the previous schedule for an employee. For example, if there is a maximum number of consecutive night shifts, then we need to know if the employee had any night shifts at the end of the previous scheduling period before assigning any at the beginning of the current one. This information is provided in this section. An example is shown in Figure 4.6.

```
<EmployeeHistory EmployeeID="T">
  <LastDayType>WorkingDay</LastDayType>
  <LastDayShifts>
     <Shift>D1</Shift>
  </LastDayShifts>
  <PreviousConsecutiveWorkingDays>4
  </PreviousConsecutiveWorkingDays>
  <PreviousConsecutiveWorkingDaysAndHoliday>4
  </PreviousConsecutiveWorkingDaysAndHoliday>
  <PreviousConsecutiveFreeDays>0</PreviousConsecutiveFreeDays>
```

```
  <PreviousConsecutiveWorkingWeekends>0
  </PreviousConsecutiveWorkingWeekends>
  <PreviousWorkingBankHolidays>0</PreviousWorkingBankHolidays>
  <WeekendWorkedThreeWeeksAgo>false</WeekendWorkedThreeWeeksAgo>
  <WeekendWorkedTwoWeeksAgo>true</WeekendWorkedTwoWeeksAgo>
  <WeekendWorkedOneWeekAgo>false</WeekendWorkedOneWeekAgo>
  <PreviousSaturdayWorked>false</PreviousSaturdayWorked>
  <PreviousSundayWorked>false</PreviousSundayWorked>
  <PreviousSaturdayRequestedHoliday>false
  </PreviousSaturdayRequestedHoliday>
  <PreviousSundayRequestedHoliday>false
  </PreviousSundayRequestedHoliday>
  <NightShiftThursday>false</NightShiftThursday>
  <NightShiftFriday>false</NightShiftFriday>
  <PreviousFridayWorked>true</PreviousFridayWorked>
  <PreviousNightShift>false</PreviousNightShift>
  <PreviousFreeDaysAfterNightShift>0
  </PreviousFreeDaysAfterNightShift>
  <PreviousConsecutiveHolidayDaysOff>0
  </PreviousConsecutiveHolidayDaysOff>
  <PreviousConsecutiveShifts>
    <PreviousConsecutiveShift>
      <GeneralShiftTypeID>D1</GeneralShiftTypeID>
      <Count>1</Count>
    </PreviousConsecutiveShift>
  </PreviousConsecutiveShifts>
  <PreviousOvertime>0</PreviousOvertime>
</EmployeeHistory>
```

Figure 4.6 Example scheduling history definitions

4.3.7 Miscellaneous Information

There is a large amount of other information which may also need to be provided. For example, the start and end dates of the planning period. Groups of shifts may need to be defined if they are used by some of the constraints. Bank holidays need to be specified if the maximum working bank holidays constraint is used. If the constraint specifying employees who should or should not work together is used, then this also needs to be provided. Examples of some of this information are shown in Figure 4.7.

```
  <StartDate>2007-01-01</StartDate>

  <EndDate>2007-01-28</EndDate>

  <ShiftGroup ID="EarlyAndNight">
    <Shift>Early</Shift>
    <Shift>Night</Shift>
  </ShiftGroup>

  <Partnership Type="Tutorship">
```

```
    <Employee1ID>Tutor</Employee1ID>
    <Employee2ID>Tutee</Employee2ID>
  </Partnership>

  <BankHoliday>
    <Name>Christmas Day</Name>
    <Date>2006-12-25</Date>
  </BankHoliday>
```

Figure 4.7 Example miscellaneous information

The examples given here are very small compared to a full instance description. For examples of whole instances see the benchmark website http://www.cs.nott.ac.uk/~tec/NRP/ .

4.4 A Problem Definition

All the data is freely available for any research purposes. In order to use it as benchmark nurse rostering instances though, a problem must be defined. The definition derives from ANROM. Firstly, there are only three hard constraints:

1. A nurse cannot be assigned more than one of the same shift type per day.
2. Shifts which require certain skills can only be covered by (i.e. assigned to) nurses who have those skills.
3. The shift coverage requirements must be satisfied. For example, if a certain day requires three night shifts then there must be three employees present at that time to work during that shift. Over coverage is not permitted.

All other constraints are soft. The objective is to minimise a weighted sum of penalties due to soft constraint violations. The penalty for each constraint is calculated according to the functions given in [238].

As there are many constraints and their evaluation functions can be relatively complex, it may be naïve to assume that someone who independently implements

the functions found in [238] will produce an identical objective function. This would make any comparisons between solutions independently found, unreliable because, effectively, different problems are being examined. To avoid this, the source code is publicly available. An executable is also provided for checking solutions (described in an XML format). This will aid the process of checking and verifying new solutions. Researchers may also directly incorporate the source code into their own solvers rather than having to rewrite the objective function (a time consuming task).

When the format is extended to include different versions of the problem an option will be added to the format to select/define the objective function.

4.5 Research Aids

As discussed, it is important to make available the source code for the objective function for increasing the accuracy of new results. As the data files can be large and complicated, the source code for the parsers and the data structures used by the latest solvers is online too. This should considerably reduce the time required to start examining the problem.

The source code for the latest solvers is also made available. This is particularly useful for two reasons. Firstly, if a new problem is introduced (perhaps due to changing real world requirements) then there may be no benchmarks available. However, it may be possible to use a solver for an earlier version (perhaps with a little modification) to create some benchmark results for the new instances. Even if the results are not as strong on the new instances, it does create a source of comparison.

Secondly, providing the code for the solvers is a partial solution to the problem of comparing experiments performed on different machines. If two solvers are

compared using the same computer, it removes unreliability due to differences in computing power. Of course, one algorithm may still be more effective due to a more efficient implementation but at least one variable in the experiments has been removed.

Another tool provided converts solutions in an XML format into an HTML format for viewing the schedules in a web browser. This provides a visualisation of a solution by displaying the roster and schedules for each employee. Constraint violations are highlighted and soft constraints and employee requests presented. The penalties for soft constraint violations are also broken down and explained. This is a practical way of sharing solutions which can be viewed and understood by humans without having to download and install separate software. Figure 4.8 is a screenshot of a roster displayed using HTML (shifts in the roster are colour coded to help differentiate them).

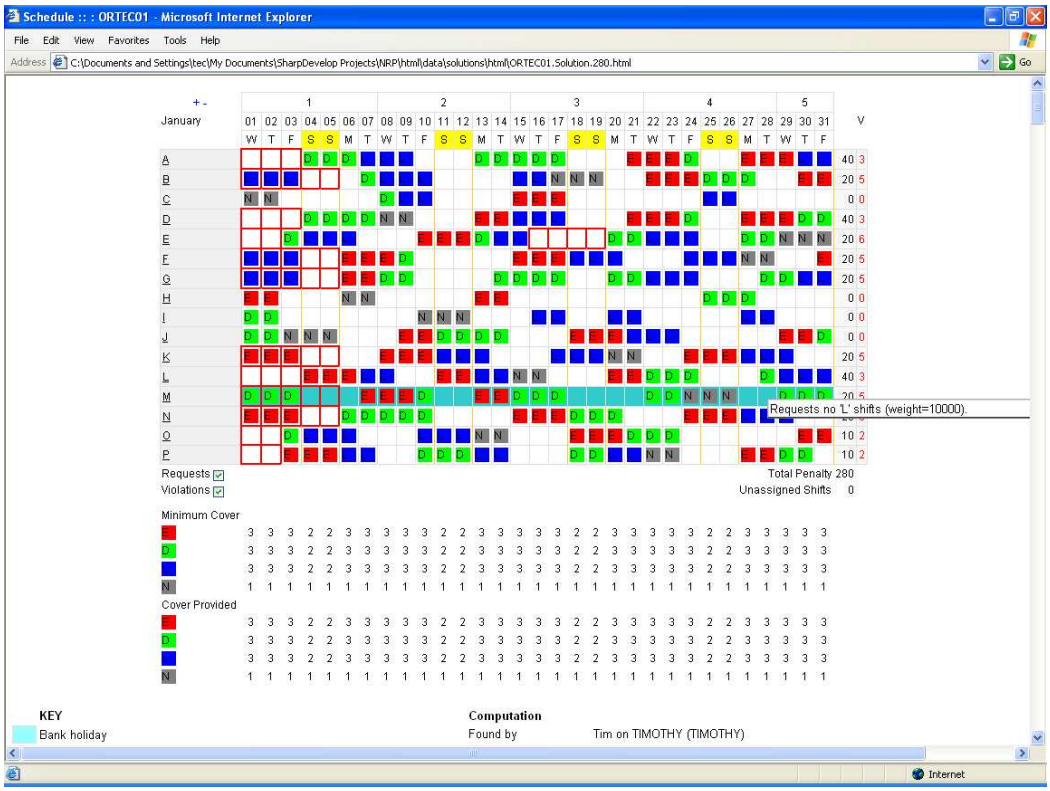

Figure 4.8 Screenshot of a schedule displayed in a web browser

A graphical user interface has also been developed to allow manual changes to solutions and provides an alternative view of schedules, their violations and penalty explanations. This gives a better ‘feel’ for the problems and an appreciation of their complexity. Figure 4.9 is a screenshot of the application. It is publicly available on the research website for download.

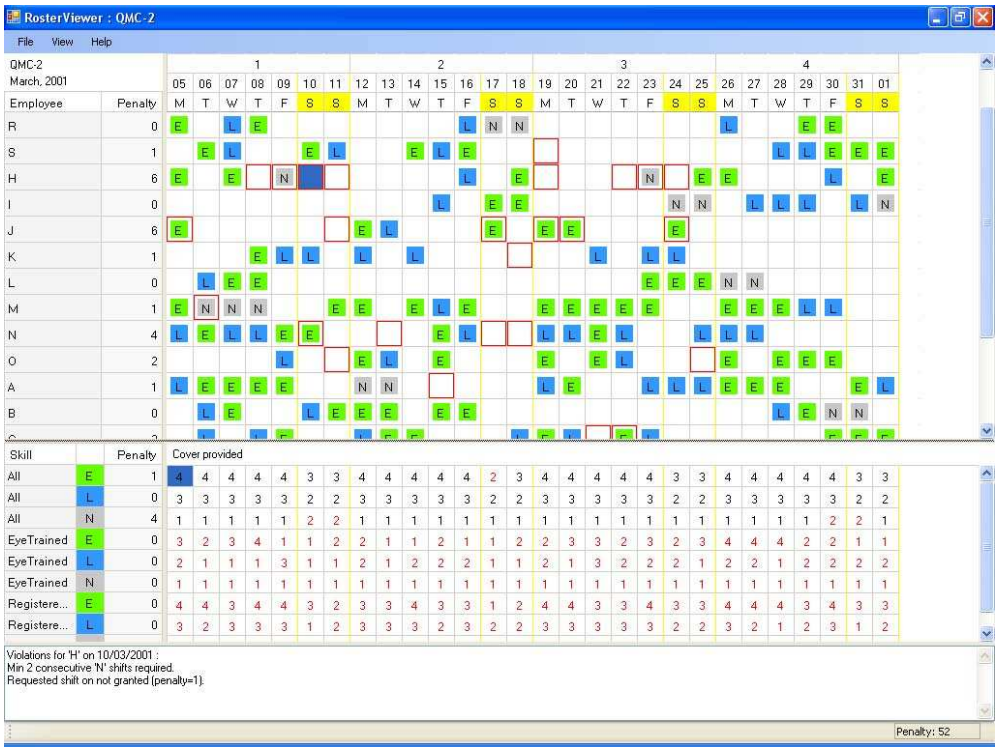

Figure 4.9 Screenshot of Nurse Rostering GUI

Another tool created whilst developing solvers provides a real time animation of the search algorithms. For neighbourhood searches and constructive heuristics, it is possible to create visualisations of the algorithms by updating a graphical view of the current schedule whenever a new shift has been assigned or shifts are swapped between employees. This was particularly helpful when developing more complex approaches. For some complicated searches (such as the one described in chapter 5) it is relatively easy to introduce errors or miss areas of inefficiency and redundancy. For example, re-visiting solutions, cycling, examining bad solutions which a simple heuristic would have avoided etc. Being able to watch an animation of the searches helped to spot some of these things and improve the efficiency of the searches. To reduce the impact of the animation on the speed of the searches, the power of the latest video cards was exploited by using Microsoft's DirectX libraries to develop the animation software. At normal

running speed of the algorithms though, the animations were very fast and difficult to follow. A control was therefore added to dynamically adjust the speed of the algorithm and even pause it during its execution. The software also allows the rosters to be manually altered (when the animation is not running) by moving shifts between employees (using the mouse) to see the effect on the roster's quality. A screenshot is shown in Figure 4.10.

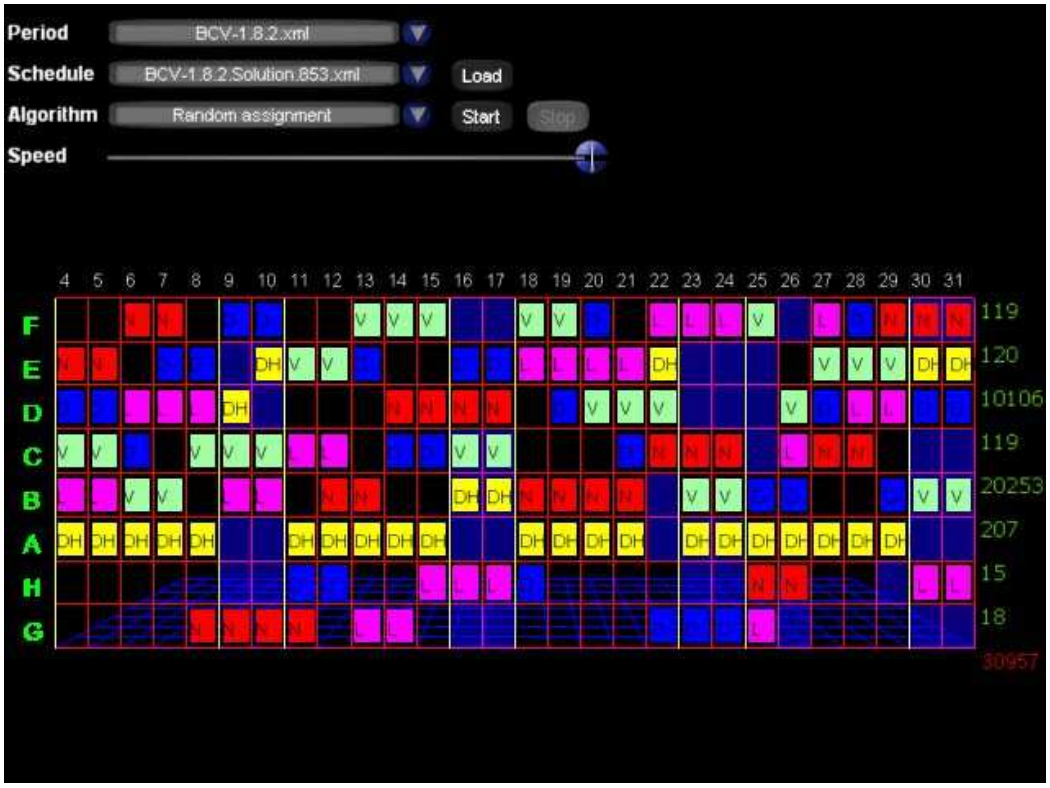

Figure 4.10 Screenshot of algorithm visualisation software

4.6 Guidelines for Experiments

The data was primarily compiled to create benchmark instances to real world nurse rostering problems. However, it is available for any research purposes. If it is used as a benchmark problem, a few guidelines for experiments and presenting results are suggested. These are only recommendations though which may help future research and make results more reliable. Most of these suggestions are well

known and generally accepted as good practice. For example, Schaerf and Di Gaspero [220] recently highlighted some of these ideas at PATAT06 and they can also be found in Johnson's well known paper [143].

- Provide solutions for verification and future use. An XML format has also been defined for presenting solutions.

- Provide an executable for a new solver. Ideally the source code should be provided as well. This may not always be possible though if it is commercially sensitive.

- For neighbourhood searches, record the number of solutions examined (Vanhoucke and Maenhout [240] even use a maximum number of solutions visited as the stop criterion for their benchmarks). This is a metric which is unaffected by the speed and technology of the machine executing the algorithm.

4.7 Research Uses and Possibilities

There are a number of possible research uses for the benchmark instances. The most obvious is to create better algorithms. It is unlikely that one algorithm will dominate over all instances. However, useful insights may be obtained by examining algorithms that are particularly strong on certain instances or groups of instances. These insights may be useful in creating more robust solvers that are effective over a wide range of instances.

Another research avenue that does not receive a lot of attention is improving the speed and efficiency of evaluation functions. These functions in nurse rostering

problems are usually long, complicated and subsequently slow and there is considerable benefit in making them faster. Burke et al. [53] have performed some research in this area, but there is scope for further investigation. Benchmark instances would confirm any improvements, especially if the code is publicly available.

Work has begun to try and identify optimal solutions to these instances, Even if this is not possible, good lower bounds may be found. Better lower bounds and optimal solutions would help gauge the strength of other results.

4.8 Design Features

The data format and software have been designed to make it as simple as possible to add new constraints and objectives, an exercise that will occur frequently as they are extended to handle a wider variety of nurse rostering problems. Within the format, nearly all the information is optional.

A few relatively simple steps are required to model a new problem.

1. Extend the format definition to allow any new information which varies between instances to be included.
2. Modify the parser and create any new data structures required by the new constraints or objectives.
3. Write the constraints and objectives.
4. If necessary, modify an existing solver to handle the changes.

None of the above steps are beyond the ability of an average programmer. The fourth step could be challenging depending upon the solver and the type of changes made. However, it would only be necessary if no solver existed for this new problem or an alternative solver was wanted for comparison purposes.

The software has been designed to allow new changes to be built on top rather than requiring everything to be rewritten each time. This also ensures everything is backwards compatible with the older versions.

In effect, the software can be regarded as an API (application programming interface) or nurse rostering engine. As discussed in section 4.5, it is possible to build a user interface (web based or graphical etc) around the software in order to view schedules, enter new data, run solvers etc. When the user interface reaches a quality suitable for use in practice, it could be an effective tool for obtaining new real world problem instances. A function could be built into the interface to anonymize any data entered and make it available for research. The users would then benefit by having researchers working on their latest problems. This would be a mutually beneficial and direct link between research and practice.

4.9 Instances Currently Available

A variety of instances described using the XML format are already available on the research website. Table 4.1 lists some of the data sets and their characteristics. They vary in the number of nurses, cover requirements, shift types, constraint types and priorities, personal requests and planning horizon.

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

Table 4.1 Example benchmark instances

Data sets BCV-1 to BCV-8 are all based on real world data and were originally modelled using ANROM. As the data was taken from a commercial system used in real world environments it has been anonymized and any confidential information removed. Data set BCV-A.12.1 is a fictional test problem that uses all the possible constraint types available and contains many conflicting requests. ORTEC01 is instance 12 for the problem presented in chapter 3 (the instance which the extra tests were performed on and the one commonly used by ORTEC). To model the problem with this system, some of its hard constraints have been changed to soft constraints with weights of 10,000. Therefore all solutions with penalty below 10,000 are feasible solutions to the original problem (solutions with penalties above 10,000 are not necessarily infeasible for the original problem though). Care was taken and a lot of testing and verification performed to ensure that the objective function is identical, so enabling comparisons to the original results.

Best known solutions to these instances (in the XML and HTML formats) are also available at the research website.

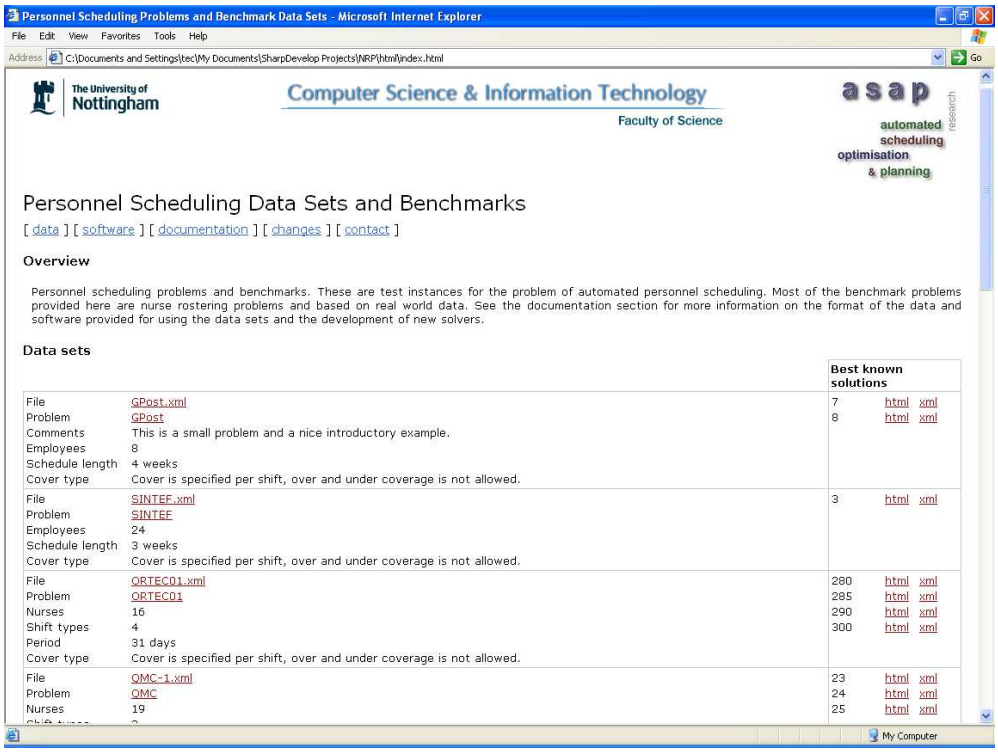

Figure 4.11 Screenshot of benchmark website

4.10 Conclusion

There is a large number of publications on solving nurse rostering problems. A problem with some of these published algorithms though is that there is rarely any way of knowing how ‘good’ they actually are. Readers have to place a certain amount of trust in the conclusions and anyone wanting to know the best way of solving a nurse rostering problem will have to use some guess work. This is not an ideal situation. As a potential solution to this problem, a flexible and extensible system has been proposed for sharing real world instances and benchmarks for nurse rostering problems. These benchmark instances will allow researchers to compare their algorithms to other approaches that have been independently implemented. This will increase the credibility of results and conclusions and help reviewers better gauge the strength of new methods.

Validation will be provided other than the usual “the rosters were better than those produced manually” or “the nurses were satisfied with the rosters”.

This need for benchmark nurse rostering problems has been expressed in a number of publications e.g. [60, 73, 228]. As well as defining a format for describing the problems, from the outset, the intention was to also provide software (and source code), in order to facilitate and encourage the use of this format. In effect, a nurse rostering engine has been created which is freely available for research purposes. The engine accepts problem data in a standard format and outputs a solution in a standard format. Using the engine’s API, user interfaces (e.g. web based or GUI) can be built on top of it. When these interfaces are used in real world environments, direct links between research and practice will be created. The latest problem data can then be exported to researchers. Once the scientific challenges behind the current problems have been analysed and solvers developed in order to handle them, the end users’ software could be updated with the new algorithms at the click of a button (e.g. via the internet). This would be a mutually beneficial relationship and a method of “bridging the gap” between academia and practice.

The engine could also be used in an alternative scenario. When a researcher solves a nurse rostering problem with a certain method and he/she would like some method of validating the method, it should be possible to convert the problem to the benchmark format and solve it using a publicly available (perhaps published) method. If their method is better this would have provided validation and information on how to improve the other solver. If their method is worse they

may be able to find ways of improving it by looking at the better approach. They will also have contributed a new benchmark instance for other researchers.

Finally, research into solving unnatural, computer generated problems is often justified by tenuous links to practice or to obtain insight into a particular problem or approach. This is justifiable to a degree. For example, many notable and hugely beneficial outputs of scientific research would not have been produced if the conductors were purely interested in practical benefits. However, this should not negate the scientific challenges and obvious contributions of tackling the inherent complexity of real world problems. Obtaining real world data though is not always an easy task and collating it into a usable format is not trivial either. This project has lessened these burdens and provided a platform for future research.

The remaining chapters in this thesis present algorithms developed for the benchmark instances currently available. The source code for the algorithms can be found in the 'Solvers' module of the rostering engine. The engine, all the data sets, best solutions, and more documentation are all available online at the research website (http://www.cs.nott.ac.uk/~tec/NRP/).

5 A Time Predefined Variable Depth Search

As mentioned in a recent survey paper by Ahuja et al. [9] many successful very-large scale neighbourhood search techniques have appeared in various forms in the field of Operations Research. They commented, for example, that the well known Lin-Kernighan algorithm for the travelling salesman problem can be viewed as a very large-scale neighbourhood search technique. Ahuja et al. categorised very large-scale neighbourhood methods into three similar classes, one of which are variable depth methods. Variable depth searches (including some ejection chain methods [114]) have been effectively applied to a number of optimisation problems, for example the vehicle routing problem [214] and the generalised assignment problem [249]. Many more examples of successful very-large scale neighbourhood searches can be found in the survey paper of Ahuja et al. Perhaps the problem closest to nurse rostering that these techniques have been applied to is exam timetabling [5, 6, 180]. There are, however, very few large-scale neighbourhood searches applied to nurse rostering problems.

One paper that does introduce the application of such techniques to nurse rostering is that of Dowsland [91]. In her approach to providing an automated nurse rostering system, a tabu search is used that oscillates between decreasing cover violation and increasing roster quality. In each of these phases, two types of ejection chains are used. The first consists of a sequence of on/off day swaps between nurses and the other is made up of sequences of swapping week long work patterns between nurses. The chains are able to escape from poor local optima that single on/off day or pattern swaps would not be able to escape from.

Another example of using such techniques in solving a nurse rostering problem is the method of Louw et al. [166] who also use an ejection chain approach. The compound move used is similar to Dowsland's chain of on/off day swaps. They noted that "*the compound move was able to achieve far superior reductions in the objective function value when compared to any of the elementary move types*".

Very large-scale neighbourhood searches face the problem of exploring an exponentially large neighbourhood. Therefore, the key to developing effective ones is identifying heuristics and other mechanisms which can efficiently narrow or direct the search. This chapter presents a variable depth search for nurse rostering and describes the heuristics and other features that make it successful.

The next section investigates various search neighbourhoods that have been used to solve nurse rostering problems. Section 5.2 presents the variable depth search and Section 5.3 contains the results from a number of experiments using this algorithm and comparisons to other approaches.

5.1 Search Neighbourhoods for Nurse Rostering

This section describes two types of search neighbourhood that have been used to solve nurse rostering problems. Their applicability to the benchmark instances are then investigated. The results from this preliminary investigation are relevant to the variable depth search presented in Section 5.2.

5.1.1 The Single Shift Neighbourhood

Included in this category are all neighbourhoods that are identified by moves or swaps that change the assignment of up to two shifts, days on/off or variables at a time. Depending on the type of cover constraints (are there minimum and/or

maximum shift cover requirements and are they hard or soft constraints?), these moves may involve either one nurse (e.g. [26, 39, 88]) or two nurses (e.g. [56, 58, 78, 141, 159, 176, 205]). The neighbourhoods described in chapter 3 also fall under this category.

In the benchmark problems, the shift cover requirements are hard constraints and neither over nor under coverage are permitted. Therefore, once an initial feasible roster is constructed, only swaps or moves between two nurses are allowed. This ensures that the coverage constraint is not violated. Examples of these swaps are illustrated in Figure 5.1 and Figure 5.2. Figure 5.1 shows a section of a roster where L, N, D and DH are shifts and G, H and A are nurses. Move *a* involves the swapping of shift L and shift N on the 5th of December between nurses G and H to give the resulting roster on the right. In Figure 5.2, move *b* involves the assigning of shift L to nurse G from nurse H on 5th December. This could be alternatively phrased as “swapping shift L and an empty shift between nurses G and H on the 5th of December.”

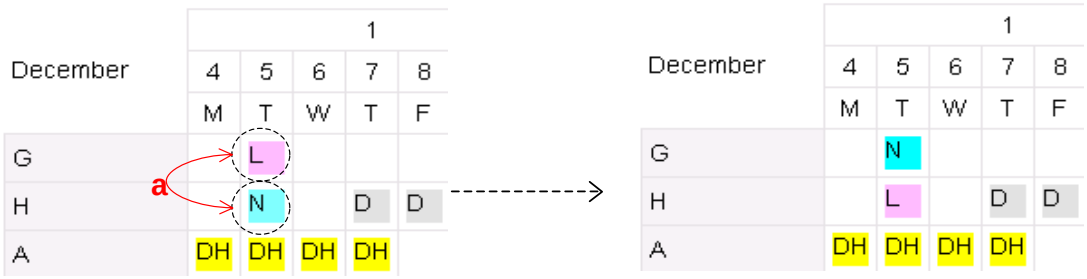

Figure 5.1 Example move *a* in the single shift neighbourhood

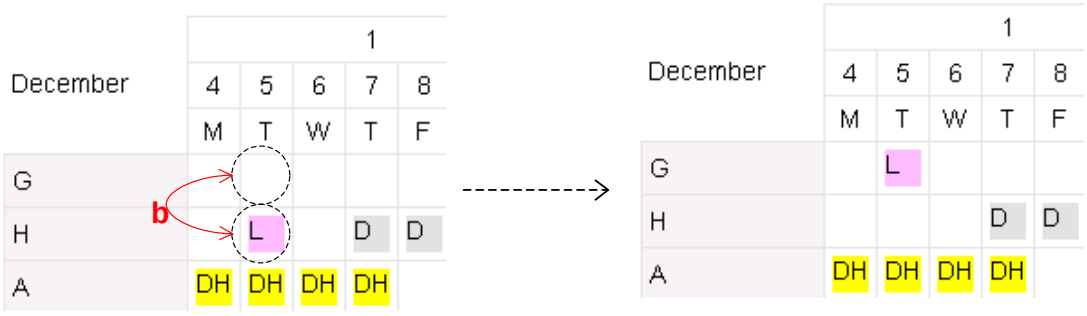

Figure 5.2 Example move *b* in the single shift neighbourhood

The single shift neighbourhood is the most commonly used neighbourhood in solving nurse rostering problems. It is a relatively small neighbourhood and easy to implement in a search algorithm. Even for the largest instances examined, using today's average desktop computer, this neighbourhood can be exhaustively searched and a local optimum can be reached quickly using a hill climber. Rosters produced using this approach, however, are not always of satisfactory quality and can usually be improved by an experienced human scheduler. Therefore, this neighbourhood is often, either incorporated into a more sophisticated method such as a metaheuristic (as in chapter 3) and/or replaced with a larger neighbourhood.

5.1.2 The Block Neighbourhood

The single shift neighbourhood, on its own, is often not effective enough. Meyer auf'm Hofe [179] highlights its weakness with a specific example in which this neighbourhood, even if combined with a tabu list, would be unlikely to remove a particular violation as it requires the simultaneous change of eight (and only these eight) specific variables.

This section describes a larger neighbourhood defined by moves which would have been able to repair that particular violation. The neighbourhoods have

recently been incorporated into search algorithms for nurse rostering and that approach has been shown to be very effective.

Included in this category of neighbourhoods are those defined by the swapping of all assignments on two or more adjacent days between two nurses. Examples of these swaps can be seen in Figure 5.3 and Figure 5.4. Figure 5.3 illustrates a move involving the swapping of a block of two adjacent days. On the 5th and 6th December, the N shifts of nurse A are assigned to nurse G and the L shifts of nurse G are assigned to nurse A. Here the block size is two as it involves two adjacent days. Figure 5.4 shows a move involving the swapping of a block of adjacent days of length three. Note that on 6th December, nurse A had no shift but this is still labelled as a swap with a block of days of length three.

As the block neighbourhood is a larger neighbourhood, its practical use was previously restricted by computational limitations. However, the recent dramatic increases in computing power have made a more aggressive use of this neighbourhood much more viable, as will be shown.

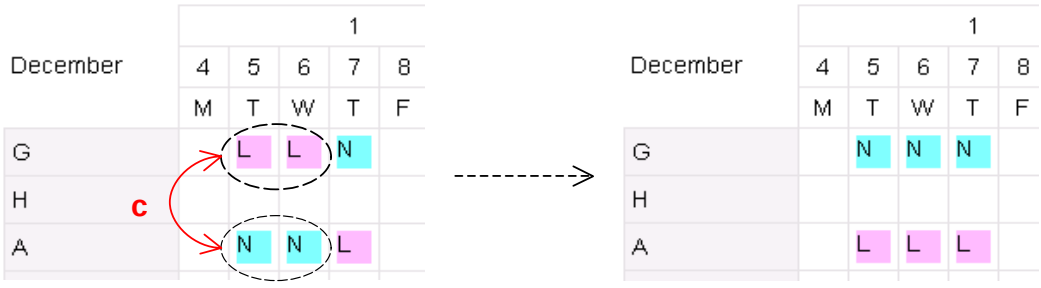

Figure 5.3 Example move *c* in the block neighbourhood

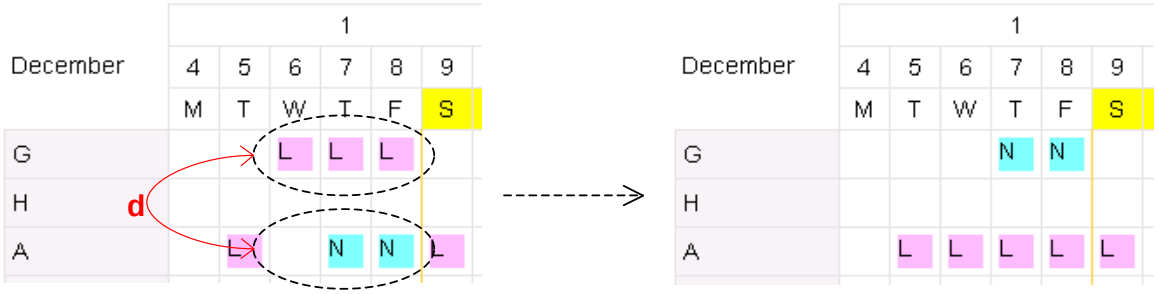

Figure 5.4 Example move *d* in the block neighbourhood

An early example of the use of changing blocks of shifts can be found in [148], where, although a block neighbourhood is not actually used, varying size blocks of shifts are assigned to nurses to create initial rosters. In [140], Jan et al. swap assignments on blocks of days between nurses as a mutation operator (called *escape*) in a genetic algorithm but again do not use it as a search neighbourhood. The first use of this type of swap in a search neighbourhood can be found in [58] and subsequently [49] and [56] in which they are called *shuffle* neighbourhoods. The authors commented that although these larger neighbourhoods were very time consuming and computationally intensive to use, the solutions they produced were significantly better and almost impossible to improve by hand. A very similar neighbourhood is also used by Valouxis and Housos [237] and later, Bellanti et al. [37] also describe a neighbourhood search which partially uses similar moves.

During development and testing, it was noticed that an expert human planner will often make improvements to a roster using similar moves. This may explain why an expert has such difficulty in improving a roster produced using this search neighbourhood.

5.1.3 Comparing the Single Shift and Block Neighbourhood

To examine how effective these neighbourhoods are and how quickly they can be searched using today's desktop personal computer, some experiments were conducted using a simple first improvement hill climber on the benchmark instances introduced in chapter 4. (Note that it is actually a hill descender as the penalty is being minimised but it is referred to as a hill climber as this is, generally, a more familiar term). The search uses neighbourhoods defined by swaps of up to a maximum length block of days. The pseudocode is given in Figure 5.5. This pseudocode is simply to provide an outline of the process. The actual implementation contains many more lines of code that increase efficiency and avoid redundancy (e.g. not visiting a solution already examined, etc).

```
1.  WHILE there are untried swaps
2.    FOR BlockLength = 1 to MAX_BLOCK_LENGTH
3.      FOR each employee (E1) in the roster
4.        FOR each day (D1) in the planning period
5.          FOR each employee (E2) in the roster

6.            Swap all assignments between E1 and E2 on D1 up
              to D1+BlockLength

7.            IF an improvement in roster penalty THEN
8.              Break from this loop and move on to the next day
9.            ELSE
10.             Reverse the swap
11.           ENDIF

12.         ENDFOR
13.       ENDFOR
14.     ENDFOR
15.   ENDFOR
16. ENDWHILE
```

Figure 5.5 Pseudocode for the hill climber

The initial roster is created using a randomized greedy assignment method. It operates as follows: for each shift which needs to be covered, assign it to the nurse who incurs the least gain in penalty for their individual schedule (or who receives the greatest decrease in penalty) on receiving this shift.

In order to provide different starting solutions and allow the search to also be used with random restarts, the set of shifts to be assigned is randomly shuffled. The quality of the initial rosters created using this greedy algorithm is usually very poor.

As there are few hard constraints, it is not difficult to construct a feasible roster. Also, the hard constraints are all related to coverage and it is possible to pre-calculate whether a feasible solution can be built. If a feasible solution does not exist, the user is notified that the cover requirements need to be reduced or extra staff need to be added. This is important as the hill climber operates over the feasible solution space.

Table 5.1 presents the results of the hill climber (outlined in Figure 5.5) when the maximum block length parameter (MBL) is set from one to ten. Note that when MBL=1, it is effectively the single shift neighbourhood.

Each experiment is repeated five times using different initial rosters (the same initial rosters are used for each MBL setting). The best, average and worst solutions, the average number of solutions examined per repeat and the average computation time per repeat are recorded for each instance. Table 5.1 contains the averages of these over all instances (for quick reference) and the results for each instance can be found in Table 5.2. The experiments were performed using a desktop PC with an Intel P4 2.4GHz processor.

Repeats	MBL	Best	Average	Worst	Average No. solutions examined per repeat	Average computation time per repeat (secs)
5	1	2064	2820	3576	49200	3.8
5	2	1565	2245	2614	86095	7.5
5	3	1452	1840	2224	100472	9.2
5	4	1413	1634	1893	115241	11.0
5	5	1253	1573	2005	127452	12.4
5	6	1191	1570	1891	138627	13.7
5	7	1341	1526	1829	159760	16.3
5	8	1160	1452	1652	171433	17.7
5	9	1165	1462	1712	183773	19.2
5	10	1165	1458	1721	185903	19.7
25	1	1589	2613	3888	50513	4.0

Table 5.1 Results of varying MAX_BLOCK_LENGTH (MBL)

Instance	Max block length	Best	Ave.	Worst	Average No. solutions examined	Average computation time (seconds)	Instance	Max block length	Best	Ave.	Worst	Average No. solutions examined	Average computation time (seconds)
ORTEC01	1	8927	14641	20192	55752	2.6	BCV-5.4.1	1	487	725	933	1560	0.1
ORTEC01	2	5917	11046	13185	83462	4.1	BCV-5.4.1	2	193	369	633	2389	0.1
ORTEC01	3	5120	7735	9771	109864	5.9	BCV-5.4.1	3	48	196	487	2433	0.1
ORTEC01	4	5020	5706	6865	126689	7.1	BCV-5.4.1	4	48	108	205	2670	0.1
ORTEC01	5	3355	5470	8045	144511	8.1	BCV-5.4.1	5	48	106	195	2680	0.1
ORTEC01	6	2745	5303	6915	167063	9.7	BCV-5.4.1	6	48	106	195	2831	0.1
ORTEC01	7	4257	4848	6470	173036	10.2	BCV-5.4.1	7	48	106	195	3032	0.1
ORTEC01	8	2475	4022	4700	185363	11.1	BCV-5.4.1	8	48	106	195	3268	0.1
ORTEC01	9	2525	4174	5295	202584	12.4	BCV-5.4.1	9	48	106	195	3452	0.2
ORTEC01	10	2525	4131	5345	188767	11.6	BCV-5.4.1	10	48	106	195	3587	0.2
BCV-1.8.1	1	328	491	698	9647	0.4	BCV-6.13.1	1	1024	1324	1597	19883	1.0
BCV-1.8.1	2	291	448	650	14206	0.6	BCV-6.13.1	2	1019	1223	1287	29736	1.6
BCV-1.8.1	3	288	332	470	20461	0.9	BCV-6.13.1	3	994	1131	1286	34222	1.9
BCV-1.8.1	4	273	330	464	22138	1.1	BCV-6.13.1	4	954	1057	1257	41324	2.4
BCV-1.8.1	5	287	338	470	22226	1.1	BCV-6.13.1	5	954	1057	1257	46422	2.8
BCV-1.8.1	6	287	336	470	25884	1.3	BCV-6.13.1	6	954	1057	1257	50864	3.1
BCV-1.8.1	7	287	337	471	30805	1.6	BCV-6.13.1	7	954	1011	1101	60173	3.8
BCV-1.8.1	8	287	337	471	33242	1.8	BCV-6.13.1	8	954	1011	1101	64769	4.2
BCV-1.8.1	9	287	329	471	35425	2.0	BCV-6.13.1	9	954	1011	1101	69157	4.5
BCV-1.8.1	10	287	327	471	37639	2.1	BCV-6.13.1	10	954	1011	1101	73355	4.9
BCV-2.46.1	1	1704	1715	1726	140592	12.4	BCV-7.10.1	1	403	555	662	10259	0.4
BCV-2.46.1	2	1618	1674	1716	246792	24.1	BCV-7.10.1	2	381	514	606	13863	0.6
BCV-2.46.1	3	1618	1663	1701	302478	31.4	BCV-7.10.1	3	381	505	596	18048	0.9
BCV-2.46.1	4	1618	1663	1701	335523	35.8	BCV-7.10.1	4	381	505	596	19851	1.1
BCV-2.46.1	5	1618	1663	1701	361902	39.5	BCV-7.10.1	5	381	505	596	21480	1.1
BCV-2.46.1	6	1618	1663	1701	387107	42.8	BCV-7.10.1	6	381	505	596	23038	1.2
BCV-2.46.1	7	1618	1663	1701	411298	46.4	BCV-7.10.1	7	381	505	596	24540	1.3
BCV-2.46.1	8	1618	1663	1701	434607	49.8	BCV-7.10.1	8	381	505	596	25982	1.4
BCV-2.46.1	9	1618	1663	1701	456895	52.9	BCV-7.10.1	9	381	505	596	27368	1.6
BCV-2.46.1	10	1618	1663	1701	478249	55.9	BCV-7.10.1	10	381	505	596	28691	1.7

BCV-3.46.1	1	3883	3969	4094	200929	16.8	BCV-8.13.1	1	236	268	334	17229	0.8
BCV-3.46.1	2	3607	3675	3801	392147	37.2	BCV-8.13.1	2	148	236	333	22756	1.1
BCV-3.46.1	3	3464	3525	3605	420806	41.7	BCV-8.13.1	3	148	198	236	28734	1.5
BCV-3.46.1	4	3474	3507	3554	483621	49.7	BCV-8.13.1	4	148	198	235	34321	1.9
BCV-3.46.1	5	3443	3463	3493	535836	56.8	BCV-8.13.1	5	148	198	235	37471	2.1
BCV-3.46.1	6	3432	3463	3486	581235	62.8	BCV-8.13.1	6	148	198	235	40455	2.3
BCV-3.46.1	7	3430	3456	3470	737642	82.7	BCV-8.13.1	7	148	198	235	43316	2.5
BCV-3.46.1	8	3408	3446	3470	800863	90.5	BCV-8.13.1	8	148	198	235	48029	2.9
BCV-3.46.1	9	3409	3446	3470	857518	98.4	BCV-8.13.1	9	148	198	235	50671	3.1
BCV-3.46.1	10	3409	3454	3507	852724	99.3	BCV-8.13.1	10	148	198	235	53218	3.3
BCV-4.13.1	1	75	110	189	12284	0.6	BCV-A.12.1	1	3570	4397	5335	23867	2.9
BCV-4.13.1	2	17	53	75	20875	1.0	BCV-A.12.1	2	2463	3210	3858	34728	4.6
BCV-4.13.1	3	22	54	75	21729	1.1	BCV-A.12.1	3	2433	3060	4015	45941	6.4
BCV-4.13.1	4	22	54	75	24019	1.3	BCV-A.12.1	4	2190	3207	3980	62257	9.2
BCV-4.13.1	5	22	52	74	31412	1.8	BCV-A.12.1	5	2275	2878	3980	70575	10.4
BCV-4.13.1	6	22	52	74	33390	2.0	BCV-A.12.1	6	2275	3019	3980	74401	11.2
BCV-4.13.1	7	22	52	74	35330	2.1	BCV-A.12.1	7	2265	3082	3980	78429	11.9
BCV-4.13.1	8	15	50	74	38808	2.4	BCV-A.12.1	8	2265	3178	3980	79397	12.3
BCV-4.13.1	9	13	49	74	45282	2.8	BCV-A.12.1	9	2265	3134	3980	89375	14.0
BCV-4.13.1	10	13	49	74	47293	3.0	BCV-A.12.1	10	2265	3134	3980	95509	15.0

Table 5.2 Results of varying MBL in the hill climber (5 repeats for each instance)

Instance	Max block length	Best	Ave.	Worst	Average No. solutions examined	Average computation time (seconds)
ORTEC01	1	5431	12505	21199	56621	2.6
BCV-1.8.1	1	312	432	698	9168	0.4
BCV-2.46.1	1	1634	1694	1799	158093	14.2
BCV-3.46.1	1	3694	3899	4094	199456	16.9
BCV-4.13.1	1	18	135	411	12140	0.6
BCV-5.4.1	1	194	714	1099	1676	0.1
BCV-6.13.1	1	1024	1360	1749	19574	1.0
BCV-7.10.1	1	403	535	742	9379	0.4
BCV-8.13.1	1	149	243	379	16799	0.8
BCV-A.12.1	1	3030	4609	6708	22223	2.8
Average		1589	2613	3888	50513	4.0

Table 5.3 Results for hill climber, 25 repeats with MBL=1

The results in Table 5.2 and Table 5.1 show that the single shift neighbourhood (i.e. MBL=1) is not as effective as when MBL>1. It is logical to question whether this is simply due to less solutions being examined when MBL=1. To provide a fairer comparison, the experiments were repeated for MBL=1 but with 25 instead of 5 repeats (Table 5.3 and the bottom row of Table 5.1). This ensures that the searches with MBL=1 receive at least the same time (and in most cases more) as

each experiment with MBL>1. Although giving a longer computation time did improve the best solution found, it was still worse than MBL=2 on all but one instance. Testing the null hypothesis (H_0) that the difference in objective values of the solutions produced by the two algorithms are symmetrically distributed around the central point of zero, for the Wilcoxon signed rank test the probability was greater than 0.05 and for the Sign test less than 0.05. Compared against MBL=5, MBL=1 was also worse on all instances but one. For MBL=5 versus MBL=1, H_0 had a probability of less than 0.05 using the Wilcoxon signed rank test.

As shown, increasing MBL increases the quality of the results but at the cost of extra computation time. However, when MBL>8, any increases in performance become less clear. Although MBL could range up to the number of days in the planning period, the results suggest that setting MBL>8 does not yield better results, especially in relation to the extra computation time required. In fact, when MBL>8 the results deteriorate slightly for some instances. This would have been a strange result if line one of the pseudocode was not present. However, what is happening is that a move is being made when the block length=9 that would obviously not have been made if MBL<9 and hence in the next iteration of the loop at line 1 the current solution is slightly different.

It can also be seen that the increase in computation time is approximately linear in relation to MBL.

On average, increasing MBL will yield better solutions. However, if the results for each instance are studied, the benefits of using larger blocks on some instances is less noticeable. For example, on instances BCV-2.46.1 and BCV-7.10.1, MBL=2 is only slightly better than MBL=1 and increasing MBL above three gives no further

improvement. Similarly, increasing MBL above five for instance BCV-5.4.1 is not worthwhile. Although the reasons for this are not obvious, it could possibly be linked to which types of constraints and their priorities are used in the schedules. Although it may be possible to estimate the suitability of neighbourhoods for a particular instance based on its constraints, it would also be difficult, as potentially each nurse could request a different set of constraint types with different parameters for any one scheduling period.

The computation times for each instance range from less than one second to approximately 90 seconds. As would be expected, the longer computation times are required for the rosters with more employees, longer planning horizons and also those instances which utilise a larger set of the available soft constraint types for each employee (e.g instance BCV-A.12.1).

As can be seen, on the machine used, the search, on average, examines approximately 10,000 solutions per second. Examining the results for each instance reveals that the solutions examined per second ranges from approximately 28,000 for the instances with fewer soft constraints to around 6,000 for the instances which use all the available soft constraint types. Evaluating soft constraints is by far the most time consuming function in the search and so a large amount of effort was spent streamlining them to ensure that they were fast and efficient as well as accurate. It is possible to obtain large increases in search performance through writing faster code than any new heuristic or search mechanism may be able to achieve. This is not always appreciated and the challenge and importance of writing fast and efficient

evaluation functions is often underestimated. Also, metrics such as the performance per number of solutions evaluated are often analysed less, if at all. In the results, the number of rosters examined as well as computation times are provided. In the author's opinion this is a more revealing, reliable and future-proof measure of a search method's performance.

5.2 The Variable Depth Search

As the results in the previous section show, using today's average desktop PC, a local search employing the block neighbourhood can be completed on the larger instances in less than 90 seconds. These solutions are very difficult to improve by hand. However, due to the complexity of the problems, they are still very often local optima (albeit high quality ones). Therefore, end users may wish to use idle computer time (e.g. during a lunch break or over night) to try and find even higher quality rosters. Perhaps the simplest way to provide this option is by restarting the hill climber as many times as possible in the allotted time with different initial rosters, in the form of a basic iterated local search [165]. This is something that was tested and the results are provided in section 5.3. However, the main focus of this chapter is a variable depth search which will now be introduced.

The first step of the algorithm is to create an initial roster. This is done using the greedy assignment method introduced in section 5.1.3. As mentioned, these initial rosters can be constructed very quickly (in less than a second) but are generally of poor quality. It was found, however, that the quality of the initial roster had relatively little impact on the final roster. Once the initial roster is created, it is possible to proceed with the variable depth search which, like the hill climber,

also operates over the feasible solution space. Figure 5.6 provides an outline of the algorithm.

The search is similar to a method used when attempting to manually improve rosters. When improving rosters by hand it was observed that first we would try to improve one nurse's individual schedule (that is lower the penalty for that nurse's schedule). Improving this nurse's schedule would usually be at the expense of another nurse so we then try to improve their schedule. If the second nurse's schedule is improved it may be at the expense of a third nurse's schedule so we then move on to the third nurse and so on until (hopefully) we have an overall roster penalty that is lower than the original penalty. If not, we would reverse all the changes we have just made and try a different path. This is the basic idea behind the algorithm.

$penalty_r$ = the penalty for roster *r*.

$penalty_{r,n}$ = the penalty for the schedule of nurse *n* in roster *r*.

```
0.   set best roster    := the current roster

1.   set current roster := an unvisited neighbour in neighbourhood
                           for best roster

2.   if no unvisited neighbour available
         stop and return best roster

3.   if penalty_current roster < penalty_best roster
         goto 0.

4.   if neither of the penalties decrease for the individual schedules of
     the two employees involved in the swap OR maximum depth <= 1
         goto 1.

5.   set E1 := the employee with increased penalty
     set current depth := 1

6.   In the neighbourhood for the current roster where considering swaps
     of blocks between employee E1 and all other employees (E2)

     set current roster := neighbouring roster with lowest penalty where
     penalty_neighbour < penalty_best roster or
     penalty_neighbour - penalty_neighbour,E2 + penalty_current roster,E2
     < penalty_best roster

7.   if no such neighbour
         goto 1.
8.   else if current roster's penalty < best roster's penalty
         goto 0.
9.   else if current depth < a preset maximum depth
         set E1 := E2
         set current depth := current depth + 1;
         goto 6.
10.  else
         goto 1.
```

Figure 5.6 Variable depth search outline

The neighbourhoods referred to in Figure 5.6 are identified by swaps of blocks up to a maximum block length (MBL). The neighbourhood at step 1 is defined by all possible swaps of blocks, on all days of the planning period, between all nurses. At step 6, the swaps are just between two nurses on all days of the planning period. It was found to be generally more efficient to set MBL at step 1 lower than at step 6 (e.g at step 1, use 2 or 3 and at step 6, use 5 or 6).

At various points in the algorithm (e.g. steps 4 and 6), it is necessary to analyse the change in penalty for a nurse's individual schedule after a swap has been performed. After any swap, at least one but no more than two of the nurse's individual schedules will have been altered. However, the penalties for other nurses' schedules may also have changed even though their schedule has not been modified. This occurs in instances which use the so called 'vertical' constraints of tutorship and ensuring that certain nurses work separately. Therefore, when analysing the change in penalty for any individual nurse's schedule that has just been altered, what we actually use is the net change in penalties for this nurse and all other nurses that are directly linked to this nurse by 'vertical' constraints.

5.2.1 Heuristics

It was stated that the search operates over the feasible solution space. However, it was discovered that it was beneficial to treat the hard constraint that a nurse must have the skills required to perform a shift as a soft constraint. This can be achieved by assigning a sufficiently high weight to the hard constraint violation (e.g. giving it the same value as the penalty of the initial roster) thus ensuring that a solution with this constraint violated will not be returned. Using it as a soft constraint though, allows greater exploration of the search space. This happens because rosters with this hard constraint violated are sometimes used as intermediate solutions in a chain of moves and a better local optimum may be reached via them.

Step 6 is perhaps the most important step in the algorithm. Step 6 specifies which moves to examine as potential candidates to be added to the current chain of moves and also defines the rule for deciding which one (if any) to select. As

outlined in Figure 5.6, a swap is only selected as a potential move to add to the current chain if *ignoring the change in E2's penalty, the neighbour's penalty is less than the best roster's penalty*. This rule is similar to, and inspired by, the 'Gain Criterion' of the Lin-Kernighan algorithm for the travelling salesman problem [163]. In section 5.3, the results of a number of experiments are presented in which this rule is removed to investigate its benefit.

The number of moves to examine in order to select candidates for continuing the chain can have a significant effect on the performance of the algorithm. If too many moves are tested, then the algorithm's run time will increase. If too few are selected, then there is a smaller chance of a successful one being found and the algorithm will become less effective. In Figure 5.6, all swaps up to a maximum block length, on all days of the planning period, between one nurse and all the others are tested. Reducing the run time by limiting the number of nurses to test swaps between and reducing the number of days adjacent to the swap at step 1 over which to test swaps was evaluated. As expected, the run time was improved but at the cost of roster quality. To try to increase efficiency, two heuristics for selecting candidate moves were developed and tested instead.

In the first heuristic (**violation flag heuristic**), all days which need changing either through the removal, addition or changing of shift assignments, in order to remove a soft constraint violation are flagged during penalty recalculations. Only the swaps which involve at least one of these days are then tested. This heuristic is also applied at step 1. Only focusing on parts of a solution that have violations and need repairing is a common heuristic. For example, Nonobe and Ibaraki [195]

use a similar heuristic in a tabu search approach tested on a nurse rostering problem formulated as a constraint satisfaction problem.

In the second heuristic (**worsened days heuristic**), an array of penalties due to soft constraint violations for each day is maintained for each nurse's schedule during penalty calculations. Using these arrays, the moves in step 6 are then restricted to only those blocks that contain days which were made worse (i.e. penalty increased) after the last move. This is a more restrictive heuristic as days which contain violations will be ignored if they were not affected by the last swap in the chain.

5.2.2 Predefined Run Time

The running time for the algorithm depends on the size of the neighbourhoods at steps 1 and 6, the maximum depth used at step 9 and the structure of the instance being addressed. The size of the neighbourhoods at steps 1 and 6 depends upon the number of nurses, the number of days in the planning period and the maximum block length. The effects of the third factor (the instance structure) on the running time cannot be as easily predicted as factors such as the number of nurses and days. For some instances, it is possible that the structure (determined more by the soft constraints and their weights) is such that there is very often a valid neighbour found at step 6 with which to replace the current roster but which is not better than the best roster. This can mean that the search sometimes reaches great depths which obviously affects the running time.

To reduce this effect, a maximum depth which is set beforehand is used at step 9. Initially the depth was set using a trial and error method of running the algorithm for a short time and observing its progress on the particular instance. Then

altering the maximum depth value until a suitable setting is found (that is estimated) will restrict the algorithm to a satisfactory running time. This is obviously not a suitable approach for practical use. Therefore, an additional mechanism was added which takes the preferred running time as a parameter and attempts to use that time efficiently.

This mechanism works as follows: for the algorithm to finish, every neighbour in the neighbourhood at step 1 needs to be examined and potentially used as the first solution in a chain of moves. It is possible to calculate the size of the neighbourhood at step one using the number of nurses, the maximum block length and the number of days in the planning horizon. Given a preferred running time and the number of solutions to evaluate at step 1 (updated each time a new best solution is found), it is possible to calculate an average time to spend using each neighbour at step 1 as the first solution in a chain. Then at step 9, instead of testing whether a maximum depth will be exceeded in continuing the chain, we test whether the average time per chain will be exceeded if it continues.

In the results section where this heuristic is not used but a maximum running time is set, the search immediately terminates and returns the best solution when the time limit is reached. If the algorithm naturally terminates and the preferred running time has not been exceeded, then at step 2, instead of returning the best roster, a new initial roster is created and the algorithm restarts at step 0.

Figure 5.7 shows an example of an improving chain of moves. The change in the roster consists of seven moves which, when performed simultaneously, provide an overall reduction in the roster's penalty. It can be seen that the second nurse of

a swap is always the first nurse in the next swap. Note that Figure 5.7 is just used to illustrate the idea of a chain of moves. For the roster shown (which is far from optimal), there are many other chains which would also improve the roster.

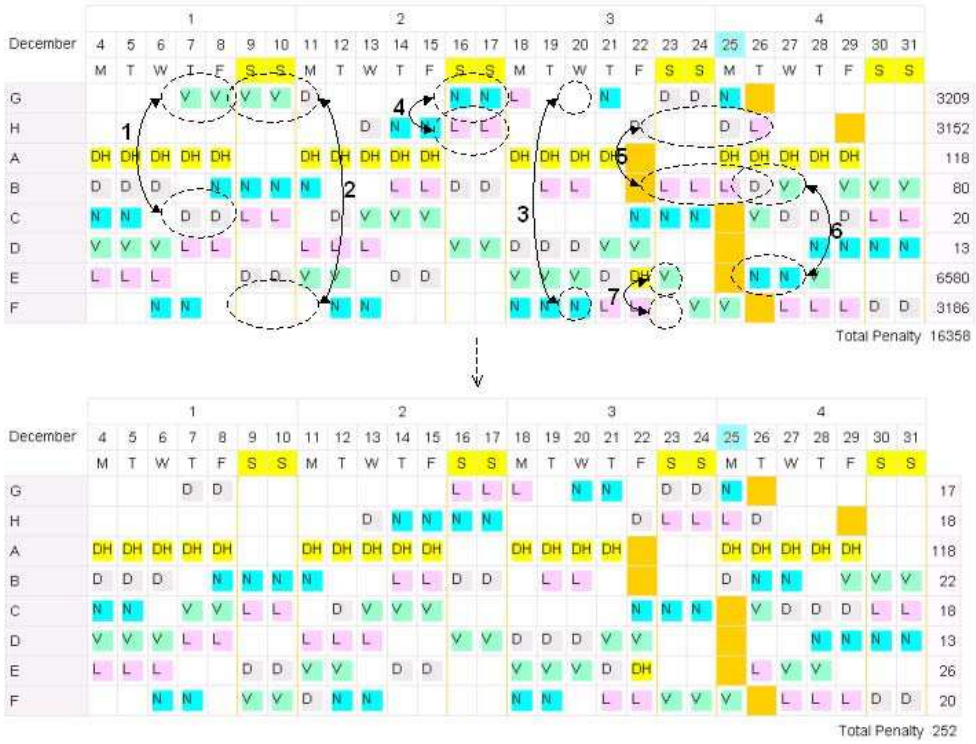

Figure 5.7 Example chains of swaps

From observing the logs after searches, the lengths of improving chains of moves varies greatly. Note that poor rosters are easily improved by single moves. When the local optima start to be found less frequently, (and the penalty is approaching better values) even for the smaller instances, improving chains of moves with lengths over one hundred are not uncommon.

5.2.3 Efficient Implementation

At step 6, there is a possibility that a neighbouring solution will be selected that has been visited previously and cycling could occur. Two different methods were tested to remove this risk. In the first method, a history of solutions visited along

the current path (i.e. since the last use of step 1) is maintained and used to ensure that they are not revisited. This is fast and prevents cycling but does not guarantee that solutions are not revisited at other points in the algorithm, for example visiting a solution at step 1 that has already been used at step 6. The second method maintains a hash table of all solutions visited during the run of the algorithm and so also prevents all solutions being revisited. Testing showed that the first, simpler method, produced better results. This appears to be because the probability of visiting a duplicate solution at the points which the first method does not prevent is small and much lower than the probability of cycling at step 6. Therefore, using the faster method which prevents the majority of cycling and revisiting duplicate solutions was more efficient than the slower approach which guaranteed no cycling or duplicate paths.

As discussed earlier, increases in performance can be achieved as effectively through making the algorithm faster and more efficient as by using better heuristics. We have already mentioned the importance of avoiding cycling. There are also some other efficiency measures which are worth highlighting. Firstly, when a nurse's schedule has been altered it is only necessary to re-evaluate their schedule and any other nurses' schedules which may be linked by vertical constraints to recalculate the new roster's penalty. Secondly, by far the most time consuming operation is calculating penalties (i.e. soft constraint evaluations). If there is a likelihood that a solution will be returned to, then the algorithm caches penalties to avoid having to recalculate them. Finally, some soft constraint calculations can be speeded up by using data structures that are modified as assignments are made. A simple example is to update the total number of hours

worked when a shift is (un)assigned rather than to add all the hours up when calculating the penalty. Some of the more complicated constraints benefit from a similar approach.

5.3 Results

The algorithm was tested on the ten publicly available data sets introduced in chapter 4.

5.3.1 Comparing Heuristics

In the experiments, the following parameters were used: Maximum depth = 1000, maximum block length at step 1 = 2, maximum block length at step 6 = 5. As described, if the algorithm finishes before the maximum running time is reached, a new initial solution is constructed and the search restarts.

Table 5.4, Table 5.5, Table 5.6 and Table 5.7 contain the results of the variable depth search when different heuristics are used. Predefined maximum run times of 1 minute, 2 minutes and 5 minutes were tested with each instance and repeated five times using different random seeds (but which were the same for each corresponding trial). The best, worst and average of these runs were recorded. The penalties in Table 5.4 are the averages over all instances. The results for each instance given in Table 5.5, Table 5.6 and Table 5.7.

The heuristics are summarised below:

No heuristics Step 4 is removed and step 5 is applied twice but with E1 set as the other nurse each time. At step 6 the only rule is to select the best neighbour.

PG Partial gain heuristic (only add a swap to the end of the current chain if it satisfies the partial improvement criterion).

VF Violation flag heuristic (only test swaps which include at least one day which is flagged as having a violation). This is used at steps 1 and 6.

WD Only swap worsened days heuristic (only test swaps which include at least one day which has a violation and the violation's penalty was increased or the violation occurred after the last swap). This is used at step 6.

TR Time restriction heuristic (ensure that a pre-calculated average amount of time is spent on every chain).

ILS Hill climber with restarts. Maximum depth is set to 1 and maximum block size at step 1 is increased to 5.

	1 Minute			2 Minutes			5 Minutes		
Heuristics	Best	Avg.	Worst	Best	Avg.	Worst	Best	Avg.	Worst
No heuristics	1390	1561	1879	1196	1454	1825	1154	1394	1766
PG	917	1156	1461	888	1078	1271	872	1034	1225
VF	1222	1629	2050	1113	1329	1611	973	1111	1232
WD	1046	1241	1515	1008	1131	1251	972	1087	1234
PG + VF	1003	1228	1680	894	1163	1619	872	1016	1303
PG + WD	900	1016	1205	895	973	1083	877	946	1031
TR	1153	1454	1829	1175	1450	1654	1008	1232	1386
TR + PG + VF	1014	1122	1277	888	994	1141	874	978	1141
TR + PG + WD	899	1068	1260	862	935	1027	858	889	938
ILS	1164	1266	1405	1048	1174	1290	1012	1067	1161

Table 5.4 Comparison of heuristic combinations

Time restriction = 1 minute. Max. depth = 1000. Max. block length at step 6 = 5. Max. block length at step 1 = 2.											
		No heuristics	PG	VF	WD	PG + VF	PG + WD	TR	TR + PG + VF	TR + PG + WD	ILS
ORTEC01	Average	5768	2358	6560	3234	3347	1103	4605	2242	1910	3573
	Best	4826	506	3245	1655	1421	425	2746	1430	555	2825
BCV-1.8.1	Average	263	263	267	265	263	259	278	257	261	266
	Best	255	253	259	257	254	254	271	254	255	261
BCV-2.46.1	Average	1679	1667	1643	1662	1659	1683	1657	1605	1598	1641
	Best	1653	1616	1595	1633	1616	1646	1618	1593	1574	1618
BCV-3.46.1	Average	3570	3478	3546	3515	3501	3481	3588	3477	3411	3464
	Best	3483	3411	3456	3505	3460	3422	3531	3442	3380	3443
BCV-4.13.1	Average	11	13	10	11	11	10	22	11	11	12
	Best	10	10	10	10	10	10	11	10	10	11
BCV-5.4.1	Average	48	48	48	48	48	48	48	48	48	48
	Best	48	48	48	48	48	48	48	48	48	48
BCV-6.13.1	Average	941	873	793	770	814	769	844	768	770	908
	Best	827	768	768	768	768	768	769	768	768	822
BCV-7.10.1	Average	419	455	385	384	383	384	392	384	384	381
	Best	381	381	381	381	381	381	381	381	381	381
BCV-8.13.1	Average	180	164	148	148	148	148	196	148	148	148
	Best	148	148	148	148	148	148	148	148	148	148
BCV-A.12.1	Average	2731	2240	2890	2370	2103	2271	2912	2284	2135	2215
	Best	2265	2033	2309	2050	1919	1900	2009	2068	1870	2085

Table 5.5 Variable depth search heuristics with maximum run time 1 minute

Time restriction = 2 minutes. Max depth = 1000. Max. block length at step 6 = 5. Max. block length at step 1 = 2.											
		No heuristics	PG	VF	WD	PG + VF	PG + WD	TR	TR + PG + VF	TR + PG + WD	ILS
ORTEC01	Average	5023	1931	4246	2529	2893	860	4990	1262	868	2713
	Best	3110	485	2450	1605	520	425	3099	470	480	1691
BCV-1.8.1	Average	263	261	265	262	260	255	269	257	257	265
	Best	255	252	257	257	254	253	263	253	253	261
BCV-2.46.1	Average	1679	1664	1642	1652	1649	1680	1627	1597	1600	1620
	Best	1653	1616	1595	1633	1596	1646	1616	1572	1574	1615
BCV-3.46.1	Average	3486	3433	3477	3433	3477	3457	3523	3410	3380	3446
	Best	3449	3392	3419	3414	3399	3372	3479	3380	3338	3413
BCV-4.13.1	Average	11	11	10	11	10	10	11	11	10	12
	Best	10	10	10	10	10	10	10	10	10	11
BCV-5.4.1	Average	48	48	48	48	48	48	48	48	48	48
	Best	48	48	48	48	48	48	48	48	48	48
BCV-6.13.1	Average	904	842	780	769	768	769	836	768	768	889
	Best	827	768	768	768	768	768	774	768	768	822
BCV-7.10.1	Average	419	437	384	384	382	384	403	384	384	381
	Best	381	381	381	381	381	381	381	381	381	381
BCV-8.13.1	Average	164	148	148	148	148	148	148	148	148	148
	Best	148	148	148	148	148	148	148	148	148	148

BCV-A.12.1	Average	2540	2007	2293	2071	1992	2115	2647	2051	1882	2213
	Best	2075	1775	2050	1819	1820	1900	1928	1845	1620	2085

Table 5.6 Variable depth search heuristics with maximum run time 2 minutes

Time restriction = 5 minutes. Max depth = 1000. Max. block length at step 6 = 5. Max. block length at step 1 = 2.

		No heuristics	PG	VF	WD	PG + VF	PG + WD	TR	TR + PG + VF	TR + PG + WD	ILS
ORTEC01	Average	4763	1785	2421	2234	1570	857	3150	1147	448	1901
	Best	3000	460	1390	1410	435	425	1420	360	420	1590
BCV-1.8.1	Average	259	258	262	255	255	254	264	257	254	263
	Best	254	252	256	253	253	253	255	253	253	257
BCV-2.46.1	Average	1664	1654	1621	1635	1647	1669	1622	1596	1586	1607
	Best	1633	1614	1595	1594	1595	1613	1595	1572	1572	1595
BCV-3.46.1	Average	3468	3404	3465	3392	3448	3423	3443	3392	3378	3427
	Best	3427	3374	3411	3382	3399	3372	3399	3347	3355	3413
BCV-4.13.1	Average	11	11	10	10	10	10	12	10	10	11
	Best	10	10	10	10	10	10	10	10	10	10
BCV-5.4.1	Average	48	48	48	48	48	48	48	48	48	48
	Best	48	48	48	48	48	48	48	48	48	48
BCV-6.13.1	Average	857	818	768	769	768	768	814	768	768	853
	Best	768	768	768	768	768	768	770	768	768	777
BCV-7.10.1	Average	401	401	382	383	382	383	384	383	383	381
	Best	381	381	381	381	381	381	381	381	381	381
BCV-8.13.1	Average	148	148	148	148	148	148	148	148	148	148
	Best	148	148	148	148	148	148	148	148	148	148
BCV-A.12.1	Average	2320	1814	1985	2000	1881	1900	2437	2028	1867	2034
	Best	1869	1664	1720	1729	1679	1750	2053	1849	1620	1900

Table 5.7 Variable depth search heuristics with maximum run time 5 minutes

The results indicate that a good combination of heuristics is TR+PG+WD. For example, comparing this combination against no heuristics over all tests it outperformed or was equal on all but four (out of 150). The null hypothesis that the difference in objective values of the solutions produced by the two algorithms are symmetrically distributed around the central point of zero, using the Wilcoxon signed rank test, had a probability less than 0.05. Similarly TR+PG+WD could be considered “better” than TR+PG+VF at the same confidence level using the Wilcoxon signed rank test and using the Sign test. The better results all use PG For example, using the Wilcoxon signed rank test, comparing PG against no heuristics, the null hypothesis could be rejected at the 0.05 confidence level,

indicating the efficacy of PG. TR is improved if combined with other heuristics and similarly VF and WD also perform better when not used on their own. As would be expected, VF and WD perform similarly (WD is a more restrictive form of VF). ILS performs similarly to TR, VF and WD when they are used on their own. All other combinations outperform ILS.

Examining the results for each instance is also interesting. VF is very effective on instance BCV-4.13.1, even when used on its own and WD+PG works well on instance ORTEC01. However, we were unable to draw any definite conclusions on why this should be the case after examining the characteristics of these instances more closely. Due to its simplicity ILS is able to examine more solutions in the allocated time but its lack of heuristics makes it less effective.

5.3.2 Comparisons with Other Methods

Brucker et al. [46] developed a heuristic constructive approach and tested it on the benchmark instances. As it is a constructive method it is not possible to provide a comparison to the variable depth search by using the number of solutions examined metric. However, their experiments were performed on the same machine and a comparison can be provided by using computation times. The results in Table 5.8 are Brucker et al's best results from all experiments. The total computation time in obtaining these solutions for each instance was then set as the maximum run time for the variable depth search with the heuristic combination TR+PG+VF.

Burke et al's result for ORTEC01 using the hybrid variable neighbourhood search [50] had a computation time of twelve hours (this is the method and results presented in chapter 3). The result for the variable depth search on this instance is the best of the five, five minute tests using the heuristic combination TR+PG+VF.

As can be seen, the variable depth search outperforms the constructive method, over the same computation times, on all instances except one, on which they are equal. Using the Sign test, these results were significant at the 0.05 level (testing the same null hypothesis as in section 5.3.1) but not significant using the Wilcoxon test. It also beats the hybrid method of Burke et al. on instance ORTEC01.

	Time	Brucker et al. [46]	Burke et al. [50]	Variable Depth Search
ORTEC01	12 hrs	-	541	360
BCV-1.8.1	136 sec	323	-	253
BCV-2.46.1	3424 sec	1594	-	1572
BCV-3.46.1	2888 sec	3601	-	3324
BCV-4.13.1	208 sec	18	-	10
BCV-5.4.1	16 sec	200	-	48
BCV-6.13.1	304 sec	890	-	768
BCV-7.10.1	216 sec	396	-	381
BCV-8.13.1	224 sec	148	-	148
BCV-A.12.1	944 sec	3335	-	1843

Table 5.8 Comparison of VDS with other algorithms

To provide further comparisons, the hybrid tabu search of Burke et al. [49, 58] was implemented and tested on the benchmark data sets. The best version of their tabu search (TS2) was applied five times to each instance. Table 5.9 contains the best and average results. The average execution time on each instance was also recorded. The variable depth search was then set a maximum run time identical to that used by the tabu search for each instance. Five repeats of the variable depth search were then performed to obtain average and best results. Heuristics TR+PG+WD were used.

	Variable depth search			TS2 [49, 58]			
Instance	Best	Average	Avg. Evals.	Best	Average	Avg. Evals.	Time (secs)
ORTEC01	480	1120	1,852,788	1581	3201	2,363,828	108
BCV-1.8.1	262	269	159,379	293	350	140,690	9
BCV-2.46.1	1574	1593	1,563,865	1573	1596	1,557,905	167
BCV-3.46.1	3334	3346	4,486,399	3410	3453	5,088,206	427
BCV-4.13.1	10	11	152,182	11	25	150,639	9
BCV-5.4.1	48	48	21,208	48	48	17,580	1
BCV-6.13.1	769	817	356,891	1010	1154	345,595	24
BCV-7.10.1	381	427	117,224	391	458	93,817	7
BCV-8.13.1	148	148	248,927	148	165	215,524	15
BCV-A.12.1	1835	1942	649,926	2065	2831	718,354	108

Table 5.9 Comparison of the variable depth search with a hybrid tabu search

The results in Table 5.8 and Table 5.9 show the variable depth search nearly always outperforms or is equal to previous methods in comparable tests over all

instances. The only time it was beaten was when TS2 found a best solution for BCV-2.46.1 with penalty 1573. The variable depth search could still manage a best with penalty 1574 though. Note that the variable depth search is actually dynamically adjusting to the run time of the other approaches. Using the Sign test, these results were statistically significant at the 0.05 level (testing the same null hypothesis as in section 5.3.1).

5.3.3 Longer Computation Times

Further experiments were conducted to examine the potential benefit of a longer execution time. The variable depth search with the same parameters as before and using heuristics TR+PG+WD was tested on each instance with a time limit of one hour. ILS was also tested for comparative purposes.

	TR+PG+WD		ILS	
	Pen	Evals	Pen	Evals
ORTEC01	435	58,703,568	630	69,151,556
BCV-1.8.1	254	62,769,622	255	72,216,494
BCV-2.46.1	1574	29,767,138	1594	39,255,535
BCV-3.46.1	3302	36,422,606	3414	42,504,506
BCV-4.13.1	10	58,613,414	10	68,301,984
BCV-5.4.1	48	73,021,195	48	92,071,328
BCV-6.13.1	768	54,683,305	814	62,509,316
BCV-7.10.1	381	59,704,319	381	69,957,265
BCV-8.13.1	148	59,084,678	148	66,960,140
BCV-A.12.1	1564	23,675,881	1808	25,506,048
Average	848		910	

Table 5.10 Experiments with VDS using longer computation times

The increase in computation time leads to an improvement for both methods. Again the simpler ILS examines more solutions in the allotted time but is still not as effective as the variable depth search. In fact, ILS after one hour on each instance is, on average, still worse than the best of five, one minute repeats of the variable depth search with heuristics PG + WD.

5.4 Conclusion

This chapter has briefly reviewed search neighbourhoods that have been previously used to solve nurse rostering problems. They were tested using new benchmark nurse rostering problems and based on the results, a variable depth search was created.

The block neighbourhood is very effective for the majority of the instances. Today's technology allows these larger neighbourhoods to be exhaustively searched very quickly. Even a simple hill climber which uses these neighbourhoods will produce satisfactory rosters and combining the hill climber with the greedy construction, restart method further improves the quality of solutions produced.

However, the variable depth search can still improve upon this basic iterated local search. The variable depth search works by chaining together the block moves using a number of heuristics to select the next move (link in the chain). The results show the best combination of heuristics to use is PG (the positive gain criterion) with TR (the time restriction heuristic) and WD (selecting moves on days that have violations which occurred after the last move).

It is also worth noting that, although the variable depth search is more effective, it is also more complicated to implement with an increased potential for introducing errors.

If the hill climber using the block neighbourhoods or the variable depth search were used in a population based approach such as a memetic algorithm [152] or a scatter search [118], then even better results may be possible. These are good methods for adding extra diversification to the search, especially over extended

execution times. This is investigated in the next chapter where the variable depth search is used as the improvement method in a scatter search.

6 A Scatter Search

This chapter describes a scatter search for the nurse rostering problem. Like genetic algorithms [104, 124, 132, 219], memetic algorithms [151, 152, 186] particle swarm optimisation [145, 177] and ant colony optimisation [89, 90], a key feature of scatter search is the maintenance of a population of solutions. This is in contrast to many other metaheuristics which generally work with one solution, for example simulated annealing [1, 147], tabu search [111, 117], GRASP [101], variable neighbourhood search [130, 185] etc. In genetic algorithms, these sets of solutions are often referred to as populations. To continue the metaphor, the individuals within a population may be labelled as parents and new solutions are usually called offspring. New solutions are generally created from two parents in the population through crossover and mutation operations. Although, for different problems, the details of the crossover and mutation functions can vary, there is typically some stochastic element to their operation. This contrasts with scatter search in which the method for forming new solutions is designed to minimize (if not eliminate) decisions being allocated to random (or more usually pseudo-random) chance. The idea is to try to replace calls to the random number function with "*systematic and strategically designed rules*" [118].

Another difference is found in the way that new solutions are added to the population or reference set. In many genetic algorithms, new solutions are allowed to enter the current population if their quality (usually determined by an objective function) is greater than the worst member of the current population. In scatter search, a method for comparing the similarity of two solutions is used to

measure the reference set's overall diversity. Whether or not a new solution enters the reference set may then be decided by not only its quality, but also its contribution to the reference set's diversity. The goal is to try to maintain the highest quality, yet diverse, reference set.

Some genetic algorithms also use a local search or other optimisation method on each of the new solutions between generations in order to improve their quality. These methods may also act to repair the solutions if they were incomplete or infeasible after the crossover stage. These genetic algorithms plus local search are often labelled as memetic algorithms but may also be referred to as hybrid genetic algorithms and genetic local search. The idea of using a heuristic improvement process on new solutions is also common to scatter search. These improvers/repairers can be a noticeable bottleneck in the algorithm though. Also, as new solutions can be created from more than one reference solution (in contrast to genetic algorithms), even with a small reference set, many new solutions can be created at each iteration. Therefore, the reference set is typically a lot smaller than the corresponding population in a genetic algorithm.

However, as the boundaries between metaheuristic algorithm classification sometimes overlap and as different metaheuristic approaches are often hybridised, so also, features of scatter search may appear in genetic algorithms and vice versa. The comparisons between genetic algorithms and scatter search described here are just a basic introduction. For further information on scatter search and a more in depth analysis see [115, 118, 119, 155].

Population based optimisation methods have previously and successfully been applied to employee timetabling problems in various forms. Example approaches

include memetic algorithms [49] and genetic algorithms [12, 13, 93, 140, 144, 190, 229]. These papers were reviewed in chapter 2.

At the time the work presented here was undertaken, there had been very little research into investigating scatter search for personnel scheduling and no known applications of it to nurse rostering. This made it an appealing method to test, especially considering how successful evolutionary approaches for nurse rostering have been previously. This is a conclusion which was simultaneously (but independently) made by Maenhout and Vanhoucke. They have also implemented and tested a scatter search for the nurse rostering problem [167]. Surprisingly, their implementation of Glover's template is almost entirely different to the one presented here. However, they also were able to achieve successful results albeit on a variation of the nurse rostering problem. Scatter search and path relinking strategies have been applied to a considerable variety of problems other than nurse scheduling though. For example, arc routing [125], linear ordering [71], quadratic assignment [81], mixed integer programming [120] and exam proctor assignment [164]. All of these studies have demonstrated promising results.

The next section describes the scatter search implementation and section 6.2 contains the results of testing this algorithm on the benchmark instances introduced in chapter 4. To help draw conclusions, the scatter search has been compared against Brucker at al.'s constructive method [46] and the memetic algorithm of Burke at al. [49]. The variable depth search presented in chapter 5 is also used for comparisons as well.

6.1 The Algorithm

In Glover's template for scatter search [115], five component subroutines for the overall process are outlined. The following sections describe an implementation of these subroutines for the nurse rostering problem. The overall scatter search is outlined in Figure 6.1.

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

Figure 6.1 Scatter search overview

6.1.1 The Diversification Generation Method

A diversification generation method is required to create a diverse set of solutions. These solutions are then improved (according to the objective function) and added to the initial reference set subject to certain criteria. When creating the diverse set of solutions, the objective value for each solution is not relevant, only its similarity to other solutions in the set is of interest. A number of methods were tested for creating diverse solution sets with varying degrees of success. Of these methods, the one outlined in Figure 6.2 consistently produced the most diverse set of solutions.

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

Figure 6.2 Pseudocode for the scatter search initial set creation

To measure the similarity of two rosters, a simple but effective method is used: counting the number of *nurse to shift* assignments in common. An example of this is given in Figure 6.3 which shows the individual schedules of three nurses (labelled D, E, and F) from two different rosters. Identical assignments are highlighted, for example nurse E has a late shift (L) on Monday 4th in both schedules. In this example, just looking at these three nurses' schedules, there are seventeen identical assignments in the two rosters.

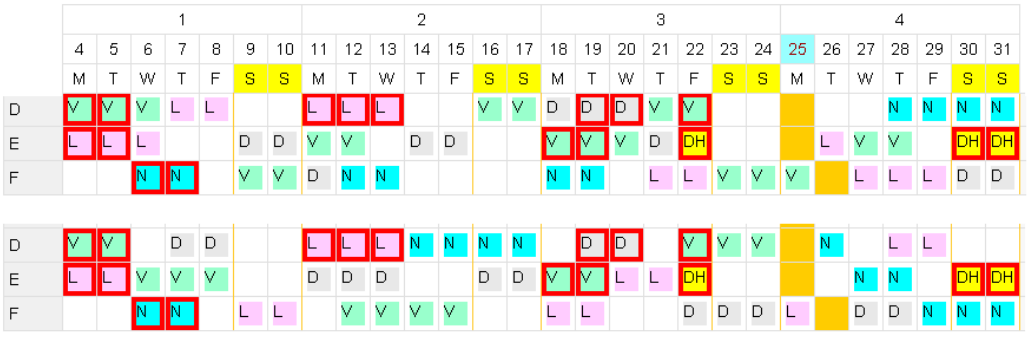

Figure 6.3 Example of the roster similarity measure

To demonstrate the success of the method, we compare it to a purely random approach which simply assigns randomly selected shifts to randomly chosen nurses. The assignments are made subject to no hard constraint violations (e.g. skills needed to perform certain shifts) until total cover is satisfied. A set of ten solutions was generated using both methods and the total number of shift assignments in common between all solutions in the set counted. Five repeats were performed on each instance and the average numbers of common assignments for each instance are given in Table 6.1. The results show that, on nine out of the ten instances, the random method made approximately twice as many common assignments. On the other instance, the random method made roughly 30% more common assignments.

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

Table 6.1 Comparison of common assignments by different generation methods

6.1.2 Improvement Method

The goal is to try to improve any solutions according to their objective function. If necessary, it may also repair solutions. The solutions which it works upon may be those produced by the diversification generation method (see section 6.1.1) or the solution combination method (see section 6.1.5).

The improvement method used is the time predefined variable depth search presented in chapter 5. Experiments were performed using a number of different

maximum run times to investigate how it affected the performance of the scatter search. The results are provided in section 6.2. For the experiments, the same settings as described in section 5.3.1 were used.

6.1.3 Reference Set Update Method

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

The reference set is initialised in a similar manner to that used by Glover at al. [119]. After all the solutions produced by the diversification method are improved by the variable depth search, they are ranked according to the objective function. The best b_1 of these solutions are then added to the reference set. From the remaining solutions, b_2 are selected and added, based on their contribution to the diversity of the reference set. Figure 6.4 outlines the process. b_1, b_2 and the number of diverse solutions to initially generate are all parameters which may affect the running time of the algorithm and the quality of schedules produced. In section 6.2, the results of varying these parameters are presented.

P is the set of solutions created using the diversification generation method.
RefSet is the reference set and is initially empty.
b_1 and b_2 are algorithm specific parameters (integers >= 0).

1. FOR 1 to b_1
2. Select from *P* the best solution according to the objective function
3. Remove the solution from *P* and add it to *RefSet*
4. ENDFOR
5. FOR 1 to b_2
6. For each solution in *P* calculate its total similarity to all the solutions currently in *RefSet* (using the similarity function)
7. Select the least similar solution (the schedule with least assignments in common with other rosters in *RefSet*)
8. Remove the solution from *P* and add to *RefSet*
9. ENDFOR

Figure 6.4 Scatter search reference set initialisation

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

6.1.4 Subset Generation Method

The subset generation method is used to identify the subsets of solutions in the reference set that will be used by the combination method to create new solutions. A commonly used subset generation method in scatter search is that suggested by Glover [115]. This approach is also adopted here. Using this method, four different types of subsets of increasing size are identified. They are:

1. All unique subsets of the reference set containing 2 elements.
2. Subsets of size 3 identified by adding to each 2-element subset (above) the best solution not already in this subset.

3. Subsets of size 4 identified by adding to each 3-element subset (above) the best solution not already in this subset.
4. Subsets containing the best *i* solutions, for i = 5 to |RefSet|

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

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

6.1.5 The Solution Combination Method

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

Although the subset generation method can be easily adapted to a wide range of problems, the solution combination method is often more specifically designed for each problem. Glover et al. [118] discuss a number of forms that the solution combinations or path relinking could take. The solution combination method developed here is categorized in their paper as a constructive neighbourhood approach "*where the guiding solutions vote for attributes to be included in the initiating solution*" [118]. In our case, the attributes are *shift to nurse* allocations within the rosters.

A solution can be regarded as simply a number of *shift to nurse* assignments. In the solution combination method, each *shift to nurse* allocation in each solution to be combined is regarded as a *vote* for a *candidate*. The candidates available for

selection are all the possible *shift to nurse* assignments for the problem instance. All these votes are then analysed and used to construct a new solution. The shift assignments (candidates) for the new solution are made according to the number of votes they received from the guiding solutions. The pseudocode in Figure 6.5 outlines the process.

In Figure 6.5, a candidate is a *shift to nurse* assignment on a specific day. The voters are the guiding solutions, each solution is a voter and each assignment within that solution is a vote for a specific candidate. The first step in the process is to create a new solution which initially has no assignments.

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

Figure 6.5 An outline of the scatter search solution combination method

At step 5 of Figure 6.5, the list of candidates (that is, *shift to nurse* assignments) is sorted by the number of votes they received. As the guiding sets of solutions are small (see the subset definitions), the candidates are often tied. If this is the case, two tie-breakers are used. If two candidates receive the same number of votes, the candidate whose voters have had the least total number of successful votes is ordered first. If this does not differentiate between the two candidates,

then the candidate whose voters have the lowest sum of objective function values comes first (lowest as it is a minimisation problem).

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

6.2 Results

The algorithm was tested using the benchmark data sets introduced in chapter 4.

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

To conduct these experiments, a reference set of size 5 (b_1=3, b_2=2, initial number of solutions=8) was used and the variable depth search was given 5 seconds, 30 seconds and 5 minutes. An additional experiment was conducted where the variable depth search was replaced with a hill climbing algorithm which uses the single shift neighbourhood introduced in section 5.1.1. As shown, this local search typically has a very short execution time (less than one second on smaller instances).

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

The results of these experiments are shown in Table 6.2. As well as the computation time, the number of solutions evaluated is also given. The experiments were performed on desktop PC with an Intel Pentium 4 2.4GHz processor.

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

Table 6.2 Results of varying the reference set size and the improvement method

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

Using these settings, further tests were performed to compare this approach to the memetic algorithm, MEH, of Burke et al [49]. MEH is a hybrid approach which performs a tabu search on individuals in the population between generations and a greedy shuffling step on the best solution at the end. It was shown to be a robust approach and the best method on the more difficult instances. The same settings as described in the original paper were used (underlying memetic algorithm ME4, population size of twelve and stop criterion of no improvement during two generations). Five repeats of both the scatter search and MEH were executed. The best and average solutions and average computation times are shown in Table 6.3. The scatter search with RefSet=5 and using the hill climbing improver was also tested for comparative purposes.

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

Table 6.3 Scatter search compared to MEH

The results show that the scatter search using the hill climber as the improver (SS1) produces average solutions which are found in less time and are better than MEH on seven out of the ten instances. For the other three instances, the solutions are better for two of them but used slightly more time and worse on one but using less time. The Wilcoxon signed rank test was used to test the null hypothesis (H_0) that the difference in objective values of all the solutions produced by the two algorithms are symmetrically distributed around the central point of zero. It had a probablity of greater than 0.05.

The scatter search using the variable depth search with a maximum run time of five seconds as the improver (SS2) outperforms SS1, but with a longer run times for all instances except BCV-3.46.1. SS2's average results compared to SS1 are better for eight of the instances, equal on one and were worse on the other.

For further comparisons, Table 6.4 contains the average results of SS1, the best results of Brucker et al's heuristic constructive approach on the BCV data sets and Burke et al's best result on the ORTEC01 data set. Brucker et al.'s experiments were all performed on the same machine as SS1.

As can be seen, SS1 outperforms the constructive approach on all but two instances (for one of which they are equal). It can also be noted that SS1 uses less computation time on all instances but one. SS1 does not outperform the result of Burke at al. on instance ORTEC01 although it does use considerably less computation time. SS2 does produce a better result in less time for ORTEC01 though. Comparing SS1 with the constructive approach, H_0 (as defined above) had a probability of greater than 0.05 using the Wilcoxon signed rank test and the Sign test.

	SS1		Brucker et al. [46]		Burke et al. [50]	
	Penalty	Time (s)	Penalty	Time (s)	Penalty	Time
ORTEC01	1707	713	-	-	541	12 hours
BCV-1.8.1	268	66	323	136	-	-
BCV-2.46.1	1588	1665	1594	3424	-	-
BCV-3.46.1	3396	5226	3601	2888	-	-
BCV-4.13.1	12	114	18	208	-	-
BCV-5.4.1	135	9	200	16	-	-
BCV-6.13.1	904	207	890	304	-	-
BCV-7.10.1	385	76	396	216	-	-
BCV-8.13.1	148	123	148	224	-	-
BCV-A.12.1	1813	518	3335	944	-	-

Table 6.4 Comparisons of scatter search with other algorithms

For a final comparison, the variable depth search (VDS) was tested on its own but with a predefined maximum run time equal to the average time used by MEH and then SS1. Five repeats were also performed and the best and average results are shown in Table 6.5.

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

Table 6.5 VDS with the same maximum run times as MEH and SS1

Looking at average results, the variable depth search outperforms MEH on seven of the ten instances, is equal on one and worse on the other two. Using best results, VDS is better on seven out of ten instances and equal on three. When compared to SS1 and using identical run times, for the average results, VDS is better on six out of ten instances, equal for two and worse on two. Comparing best results, VDS is better than SS1 on five instances, equal on three and worse on two. Comparing MEH with VDS, H_0 had probability of greater than 0.05 using the Wilcoxon signed rank test. Comparing SS1 with VDS, H_0 also has a probability greater than 0.05.

6.3 Conclusion

A scatter search has been presented for the nurse rostering problem and tested using benchmark instances. Scatter search is similar to a memetic algorithm in that it is a population based, evolutionary approach which improves individuals between generations using a heuristic or exact method. Scatter search differs in that new schedules can be constructed with guidance from more than two solutions and random decisions are replaced with strategic rules. The first experiments showed that the scatter search is more powerful when using the

variable depth search as the improvement method rather than a hill climber operating over the single shift neighbourhood. However, this improvement in solution quality comes at a cost of extra computation time. Setting a maximum run time for the VDS of 5 seconds though is a good compromise for most users on today's PCs. Even on the largest instances, this produced a run time of less than a couple of hours.

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

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

Interestingly, there is one instance on which the scatter search outperforms the VDS and is consistently strong; instance BCV-A.12.1. This is highlighted by the solution for BCV-A.12.1 with penalty 1378 which was found in the first set of experiments. 1378 is a new record and was particularly impressive considering the days of computation that had been used on it previously by other methods; for example the VDS during its development and testing. It is not clear why the scatter search should be so suited to this instance. BCV-A.12.1 does contain

constraint types which do not appear in the other instances (for example the tutorship constraint), which may be a possible explanation.

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

7 Conclusions

In Section 1.1 the central research question of the thesis was defined: “To what extent can the state of the art metaheuristic approaches to nurse rostering be improved upon, particularly to meet today’s real world needs in complex operating environments?” To answer this question, six hypotheses were formed. We will now attempt to resolve these hypotheses based on the investigations, experiments and interpretation of results presented in Chapters 3-6.

Hypothesis 1: Based on recent advances in metaheuristic approaches to nurse rostering, improvements can be made on the genetic algorithm in ORTEC’s software Harmony. To test this hypothesis a hybrid heuristic ordering and variable neighbourhood search was developed. When compared against the genetic algorithm, a statistical analysis of the results leads us to conclude that with a high level of probability the hybrid method is better on the smaller instances (up to twenty nurses). On the larger instances it is not possible to make confident conclusions on either algorithm’s superiority. Therefore, although the hybrid variable neighbourhood search is a successful algorithm which incorporates novel and effective ideas, this hypothesis can be neither entirely accepted or rejected.

Hypothesis 2: The research community will significantly benefit from the development of a collection of real world benchmark data sets. As recognised at the start of the research, this hypothesis will have to be tested in a time frame beyond the scope of this project. However, the data sets and related software have been created. Extensibility, accessibility and the ability to bridge the gap between

research and practice were key design goals which have been achieved. As a result we believe this hypothesis will eventually be accepted.

Hypothesis 3: Very large scale neighbourhood search techniques can be successfully applied to nurse rostering. The variable depth search confirms this hypothesis. When it was compared against a number of state-of-the-art metaheuristic approaches the results show this is a very effective algorithm.

Hypothesis 4: A successful time predefined algorithm can be developed for the nurse rostering problem. Again the variable depth search confirms this hypothesis. It uses a heuristic which, based on the execution time remaining, dynamically alters the length of chains of swaps examined when looking for an improvement. In effect this prevents over intensification and ensures diversification during the search. The algorithm was competitive with and often superior to other approaches even when adjusting to their computation times.

Hypothesis 5: A class of search neighbourhoods that are known to be very effective for the nurse rostering problem but are computationally intensive to use, can now be applied equally successfully but with much shorter computation times. This hypothesis was confirmed in Section 5.1.3 and was a result of significant practical importance. These neighbourhood operators were previously known to be very effective but their use had been very restricted due to their increased size and subsequent high computational expense. However, it is now possible to search these neighbourhoods exhaustively in very short computation

times and produce high quality rosters. They may also be incorporated in more sophisticated approaches (as in the variable depth search).

Hypothesis 6: If a successful very large scale neighbourhood search algorithm can be developed, it will be possible to incorporate it in a novel evolutionary algorithm for increased robustness. To test this hypothesis a scatter search was developed and the variable depth search was used as the improvement method applied to solutions between generations. The scatter search was shown to be successful when compared against a previously published evolutionary approach. It produced very high quality solutions on all instances and a new record on a particularly complex instance. When compared against the variable depth search though, over all instances, it could not be concluded with a high level of confidence that one method outperformed. Hence, although the first part of the hypothesis can be confirmed, there is not clear evidence that the scatter search is in fact more robust.

Based on these investigations we can now attempt to answer the main research question: “To what extent can the state of the art metaheuristic approaches to nurse rostering be improved upon, particularly to meet today’s real world needs in complex operating environments?”. We can conclude that the hybrid variable neighbourhood outperforms Harmony’s genetic algorithm on smaller instances. We cannot make confident conclusions about their performance on larger instances. We can conclude that the variable depth search is better than the tabu search of Burke et al. [58] and the hybrid constructive approach of Brucker at al. [46]. Although the variable depth search outperforms or is equal to the memetic

algorithm of Burke at al. [49] on nearly all test instances, based on the statistical analysis we cannot confidently conclude that it is in fact superior. Similarly, the scatter search produces better results than the memetic algorithm on most of the benchmark data sets. However, based on the statistical anlaysis there is not enough evidence to make confident conclusions on which is the better algorithm.

7.1 Contributions

As well as answering the main research question this thesis has also been aimed at advancing research into the highly practical and scientifically challenging nurse rostering problem. It makes a number of significant research contributions.

Chapter 2 contains a thorough review of the automated nurse rostering literature and associated publications. Although literature reviews on this topic already exist e.g. [30, 60, 98], a surprisingly large number of publications have appeared since the last of these surveys was published in 2004 (over 30). Also, with the previous summaries to refer to, chapter 2 includes interesting contributions, strengths and weaknesses that were not previously discussed. An effort was made to ensure that the review gives a different angle on the publications, highlighting key points not previously mentioned. Hence, it contributes to and strengthens the existing library of nurse rostering surveys.

The review begins with a general overview of personnel scheduling, an introduction to the different scheduling problems and definitions of key terms used in personnel scheduling. The variety of approaches to solving rostering problems are then organised into ten distinct categories, beginning with mathematical programming and ending with hyperheuristics. Each of these general methodologies are briefly examined and all significant publications

falling within each category are reviewed in a chronological order. Related nurse rostering publications, overviews and surveys are also discussed. The review concludes with a short critical discussion of key papers, an analysis of the progress in automated nurse rostering and the relation of the research in this thesis to the large body of previous work.

The research conducted in collaboration with ORTEC makes a number of contributions. Firstly, a novel and successful hybridisation of heuristic ordering and variable neighbourhood search was developed. This was validated on a challenging real world problem against a commercially successful genetic algorithm. As a result the scheduling software Harmony is also now no longer a ‘black box’ to other researchers. Instead, benchmark results have been provided which can be used to provide validation for other approaches. The ORTEC problem has already attracted the attention of other researchers e.g. [64].

The algorithm is an iterative procedure of applying a variable neighbourhood descent, heuristically disrupting the roster and then repairing it using heuristic ordering. The variable neighbourhood search can be regarded as an intensification phase in the overall search and the disruption and repair as a diversification mechanism. The disruption method is a simple but effective idea, unassigning the shifts of employees who have poor quality schedules. The heuristic ordering phase then attempts to improve the quality of these schedules by ranking the unassigned shifts in order of difficulty (to assign) before reassigning them over all nurses using a greedy method. For example, night shifts and weekend shifts have a number of high priority constraints associated with them so it makes sense to try

to assign them first. The variable neighbourhood descent is then reapplied from this point in the search space to try to further improve the roster.

Experiments were conducted to examine the effect on solution quality of increasing the size of the disruption (i.e. the number of employees' schedules to unassign shifts from). Interestingly it was discovered that the optimum disruption size appears to be unrelated to the size of the instance being solved. The best number of nurses' schedules to unassign was around 3-5 regardless of the number of nurses in the roster.

Experimental tests showed this approach was consistently more successful than Harmony's existing genetic algorithm on instances with less than twenty nurses and competitive on larger instances. As a result, the hybrid variable neighbourhood search was incorporated in the latest product versions of Harmony alongside the genetic algorithm.

The new benchmark problems and related software have provided a solid platform from which nurse rostering research can build upon well into the future. Although this was a significant amount of work (all the software alone is over 35,000 lines of code at a recent count), the invested time and effort will benefit other researchers and be rewarded by the quality of new algorithms and related research it will produce. This, in turn, will lead to an increase in the adoption and impact of automated personnel scheduling in practice.

To describe and share nurse rostering instances, an XML format has been defined (using schema). Although a large variety of instances can already be presented, the format has been designed with flexibility and future expansion in mind. A number of rostering instances taken from real world scenarios have been

converted into this new format and made publicly available for other researchers (see http://www.cs.nott.ac.uk/~tec/NRP/). These instances have also been used to establish benchmark results and the best known solutions are also available online. Benchmarks will help validate new algorithms and methodologies and encourage competition and collaboration. This will result in more powerful solvers for practical, real world scenarios.

To help and encourage other researchers a number of software tools have been created and made available (including source code). These include parsers, helpful data structures, objective functions and solvers. Rosters may also be converted to/saved as HTML for examining solutions, their violations and explanations of penalties via a web browser (Figure 7.1). An application has also been created for viewing rosters and adjusting them by hand. This gives a better feel for the complexity of the problems and can also be used to verify new solutions to the benchmark instances (Figure 7.2).

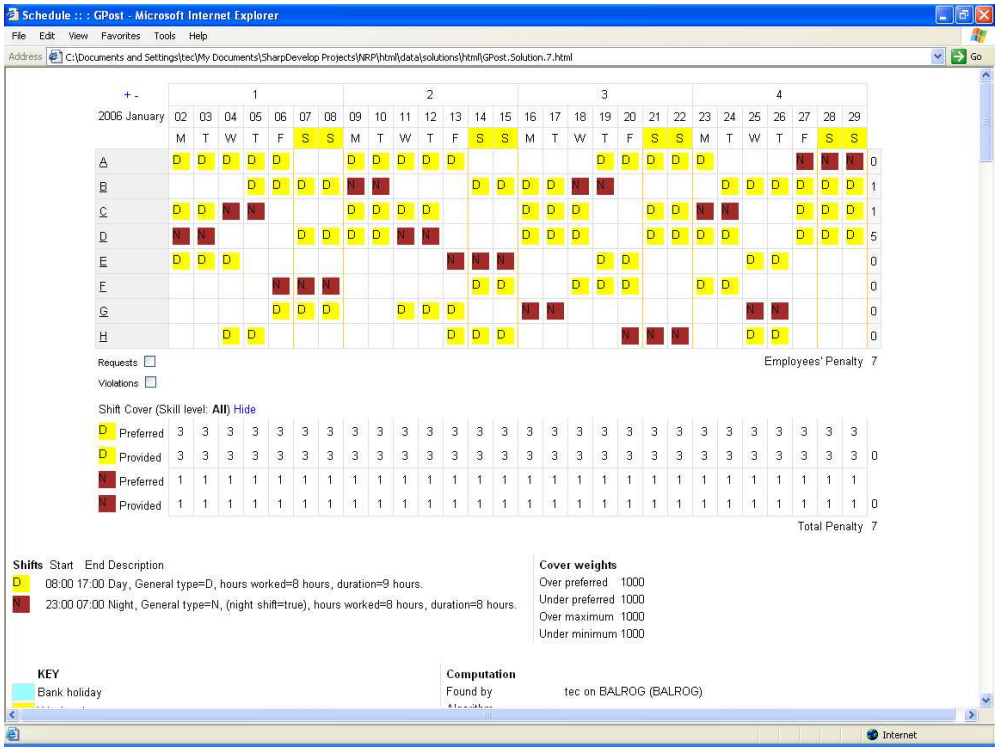

Figure 7.1 Roster displayed using HTML

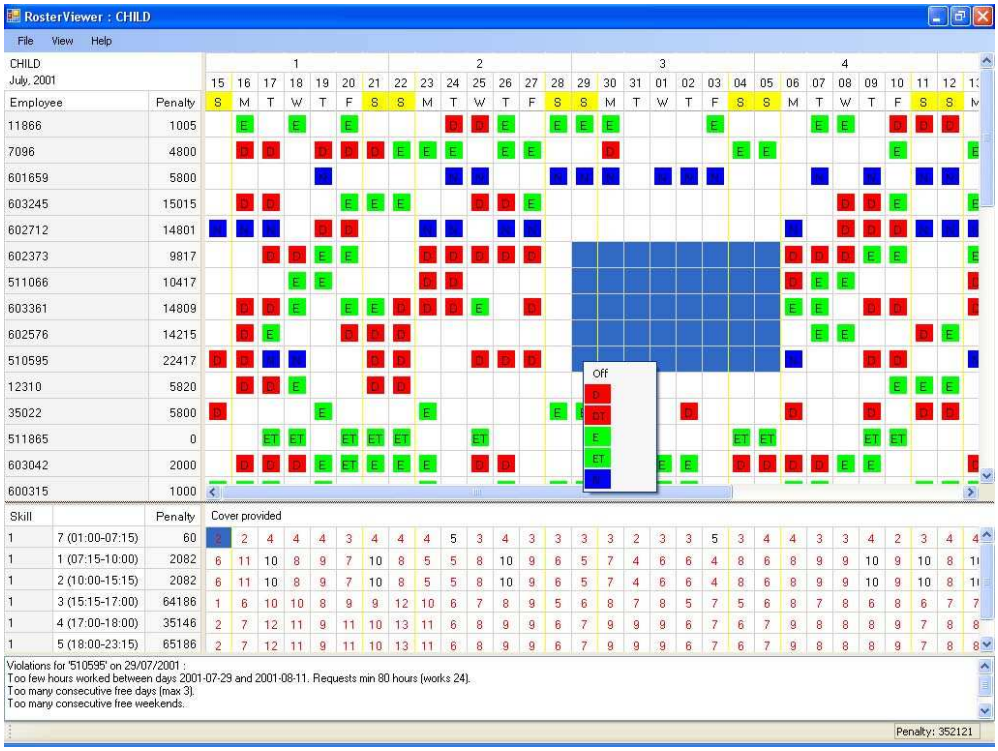

Figure 7.2 Screenshot of roster viewer application

Chapter 5 begins with an analysis of the very effective, block neighbourhood for nurse rostering problems. Although this neighbourhood was already known, its previous use was severely restricted due to slower computers and its larger size (compared to the single shift/day neighbourhood). It was shown that these neighbourhoods can now be exhaustively searched very quickly, producing high quality solutions. The approach in chapter 5 takes this even further by chaining these block moves together in a variable depth search (a class of very large scale neighbourhood search). A number of heuristics for achieving this were developed and tested. The result was a very competitive algorithm. Again, it was validated against the state of the art, previously published methods on our new benchmark instances. The most efficient heuristics for building the chains was a positive gain heuristic and a heuristic which only tests swaps which are flagged as containing violations. The positive gain heuristic was inspired by the ‘Gain Criterion’ of the Lin-Kernighan algorithm for the travelling salesman problem. It ensures the chain is only continued if the last but one move resulted in an improved (although infeasible) solution.

A particularly novel feature of this algorithm was a mechanism for allowing the user to specify a maximum run time which the algorithm dynamically reacts to, in order to use its time more effectively. It does this by altering the maximum length of chains it may examine. Hence, it avoids over intensification and ensuring diversification.

Finally, the variable depth search has been incorporated into an evolutionary approach. The result is a robust method which produces solid results on all

instances. The algorithm was particularly effective on one of the more complex instances. This is also one of the first applications of scatter search to the nurse rostering problem.

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

All the source code for the algorithms developed for the benchmark instances is publicly available. This is also a novel contribution, especially within the field of nurse rostering. Writing code to be shared with other researchers, requires a

discipline and standard which is not always necessary for code written for private use.

7.2 Publications

The research presented in this thesis has been published (or is currently under review) as follows:

[50] Burke, E.K., T. Curtois, G. Post, R. Qu, and B. Veltman, *A Hybrid Heuristic Ordering and Variable Neighbourhood Search for the Nurse Rostering Problem.* European Journal of Operational Research, Accepted for publication, to Appear 2008.

[51] Burke, E.K., T. Curtois, R. Qu, and G. Vanden Berge, *A Scatter Search for the Nurse Rostering Problem.* 2007, Under journal review.

[52] Burke, E.K., T. Curtois, R. Qu, and G. Vanden Berge, *A Time Predefined Variable Depth Search for Nurse Rostering.* 2007, Under journal review.

The author of this thesis is also the main author of the papers listed above. During this project the author also contributed to the following related work which has been published or is currently under review. The author's contributions to these papers were implementing (but not designing) algorithms and/or conducting experiments.

[46] Brucker, P., E.K. Burke, T. Curtois, R. Qu, and G. Vanden Berghe, *Adaptive Construction of Nurse Schedules: A Shift Sequence Based Approach and New Benchmarks.* Under journal review, 2006.

[82] Curtois, T., L. Fijn van Draat, J.-K. van Ommeren, and G. Post. *Progress Control in Variable Neighbourhood Search*, in *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling*. E.K. Burke and H. Rudova, Editors. 2006. Brno, Czech Republic. pp. 376-380.

7.3 Future Research

This section discusses possible future research directions which are directly related to the specific research presented in this thesis and for nurse rostering in general.

7.3.1 Future Research Directly Related to this Thesis

A large number of nurse rostering problems described in the literature could be presented in the current version of the data format. However, the format will continue to be extended to include a wider variety of instances with varying objective functions. Currently, many of the constraints are soft and to model a problem where one of these constraints is actually hard, a very high weight is set. Although this has proved satisfactory, it may be an improvement to include an option to identify a constraint as strictly hard. This would reduce the risk of returning infeasible solutions.

Since this thesis was completed, two more problems have been added to the database. One from Queen's Medical Centre, Nottingham and another from SINTEF (a Norwegian research organisation).

The format is also currently being extended to allow the inclusion of instances (including physician scheduling) from hospitals in the area of Montreal, Canada.

These instances allow minimum, maximum and preferred levels of cover to be specified. The objective function includes a measure of deviation from preferred levels which has to be minimised. To accommodate these instances, the format also now allows cover constraints to be specified per periods of the day as well as per shift. The shifts therefore can include definitions of which periods of the day they cover. A number of new employee constraints particular to these instances have also been added. This work, which is being carried out in collaboration with the Université de Montréal, will allow us to make comparisons between different algorithmic approaches on shared instances.

Relating to the algorithms presented, there may be potential for further investigation. For example, in the variable depth search, specific neighbourhoods (i.e. maximum block size settings) and heuristics are particularly effective on certain instances. A method which can exploit this by, for example, intelligently selecting neighbourhoods and heuristics, may be able to contribute gains in performance. One possibility may be an algorithm which runs some short preliminary tests on the instance, testing different parameters and heuristics in order to estimate a good set of parameters for the variable depth search. An alternative approach may be to dynamically adjust the algorithm's set of heuristics and parameters as the search progresses. This is somewhat akin to hyperheuristics [63, 217].

Another interesting idea with potential benefit is related to analysing and optimising constraint evaluation functions used in neighbourhood searches. As discussed in section 5.2.3 improvements in the speed of the evaluation functions

can contribute to increased performance of the search algorithms (more solutions examined per second). Some work has been initiated in this direction e.g. [53]. However, there may be scope for further improvements. Pattern matching algorithms could have possible application here. Again, the benchmark instances and publicly available code could be used for experiments and validation.

7.3.2 Future Research Directions for Nurse Rostering

In the survey papers [60, 73, 99] a variety of promising future research directions related to nurse scheduling in general are suggested. Some of the key ideas include:

Integrating staffing and rostering models. If the staffing and rostering problems are combined into a single model, it may yield benefits in terms of cost savings and increased satisfaction with work schedules. For example, a combined model may help decision makers analyse the relationship between the workforce size (and related costs) and the quality of rosters that could be produced under different demand scenarios. A multiobjective formulation may be most appropriate here.

Pareto optimisation has also been suggested as having utility in nurse rostering. So far, though, the number of investigations using these methods has been limited, [141, 158] being the only real applications. If a decision maker wishes to view or choose from the trade off between two or more conflicting objectives (or groups of objectives), then a Pareto optimisation approach would be an obvious choice.

Producing more robust schedules which can adapt more easily to unforeseen circumstances would also be an interesting scientific challenge with practical benefits. There has been a variety of research into how to assign pool or float nurses or how to adjust schedules when employees are absent e.g. [188-190, 234]. However, there has been very little work into how to make these potential problems easier by building robustness and flexibility into the original rosters.

Parameterless algorithms. Naturally, end users do not want to fiddle with meaningless parameters to achieve the highest quality rosters. They just want to click a button and see the solutions appear. The time predefined variable depth search in chapter 5 shows how it is possible to accept a preferred run time and dynamically adjust to this requirement without requiring manual tinkering with settings, yet still produce excellent rosters. In section 7.3.1, methods which may enhance solution quality even further through automatic and intelligent parameter selection are also suggested. In the future, approaches which minimise the reliance on manual parameter/heuristic selection, should be aimed for. (This is also a recent research direction in exam timetabling [48]).

Improved user interfaces. Increasing the attractiveness and ease of use of user interfaces will increase the adoption of automated rostering systems in hospitals. As discussed, nurse rostering problems are complicated and as such require a lot of data input. Entering and modifying this data should be fast and intuitive for the typical end users. There is a considerable amount of work in researching and designing such complex user interfaces. However, appealing and user friendly interfaces will significantly increase the uptake of rostering systems.

Algorithmic improvements. This thesis has introduced a number of algorithms which have been shown to outperform commercial and previously published approaches. Easily accessible and practically oriented benchmark instances have also been introduced in order to allow future comparisons against these new methods. Predicting the shape and form of future solvers is difficult. However, problem decomposition and hybrid methods have both been suggested as having potential for the nurse rostering problem [60]. Again, problem decomposition has already been successfully investigated for exam timetabling [66].

References

1. Aarts, E., J. Korst, and W. Michiels, *Simulated Annealing,* in *Search Methodologies. Introductory Tutorials in Optimization and Decision Support Techniques,* E.K. Burke and G. Kendall, Editors. 2005, Springer. pp. 187-210.
2. Aarts, E. and J.K. Lenstra, Editors. *Local Search in Combinatorial Optimization*. 1997, Wiley.
3. Abdennadher, S. and H. Schlenker. *INTERDIP An Interactive Constraint Based Nurse Scheduler,* in *Proceedings of The First International Conference and Exhibition on The Practical Application of Constraint Technologies and Logic Programming*. 1999.
4. Abdennadher, S. and H. Schlenker. *Nurse scheduling using constraint logic programming*, in *Proceedings of the Eleventh Conference on Innovative Applications of Artificial Intelligence*. 1999: AAAI Press. pp. 838-843.
5. Abdullah, S., S. Ahmadi, E.K. Burke, and M. Dror, *Investigating Ahuja-Orlin's Large Neighbourhood Search Approach for Examination Timetabling.* OR Spectrum, 2007. **29**(2): pp. 351-372.
6. Abdullah, S., S. Ahmadi, E.K. Burke, M. Dror, and B. McCollum, *A Tabu Based Large Neighbourhood Search Methodology for the Capacitated Examination Timetabling Problem.* Journal of the Operational Research Society, 2007 (to appear).
7. Abernathy, W.J., N. Baloff, J.C. Hershey, and S. Wandel, *A Three-Stage Manpower Planning and Scheduling Model - A Service-Sector Example* Operations Research, 1973. **21**(3): pp. 693-711.

8. Ahuja, H. and R. Sheppard, *Computerized nurse scheduling.* Industrial Engineering, 1975. **7**: pp. 24-29.

9. Ahuja, R.K., Ö. Ergun, J.B. Orlin, and A.P. Punnen, *A survey of very large-scale neighborhood search techniques.* Discrete Applied Mathematics, 2002. **123**(1-3): pp. 75-102.

10. Aickelin, U. *Genetic Algorithms for Multiple-Choice Problems.* PhD thesis, University of Wales Swansea, 1999.

11. Aickelin, U., E.K. Burke, and J. Li, *An Estimation of Distribution Algorithm with Intelligent Local Search for Rule-based Nurse Rostering.* Journal of the Operational Research Society, 2007(in print).

12. Aickelin, U. and K.A. Dowsland, *Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem.* Journal of Scheduling, 2000. **3**(3): pp. 139-153.

13. Aickelin, U. and K.A. Dowsland, *An Indirect Genetic Algorithm for a Nurse Scheduling Problem.* Computers and Operations Research, 2003. **31**(5): pp. 761-778.

14. Aickelin, U. and J. Li, *An Estimation of Distribution Algorithm for Nurse Scheduling.* Annals of Operations Research, 2007(in print).

15. Aickelin, U. and P. White, *Building Better Nurse Scheduling Algorithms.* Annals of Operations Research, 2004. **128**: pp. 159-177.

16. Al-Zubaidi, H. and A.H. Christer, *Maintenance manpower modelling for a hospital building complex.* European Journal of Operational Research, 1997. **99**(3): pp. 603-618.

17. Alfares, H.K., *Survey, Categorization, and Comparison of Recent Tour Scheduling Literature.* Annals of Operational Research, 2004. **127**: pp. 145–175.

18. Anzai, M. and Y. Miura, *Computer program for quick work scheduling of nursing staff.* Medical Informatics, 1987. **12**: pp. 43-52.

19. Arabeyre, J.P., J. Fearnley, F.C. Steiger, and W. Teather, *The Airline Crew Scheduling Problem: A Survey*. Transportation Science, 1969. **3**(2): pp. 140-163.

20. Arthur, J.L. and A. Ravindran, *A multiple objective nurse scheduling model.* AIIE Transactions, 1981. **13**(1): pp. 55-60.

21. Azaiez, M.N. and S.S. Al Sharif, *A 0-1 goal programming model for nurse scheduling.* Computers and Operations Research, 2005. **32**(3): pp. 491 - 507.

22. Bailey, J., *Integrated days off and shift personnel scheduling.* Computing and Industrial Engineering, 1985. **9**(4): pp. 395-404.

23. Bailey, J. and J. Field, *Personnel scheduling with flexshift models.* Journal of operations management 1985. **5**(3): pp. 327-338.

24. Baker, K.R., R.N. Burns, and M. Carter, *Staff Scheduling with Day-Off and Workstretch Constraints.* AIIE Transactions, 1979. **11**(4): pp. 286-292.

25. Bard, J.F. and H.W. Purnomo, *A column generation-based approach to solve the preference scheduling problem for nurses with downgrading.* Socio-Economic Planning Sciences, 2005. **39**(3): pp. 193-213.

26. Bard, J.F. and H.W. Purnomo, *Preference scheduling for nurses using column generation.* European Journal of Operational Research, 2005. **164**(2): pp. 510-534.

27. Bard, J.F. and H.W. Purnomo, *Cyclic Preference Scheduling of Nurses Using A Lagrangian-Based Heuristic.* Journal of Scheduling, 2007. **10**(1): pp. 5-23.

28. Barták, R. *Constraint Programming: In Pursuit of the Holy Grail*, in *Proceedings of the Week of Doctoral Students (WDS99).* 1999. Prague, Czech Republic: MatFyzPress. pp. 555-564.

29. Beaulieu, H., J.A. Ferland, B. Gendron, and P. Michelon, *A mathematical programming approach for scheduling physicians in the emergency room.* Health Care Management Science, 2000. **3**(3): pp. 193-200.

30. Beddoe, G.R. *Case-Based Reasoning in Personnel Rostering*. PhD Thesis, University of Nottingham, UK, 2004.

31. Beddoe, G.R. and S. Petrovic. *A novel approach to finding feasible solutions to personnel rostering problems*, in *Proceedings of the 14th Annual Conference of the Production and Operations Management Society (POM), Savannah, Georgia, United States*. 2003.

32. Beddoe, G.R. and S. Petrovic, *Selecting and Weighting Features Using a Genetic Algorithm in a Case-Based Reasoning Approach to Personnel Rostering.* European Journal of Operational Research, 2006. **175**(2): pp. 649-671.

33. Beddoe, G.R. and S. Petrovic, *Enhancing Case-Based Reasoning for Personnel Rostering with Selected Tabu Search Concepts.* Journal of the Operational Research Society, To appear.

34. Begur, S.V., D.M. Miller, and J.R. Weaver, *An Integrated Spatial DSS for Scheduling and Routing Home-Health-Care Nurses.* Interfaces, 1997. **27**(4): pp. 35-48.

35. Belien, J. and E. Demeulemeester, *A branch-and-price approach for integrating nurse and surgery scheduling.* European Journal of Operational Research, To appear in 2007.

36. Bell, P.C., G. Hay, and Y. Liang, *A visual interactive decision support system for workforce (nurse) scheduling.* Information Systems and Operational Research, 1986. **24**(2): pp. 134-145.

37. Bellanti, F., G. Carello, F.D. Croce, and R. Tadei, *A greedy-based neighborhood search approach to a nurse rostering problem.* European Journal of Operational Research, 2004. **153**: pp. 28–40.

38. Berman, O., R.C. Larson, and E. Pinker, *Scheduling workforce and workflow in a high volume factory.* Management Science, 1997. **43**(2): pp. 158-172.

39. Berrada, I., J.A. Ferland, and P. Michelon, *A multi-objective approach to nurse scheduling with both hard and soft constraints.* Socio-Economic Planning Sciences, 1996. **30**(3): pp. 183-193.

40. Blake, J.T. and M.W. Carter, *A goal programming approach to strategic resource allocation in acute care hospitals.* European Journal of Operational Research, 2002. **140**(3): pp. 541-561.

41. Blau, R. and A. Sear, *Nurse Scheduling with a Microcomputer.* Journal of Ambulance Care Management, 1983: pp. 1-13.

42. Blöchliger, I., *Modeling staff scheduling problems. A tutorial.* European Journal of Operational Research, 2004. **158**(3): pp. 533-542.

43. Bosch, R. and M. Trick, *Integer Programming*, in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques* E.K. Burke and G. Kendall, Editors. 2005, Springer. pp. 69-96.

44. Bradley, D.J. and J.B. Martin, *Continuous personnel scheduling algorithms: a literature review.* Journal of the Society for Health Systems, 1991. **2**(2): pp. 8-23.

45. Brooks, I. and S. Swailes, *Analysis of the relationship between nurse influences over flexible working and commitment to nursing.* Journal of Advanced Nursing, 2002. **38**(2): pp. 117-126.

46. Brucker, P., E.K. Burke, T. Curtois, R. Qu, and G. Vanden Berghe, *Adaptive Construction of Nurse Schedules: A Shift Sequence Based Approach and New Benchmarks.* Under journal review, 2006.

47. Burke, E.K., Y. Bykov, J. Newall, and S. Petrovic, *A Time-Predefined Approach to Course Timetabling.* Yugoslav Journal of Operations Research, 2003. **13**(2): pp. 139-151.

48. Burke, E.K., Y. Bykov, J. Newall, and S. Petrovic, *A Time-Predefined Local Search Approach to Exam Timetabling Problems.* IIE Transactions, 2004. **36**(6): pp. 509-528.

49. Burke, E.K., P. Cowling, P. De Causmaecker, and G. Vanden Berghe, *A Memetic Approach to the Nurse Rostering Problem.* Applied Intelligence, 2001. **15**(3): pp. 199-214.

50. Burke, E.K., T. Curtois, G. Post, R. Qu, and B. Veltman, *A Hybrid Heuristic Ordering and Variable Neighbourhood Search for the Nurse Rostering Problem.* European Journal of Operational Research, Accepted for publication, to Appear 2007.

51. Burke, E.K., T. Curtois, R. Qu, and G. Vanden Berge, *A Scatter Search for the Nurse Rostering Problem.* 2007, Under journal review.

52. Burke, E.K., T. Curtois, R. Qu, and G. Vanden Berge, *A Time Predefined Variable Depth Search for Nurse Rostering.* 2007, Under journal review.

53. Burke, E.K., P. De Causmaecker, S. Petrovic, and G. Vanden Berghe. *Fitness Evaluation for Nurse Scheduling Problems,* in *Proceedings of the Congress on Evolutionary Computation (CEC2001).* 2001. Seoul, Korea: IEEE Press. pp. 1139-1146.

54. Burke, E.K., P. De Causmaecker, S. Petrovic, and G. Vanden Berghe. *Variable Neighbourhood Search for Nurse Rostering Problems,* in *Proceedings of the 4th Metaheuristics Internation Conference (MIC 2001).* 2001. Porto, Portugal. pp. 755-60.

55. Burke, E.K., P. De Causmaecker, S. Petrovic, and G. Vanden Berghe. *A Multi-Criteria Meta-heuristic Approach to Nurse Rostering,* in *Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002).* 2002. Hawaii, USA: IEEE Press. pp. 1197-1202.

56. Burke, E.K., P. De Causmaecker, S. Petrovic, and G. Vanden Berghe, *Variable Neighborhood Search for Nurse Rostering Problems,* in *Metaheuristics: Computer Decision-Making,* M.G.C. Resende and J.P. de Sousa, Editors. 2004, Kluwer. pp. 153-172.

57. Burke, E.K., P. De Causmaecker, S. Petrovic, and G. Vanden Berghe, *Metaheuristics for Handling Time Interval Coverage Constraints in Nurse Scheduling.* Applied Artificial Intelligence, 2006. **20**(3).

58. Burke, E.K., P. De Causmaecker, and G. Vanden Berghe. *A Hybrid Tabu Search Algorithm for the Nurse Rostering Problem,* in *Simulated*

Evolution and Learning, Selected Papers from the 2nd Asia-Pacific Conference on Simulated Evolution and Learning, SEAL 98, Springer Lecture Notes in Artificial Intelligence Volume 1585. B. McKay, et al., Editors. 1999: Springer. pp. 187-194.

59. Burke, E.K., P. De Causmaecker, and G. Vanden Berghe, *Novel Meta-heuristic Approaches to Nurse Rostering Problems in Belgian Hospitals*, in *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, J. Leung, Editor. 2004, CRC Press.

60. Burke, E.K., P. De Causmaecker, G. Vanden Berghe, and H. Van Landeghem, *The State of the Art of Nurse Rostering.* Journal of Scheduling, 2004. **7**(6): pp. 441 - 499.

61. Burke, E.K., E. Hart, G. Kendall, J. Newall, P. Ross, and S. Schulenburg, *Hyper-Heuristics: An Emerging Direction in Modern Search Technology*, in *Handbook of Meta-Heuristics*, F. Glover and G. Kochenberger, Editors. 2003, Kluwer. pp. 457-474.

62. Burke, E.K. and G. Kendall, Editors. *Search Methodologies. Introductory Tutorials in Optimization and Decision Support Techniques*. 2005, Springer.

63. Burke, E.K., G. Kendall, and E. Soubeiga, *A Tabu-Search Hyperheuristic for Timetabling and Rostering.* Journal of Heuristics, 2003. **9**(6): pp. 451 - 470.

64. Burke, E.K., J. Li, and R. Qu, *A Hybrid Model of Integer Programming and Variable Neighbourhood Search for Highly-constrained Nurses Rostering Problems.* Under review, 2006.

65. Burke, E.K., B. MacCarthy, S. Petrovic, and R. Qu, *Multiple-retrieval case-based reasoning for course timetabling problems.* Journal of the Operational Research Society, 2006. **57**: pp. 148–162.

66. Burke, E.K. and J. Newall, *A Multi-Stage Evolutionary Algorithm for the Timetable Problem.* IEEE Transactions on Evolutionary Computation, 1999. **13**(1): pp. 63-74.

67. Burke, E.K. and S. Petrovic, *Recent Research Directions in Automated Timetabling.* European Journal of Operational Research, 2002. **140**(2): pp. 266-280.

68. Burke, E.K. and S. Petrovic, *University Timetabling (Chapter 45)*, in *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, J.Y.-T. Leung, Editor. 2004, CRC Press.

69. Burke, E.K., S. Petrovic, and R. Qu, *Case-based heuristic selection for timetabling problems.* Journal of Scheduling, 2006. **9**(2): pp. 115-132.

70. Cai, X. and K.N. Li, *A genetic algorithm for scheduling staff of mixed skills under multi-criteria.* European Journal of Operational Research, 2000. **125**(2): pp. 359-369.

71. Campos, V., F. Glover, M. Laguna, and R. Martí, *An Experimental Evaluation of a Scatter Search for the Linear Ordering Problem.* Journal of Global Optimization, 2001. **21**(4): pp. 397-414.

72. Carter, M.W. and S.D. Lapierre, *Scheduling Emergency Room Physicians.* Health Care Management Science, 2001. **4**(4): pp. 347-360.

73. Cheang, B., H. Li, A. Lim, and B. Rodrigues, *Nurse rostering problems—a bibliographic survey.* European Journal of Operational Research, 2003. **151**(3): pp. 447-460.

74. Cheeseman, P., B. Kanefsky, and W.M. Taylor. *Where the really hard problems are*, in *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*. J. Mylopoulos and R. Reiter, Editors. 1991. San Mateo, CA: Morgan Kaufmann. pp. 331-337.

75. Chen, J.-G. and T.W. Yeung, *Hybrid expert-system approach to nurse scheduling.* Computers in Nursing, 1993. **11**(4): pp. 183-190.

76. Cheng, B.M.W., J.H.M. Lee, and J.C.K. Wu. *A Constraint-Based Nurse Rostering System Using a Redundant Modeling Approach*, in *Proceedings of Eighth IEEE International Conference on Tools with Artificial Intelligence*. 1996. pp. 140-148.

77. Cheng, B.M.W., J.H.M. Lee, and J.C.K. Wu, *A nurse rostering system using constraint programming and redundant modeling.* IEEE Transactions on Information Technology in Biomedicine, 1997. **1**(1): pp. 44-54.

78. Chiarandini, M., A. Schaerf, and F. Tiozzo. *Solving Employee Timetabling Problems with Flexible Workload using Tabu Search*, in *Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT-2000)*. E.K. Burke and W. Erben, Editors. 2000. Konstanz, Germany. pp. 298-302.

79. Chun, A.H.W., S.H.C. Chan, G.P.S. Lam, F.M.F. Tsang, J. Wong, and D.W.M. Yeung. *Nurse Rostering at the Hospital Authority of Hong Kong*, in *Proceedings of the Twelfth Conference on Innovative Applications of Artificial Intelligence*. 2000. pp. 951 - 956.

80. Cowling, P., G. Kendall, and E. Soubeiga. *Hyperheuristics: A robust optimisation method applied to nurse scheduling*, in *Proceedings of*

Parallel Problem Solving from Nature VII. 2002. Granada, Spain: Springer-Verlag. pp. 851-860.

81. Cung, V.-D., T. Mautor, P. Michelon, and A. Tavares. *A Scatter Search Based Approach for the Quadratic Assignment Problem*, in *Proceedings of IEEE International Conference on Evolutionary Computation*. 1997. pp. 165-169.

82. Curtois, T., L. Fijn van Draat, J.-K. van Ommeren, and G. Post. *Progress Control in Variable Neighbourhood Search*, in *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling*. E.K. Burke and H. Rudova, Editors. 2006. Brno, Czech Republic. pp. 376-380.

83. Dantzig, G.B., *A Comment on Edie's "Traffic Delays at Toll Booths"*. Journal of the Operations Research Society of America, 1954. **2**(3): pp. 339-341.

84. Dantzig, G.B., *Linear Programming and Extensions*. 1998: Princeton University Press.

85. Darmoni, S.J., A. Fajner, N. Mahé, A. Leforestier, M. Vondracek, O. Stelian, and M. Baldenweck, *Horoplan: computer-assisted nurse scheduling using constraint-based programming* Journal of the Society for Health Systems, 1995. **5**: pp. 41-54.

86. De Causmaecker, P. and G. Vanden Berge. *Relaxation of Coverage Constraints in Hospital Personnel Rostering*, in *Selected Revised Papers of 4th International Conference on Practice and Theory of Automated Timetabling, Springer Lecture Notes in Computer Science 2740*. E.K. Burke and P. De Causmaecker, Editors. 2003. pp. 129-147.

87. Deb, K., *Multi-objective Optimization*, in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, E.K. Burke and G. Kendall, Editors. 2005, Springer. pp. 273-316.

88. Dias, T.M., D.F. Ferber, C.C. de Souza, and A.V. Moura, *Constructing nurse schedules at large hospitals.* International Transactions in Operational Research, 2003. **10**(3): pp. 245-265.

89. Dorigo, M., V. Maniezzo, and A. Colorni, *Ant system: optimization by a colony of cooperating agents.* IEEE Transactions on Systems, Man, and Cybernetics. Part B: Cybernetics, 1996. **26**(1): pp. 29-41.

90. Dorigo, M. and T. Stützle, *Ant Colony Optimization*. 2004: The MIT Press.

91. Dowsland, K.A., *Nurse scheduling with tabu search and strategic oscillation.* European Journal of Operational Research, 1998. **106**(2): pp. 393-407.

92. Dowsland, K.A. and J.M. Thompson, *Solving a nurse scheduling problem with knapsacks, networks and tabu search.* Journal of the Operational Research Society, 2000. **51**(7): pp. 825-833.

93. Easton, F.F. and N. Mansour. *A Distributed Genetic Algorithm for Employee Staffing and Scheduling Problems,* in *Proceedings of the 5th International Conference on Genetic Algorithms*. 1993. San Mateo. pp. 360-367.

94. Easton, F.F., D.F. Rossin, and W.S. Borders, *Analysis of Alternative Scheduling Policies for Hospital Nurses.* Production and Operations Management, 1992. **1**(2): pp. 159-174.

95. Easton, K., G. Nemhauser, and M. Trick, *Sports Scheduling (Chapter 52)*, in *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, J.Y.-T. Leung, Editor. 2004, CRC Press.

96. Edie, L.C., *Traffic Delays At Toll Booths.* Journal of the Operations Research Society of America, 1954. **2**(2): pp. 107-138.

97. Ernst, A.T., P. Hourigan, M. Krishnamoorthy, G. Mills, H. Nott, and D. Sier. *Rostering Ambulance Officers*, in *Proceedings of the 15th National Conference of the Australian Society for Operations Research*. 1999. pp. 470–481.

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

99. Ernst, A.T., H. Jiang, M. Krishnamoorthy, and D. Sier, *Staff scheduling and rostering: A review of applications, methods and models.* European Journal of Operational Research, 2004. **153**(1): pp. 3-27.

100. Eveborn, P. and M. Rönnqvist, *Scheduler – A System for Staff Planning.* Annals of Operations Research, 2004. **128**: pp. 21–45.

101. Feo, T.A. and M.G.C. Resende, *Greedy Randomized Adaptive Search Procedures.* Journal of Global Optimization, 1995. **6**(2): pp. 109-133.

102. Ferland, J.A., I. Berrada, I. Nabli, B. Ahiod, P. Michelon, V. Gascon, and É. Gagné, *Generalized Assignment Type Goal Programming Problem: Application to Nurse Scheduling.* Journal of Heuristics, 2001. **7**(4): pp. 391-413.

103. Fitzpatrick, J.M., A.E. While, and J.D. Roberts, *Shift work and its impact upon nurse performance: current knowledge and research issues.* Journal of Advanced Nursing, 1999. **29**(1): pp. 18-27.

104. Fogel, D.B., *Evolutionary Computation: The Fossil Record* 1998: Wiley-IEEE Press.

105. Franz, L.S., H.M. Baker, G.K. Leong, and T.R. Rakes, *A mathematical model for scheduling and staffing multiclinic health regions.* European Journal of Operational Research, 1989. **41**(3): pp. 277-289.

106. Freuder, E.C. and M. Wallace, *Constraint Programming*, in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques* E.K. Burke and G. Kendall, Editors. 2005, Springer. pp. 239-272.

107. Fries, B.E., *Bibliography of Operations Research in Health-Care Systems.* Operations Research, 1976. **24**(5): pp. 801-814.

108. Gans, N., G. Koole, and A. Mandelbaum, *Telephone Call Centers: Tutorial, Review, and Research Prospects.* Manufacturing & Service Operations Management, 2003. **5**(2): pp. 79-141.

109. Gascon, V., S. Villeneuve, P. Michelon, and J.A. Ferland, *Scheduling the flying squad nurses of a hospital using a multi-objective programming model.* Annals of Operations Research, 2000. **96**: pp. 149-166.

110. Gendreau, M., J.A. Ferland, B. Gendron, N. Hail, B. Jaumard, S.D. Lapierre, G. Pesant, and P. Soriano. *Physician Scheduling in Emergency Rooms*, in *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling*. E. Burke and H. Rudova, Editors. 2006. Brno, Czech Republic.

111. Gendreau, M. and J.-Y. Potvin, *Tabu Search*, in *Search Methodologies. Introductory Tutorials in Optimization and Decision Support Techniques*, E.K. Burke and G. Kendall, Editors. 2005, Springer. pp. 165-186.

112. Glover, F., *Tabu Search - Part I.* ORSA Journal on Computing, 1989. **1**(3): pp. 190-206.

113. Glover, F., *Tabu Search - Part II.* ORSA Journal on Computing, 1990. **4**(1): pp. 4-32.

114. Glover, F., *Ejection Chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problems.* Discrete Applied Mathematics, 1996. **65**(1-3): pp. 223-253.

115. Glover, F., *A Template for Scatter Search and Path Relinking*, in *Lecture Notes in Computer Science, 1363*, J.K. Hao, et al., Editors. 1998, Springer. pp. 13-54.

116. Glover, F. and M. Laguna, *Tabu Search*, in *Modern Heuristic Techniques for Combinatorial Problems*, C.R. Reeves, Editor. 1993, Blackwell Scientific Publications: Oxford. pp. 70-150.

117. Glover, F. and M. Laguna, *Tabu Search*. 1997: Kluwer Academic Publishers.

118. Glover, F., M. Laguna, and R. Martí, *Fundamentals of Scatter Search and Path Relinking.* Control and Cybernetics, 2000. **29**(3): pp. 653-684.

119. Glover, F., M. Laguna, and R. Martí, *Scatter Search*, in *Advances in Evolutionary Computing*, A. Ghosh and S. Tsutsui, Editors. 2003, Springer. pp. 519-539.

120. Glover, F., A. Løkketangen, and D.L. Woodruff, *Scatter Search to Generate Diverse MIP Solutions*, in *OR Computing Tools for Modeling,*

Optimization and Simulation: Interfaces in Computer Science and Operations Research, M. Laguna and J.L. González-Velarde, Editors. 2000, Kluwer. pp. 299-317.

121. Glover, F. and C. McMillan, *The general employee scheduling problem: an integration of MS and AI.* Computers and Operations Research, 1986. **13**(5): pp. 563-573.

122. Glover, F., C. McMillan, and R. Glover, *A Heuristic Programming Approach to the Employee Scheduling Problem and Some Thoughts on "Managerial Robots"*. Journal of Operations Management, 1984. **4**(2): pp. 113-128.

123. Glover, F.W. and G.A. Kochenberger, Editors. *Handbook of Metaheuristics* 2003, Kluwer Academic Publishers.

124. Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*. 1989: Addison-Wesley.

125. Greistorfer, P., *A Tabu Scatter Search Metaheuristic for the Arc Routing Problem.* Computers & Industrial Engineering, 2003. **44**: pp. 249–266.

126. Gutjahr, W.J. and M.S. Rauner, *An ACO algorithm for a dynamic regional nurse-scheduling problem in Austria.* Computers & Operations Research, 2007. **34**: pp. 642–666.

127. Hansen, P. and N. Mladenović, *An introduction to variable neighborhood search*, in *Meta-heuristics: Advances and trends in local searchs paradigms for optimization*, S. Voss, et al., Editors. 1999, Kluwer Academic Publishers. pp. 433-458.

128. Hansen, P. and N. Mladenović, *Variable neighborhood search: Principles and applications.* European Journal of Operational Research 2001. **130**(3): pp. 449-467.

129. Hansen, P. and N. Mladenović, *Variable Neighborhood Search*, in *Handbook of Metaheuristics*, F. Glover and G.A. Kochenberger, Editors. 2003, Springer. pp. 145-184.

130. Hansen, P. and N. Mladenović, *Variable Neighborhood Search*, in *Search Methodologies. Introductory Tutorials in Optimization and Decision Support Techniques*, E.K. Burke and G. Kendall, Editors. 2005, Springer. pp. 211-238.

131. Hogg, T., B.A. Huberman, and C. Williams, *Phase transitions and the search problem.* Artificial Intelligence, 1996. **81**: pp. 1-15.

132. Holland, J.H., *Adaptation in Natural and Artificial Systems*. 1992: The MIT Press.

133. Hoos, H.H. and T. Stützle. *SATLIB: An Online Resource for Research on SAT*, in *SAT 2000*. I.P. Gent, H.v. Maaren, and T. Walsh, Editors. 2000: IOS Press. pp. 283-292.

134. Hung, R., *Hospital nurse scheduling.* Journal of Nursing Administration, 1995. **25**(7-8): pp. 21-23.

135. Ikegami, A. and A. Niwa, *A Subproblem-centric Model and Approach to the Nurse Scheduling Problem.* Mathematical Programming, 2003. **97**(3): pp. 517-541.

136. ILOG, *CPLEX Mathematical programming optimizer (http://www.ilog.com/products/cplex/ retrieved 27-June-2007).* 2006.

137. ILOG, *ILOG Constraint programming engine (http://www.ilog.com/products/cp/ retrieved 27-June-2007)*. 2006.

138. Inoue, T., T. Furuhashi, H. Maeda, and M. Takaba, *A Proposal of Combined Method of Evolutionary Algorithm and Heuristics for Nurse Scheduling Support System.* IEEE Transactions on Industrial Electronics, 2003. **50**(5): pp. 833- 838.

139. Isken, M. and W. Hancock, *A Heuristic Approach to Nurse Scheduling in Hospital Units with Non-Stationary, Urgent Demand, and a Fixed Staff Size*. Journal of the Society for Health Systems, 1990. **2**(2): pp. 24-41.

140. Jan, A., M. Yamamoto, and A. Ohuchi. *Evolutionary Algorithms for Nurse Scheduling Problem*, in *Proceedings of the 2000 Congress on Evolutionary Computation*. 2000. California, USA: IEEE Press. pp. 196-203.

141. Jaszkiewicz, A., *A metaheuristic approach to multiple objective nurse scheduling.* Foundations of Computing and Decision Sciences, 1997. **22**(3): pp. 169-183.

142. Jaumard, B., F. Semet, and T. Vovor, *A Generalized Linear Programming Model for Nurse Scheduling.* European Journal of Operational Research 1998. **107**(1): pp. 1-18.

143. Johnson, D.S. *A Theoretician's Guide to the Experimental Analysis of Algorithms*, in *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*. M.H. Goldwasser, D.S. Johnson, and C.C. McGeoch, Editors. 2002. pp. 215-250.

144. Kawanaka, H., K. Yamamoto, T. Yoshikawa, T. Shinogi, and S. Tsuruoka. *Genetic algorithm with the constraints for nurse scheduling problem,* in *Proceedings of the 2001 Congress on Evolutionary Computation*. 2001. Seoul, South Korea: IEEE Press. pp. 1123-1130.

145. Kennedy, J. and R. Eberhart. *Particle swarm optimization,* in *Proceedings of the 1995 IEEE International Conference on Neural Networks*. 1995. Perth, Australia. pp. 1942-1948.

146. Khoong, C.M., H.C. Lau, and L.W. Chew, *Automated Manpower Rostering: Techniques and Experience.* International Transactions in Operational Research, 1994. **1**(3): pp. 353-361.

147. Kirkpatrick, S., C.D. Gelatt, and M.P. Vecchi, *Optimization by Simulated Annealing.* Science, 1983. **220**(4598): pp. 671-680.

148. Kostreva, M.M. and K.S.B. Jennings, *Nurse scheduling on a microcomputer.* Computers and Operations Research, 1991. **18**(9): pp. 731-739.

149. Koza, J.R., *Genetic Programming*. 1992: MIT Press.

150. Kragelund, L.V., *Solving a timetabling problem using hybrid genetic algorithms.* Software: Practice and Experience, 1997. **27**(10): pp. 1121-1134.

151. Krasnogor, N., W. Hart, and J. Smith, Editors. *Recent Advances in Memetic Algorithms, Studies in Fuzziness and Soft Computing*. 2004, Springer.

152. Krasnogor, N. and J. Smith, *A Tutorial for Competent Memetic Algorithms: Model, Taxonomy and Design Issues.* IEEE Transactions on Evolutionary Computation, 2005. **9**(5): pp. 474-488.

153. Kwak, N.K. and C. Lee, *A Linear Goal Programming Model for Human Resource Allocation in a Health-Care Organization.* Journal of Medical Systems, 1997. **21**(3): pp. 129-140.

154. Kwan, R., *Bus and Train Driver Scheduling (Chapter 51),* in *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, J.Y.-T. Leung, Editor. 2004, CRC Press.

155. Laguna, M. and R. Martí, *Scatter Search. Methodology and Implementation in C*. 2003: Kluwer Academic Publishers.

156. Landa Silva, J.D., E.K. Burke, and S. Petrovic, *An Introduction to Multiobjective Metaheuristics for Scheduling and Timetabling*, in *Metaheuristics for Multiobjective Optimisation, Lecture Notes in Economics and Mathematical Systems*, X. Gandibleux, et al., Editors. 2004, Springer. pp. 91-129.

157. Lau, H.C., *On the Complexity of Manpower Shift Scheduling.* Computers & Operations Research, 1996. **23**(1): pp. 93-102.

158. Le, K.N. and D. Landa-Silva, *Simple Evolutionary Algorithm with Self-Adaptation for Multi-Objective Optimisation*. 2007, School of Computer Science and IT, University of Nottingham. Working Paper.

159. Li, H., A. Lim, and B. Rodrigues. *A Hybrid AI Approach for Nurse Rostering Problem*, in *Proceedings of the 2003 ACM symposium on Applied computing*. 2003. pp. 730-735.

160. Li, J. and U. Aickelin. *A Bayesian Optimization Algorithm for the Nurse Scheduling Problem*, in *Proceedings of 2003 Congress on Evolutionary Computation (CEC2003)*. 2003. Canberra, Australia: IEEE Press. pp. 2149-2156.

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

162. Liao, C.-J. and C.-Y. Kao, *Scheduling nursing personnel on a microcomputer*. Health Manpower Management, 1997. **23**(3): pp. 100-106.

163. Lin, S. and B.W. Kernighan, *An Effective Heuristic Algorithm for the Traveling-Salesman Problem.* Operations Research, 1973. **21**(2): pp. 498-516.

164. Lourenço, H., M. Laguna, and R. Martí, *Assigning Proctors to Exams with Scatter Search*, in *Computing Tools for Modeling, Optimization and Simulation*, M. Laguna and J.L. González-Velarde, Editors. 2000, Springer. pp. 215-228.

165. Lourenço, H.R., O.C. Martin, and T. Stützle, *Iterated Local Search*, in *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Editors. 2003, Kluwer. pp. 321-353.

166. Louw, M.J., I. Nieuwoudt, and J.H. van Vuuren, *Finding Good Nursing Duty Schedules: A Case Study*.

167. Maenhout, B. and M. Vanhoucke, *New computational results for the nurse scheduling problem: a scatter search algorithm*, in *Lecture notes in Computer Science, 3906*. 2006, Springer. pp. 159 -170.

168. Maier-Rothe, C. and H.B. Wolfe, *Cyclical scheduling and allocation of nursing staff.* Socio-Economic Planning Sciences, 1973. **7**: pp. 471-487.

169. Mason, A.J. and M.C. Smith. *A Nested Column Generator for solving Rostering Problems with Integer Programming*, in *International Conference on Optimisation: Techniques and Applications*. L. Caccetta, et al., Editors. 1998. Perth, Australia. pp. 827-834.

170. McCollum, B. *University Timetabling: Bridging the Gap between Research and Practice*, in *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling*. E.K. Burke and H. Rudova, Editors. 2006. Brno, Czech Republic. pp. 15-35.

171. McVicar, A., *Workplace stress in nursing: a literature review*. Journal of Advanced Nursing, 2003. **44**(6): pp. 633–642.

172. Meisels, A., E. Gudes, and G. Solotorevsky. *Employee Timetabling, Constraint Networks and Knowledge-Based Rules: A Mixed Approach*, in *Selected papers from the First International Conference on Practice and Theory of Automated Timetabling*. 1996: Springer-Verlag. pp. 93-105.

173. Meisels, A., E. Gudes, and G. Solotorevsky, *Combining rules and constraints for employee timetabling* International Journal of Intelligent Systems, 1997. **12**(6): pp. 419-439.

174. Meisels, A. and N. Lusternik. *Experiments on Networks of Employee Timetabling Problems*, in *E. Burke and M. Carter (Eds.) Selected papers from the Second International Conference on Practice and Theory of Automated Timetabling*. E. Burke and M. Carter, Editors. 1997: Springer-Verlag. pp. 130-141.

175. Meisels, A. and A. Schaerf, *ETP - Problem structure and File formats*. 1999.

176. Meisels, A. and A. Schaerf, *Modelling and solving employee timetabling problems.* Annals of Mathematics and Artificial Intelligence, 2003. **39**: pp. 41-59.

177. Merkle, D. and M. Middendorf, *Swarm Intelligence*, in *Search Methodologies. Introductory Tutorials in Optimization and Decision Support Techniques*, E.K. Burke and G. Kendall, Editors. 2005, Springer. pp. 401-436.

178. Meyer auf'm Hofe, H. *ConPlan/SIEDAplan: Personnel Assignment as a Problem of Hierarchical Constraint Satisfaction*, in *PACT-97: Proceedings of the Third International Conference on the Practical Application of Constraint Technology*. 1997. pp. 257-272.

179. Meyer auf'm Hofe, H., *Solving Rostering Tasks as Constraint Optimization*, in *Selected papers from the Third International Conference on Practice and Theory of Automated Timetabling, Springer Lecture Notes in Computer Science Volume 2079* E.K. Burke and W. Erben, Editors. 2000. pp. 191-212.

180. Meyers, C. and J.B. Orlin. *Very Large-Scale Neighborhood Search Techniques in Timetabling Problems*, in *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling*. E.K. Burke and H. Rudova, Editors. 2006. Brno, Czech Republic. pp. 36-52.

181. Millar, H.H. and M. Kiragu, *Cyclic and non-cyclic scheduling of 12 h shift nurses by network programming* European Journal of Operational Research, 1998. **104**(3): pp. 582-592.

182. Miller, H.E., W.P. Pierskalla, and G.I. Rath, *Nurse Scheduling Using Mathematical Programming.* Operations Research, 1976. **24**(5): pp. 857-870.

183. Mitchell, D.G., B. Selman, and H.J. Levesque. *Hard and Easy Distributions of SAT Problems*, in *Proceedings of the Tenth National Conference on Artificial Intelligence*. 1992. San Jose, CA: AAAI Press/MIT Press. pp. 459-465.

184. Mladenović, N. *A variable neighborhood algorithm - a new metaheuristic for combinatorial optimization*, in *Abstracts of papers presented at Optimization Days*. 1995. Montreal.

185. Mladenović, N. and P. Hansen, *Variable neighborhood search.* Computers and Operations Research, 1997. **24**(11): pp. 1097-1100.

186. Moscato, P., *On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms.* 1989, Caltech Concurrent Computation Program Report 826, California Institute of Technology.

187. Moscato, P. and C. Cotta, *A Gentle Introduction to Memetic Algorithms*, in *Handbook of metaheuristics*, F. Glover and G. Kochenberger, Editors. 2003, Kluwer. pp. 105-144.

188. Moz, M. and M.V. Pato, *An Integer Multicommodity Flow Model Applied to the Rerostering of Nurse Schedules* Annals of Operations Research, 2003. **119**: pp. 285-301.

189. Moz, M. and M.V. Pato, *Solving the Problem of Rerostering Nurse Schedules with Hard Constraints: New Multicommodity Flow Models* Annals of Operations Research, 2004. **128**: pp. 179-197.

190. Moz, M. and M.V. Pato, *A genetic algorithm approach to a nurse rerostering problem.* Computers & Operations Research, 2007. **34**: pp. 667–691.

191. Musa, A. and U. Saxena, *Scheduling nurses using goal-programming techniques.* IIE transactions, 1984. **16**: pp. 216-221.

192. Musliu, N., J. Gärtner, and W. Slany, *Efficient generation of rotating workforce schedules.* Discrete Applied Mathematics, 2002. **118**(1-2): pp. 85-98.

193. Musliu, N., A. Schaerf, and W. Slany, *Local search for shift design.* European Journal of Operational Research, 2004. **153**(1): pp. 51-64.

194. Nemhauser, G.L. and L.A. Wolsey, *Integer and Combinatorial Optimization* 1999: Wiley.

195. Nonobe, K. and T. Ibaraki, *A tabu search approach to the constraint satisfaction problem as a general problem solver.* European Journal of Operational Research, 1998. **106**(2-3): pp. 599-623.

196. Nooriafshar, M., *A heuristic approach to improving the design of nurse training schedules.* European Journal of Operational Research, 1995. **81**: pp. 50-61.

197. Okada, M., *An Approach to the Generalized Nurse Scheduling Problem - Generation of a Declarative Program to Represent Institution-Specific Knowledge.* Computers and Biomedical Research, 1992. **25**: pp. 417-434.

198. Okada, M. and M. Okada, *Prolog-based system for nursing staff scheduling implemented on a personal computer.* Computers and Biomedical Research, 1988. **21**: pp. 53-63.

199. Osman, I.H. and G. Laporte, *Metaheuristics: A bibliography* Annals of Operations Research, 1996. **63**(5): pp. 511-623.

200. Özcan, E. *Towards an XML based standard for Timetabling Problems: TTML*, in *Proceedings of First Multidisciplinary International Conference on Scheduling: Theory and Applications*. G. Kendall, E. Burke, and S. Petrovic, Editors. 2003. Nottingham, UK. pp. 566-570.

201. Özcan, E. *Memetic Algorithms for Nurse Rostering*, in *The 20th International Symposium on Computer and Information Sciences*. 2005: Springer-Verlag. pp. 482-492.

202. Ozkarahan, I., *A disaggregation model of a flexible nurse scheduling support system.* Socio-Economic Planning Sciences, 1991. **25**(1): pp. 9-26.

203. Ozkarahan, I. and J.E. Bailey, *Goal programming model subsystem of a flexible nurse scheduling support system.* IIE transactions, 1988. **20**(3): pp. 306-316.

204. Petrovic, S., G.R. Beddoe, and G. Vanden Berghe, *Case-based reasoning in employee rostering: learning repair strategies from domain experts*. 2002, Technical Report, Automated Scheduling Optimisation and Planning Research Group, School of Computer Science and Information Technology, University of Nottingham.

205. Petrovic, S., G.R. Beddoe, and G. Vanden Berghe. *Storing and adapting repair experiences in employee rostering*, in *Selected Papers from the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2002), Springer Lecture Notes in Computer Science*

Volume 2740. E.K. Burke and P. De Causmaecker, Editors. 2003. pp. 149-166.

206. Petrovic, S., Y. Yang, and M. Dror, *Case-based Selection of Initialisation Heuristics for Metaheuristic Examination Timetabling*. Expert Systems With Applications, 2007 (to appear). **33**(3).

207. Poissonnet, C.M. and M. Véron, *Health effects of work schedules in healthcare professions.* Journal of Clinical Nursing, 2000. **9**: pp. 13-23.

208. Post, G. and B. Veltman. *Harmonious Personnel Scheduling*, in *Proceedings of the 5th International Conference on the Practice and Automated Timetabling (PATAT 2004)*. E.K. Burke and M. Trick, Editors. 2004. Pittsburgh, PA. USA. pp. 557-559.

209. Punnakitikashem, P., J.M. Rosenberger, and D.B. Behan, *Stochastic Programming for Nurse Assignment* 2005, Technical Report COSMOS 05-01, The University of Texas at Arlington. Arlington, TX.

210. Purnomo, H.W. and J.F. Bard, *Cyclic preference scheduling for nurses using branch and price.* Naval Research Logistics, 2007. **54**(2): pp. 200-220.

211. Qu, R., *Benchmark Data Sets in Exam Timetabling (http://www.cs.nott.ac.uk/~rxq/data.htm retrieved 27-June-2007)*. 2007.

212. Qu, R., E. Burke, B. McCollum, L.T.G. Merlot, and S.Y. Lee, *A Survey of Search Methodologies and Automated Approaches for Examination Timetabling. Computer Science Technical Report No. NOTTCS-TR-2006-4*. 2006, School of Computer Science and Information Technology. University of Nottingham.

213. Randhawa, S.U. and D. Sitompul, *A heuristic-based computerized nurse scheduling system.* Computers and Operations Research, 1993. **20**(9): pp. 837-844.

214. Rego, C., *A Subpath Ejection Method for the Vehicle Routing Problem.* Management Science, 1998. **44**(10): pp. 1447-1459.

215. Reinelt, G., *TSPLIB95 (http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/ retrieved 27-June-2007).* 2007.

216. Rosenbloom, E.S. and N.F. Goertzen, *Cyclic nurse scheduling.* European Journal of Operational Research, 1987. **31**: pp. 19-23.

217. Ross, P., *Hyper-heuristics,* in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, E.K. Burke and G. Kendall, Editors. 2005, Springer. pp. 529-556.

218. Rousseau, L.-M., G. Pesant, and M. Gendreau, *A General Approach to the Physician Rostering Problem.* Annals of Operations Research, 2002. **115**: pp. 193-205.

219. Sastry, K., D. Goldberg, and G. Kendall, *Genetic Algorithms*, in *Search Methodologies. Introductory Tutorials in Optimization and Decision Support Techniques* E.K. Burke and G. Kendall, Editors. 2005, Springer. pp. 97-126.

220. Schaerf, A. and L. Di Gaspero. *Measurability and Reproducibility in Timetabling Research: State-of-the-Art and Discussion*, in *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling*. E.K. Burke and H. Rudova, Editors. 2006. Brno, Czech Republic. pp. 53-62.

221. Schaerf, A. and A. Meisels. *Solving Employee Timetabling Problems by Generalized Local Search*, in *Proceedings of the 6th Congress of the Italian Association for Artificial Intelligence on Advances in Artificial Intelligence*. 1999: Springer-Verlag. pp. 380-389.

222. Scott, S. and R. Simpson. *Case-Bases Incorporating Scheduling Constraint Dimensions - Experiences in Nurse Rostering*, in *Proceedings of the 4th European Workshop on Advances in Case-Based Reasoning, Lecture Notes In Computer Science; Vol. 1488*. 1998: Springer-Verlag. pp. 392-401.

223. Siferd, S.P. and W.C. Benton, *Workforce Staffing and Scheduling : Hospital Nursing Specific Models.* European Journal of Operational Research, 1992. **60**: pp. 233-246.

224. Silvestro, R. and C. Silvestro, *An evaluation of nurse rostering practices in the National Health Service.* Journal of Advanced Nursing, 2000. **32**(3): pp. 525-535.

225. Sitompul, D. and S.U. Randhawa, *Nurse scheduling models: a state-of-the-art review*. Journal of the Society of Health Systems, 1990. **2**(1): pp. 62-72.

226. Smith, L.D., D. Bird, and A. Wiggins, *A computerized system to schedule nurses that recognizes staff preferences.* Hospital & health services administration, 1979. **24**(4): pp. 19-35.

227. Smith, L.D. and A. Wiggins, *A Computer-Based Nurse Scheduling System.* Computers and Operations Research, 1977. **4**(3): pp. 195-212.

228. Spyropoulos, C.D., *AI planning and scheduling in the medical hospital environment.* Artificial Intelligence in Medicine, 2000. **20**(2): pp. 101-111.

229. Tanomaru, J. *Staff scheduling by a genetic algorithm with heuristic operators*, in *Proceedings of the IEEE Conference on Evolutionary Computation*. 1995. pp. 456-461.

230. Taylor, P.E. and S.J. Huxley, *A Break from Tradition for the San Fransisco Police: Patrol Officer Scheduling Using an Optimization-Based Decision Support System.* Interfaces, 1989. **19**(1): pp. 4-24.

231. Thompson, G.M., *A simulated-annealing heuristic for shift scheduling using non-continuously available employees.* Computers and Operations Research, 1996. **23**(3): pp. 275-288.

232. Thornton, J. and A. Sattar. *Nurse Rostering and Integer Programming Revisited*, in *International Conference on Computational Intelligence and Multimedia Applications*. B. Verma and X. Yao, Editors. 1997. pp. 49-58.

233. Tien, J.M. and A. Kamiyama, *On manpower scheduling algorithms.* SIAM Review, 1982. **24**(3): pp. 275-287.

234. Trivedi, V.M. and M.D. Warner, *A Branch and Bound Algorithm for Optimum Allocation of Float Nurses.* Management Science, 1976. **22**(9): pp. 972-981.

235. Tsang, E., J. Ford, P. Mills, R. Bradwell, R. Williams, and P. Scott, *Towards a Practical Engineering Tool for Rostering.* Annals of Operational Research, Accepted for publication, January 2006.

236. UK Government Statistical Service, *Staff in the NHS 2006 (available online at http://www.ic.nhs.uk/statistics-and-data-collections/workforce/nhs-staff-numbers retrieved 18-Jun-2007)*. 2006.

237. Valouxis, C. and E. Housos, *Hybrid optimization techniques for the workshift and rest assignment of nursing personnel.* Artificial Intelligence in Medicine, 2000. **20**: pp. 155-175.

238. Vanden Berghe, G. *An Advanced Model and Novel Meta-Heuristic Solution Methods to Personnel Scheduling in Healthcare*. Ph.D. Thesis, University of Gent, Belgium, 2002.

239. Vanhoucke, M. and B. Maenhout, *Characterisation and generation of nurse scheduling problem instances*. 2005, Vlerick Leuven Gent Management School.

240. Vanhoucke, M. and B. Maenhout, *NSPLib–A Nurse Scheduling Problem Library: A tool to evaluate (meta-) heuristic procedures*. 2005.

241. Voss, S., S. Martello, I.H. Osman, and C. Roucairol, Editors. *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. 1999, Kluwer Academic Publishers.

242. Warner, D.M., *Scheduling Nursing Personnel according to Nursing Preference: A Mathematical Programming Approach* Operations Research, 1976. **24**: pp. 842-856.

243. Warner, D.M. and J. Prawda, *A Mathematical Programming Model for Scheduling Nursing Personnel in a Hospital* Management Science, 1972. **19**(4): pp. 411-422.

244. Warner, M., B.J. Keller, and S.H. Martel, *Automated Nurse Scheduling.* Journal of the Society for Health Systems, 1990. **2**(2): pp. 66-80.

245. Weil, G., K. Heus, P. Francois, and M. Poujade, *Constraint programming for nurse scheduling.* IEEE Engineering in Medicine and Biology Magazine, 1995. **14**(4): pp. 417-422.

246. Wolpert, D.H. and W.G. Macready, *No Free Lunch Theorems for Optimization.* IEEE Transactions on Evolutionary Computation, 1997. **1**(1): pp. 67-82.

247. Wong, G.Y.C. and H.W. Chun. *Nurse Rostering Using Constraint Programming and Meta-level Reasoning*, in *Developments in Applied Artificial Intelligence: 16th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*. 2003. Loughborough, UK: Springer. pp. 712-721.

248. Wright, P.D., K.M. Bretthauer, and M.J. Cote, *Reexamining the Nurse Scheduling Problem: Staffing Ratios and Nursing Shortages.* Decision Sciences, 2006. **37**(1): pp. 39-70.

249. Yagiura, M., T. Yamaguchi, and T. Ibaraki, *A Variable Depth Search Algorithm for the Generalized Assignment Problem*, in *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, S. Voss, et al., Editors. 1999, Kluwer Academic Publishers: Boston, MA. pp. 459-471.