Faculty of Engineering and Applied Science
Griffith University Gold Coast, Queensland

AN ENHANCED CYCLIC DESCENT ALGORITHM FOR NURSE ROSTERING

by

John Richard Thornton, BBus

A Thesis submitted as partial fulfilment for
the degree of Bachelor of Science (Honours)

Date: 31st October 1995

ABSTRACT

The study introduces an enhanced cyclic descent algorithm for nurse rostering. The algorithm is compared to four other rostering algorithms and to manually generated roster solutions obtained from the Gold Coast Hospital. Three criteria are developed with which the roster generation methods are assessed: these are roster schedule quality, roster shift allocation quality and execution time. A statistical analysis shows that the enhanced cyclic descent algorithm has the best overall performance. An integer linear programming algorithm and an enhanced simulated annealing algorithm are also shown to perform well with smaller problems.

ACKNOWLEDGMENTS

The inspiration and much of the expert knowledge required for this study has come from my wife Nicolette, who works as a nurse at the Gold Coast Hospital. It is she who I first acknowledge, both for providing the raw material for the study and for her loving support during it's execution. I also wish to thank and acknowledge the following people, without whose help and advice the research would not have been possible:

Mr Colin Thorne

Ms Ann Nguyen

Dr Clyde H. Wild

Mr Marcus Randall

Mr Paul Kachel (Director of Nursing, Gold Coast Hospital)

Ms Alexandra Cherry (Clinical Nurse Consultant, Gold Coast Hospital)

Ms Anita Petrov (Nursing Supervisor, Gold Coast Hospital)

TABLE OF CONTENTS

Appendices

List of Tables

List of Figures

ABBREVIATIONS

AI	Artificial Intelligence
ANSI	American National Standards Institute
CN	Clinical Nurse
CNC	Clinical Nurse Consultant
DSS	Decision Support System
DV	Dependent Variable
EN	Enrolled Nurse
DOS	Disk Operating System
IBM®	International Business Machines Corporation
ILP	Integer Linear Programming
IV	Independent Variable
MANOVA	Multiple Analysis of Variance
MS-DOS®	MicroSoft® Disk Operating System
PC	Personal Computer
RAM	Random Access Memory
RN	Registered Nurse
SA	Simulated Annealing

STATEMENT OF ORIGINALITY

The material presented in this thesis has not been previously submitted for a degree or diploma in any university, and to the best of my knowledge contains no material previously published or written by another person except where due acknowledgment is made in the thesis itself.

John Richard Thornton

Chapter 1: Introduction

1.1 The Importance of Nurse Allocation Decisions

Two critical and conflicting objectives in the running of a hospital are the minimisation of costs and the provision of adequate patient care. Typically, nursing salaries form the largest item in a hospital budget (Sitompul 1992). The number and skill level of nurses assigned to a hospital ward is also a primary determinant of the quality of patient care. Nurse allocation decisions are therefore a central issue in hospital management, and the generation of nurse rosters is one of the main tasks of nurse allocation.

In addition, nursing personnel are a scarce resource (Hung 1995). Most developed countries experience a nursing shortage, and are required to recruit nurses from overseas. There are high turnover rates in nursing staff, and this is in part attributable to the unsocial hours nurses are expected to work. Nurse allocation policies have a direct impact on nurse satisfaction, and hence on turnover (Kostreva and Genevier 1989). The difficulty in replacing nursing staff means that attention needs to be paid to producing work schedules that reflect nurse preferences.

Health and safety considerations are an important factor in nurse allocation. Schedules requiring nurses to work long stretches without days off, or frequently alternating patterns of unsocial hours, can result in stress, exhaustion, inadequate care, absenteeism and staff turnover (Sandhu *et al.* 1992). The same effects can be caused by understaffing a hospital ward relative to the patient load. The consequences of inadequate patient care can be serious. In emergency situations, the observational skills and speed of response of nursing staff can mean the difference between life and death for a patient. Nursing staff are also responsible for the correct and regular administration of dangerous drugs. It is therefore important that a ward is adequately staffed, and that the staff are sufficiently rested between each period of duty.

To summarise, the nurse allocation process has an impact on three areas of hospital management:

- The quality and safety of patient care
- The distribution of the hospital budget
- The satisfaction and turnover of the nursing staff

1.2 Nurse Rostering within the Nurse Allocation Process

The nurse allocation process has been divided into four stages (Warner 1976):

1. The *long term allocation* of nurses to hospital wards or units, based on funding levels and predictions of the expected demand for nursing care.
2. The *medium term allocation* of when each nurse will be on or off duty, resulting in the creation of a nurse roster.
3. The *daily allocation* of additional 'floating' or 'pooled' staff to cover for unforeseen absenteeism and fluctuations in demand.
4. The *hour by hour allocation* of tasks and patients to individual nurses.

The current study is concerned with Stage 2, the specification of when each nurse on a particular ward will be on or off duty. This specification results in the creation of a nurse roster. A roster defines shift duties for a fixed time period, which can range from one week to several months. The roster not only defines the patterns of shift types and days off each nurse has to work, but also the total number of nurses working each shift of each day. The form and definition of a roster is further explained in Appendix 1.

Nurse Rostering as a Separate Issue. The four stages of the nurse allocation process are interrelated. The tasks performed in a ward (stage four), define the number of nurses required for the ward, and the number of nurses required for a shift (stages one, two and three). Nevertheless, each stage of the nurse allocation process is separated in time, with the output of one process becoming the input for another. Given that the other stages of the problem have been defined, then an individual stage can be considered in isolation. This is reflected in hospital policy. Typically, the allocation of staff to a ward is a centralised administrative decision (stages one and three), whilst the rostering of nurses and allocation of nursing tasks (stages two and four) are performed at a ward level. On the basis of this division, the current study considers nurse rostering separately from the other allocation stages.

1.3 The Complexity of Nurse Rostering

Nurse rostering is a complex problem. Given a hospital ward employing twenty-five full-time nurses, and providing round-the-clock nursing care, there are 2^{700} possible combinations of nurses and shifts for a two week period[1]. Using current computer technology, an exhaustive search of these possibilities is infeasible. However, as with other complex scheduling problems, the majority of solutions can be eliminated by applying rules associated with the problem constraints. For instance, there must be a minimum number of staff on duty during each shift, and there are legal limits to the number of consecutive shifts a nurse can work without a day off. Even given the application of such rules, realistic nurse rostering scenarios are still too complex to be solved by an exhaustive search methodology.

Sitompul (1992) notes that nurse rostering shares much in common with other difficult staff scheduling problems such as police station, fire station and telephone exchange staffing. All these problems require staff to be on duty 24 hours a day and seven days a week, with fluctuating daily demand for services and fixed regulations as to acceptable work patterns. However, the nurse rostering problem is further distinguished by the following features:

- **Multiple minimum staffing levels**: There can be four or more grades of nursing staff, each with a different skill level. Legal controls limit the tasks each grade of nurse can perform. Consequently, each shift can have a minimum staffing requirement for each grade of nurse.

[1] Given there are three shift types that can be worked in a day, a nurse can work any one of these shifts, or alternatively have a day off. Therefore, there are 4 possible states that a nurse can be in on a particular day. Over a 14 day period, this means there are 4^{14} (or 2^{28}) possible combinations of shifts and days off for one nurse. If a roster is to be calculated for 25 nurses, this means there are $2^{28x25} = 2^{700}$ possible rosters.

- **Desired staffing levels**: Beyond the provision of minimum staffing levels there are also desired staffing levels which should be met as often as possible.
- **Nurse preferences**: Due to the importance of maintaining nurse satisfaction and reducing turnover, schedules should reflect a nurse's preferences for shift patterns and days off.
- **Flexible rostering**: In order to meet changing nurse requests for particular days off, a roster should not be fixed or imposed. This means a new roster needs to be calculated in each rostering period, rather than rotating duties within an existing roster.

The main feature that emerges from these points, and that sets nurse rostering apart from other scheduling problems, is that nurse rostering has *multiple* objectives (Ozkarahan and Bailey 1988). Other sophisticated problems, such as the aircrew scheduling problem, usually have a single objective of minimising costs, after the basic constraints have been met (Graves *et al.* 1993, Hoffman and Padberg 1993). However, nurse rostering involves minimising nurse dissatisfaction with the roster, *and* minimising deviations from desired staffing levels. These two objectives can then be decomposed into a series of sub-objectives (see Appendix 3, Section A3.3).

The constraints and multiple objectives of the nurse rostering problem make it unique within the domain of staff scheduling. The situation is further complicated by the existence different policies and circumstances within different hospitals and on different wards. This has meant that existing solutions to the problem have not been widely applied[2] (Sitompul 1992). In the next chapter, existing approaches to nurse rostering are considered in more detail, through a review of the current literature.

[2]As an example of a commercial application, PolyOptimum®, a US based company owned by Microsoft®, have produced a hospital staffing, scheduling and productivity monitoring system called ProAct®. This product has gained some acceptance in NSW hospitals. Interviews conducted with nursing staff who have used the system, have produced mixed results. Whilst the system is preferred to a return to manual rostering, doubts were expressed that the original investment in the product was justified. Claims for reductions in staffing costs have not conclusively materialised, and many rosters produced by the system require extensive manual alterations.

Chapter 2: Literature Review

Computerised nurse rostering has been of interest to researchers for over twenty years. Due to the nature of the problem, the literature has tended to be more applied than theoretical. Many publications have arisen from the implementation of working hospital systems (Warner 1976). Other researchers have used the expertise and data available from the health care industry to test new rostering approaches (Ozharahan and Bailey 1988). As new computing techniques have developed, this has also been reflected in the rostering literature. The 1970s saw the use of linear and integer programming techniques, whilst the 1980s introduced the use of goal programming, decision support systems, expert systems and knowledge-based systems. More recently, researchers have given greater consideration to the users of nurse rostering systems (Kostreva and Jennings 1991) and to the development of a more flexible and generic approach to nurse rostering (Sitompul 1992).

The current chapter provides a literature review of computerised approaches to nurse rostering. Reference is made to relevant research in other areas of staff scheduling. Through an analysis of the literature, a background to the nurse rostering problem is given, and the area of intended research is introduced.

Firstly, the two basic approaches to nurse rostering covered in the literature are discussed: these are cyclic and non-cyclic rostering.

2.1 Cyclic vs Non-Cyclic Rostering

Cyclic Rostering: Cyclic nurse rostering involves generating a fixed roster that can satisfy staff requirements, without considering individual nurse requests. Nurses are then assigned schedules within the roster. The roster remains the same in each successive rostering period, with nurses being assigned different schedules within the roster. In this way a nurse will ‘cycle’ through the various schedules in the roster. Howell (1966) and

Frances (1966) laid down some basic principles for manual cyclic rostering. Howell's work was further extended and applied by Megeath (1978). In addition, Rosenbloom and Goertzen (1987) developed a computer algorithm for the generation of cyclic rosters.

Non-Cyclic Rostering: A non-cyclic roster is reformulated before each rostering period, with each schedule in the roster being matched to a particular nurse. This is done to accommodate individual nurse preferences and to allow for fluctuations in the number and type of staff assigned to a ward.

Advantages of Cyclic Rostering: The main advantage of a cyclic roster is that it can be used repeatedly during successive rostering periods. One roster can therefore be used for several months or even years. Due to this infrequent calculation, it can be more cost effective to use human expertise to generate cyclic rosters, than to develop an automated solution (Megeath 1978). In addition, a well designed cyclic roster can result in better overall roster quality, and a fairer distribution of schedules (Smith and Wiggins 1977). This is because *non*-cyclic rosters try to include nurse's special requests. This will usually result in longer work stretches and a more unbalanced distribution of shift types than would otherwise be necessary. Cyclic rosters can also incorporate the principles of circadian rhythms (Kostreva and Genevier 1989). Using this approach, nurses are given schedules that minimise physical and psychological stress caused by changing shift patterns.

Disadvantages of Cyclic Rostering: The basic problem with cyclic rostering is a lack of flexibility (Smith and Wiggins 1977). A nurse requiring a particular day off, which is not granted in the roster, must make arrangements to exchange shifts with another nurse of the same level. This may not always be possible. Nurses may also be unable to obtain their preferred holiday periods. Changes in the numbers of staff in a ward will require complicated revisions of the roster. In a ward with frequent changes in personnel and a fluctuating patient load, cyclic rostering may prove as complicated as non-cyclic rostering.

More attention has been paid in the literature to the computerised generation of non-cyclic rosters, than to the generation of cyclic rosters. This is due to the greater complexity of non-cyclic rosters and to the large and repeated investment of human effort required in their creation.

2.2 Non-Cyclic Approaches to Nurse Rostering

2.2.1 Mathematical Programming Approaches to Rostering

A mathematical programming approach to nurse rostering involves the construction of a mathematical model of the problem. This typically means the definition of an objective function or functions, and the creation of a series of constraints. Then, using a suitable computational technique, the value of the objective function is either maximised or minimised (Papadimitriou and Steiglitz 1982). A mathematical programming approach to nurse rostering is illustrated in Appendix 2.

The existing mathematical programming literature on nurse rostering can be divided into four categories according to the computational techniques employed. These techniques are linear programming, integer programming, goal programming and local search:

2.2.1.1 Linear Programming

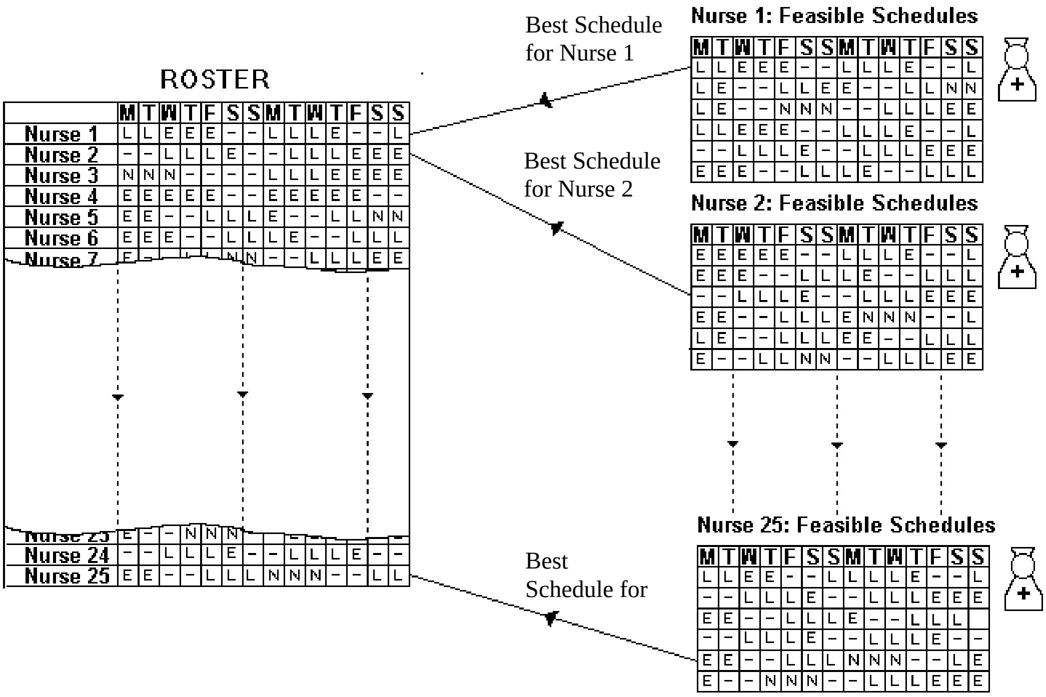
Linear programming is an algorithmic technique for finding the optimum solution to a constrained minimisation or maximisation problem (Taha 1992). Existing computerised linear programming applications can solve problems of equivalent size to the rostering problem. However, linear programming solutions are usually non-integral. Rosters asking nurses to work 0.43 of a schedule, or 1.23 of a shift have no practical meaning. Nevertheless, linear programming techniques have been successfully applied to certain staffing problems.

An early paper on the scheduling of hospital housekeeping staff was produced by Rothstein in 1973. Whilst not directly applicable to nurse rostering, Rothstein's work does present a simple method for solving the "days off" problem using linear programming. The objective of the model is to maximise the number of consecutive two day off periods granted to staff working a five day week. The problem is formulated in such a way that integer solutions are guaranteed. Baker (1976) shows that if the staff scheduling problem can be reduced to a special form (as a network), then all solutions will be integral. However, subsequent research into nurse rostering has been unable to formulate a more complete rostering problem as a network linear program.

2.2.1.2 Integer Programming

Integer programming techniques are designed to find optimal solutions to linear programming problems which have integer variable restrictions. However, integer programming algorithms are computationally expensive, and models with large numbers of variables soon become too time consuming to solve (Chow and Hui 1993). For this reason, standard integer programming approaches to the nurse rostering problem have not been popular. Instead, researchers have concentrated on developing specialised integer programming algorithms that exploit the features of the nurse rostering problem:

Multiple-Choice Programming: Warner (1976) uses a multiple-choice programming algorithm (Healy 1964) to solve a nurse rostering problem in the University of Michigan Hospital. The problem is expressed as one of finding the best combination of feasible nurse schedules (see Appendix 1, Section A1.3). Firstly, a set of feasible schedules is generated for each nurse. These schedule sets are then combined until the best staffing levels for the complete roster are found. In a second phase, the algorithm calculates the best combination of schedules according to nurse preferences. In both phases, a multiple-choice algorithm uses a linear programming method to arrive at an initial solution and then searches for the best integer solution. The basic principles of Warner's method are illustrated in the following diagram:



Key: M = Monday, T = Tuesday, etc

E = Early Shift, L = Late Shift, N = Night Shift, - = Day Off

Figure 1: Warner's feasible schedule approach to nurse rostering

Warner reports that the algorithm could solve problems with up to 400 variables within 20 to 40 seconds, using an IBM® 360/67. According to IBM® staff[3], a modern day 486 microprocessor should also be able to process Warner's algorithm. Therefore, earlier criticisms that Warner's solution required excessive computing resources, are no longer relevant (Rosenbloom and Goertzen 1987).

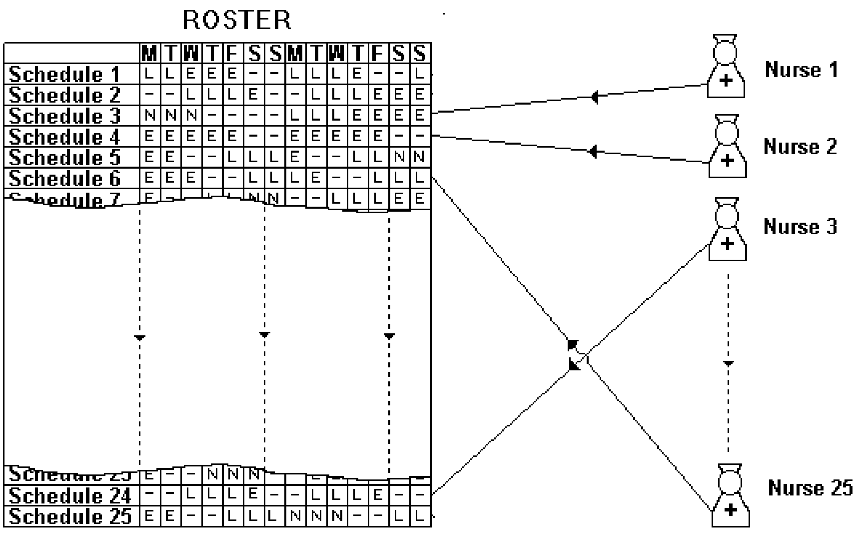
However, Warner assumes a nurse will have 10 to 20 feasible schedules per two week roster period. In Appendix 4, it is shown that nurses operating under a US rostering policy can have up to 180 feasible schedules per two week roster (in an Australian hospital, the number of feasible schedules per nurse can grow to several thousand). Warner's approach therefore relies on a degree of simplification before the problem is presented to the algorithm.

Mixed-Integer Programming: Following on from Warner's work, Kostreva *et al.* (1978) developed a mixed-integer programming formulation of the nurse rostering problem.

[3]This information was obtained via a customer service telephone call to IBM®'s Sydney offices.

Subsequently, Kostreva and Jennings (1992) used a revised version of this approach to solve nurse scheduling problems on a microcomputer. In both pieces of research, the same model is used.

Firstly, the problem is broken down into two phases. The first phase involves heuristically generating a complete roster that fulfils all the constraints of the problem. If possible, the roster meets all the staffing requirements for each shift and provides nurse schedules that meet or exceed minimum standards. When a nurse requires days off in a roster, then at least one schedule in the roster will have those days off. By providing nurses with questionnaires, a matrix of "hate points" for each nurse in relation to each schedule is calculated (Kostreva *et al*. 1978, p. 287). The second phase of the approach uses a mixed-integer programming technique to assign schedules in the roster to individual nurses. The objective of phase two is to minimise the total "hate point" score. The algorithm iterates between phase one and phase two, generating a new roster with each iteration. After running for a fixed period of time, the solution with the lowest aggregate "hate point" score is selected. The phase two assignment problem is illustrated in the following diagram:



Key: M = Monday, T = Tuesday, etc

E = Early Shift, L = Late Shift, N = Night Shift, - = Day Off

Figure 2: Kostreva et al.'s assignment approach to nurse rostering

As with Warner's study (1976), Kostreva *et al.'s* solution is problem specific, and focuses on a US nurse rostering situation. Whilst Warner's solution attempts to find an optimal overall solution, Kostreva *et al.* optimise the second phase assignment of schedules to nurses. The initial rosters, from which the assignments are made, represent feasible or "good enough" solutions to the problem. Therefore, Kostreva *et al.'s* final solutions are also feasible, but not necessarily optimal. The quality of Kostreva *et al.'s* solution depends on the roster generating heuristic. This heuristic is not specified in detail, and it's performance is not comparatively tested. For these reasons, the overall quality of the rosters generated by Kostreva *et al.'s* approach cannot be fully assessed.

2.2.1.3 Goal Programming

Goal programming is a particular case of linear (and integer) programming. In a goal programming model, different goals can be maximised or minimised, either simultaneously or in a specified order. The special feature of goal programming is that the relative importance of goals can be changed and ranked according to the preferences of the user. However, this flexibility is bought at the cost of greater computational complexity (Hillier and Lieberman 1990).

In the 1980s, criticisms arose that previous mathematical formulations of the nurse roster problem had been too inflexible (Arthur and Ravindran 1981, Musa and Saxena 1984, Ozharahan and Bailey 1988). Most early models had a fixed set of goals, typically defined by the authors of the various studies (e.g. Warner 1976, Miller *et al*. 1976). With the advent of goal programming techniques, a series of studies looked afresh at the nurse rostering problem.

Arthur and Ravindran (1981) use goal programming in a two-phase approach to the rostering problem. They define four goals for the model as:

1. Minimising deviations from minimum staffing levels
2. Minimising deviations from desired staffing levels
3. Meeting nurses' preferences
4. Meeting nurses' special requests

The first phase of the model uses goal programming to allocate the days off for the nursing staff. Individual schedules are built for each nurse using a zero-one integer programming approach. The second phase uses a heuristic procedure to allocate individual shift types. The problem is constrained by considering full-time nurses working a seven day roster period, and also by having a fixed weekend off policy. This results in each nurse having five feasible day on, day off schedules.

Musa and Saxena (1984) developed a similar zero-one goal programming model. This time their approach considers a 14 day roster, with part-time staff and a flexible week-end off policy. The problem is simplified by considering a limited number of nurses, and one shift type. This results in a 154 variable model, with 120 constraints.

Finally, Ozharahan and Bailey (1988) developed a goal programming approach to nurse rostering as part of a planned flexible decision support system. This study differs from the previous work in the area by allocating both eight and ten hour shift lengths. Integer programming techniques are used to solve the basic problem, with the allocation of starting times for individual shifts being decided in a separate heuristic procedure. As with the studies by Arthur and Ravindran (1981), and Musa and Saxena (1984), the problem complexity is constrained. This is achieved by considering a seven day rostering period, with a single grade of nurse.

In comparison to the previous studies on integer programming (Warner 1976, Kostreva *et al.* 1978), the goal programming literature has used relatively simple rostering models. This raises the question of the applicability of the goal programming techniques for more complex problems. Chow and Hui (1993) report that standard integer programming techniques, such as those used in the goal programming approaches, are unable to solve large rostering problems. Although Arthur and Ravindran (1981) discuss the extension of their model through the relaxation of constraints, the limitations of integer programming methods in more complex situations are not directly raised in the goal programming literature.

2.2.1.4 Cyclic Descent Local Search Techniques

Local search techniques use a trial and error method to find solutions to a given problem. These techniques are usually fast relative to linear and integer programming techniques. However, as the name implies, local searches find only the best of a subgroup of solutions. There is no guarantee that a local search will find the best overall solution, because the total search space is not fully explored.

The structure of a simple local search is given by the following algorithm (Papadimitriou and Steiglitz 1982):

```
local search
{
        best solution = selected starting solution
        while improve(best solution) < > ‘no’
                best solution = improve(best solution)
        return best solution
}
```

where:

$$\text{improve(best solution)} = \begin{cases} \text{any new solution within the search space such that} \\ \text{cost(new solution)} < \text{cost(best solution)} \\ \\ \text{'no' if no lesser cost solution exists} \end{cases}$$

Figure 3: A simple local search algorithm

The cost function used in figure 3 is analogous to the objective function in linear programming. It evaluates a solution in terms of the search objective(s) and returns a quantifiable cost.

Miller *et al*. (1976) used a local search technique in a mathematical programming approach to nurse rostering. A cyclic descent algorithm was employed to produce rosters by combining feasible nurse schedules (see Section 4.3). The algorithm starts by constructing a roster using one schedule for each nurse, from each nurse’s set of feasible schedules. Then holding all other schedules in the roster constant, all feasible schedules for the first nurse are tried in the roster. A cost function produces an overall roster score for each schedule in terms of how well the staffing levels are met, and the overall quality of schedules allocated. The schedule having the lowest cost is inserted into the roster, then the next nurse in the roster is selected and all feasible schedules for that nurse are tried, the best one inserted and so on. The algorithm *cycles* through each nurse, returning to the first nurse and repeating the process until no further improvement or *descent* in the cost function is found. Miller *et al.* applied the technique to a relatively small problem involving the rostering of days off for twelve nurses.

A difficulty with local search approaches is judging the quality of the solutions generated. For this reason, Miller *et al*. perform a series of tests on their algorithm. Firstly, a comparison is made with an integer programming algorithm for a small problem. Secondly, the deviations from the desired staffing levels, and the quality of schedules generated are graphically analysed. Finally, comparisons are made between the algorithm and manually generated rosters. Generally favourable results are reported for the algorithm in each test.

Blau and Sear (1983) applied a cyclic descent approach to another nurse rostering problem. Again, a simple day on, day off assignment of shifts is required. The study reports the successful implementation of the algorithm on a microcomputer, but does not evaluate the quality of the rosters generated.

2.2.2 Artificial Intelligence Approaches to Nurse Rostering

The Artificial Intelligence Domain: The domain of Artificial Intelligence (AI) encompasses a variety of techniques. The practical objective of AI research has been to develop computer programs that can exhibit intelligent and adaptive behaviour. In the area of scheduling, this has resulted in the creation of expert systems (Turban 1990), neural networks (Carling 1992), genetic algorithms (Goldberg 1989) and fuzzy logic systems (Kosho 1992).

AI criticisms of Mathematical Programming: With the emergence of computerised linear programming algorithms, and the further development of integer and goal programming, mathematical programming approaches initially dominated the area of scheduling research (for instance, see Baker 1974). More recently, Artificial Intelligence researchers have turned their attention to scheduling problems. In doing so, they have recognised two main problems with the previous approaches to scheduling:

1. Realistic scheduling problems, and specifically rostering problems, are often too complex to be solved directly using mathematical programming techniques (Chow and Hui 1993).
2. Mathematical formulations of a problem tend to be inflexible. Unlike human experts, mathematical algorithms are unable to adjust and balance conflicting requirements and constraints (Dhar and Ranganathan 1992).

The need for flexible systems, with reasoning capabilities, has lead to the application of several AI techniques to the scheduling domain. Johnston and Adorf (1992) used a neural network approach in an application to schedule observations from the Hubble Space Telescope. They also report the use of neural networks in aircrew training scheduling and school timetable construction. Genetic algorithms have been used for timetable scheduling (Colorni *et al.* 1991) and job shop scheduling (Biegel and Davern 1990). In addition, Griffith University academic, Suresh Hugenahally, is using a fuzzy logic approach to solve an applied staff scheduling problem (private communication, March 1994). However, in the area of nurse rostering, the most relevant research has focused on the use of knowledge-based and expert systems.

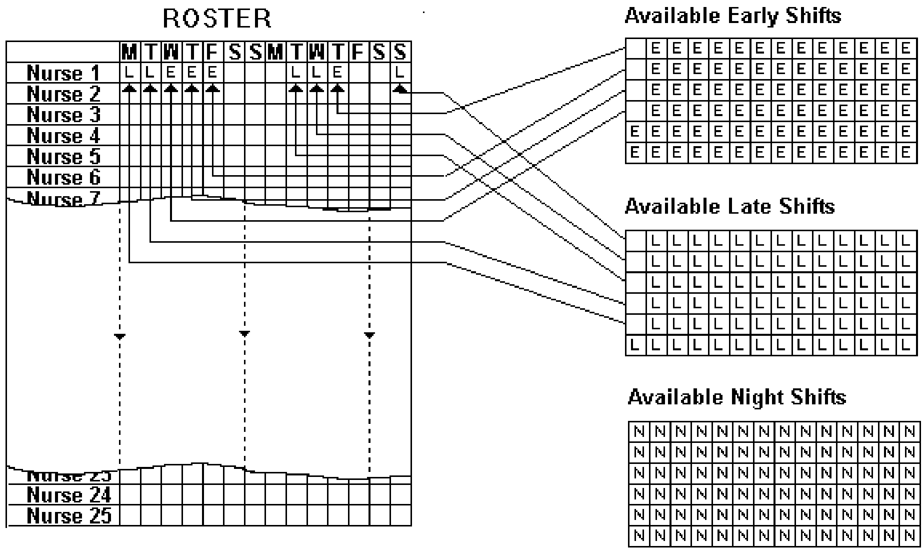
2.2.2.1 Knowledge-Based and Expert Systems

Chow and Hui (1993) report on a knowledge-based system for rostering aircraft maintenance personnel. The intent of the research is to develop a generalised approach to staff scheduling, based on the imitation of human reasoning processes. It is therefore relevant to the nurse rostering problem. In addition, Sitompul (1992, pp. 25-27) reports on Lukman’s 1986 thesis, which uses an expert system approach to develop a nurse rostering application.

Hui (1988) distinguished a knowledge-based system from an expert system by the depth of knowledge required in the problem domain: "Since the knowledge of roster scheduling is not up to expert level, the term knowledge-based system is used" (1988, p. 32). Both expert system and knowledge-based approaches to problem solving first involve extracting human knowledge from a problem domain. This knowledge is then symbolically expressed as a *knowledge base*. Typically, a computer program, known as an inference engine, will operate on a knowledge base, in order to prove or disprove a given hypothesis or goal (Hui 1988). An important distinction is that the processing algorithm (i.e. inference engine) is entirely independent of the particular problem, as expressed in the knowledge base. This differs from standard heuristic programming approaches, where the knowledge about a problem is integrated in the algorithm (Colin Thorne, private communication, October 1994).

Hui recognises that human knowledge about rostering problems is not just a series of static rules, but involves the application of *control strategies* (1988, p. 35). Control strategy knowledge is knowledge about how to approach a problem (i.e. it is procedural). Most expert systems have a predetermined control strategy such as forward and backward chaining. Hui's approach involves using multiple control strategies combined into a "conceptual inference engine" (1988, p. 38). This approach allows a developer to use a combination of control strategies to solve a particular problem. The knowledge base of Hui's system is divided into rules, constraints and heuristic operators. The heuristic operators represent the various control strategies available. Hui's aim is to provide a general purpose set of building blocks from which different types of rostering problems can be modelled and solved.

As with the goal programming approaches to rostering, Hui allocates individual shifts to staff. Schedules are built shift by shift, via the application of rules and constraints, until a complete roster is generated. The roster is then improved using a process of shift swapping. The initial shift by shift construction of a roster is illustrated in the following diagram:



Key: M = Monday, T = Tuesday, etc

E = Early Shift, L = Late Shift, N = Night Shift, - = Day Off

Figure 4: A shift by shift assignment approach to nurse rostering

The benefit of the knowledge-based approach is that constraints can be relaxed as needed. The system is capable of considering many constraints and assigning different priorities to each one. Large problems can be solved because computationally expensive mathematical algorithms are not necessary.

The price of this flexibility is that a knowledge-based system will find a satisficing, rather than optimal, problem solution. Hui does not measure the quality of rosters generated, so comparisons with other approaches are not possible. Also, it is difficult to extract and symbolically express all the expert knowledge in a given problem domain (Turban 1990). For this reason, heuristic operators are only likely to represent a subset of the rules a human expert would apply.

2.2.3 Heuristic Approaches to Nurse Rostering

Whilst the papers previously discussed have included heuristic techniques, these techniques have been secondary to the major theme of the research (e.g. Hui 1988, Kostreva and Jennings 1991). In addition to these papers, research has been conducted into nurse rostering that is based exclusively on heuristic approaches. Firstly, Smith and Wiggins (1977) developed a heuristic nurse rostering application for a US hospital. Their objective was to provide a “rough” solution “to supplement the judgement of the scheduling clerks” (Smith and Wiggins 1977, p. 198).

Secondly, Randhawa and Sitompul (1993) report on a heuristic-based decision support system for nurse scheduling. This research only considers the generation of feasible schedules, using a US shift rostering policy. The combining of schedules into rosters is not included in the study.

2.2.4 Decision Support Systems

Several researchers have mentioned the desirability of developing a decision support system (DSS) approach to nurse rostering. Both Sitompul (1992) and Ozharahan and Bailey (1988) envisage systems that would able to handle a broad range of rostering problems. Sitompul (1992) itemises the following characteristics for a nurse rostering DSS:

- An ability to handle semi-structured problems.
- The provision of several different problem solving techniques.
- An easy to use, interactive user interface.
- Sufficient flexibility and adaptability to accommodate changes in the environment and in the decision style of the user.

At present, no system reported in the literature can meet the above criteria. Bell *et al*. (1986) discuss a visual interactive decision support system for nurse scheduling. Their approach concentrates on the development of a user interface, and is able to generate partial rosters. Sitompul’s (1992) DSS application enables users to interactively specify and generate schedules, but is again unable to generate a complete roster. Finally, Ozharahan and Bailey’s (1988) work stresses the DSS aspects of goal programming, by allowing users to specify the relative importance of each rostering objective. However, their application only considers a limited and specific problem.

The above systems represent a first step towards a fully integrated DSS for nurse rostering. The idea of including multiple roster generation techniques in one system has still not been applied. In addition, the enhanced user interface environment available on today's PC has not been studied in relation to nurse rostering applications.

2.2.5 Simulated Annealing

Whilst no direct use of simulated annealing has been made in the nurse rostering literature, the technique has been successfully applied in several other scheduling domains (e.g. school timetabling, aircraft gate allocation and machine-job allocation: Abramson 1992, Lo and Bavarian 1992). Simulated annealing is a general purpose optimisation technique modelled after the physical cooling process of heated atoms (Abramson 1992). It is similar in structure to a local search algorithm (Abramson *et al*. in press). A cost function is defined and local or neighbourhood solutions are randomly generated. These solutions are automatically accepted if they cause a reduction in cost. However, if a solution causes an increase in cost (or energy), it is accepted or rejected on the basis of an annealing probability function and the given temperature of the system. This means the algorithm is able to climb out of local minima solutions that would have caused a local search algorithm to terminate (Connolly 1992). As the algorithm executes, the temperature of the system reduces and the probability of accepting a increased cost solution also decreases, until the algorithm becomes a simple local search.

So-called "classical annealing" (Lo and Bavarian 1992) uses a version of the Bolzman distribution to generate the probability of acceptance given by

$$P(accept) = e^{\frac{-\Delta E}{T}}$$

where T = temperature and ΔE = change in cost caused by accepting the new solution. Further, the temperature is systematically reduced during the execution of the algorithm using some form of cooling schedule, for example, a geometric cooling schedule:

$$T_n = T_{n-1} * R$$

where R is the cooling rate ($0 \le R < 1$) and T is a real number (Abramson 1992).

The temperature variable can be reduced after each iteration of the algorithm or after a predefined number of iterations, called a *Markov chain length*. In addition, the starting temperature of the system and an appropriate terminating condition need to be defined.

Several other forms of the simulated annealing algorithm have been developed, using different probability functions and cooling schedules (see Collins *et al.* 1988, Lo and Bavarian 1992), but all retaining the basic principle of allowing uphill climbs, with a decreasing probability of acceptance over time.

Given a slow enough cooling schedule the simulated annealing algorithm will eventually converge on an optimum solution (Lo and Bavarian 1992). However, as infinitely long cooling schedules are not practical, an optimum solution cannot be guaranteed (Abramson *et al.* in press). Further, in order to find *acceptable* solutions to complex systems very slow cooling rates still need to be employed, resulting in lengthy execution times. The slow convergence of the algorithm has caused researchers to look into specialised computer architecture for simulated annealing (Abramson 1992) and into so-called “fast” simulated annealing algorithms (Lo and Bavarian 1992).

Although not previously applied, simulated annealing is a promising candidate for a nurse rostering algorithm. It extends the possibilities of the cyclic descent algorithm by providing a mechanism to escape local minima, whilst being able to handle larger problems that could prove too complex for integer programming techniques. The main question that arises is whether a simulated annealing algorithm can converge on acceptable solutions to the nurse rostering problem within a reasonable period of time.

2.3 An Introduction to the Research Problem

The intention of the current research is to investigate the cyclic descent algorithm as a tool for solving the nurse rostering problem. Two forms of the algorithm will be tested against manual, simulated annealing and integer linear programming roster solutions. Data will be collected from two wards in an Australian hospital. Details of the rostering practices for these wards are provided in Appendix 3.

2.3.1 Australian vs United States Rostering Policy

Previous published research has concentrated on rostering in US hospitals. An important difference between US and Australian rostering practices is that Australian hospitals generally have fewer constraints in the allocation of shift types.

Generally, in US hospitals, a large proportion of nurses work only one shift type (i.e. all late shifts, all early shifts or all night shifts). Those nurses that rotate shifts, typically work only one shift type between days off, and no more than two shift types per two week period. This means that nurses are not expected to change shift types without an intervening day off or period of days off (e.g. see Sitompul 1992).

In contrast to US practice, Australian nurses are usually expected to work a mixture of early, late and night shifts without intervening days off. This adds to the complexity of the Australian rostering problem. Given the constraints detailed in Appendix 3, a full-time nurse will be able to work up to 8,000 different feasible schedules. If, as in the US, a nurse were additionally constrained to require days off between a change of shift type, the number of feasible schedules would be reduced to 180 (See Appendix 4).

2.3.2 Implications of the Literature for the Research Problem

The research requires an approach which can solve complex rostering problems. For the purposes of this study, a complex problem is defined as one involving several thousands of variables, rather than several hundred. The integer and goal programming techniques described in the literature have considered much smaller problems. Chow and Hui (1993) suggest that integer linear programming (ILP) techniques are unable to solve more complex and realistic rostering problems. However, the degree of complexity at which an ILP approach becomes inadequate is unclear. The continual advance in computer technology means that conclusions drawn in the past about ILP techniques may no longer be relevant[4]. In addition, whilst the heuristic and AI techniques considered in the

[4]Sophisticated integer programming techniques have been developed to solve the airline crew scheduling problem (Graves *et al.* 1993, Hoffman and Padberg 1993). These techniques have used up to a million variables. Whilst the large computing resources required for such approaches are not practical for the current research, the airline studies indicate that ILP techniques are capable of solving large problems.

literature can solve large problems, only satisficing or "good enough" solutions are generated. The quality of these solutions in relation to other approaches has not been measured.

This research is looking for an approach which can solve large problems *and* generate optimal or near optimal solutions. The application of an ILP algorithm is one possibility. Of the other techniques reviewed, the cyclic descent algorithm proposed by Miller *et al.* (1976), and a simulated annealing approach to rostering appear promising. This is because both techniques are capable of solving large problems, whilst *systematically* attempting to optimise the result.

Simulated Annealing and the Cyclic Descent Algorithm: As Abramson *et al.* (in press) observe, simulated annealing is closely related to the cyclic descent algorithm, and at very low temperatures the techniques become identical. The main problem with a cyclic descent strategy is that it tends to get stuck in non-optimal solutions or minima[5]. In order to circumvent these minima, simulated annealing techniques have been used to solve other types of scheduling problem (e.g. Lo and Bavarian 1992). However, simulated annealing methods tend to converge slowly on a solution, especially in complex problem situations (Abramson 1992). The approach in the current research is to develop a heuristic procedure which can move a roster solution out of a local minima in a directed and efficient manner (see Chapter 4). This avoids relying on the broader randomised search technique introduced by simulated annealing. In this way, it is intended that a near optimal solution can be generated in a relatively short time.

Problem Decomposition: Arthur and Ravindran (1981) introduced the idea of separating the allocation of shift types from the main roster optimisation problem. A preliminary study of the Australian rostering problem shows that it can also be decomposed into two

[5]A problem involving a 14 day roster and 25 nurses was given to an implementation of the cyclic coordinate descent algorithm. Each nurse was scheduled to work full time with work stretches of no more than seven days and no less than three days. All nurses were to receive two sets of two consecutive days off. The number of days worked in the previous roster was randomly generated. The algorithm was instructed to minimise the deviations from a required number of days off for each day of the shift (set at between six and eight days off for each day of the roster). The problem was set up so that a perfect solution would always exist (i.e. it was always possible for the target number of days off to be met exactly). The cyclic coordinate descent algorithm was consistently unable to find a perfect solution. An inspection of the algorithm generated solution by a human expert usually revealed that a series of simple steps could move the roster to an optimum solution.

simpler problems by separating the allocation of late and early shifts from the main body of the problem (see Section 4.1.1). Without loss of optimality, these shifts can be considered as two types of day shift with different starting times, and allocated in a separate heuristic procedure (as proposed by Ozharahan and Bailey 1988). Such an approach can significantly reduce the size of the rostering problem. However, any gains from such problem decomposition are counteracted with the introduction of flexible part-time staff into the roster (see Section 4.1.4).

Future Possibilities: The proposed approach to rostering, whilst confronting the issues of problem size and solution quality, does not fully consider the issue of flexibility. A cyclic descent technique relies on a fixed set of constraints in order to descend to a solution. Such an algorithm cannot selectively relax constraints when no immediate feasible solution is possible. In this respect, the knowledge-based techniques proposed by Hui (1988) and reviewed by Randhawa and Sitompul (1990) are superior. Bearing this in mind, it should be noted that the current research is concerned with a *part* of the nurse rostering problem, namely with the development and evaluation of a roster generating engine. A fully operational system, i.e. one that could operate without extensive human intervention, would also require a shell that could resolve conflicting and unattainable constraints. A knowledge-based or expert system approach would seem ideal for the development of such a shell.

2.4 Summary

The current literature review has looked at both cyclic and non-cyclic approaches to nurse rostering. Non-cyclic approaches have been given greater attention as they are more applicable to the research domain. The various papers concerned with non-cyclic nurse rostering are summarised by the following two tables:

Study	General Technique	Specific Technique	Important Features
Chow and Hui (1993), Hui (1988)	Non-optimising heuristics	Knowledge-based scheduling with heuristic operators	1) Able to handle large problems 2) Flexible, with abilities to selectively relax unattainable constraints 3) Applicable to multiple types of rostering problem 4) Imitates human reasoning 5) Solves general rostering problems but not tested on nurse rostering problem
Lukman (1986)	Non-optimising heuristics	Expert system	1) Able to handle large problems 2) Flexible
Smith and Wiggins (1977)	Non-optimising heuristics	List processing heuristic	1) Able to handle large problems 2) Used only as an aid for scheduling decisions, and not intended to produce workable rosters 3) Applied to a specific hospital problem
Sitompul (1992)	Non-optimising heuristics	Heuristics embedded in a Decision Support System	1) Able to flexibly generate a wide range of schedules 2) Not designed to generate a complete roster 3) Specific solution for a US rostering policy
Bell *et al*. (1986)	Non-optimising heuristics	Heuristics embedded in a Decision Support System	1) Good quality, interactive user interface 2) Used as a decision aid only, not intended to produce workable rosters 3) Applied to a specific hospital problem

Table 1: Summary of heuristic techniques for the non-cyclic nurse roster problem

Study	General Technique	Specific Technique	Important Features
Warner (1976)	Optimising	Multiple-choice (integer) programming	1) Optimises both schedule quality and deviations from desired staffing levels 2) Can solve larger problems than a standard integer programming formulation 3) Staffing constraints are flexible 4) Applied to a specific hospital problem
Arthur and Ravindran (1981)	Optimising	Goal programming with heuristics	1) Flexible to changing user priorities 2) Based on standard integer programming techniques 3) A relatively simple model is used
Musa and Saxena (1984)	Optimising	Goal programming	1) Flexible to changing user priorities 2) Based on standard integer programming techniques 3) A relatively simple model is used
Ozharahan and Bailey (1988)	Optimising	Goal programming with heuristics	1) Flexible to changing user priorities 2) Able to consider different shift starting times 3) Based on standard integer programming techniques 4) A relatively simple model is used
Miller *et al.* (1976)	Optimising	Local search cyclic coordinate descent algorithm	1) Able to handle large problems 2) Potentially flexible to changing problem priorities and formulations 3) Solutions not necessarily optimal due to limited area of search 4) Tests of the algorithm are performed
Blau and Sear (1983)	Optimising	Local search cyclic coordinate descent algorithm	1) Able to handle large problems 2) Potentially flexible to changing problem priorities and formulations 3) Solutions not necessarily optimal due to limited area of search
Kostreva and Jennings (1991), Kostreva *et al.* (1978)	Optimising	Mixed integer programming with heuristics	1) Able to handle large problems 2) Good quality user interface 3) Use of heuristics in roster generation 4) Only assignment of staff to schedules is optimised 5) Concept of "hate points" is introduced to measure nurse preferences

Table 2: Summary of optimising techniques for the non-cyclic nurse roster problem

The intent of the current research is to develop a non-cyclic approach to nurse rostering which can calculate with several thousand variables, and can also produce solutions which are comparable or superior in quality to those produced by human experts. Of the approaches to nurse rostering described in the current literature, two main limitations have been identified:

- The optimising mathematical programming techniques become impractical to use as the problem size becomes too large.
- The remaining techniques rely on heuristics, or search techniques, which may result in non-optimal solutions.

From a consideration of the existing methods, it is proposed to develop both an enhanced cyclic descent algorithm based on the work of Miller *et al.* (1976) and a simulated annealing algorithm for nurse rostering. An integer linear programming algorithm will also be used to investigate whether a mathematical optimising approach is feasible for the size of problem. It is noted that there has been a lack of comparison between existing approaches to nurse rostering within the literature. This area will be addressed by an empirical evaluation of the proposed approaches, as described in the next chapter.

Chapter 3: Methodology

3.1 An Outline of the Empirical Study

The task of the empirical study is to provide a comparative measure of the performance of five nurse rostering algorithms. The research is exploratory in nature, and is not intended to rigorously test all dimensions of the rostering problem. Instead, quantifiable criteria will be developed which will allow the scoring and comparison of the selected rostering methods.

The five algorithms considered in the study are:

1. **Basic Cyclic Descent**: A version of the cyclic coordinate descent algorithm proposed by Miller *et al.* (1976).
2. **Enhanced Cyclic Descent**: An enhanced version of the cyclic descent algorithm with a built-in hill climbing heuristic.
3. **Basic Simulated Annealing**: A "classical" simulated annealing algorithm (Lo and Bavarian 1992) using a geometric cooling schedule (Abramson 1992).
4. **Enhanced Simulated Annealing**: A simulated annealing algorithm with a built-in bias to select higher grade schedules.
5. **Integer Linear Programming**: A commercially available package employing a branch and bound algorithm (Hillier and Lieberman 1990).

Details of the development of the cyclic descent and simulated annealing algorithms are provided in Chapter 4, and the integer linear programming formulation of the problem is given in Appendix 3, Section A3.4. Each algorithm will be tested using data collected from rosters actually worked at a Queensland public hospital. Therefore results will be additionally compared with the manually generated solutions developed by hospital staff. This approach is an extension of the original study conducted by Miller *et al.* (1976), where a cyclic descent algorithm is compared with an integer programming application and with manually generated solutions.

The research introduces an enhanced version of Miller *et al.'s* cyclic descent algorithm. The purpose in developing the algorithm was to improve the searching ability of the standard cyclic descent algorithm without incurring the long execution times associated with a simulated annealing algorithm. The empirical study will test whether these objectives have been met.

Four of the algorithms used in the study (both simulated annealing and cyclic descent algorithms) have been developed and will be tested on an IBM® compatible 486 DX50 PC, running under MS-DOS®. The integer linear programming (ILP) package will be tested on a Sun®/Unix platform. It is already expected that the ILP approach will be unable to solve large scheduling problems (Hui 1988). Therefore a more powerful platform has been chosen for the ILP algorithm in order to increase the proportion of rosters that can be solved.

The inclusion of an ILP algorithm in the study is for two reasons. Firstly, the literature is unclear as to the size of rostering problem that an ILP approach can solve (within a reasonable time). By testing the ILP approach on the experimental data, an indication of a feasible problem size can be obtained (the rosters range in size from 822 to 48,782 feasible schedules). Secondly, an ILP algorithm will produce an *optimum* solution to a given set of constraints. Therefore, for those rosters that are solved by the ILP approach, an objective standard can be set by which the quality of roster solutions generated by the other methods can be measured.

The current chapter first looks at the research strategy to be used in the study. Then a more detailed description of the measurement criteria is given. This leads on to a discussion of the experimental design. Finally, the experimental hypotheses are stated and the limitations of the study are discussed.

3.2 Research Strategy

The strategy selected for the current research is to evaluate the rostering algorithms using data contained in existing manually generated rosters. The research will analyse 52 rosters obtained from two Gold Coast Hospital medical wards, spanning the complete period from January 1993 to January 1994. Using these rosters, the original problem parameters can be reconstructed. Each of the five algorithms will then attempt to generate new rosters that solve the original historical roster problems.

Having derived the data, the research will use a set of criteria to compare the rostering approaches. The three main dimensions of comparison are shift allocation quality, schedule quality and algorithm execution time. The processes of criteria generation and statistical comparison are discussed more fully in sections 3.3 and 3.4.

3.2.1 The Use of Historical Rosters

Several advantages of using historical rosters as the basis for the study have been identified:

- Objectivity: The original data is objective and can easily be converted to quantitative measures.
- Sample size: A potentially large sample size can be considered.
- Availability: The data is immediately available.
- Minimum disruption: The use of historical rosters causes a minimum disturbance to nursing staff and the operation of the hospital.

An alternative strategy of trialing computer generated rosters on one hospital ward whilst retaining manual practices on a second ward was considered. This approach was rejected for several reasons. Firstly, the task of developing an instrument that could validly measure staff satisfaction with rostering policy was seen as too ambitious for the current study. In addition, it was considered unlikely that hospital staff would agree to work untested and possibly substandard rosters. Finally, the amount of time and level of cooperation required from a hospital to mount such a study was seen as too unrealistic for research at an honours level.

3.3 Criteria for Roster Evaluation

Although there has been debate in the literature as to the relative importance of different criteria in roster evaluation, there has been broad agreement about the areas that have to be measured. Arthur and Ravindran (1981) gave the following four objectives for their goal programming model:

- Minimum staffing requirements
- Desired staffing requirements
- Nurse schedule preferences
- Nurse requests

The current study cannot properly consider nurse requests because information about requests that were *not* granted is unavailable for the historical rosters. However, measures will be developed for the remaining objectives. Interviews with nursing staff involved in rostering have confirmed that these criteria represent the most important aspects of the rostering problem.

3.3.1 Minimum and Desired Staffing Criteria

Minimum and desired staffing criteria have already been defined as constraints in the mathematical formulation of the problem (see Appendix 3). Quantitative values for deviations from minimum and desired constraints can be easily measured. However, using a simple summation procedure to generate an overall measure of deviation would not be adequate. This is because the consequences of understaffing are generally more serious than those of overstaffing. In addition, the cost of understaffing a shift by two nurses can be more than twice as costly as understaffing a shift by one. Interviews with nursing staff have indicated that relative weights or costs need to be applied to meeting staffing constraints for different shifts. When presented with this problem, the nurses responsible for rostering in the case study wards developed the following costs for each feasible staffing deviation[6] :

DAY SHIFT Number of Staff over or under Given Level	Cost of Deviation
3 over desired level	15
2 over desired level	5
1 over desired level	1
0 desired level	0
1 under desired level	10
1 under minimum level	75
2 under minimum level	250

Table 3: Costs associated with under/over staffing a day shift

NIGHT SHIFT Number of Staff over or under Desired Level	Cost of Deviation
1 over	50
0 (desired level)	0
1 under (minimum level)	100

Table 4: Costs associated with under/over staffing a night shift

SKILL MIX Staff level and shift type	Cost of shortage of 1 staff member
CN day shift	1
CN night shift	2
Senior RN day shift	5

Table 5: Costs associated with shortages of senior staff

[6]The charge sisters from the two wards considered in the study were asked to define weights for the staffing level criteria, given that an ideal solution has a zero weight. An initial weight of 10 was given to a shortage of one staff member under the desired level on a day shift, and the nurses were then asked to generate the other criteria weights on this basis. A Delphi method was used, in that the results of each nurse's scorings were fed back to the other nurse until an agreed weighting was developed.

3.3.2 Nurse Preference Criteria

Due to the infeasibility of surveying each nurse considered in the study (many no longer work at the hospital), the idea of generating unique criteria for each nurse's preferences has been rejected. Instead, criteria have been developed which represent a generalised or averaged view of schedule quality. The accepted idea of an ideal two week full-time schedule, is one with five days on and two days off, followed by another five days on and two days off period. This is shown in the following table:

Mo	Tue	We	Th	Fri	Sat	Sun	Mo	Tue	We	Th	Fri	Sat	Sun
On	*On*	*On*	*On*	*On*	**Off**	**Off**	*On*	*On*	*On*	*On*	*On*	**Off**	**Off**

Table 6: An ideal two week schedule

Within the framework of an ideal five on, two off, five on, two off schedule pattern, patterns with longer or shorter work stretches and with different distributions of days off are considered less attractive. Although part-time staff cannot work an ideal two week schedule, the same work stretch and day off preferences apply. For instance, a nurse working eight shifts in a roster would generally prefer two four day stretches and to have a least two days off between stretches. The study does not consider the granting of weekends off, as this is achieved through the requesting policy. As with the staffing level criteria, there are different weights for different kinds of violation. For instance, as work stretches get longer they become increasingly more unattractive, whereas shorter work stretches are not so unpopular. Again using interviews with the charge sisters of the two wards studied, the following weights were developed:

CONSECUTIVE DAY OFF STRETCHES Number of Consecutive Days Off	Cost of Stretch
1 day off	15
2 days off	0
3 days off	5
4 days off	10

Table 7: Costs associated with consecutive days off

CONSECUTIVE DAYS WORKED Number of Consecutive Days Worked	Cost of Stretch
1 day on	10
2 days on	5
3 days on	2
4 days on	1
5 days on	0
6 days on	1
7 days on	5
8 days on	15
9 days on	20
10 days on	30

Table 8: Costs associated with consecutive work stretches

3.3.3 Time Measurement

An additional measure will be taken of the time taken for each algorithm to find a roster solution. The dimension of time is considered important, firstly to distinguish between algorithms that are able to find equally good solutions on all other criteria, and secondly to test that a computerised solution can result in significant time savings over a manual approach.

A direct comparison between the execution time of the integer linear programming (ILP) application and the other algorithms is not possible. This is because the ILP package (LPSolve) will be tested on a Sun® workstation running under Unix, whilst the other algorithms will be tested on an IBM® compatible 486 DX50 PC running under MS-DOS®. Consequently, a program was developed in ANSI C to simulate the tasks of a roster algorithm. This program was then executed on both platforms and a platform adjustment factor obtained, based on the relative execution times. Using the platform adjustment factor, the time data for the ILP algorithm can be adjusted so that they become *approximately* comparable to the results for the PC based algorithms.

A measure is also required of the time taken by hospital staff to complete the manual rosters. As no records were kept of this data, it is assumed, based on interviews with hospital staff, that each manual roster takes approximately 3 hours to complete. This figure can be used as a yardstick with which to assess the various algorithms, but will not be included in the statistical analysis.

3.4 Research Design

3.4.1 Statistical Method

The empirical study considers the relative performance of the two cyclic descent algorithms, the two simulated annealing algorithms, the ILP algorithm and the manually generated solutions for the 52 historical rosters obtained from the hospital. The objective is to find out whether the mean criteria scores for each roster generation method are significantly different. If the criteria scores are different, then the direction of the difference is also of interest. A secondary objective is to find if there is any significant difference in mean criteria scores for the two wards considered, and whether there is any interaction effect between the ward and the method used. The problem is therefore one of testing the significance of group differences. Given that basic assumptions can be met, the appropriate statistical method would be a factorial multivariate analysis of variance (MANOVA, Tabachnick and Fidell 1989).

It is not expected that the ILP algorithm will be able to solve the larger rostering problems. Additionally, the execution times for the ILP algorithm are only approximately comparable to the other algorithms (due to the use of different platforms), and no reliable execution times are available for the manual method. Therefore the statistical analysis will be divided into three parts:

1. MANOVA 1 will consider schedule and shift score data for all 52 rosters but will omit results for the ILP algorithm (as missing results are expected), and will not consider execution times (as reliable data does not exist for the manual method).

2. MANOVA 2 will include schedule, shift and execution time data for all rosters, and consequently will omit results for the ILP and manual methods

3. MANOVA 3 will include schedule, shift and execution time data for rosters that the ILP method has been able to solve, leaving out results for the manual method.

3.4.2 Independent Variables

The independent variables in the model are the ward from which a roster originates and the method used to solve the roster. In accordance with the requirements of MANOVA, these variables are nominally scaled, as defined in the following table:

VARIABLE NAME AND VALUE	VARIABLE DESCRIPTION
Method = 1	Indicates the roster was manually generated
Method = 2	Indicates the roster was generated using the basic cyclic descent algorithm
Method = 3	Indicates the roster was generated using the enhanced cyclic descent algorithm
Method = 4	Indicates the roster was generated using the basic simulated annealing algorithm
Method = 5	Indicates the roster was generated using the enhanced simulated annealing algorithm
Method = 6	Indicates the roster was generated using the integer linear programming package
Ward = 1	Indicates the roster came from Ward 1
Ward = 2	Indicates the roster came from Ward 2

Table 9: Independent variables for the MANOVA statistical model

3.4.3 Dependent Variables

The dependent variables in the statistical model represent the various criteria upon which the wards and rostering approaches are to be compared. To calculate criteria values for each roster, a series of measures have to be made. Firstly, the number of shifts with a particular level of over- or understaffing are counted for each roster. Then, the various stretches of days on and days off for all staff and the total number of staff are counted. Using these values, the criteria values are calculated using the weights defined in the previous section. Finally, the time taken for each algorithm to find a roster solution is recorded and multiplied by the platform adjustment factor. The individual measures required to calculate the criteria are defined in the following table:

VARIABLE NAME	VARIABLE DESCRIPTION
Day_{plus3} (d_3)	Number of day shifts in roster overstaffed by 3
Day_{plus2} (d_2)	Number of day shifts in roster overstaffed by 2
Day_{plus1} (d_1)	Number of day shifts in roster overstaffed by 1
$Day_{desired1}$ ($d_{d\text{-}1}$)	Number of day shifts with 1 under desired staff level
Day_{minus1} (d_{-1})	Number of day shifts with 1 under minimum staff level
Day_{minus2} (d_{-2})	Number of day shifts with 2 under minimum staff level
$Night_{plus1}$ (n_1)	Number of night shifts in roster overstaffed by 1
$Night_{minus1}$ (n_{-1})	Number of night shifts in roster understaffed by 1
$SeniorDay_{minus1}$ (sd_{-1})	Number of day shifts with senior staff shortage of 1
$SeniorNight_{minus1}$ (sn_{-1})	Number of night shifts with senior staff shortage of 1
RN_{minus1} (rn_{-1})	Number of shifts with senior RN staff shortage of 1
$WorkStretch_1$ (w_1)	Number of 1 day work stretches in roster
$WorkStretch_2$ (w_2)	Number of 2 day work stretches in roster
$WorkStretch_3$ (w_3)	Number of 3 day work stretches in roster
$WorkStretch_4$ (w_4)	Number of 4 day work stretches in roster
$WorkStretch_5$ (w_5)	Number of 5 day work stretches in roster
$WorkStretch_6$ (w_6)	Number of 6 day work stretches in roster
$WorkStretch_7$ (w_7)	Number of 7 day work stretches in roster
$WorkStretch_8$ (w_8)	Number of 8 day work stretches in roster
$WorkStretch_9$ (w_9)	Number of 9 day work stretches in roster
$WorkStretch_{10}$ (w_{10})	Number of 10 day work stretches in roster
$DaysOff_1$ (o_1)	Number of 1 day day off stretches in roster
$DaysOff_2$ (o_2)	Number of 2 day day off stretches in roster
$DaysOff_3$ (o_3)	Number of 3 day day off stretches in roster
$DaysOff_4$ (o_4)	Number of 4 day day off stretches in roster
TotalNurses (tn)	Total number of nurses working in the roster

Table 10: Dependent variable measures for the MANOVA statistical model

Given the previously defined variable measures, the three overall criteria scores for each roster are calculated using the following formulae :

WeightedShift =

$$15d_3 + 5d_2 + d_1 + 10d_{d\text{-}1} + 75d_{\text{-}1} + 250d_{\text{-}2} + 50n_1 + 100n_{\text{-}1} + sd_{\text{-}1} + 2sn_{\text{-}1} + 5rn_{\text{-}1}$$

WeightedSchedule =

$$(10w_1 + 5w_2 + 2w_3 + w_4 + w_6 + 5w_7 + 15w_8 + 20w_9 + 30w_{10} + 15o_1 + 5o_3 + 10o_4) / tn$$

ExecutionTime = Program execution time * Platform adjustment factor

(for programs run on the IBM®/DOS platform the platform adjustment factor = 1)

3.4.4 Hypotheses

The objective of the empirical study is to generate statistical measures of the relative performance of the rostering methods considered. Of primary interest is the relative performance of the enhanced cyclic descent algorithm in comparison to the other methods. Consequently, the enhanced cyclic descent mean values will be used as the basis of comparison.

Hypotheses as to the expected performance of the algorithms can be developed from an examination of the algorithms themselves (see Chapter 4). Firstly, in the dimension of shift distribution quality (WeightedShift), it is expected that the basic cyclic descent algorithm will produce the poorest results, with all other algorithms scoring approximately equally. In the area of schedule quality (WeightedSchedule), it is expected that the ILP algorithm will produce the best results will all other algorithms again scoring approximately the same. Shortest execution times are expected for the basic cyclic descent algorithm, followed by the enhanced cyclic descent algorithm, followed by the enhanced simulated annealing algorithm and the basic simulated annealing algorithm. Execution times for the ILP algorithm are undetermined. Finally, it is expected that all the computerised methods will find better solutions than the manual method in the dimensions of schedule quality, shift distribution quality and shorter execution times. These expectations are expressed in the following hypotheses:

Hypothesis 1:

The mean value of WeightedSchedule for the ILP algorithm (method = 6) is *less than* the mean value of WeightedSchedule for the enhanced cyclic descent algorithm (method = 3) :

$$\mu_{method=6,WeightedSchedule} < \mu_{method=3,WeightedSchedule}$$

Hypothesis 2:

The mean value of WeightedShift for the basic cyclic descent algorithm (method = 2) is *greater than* the mean value of WeightedShift for the enhanced cyclic descent algorithm (method = 3) :

$$\mu_{method=2,WeightedShift} > \mu_{method=3,WeightedShift}$$

Hypothesis 3:

No *significant difference* exists between the mean values of WeightedShift for the enhanced cyclic descent algorithm (method = 3), the basic simulated annealing algorithm (method = 4), the enhanced simulated annealing algorithm (method = 5) and the ILP algorithm (method = 6) :

$$\mu_{method=3,WeightedShift} = \mu_{method=4,WeightedShift} = \mu_{method=5,WeightedShift} = \mu_{method=6,WeightedShift}$$

Hypothesis 4:

No *significant difference* exists between the mean values of WeightedSchedule for the basic cyclic descent algorithm (method = 2), the enhanced cyclic descent algorithm (method = 3), the basic simulated annealing algorithm (method = 4) and the enhanced simulated annealing algorithm (method = 5) :

$$\mu_{method=2,WeightedSchedule} = \mu_{method=3,WeightedSchedule} = \mu_{method=4,WeightedSchedule} = \mu_{method=5,WeightedSchedule}$$

Hypothesis 5:

The mean execution time for the basic cyclic descent algorithm (method = 2) is *less than* the mean execution time for the enhanced cyclic descent algorithm (method = 3) which is *less than* the mean execution time for the enhanced simulated annealing algorithm (method = 5) and the mean execution time for the basic simulated annealing algorithm (method = 4) :

$$\mu_{method=2,ExecutionTime} < \mu_{method=3,ExecutionTime} < \mu_{method=5,ExecutionTime}$$
$$\mu_{method=3,ExecutionTime} < \mu_{method=4,ExecutionTime}$$

Hypothesis 6:

The mean value of WeightedSchedule for the manual method (method = 1) is *greater than* to the mean value of WeightedSchedule for the enhanced cyclic descent algorithm (method = 3) :

$$\mu_{method=1,WeightedSchedule} > \mu_{method=3,WeightedSchedule}$$

Hypothesis 7:

The mean value of WeightedShift for the manual method (method = 1) is *greater than* to the mean value of WeightedShift for the enhanced cyclic descent algorithm (method = 3) :

$$\mu_{method=1,WeightedShift} > \mu_{method=3,WeightedShift}$$

3.5 Limitations

3.5.1 Schedule Quality and Differences between Nurses

The method of criteria generation provides an average measure of two aspects of schedule quality : 1) the distribution of days off and 2) the length of work stretch. Individual nurses will have different perceptions about the quality of a schedule which cannot be captured in an averaging approach. For example, a nurse may find a pattern with three consecutive days off and one single day off preferable to a pattern with two stretches of two days off. A human rosterer may deliberately consider such factors when selecting schedules for a nurse. In such circumstances, the criteria score will not accurately reflect the schedule quality. However, the alternative of finding schedule quality criteria weights for each nurse in the study is not considered feasible. This is firstly because of the large number of nurses involved, secondly because many of the nurses no longer work in the hospital and thirdly because nurses may be unable to remember the criteria that would have applied in previous rosters.

3.5.2 Human Preprocessing of Rosters

The computerised rostering approaches will be presented with roster problems that have already been solved. A human rosterer will have eliminated any inconsistencies in the original problem, such as infeasible requests. Also, additional staff will have been acquired for days when shortages have arisen. Therefore, the comparison between manual and computerised methods assumes that some human intervention has occurred. To fully replace human expertise, problem preprocessing software would have to be developed. This issue was previously discussed in relation to the development of an expert system shell (see Chapter 2, Section 2.3.2).

3.5.3 Measuring the Overall Roster Quality

The final judgement on the quality of a roster is subjective, and dependent on the values and priorities of the person making the judgement. It is consequently difficult to put a quantitative value on the quality of a roster (Smith and Wiggins 1977). An attempt to codify good rostering policy was made by the Australian Nurses Federation (1992). However, the principles laid down do not specify quantifiable measures. It is therefore recognised that criteria used in this study can only show that one approach differs from another within the confines of the criteria definition. As these criteria were developed for a specific rostering problem, it would be unwise generalise the results to other rostering situations.

3.5.4 Omission of Criteria

The criteria generated in the study attempt to measure the important aspects of roster quality. Some areas, such as the provision of days off after night shifts, are not included because they are automatically granted in all cases. The proportion of requests granted is also not measured because the data indicating disallowed requests is unavailable. Nevertheless, there are other dimensions of roster quality which are not assessed. For instance, there is the fairness of distribution of different shift types amongst nurses. In addition, no measure of the financial cost of staff for each shift is made. Some of these criteria are included in the algorithms, but have been omitted from the empirical study due to their lesser importance. It is noted however, that the relative importance of different rostering criteria are defined by an informal hospital rostering policy. This policy can differ from ward to ward and from time period to time period.

3.5.5 Sample Size

Out of a total of 150 rosters made available by the hospital, the study uses 52 rosters, (26 each from two wards) for the statistical analysis. The sample sizes are limited by the amount of time required to encode a roster into a format suitable for the various algorithms and by the time required to encode the roster solutions for statistical analysis (approximately 2 hours in total per roster). In addition, algorithm solution times can vary from several seconds to several hours. Given the scope and level of the current study, the sample size is considered adequate. However, it is noted that a larger sample size would be desirable, especially in comparing rosters between different years, or between different hospitals.

3.5.6 Different Platforms

The comparison of rostering algorithms across a Sun®/Unix and a PC/DOS platform introduces uncertainty as to the applicability of the integer linear programming (ILP) algorithm on a PC. A Unix-based ILP product was chosen for the research because a similar PC-based product capable of solving large problems was not immediately available. Whilst a comparison of execution times between platforms was made using a simple roster simulation program, this does not mean that the ILP program used could run on a PC. Other PC-based linear programming products were found to run slowly on a PC and were unable to access sufficient memory to solve any of the roster problems used in the research.

A better comparison of execution times between algorithms would be obtained by compiling the PC-based algorithms on a Sun®/Unix platform and then regenerating the results. Due to time constraints, this additional generation of results was not attempted. Therefore, the relative execution times will only be fully accurate between the PC-based algorithms.

3.5.7 Generalisability

The study is not intended to be directly generalised to other rostering problems. The ability of the various algorithms to adapt to different problem situations would have to be tested through direct application. This must be left for further research.

3.6 Summary

The empirical research in this study is primarily designed to show whether the enhanced cyclic descent algorithm warrants further investigation. In the first part of the study, this will be done by statistically comparing the output of the algorithm against a set of rosters already generated by human experts. The algorithm will also be compared against the cyclic descent algorithm on which it is based, against two simulated annealing algorithms and against an existing Integer Linear Programming (ILP) algorithm.

Criteria have been developed with which all the methods can be compared. These criteria quantitatively measure average schedule quality, shift allocation quality and execution times for each roster. The task of generating an overall and valid measure of schedule quality is considered too ambitious for the current study.

In the empirical study, the significance of differences between mean scores on each criteria, for each method and each ward, will be evaluated using a multivariate analysis of variance. The study hypothesises that the enhanced cyclic descent algorithm will produce scores on the criteria that are better than the scores for the manually generated rosters, better than the basic cyclic descent algorithm in the dimension of shift distribution, and as good as scores produced by the simulated annealing and ILP algorithms, but executing in a shorter time.

Chapter 4: Implementation

In order to reproduce the techniques used in the study, information is needed describing the method of schedule generation, the type of cost function used and the actual roster generating algorithms. The current chapter provides this information, and is intended to assist readers interested in replicating or extending the current study. Much of the material assumes a familiarity with rostering concepts and terms which are explained more fully in Appendices 1 to 4. The programs developed in the study were written in object-orientated Borland® C++ version 3.1 and run under MS-DOS® version 6.2 using an IBM® compatible 486 DX50 microcomputer with 8Mb of RAM. The ILP algorithm was run under Unix on a Sun® workstation having 16Mb of RAM.

4.1 Feasible Schedule Generation

A major part of the program development was spent in finding the best form of problem representation. Borrowing from Warner (1976), it was decided that all feasible schedules for each nurse should be generated and used in the rostering algorithm. It was found that for complex problems, an exhaustive iteration of feasible schedules soon uses up available memory resources. In addition, as the number of feasible schedules grows, the execution time for each iteration of a cyclic descent algorithm increases proportionately. Therefore, it became an important issue to develop a more economic form of feasible schedule representation, *without* eliminating possible solutions.

4.1.1 Separating the Allocation of Late and Early Shifts

As described in Chapter 2, the number of feasible schedules in a problem can be reduced by solving the allocation of late and early shifts as a separate heuristic procedure. This results in feasible schedules that have 3 possible shift values (day on, night on, day off), rather than 4 (early shift, late shift, night shift, day off). In an *unconstrained* 14 day schedule this reduces the number of possible schedules from 268 million to 5 million. The separation of the late/early allocation is possible because, in most cases, late and early shifts are interchangeable. To ensure a valid roster solution is obtained, all matching late and early shift constraints are summed to form new day shift constraints. For example, the minimum staff constraint for a particular day shift equals the minimum staff for an early shift plus the minimum staff for a late shift.

The allocation of early and late shifts occurs once a roster solution has been found that meets all the fixed constraints. The task is then to allocate the correct number of early and late shifts for each day, according to the constraints for each shift type, whilst minimising nurse schedule dissatisfaction. Considerations in solving this problem are that nurses generally require:

1. An early shift before days off.
2. A late shift after days off.
3. As few late shifts followed by early shifts as possible.
4. A fairly even balance of late and early shifts over the whole schedule.

This is a relatively small problem and can be formulated and solved as an integer linear program. However, in the current research, the late/early allocation is solved heuristically, providing a “good enough” rather than optimum solution. Obtaining optimality was not considered crucial as it is accepted hospital policy for nurses to renegotiate or swap late and early shifts after the roster has been posted.

4.1.2 Night Shift Representation

It is not possible to separate the allocation of night shifts in the same manner as late and early shifts because of the special constraints on a night shift, i.e. a night shift must be followed by another night shift or a day off. However, the exhaustive iteration of all night shift possibilities can be avoided by using the roster cost evaluation function to convert night shifts to day shifts where allowable and necessary (see Section 4.2). In this way, only the maximum number of nights a nurse needs to work need be expressed in a schedule, rather than each individual pattern. For example, consider a nurse who can work up to 4 night shifts in a particular roster and is constrained to working these nights in one continuous block (this is a typical situation). Given a fixed pattern of days off, the following table illustrates all possible combinations of day and night shifts for the selected nurse:

Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Day	*Day*	*Day*	*Day*	*Day*	**Off**	**Off**	*Day*	Night	Night	Night	Night	**Off**	**Off**
Day	*Day*	*Day*	*Day*	*Day*	**Off**	**Off**	*Day*	*Day*	Night	Night	Night	**Off**	**Off**
Day	*Day*	*Day*	*Day*	*Day*	**Off**	**Off**	*Day*	*Day*	*Day*	Night	Night	**Off**	**Off**
Day	*Day*	*Day*	*Day*	*Day*	**Off**	**Off**	*Day*	*Day*	*Day*	*Day*	Night	**Off**	**Off**
Day	*Day*	*Day*	*Day*	*Day*	**Off**	**Off**	*Day*	*Day*	*Day*	*Day*	*Day*	**Off**	**Off**
Day	Night	Night	Night	Night	**Off**	**Off**	*Day*	*Day*	*Day*	*Day*	*Day*	**Off**	**Off**
Day	*Day*	Night	Night	Night	**Off**	**Off**	*Day*	*Day*	*Day*	*Day*	*Day*	**Off**	**Off**
Day	*Day*	*Day*	Night	Night	**Off**	**Off**	*Day*	*Day*	*Day*	*Day*	*Day*	**Off**	**Off**
Day	*Day*	*Day*	*Day*	Night	**Off**	**Off**	*Day*	*Day*	*Day*	*Day*	*Day*	**Off**	**Off**

Table 11: All night shift combinations (block size 1, length 4) for given days off

Given the ability of the roster cost evaluation function to eliminate night shifts as needed (whilst still maintaining that all night shifts are followed by another night or a day off), the nine schedules in table 11 can be represented by the two schedules as shown in the following table:

Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Day	*Day*	*Day*	*Day*	*Day*	**Off**	**Off**	*Day*	Night	Night	Night	Night	**Off**	**Off**
Day	Night	Night	Night	Night	**Off**	**Off**	*Day*	*Day*	*Day*	*Day*	*Day*	**Off**	**Off**

Table 12: Reduced night shift representation

4.1.3 Two Phase Schedule Generation

Schedules are generated in the program in two phases:

- Phase 1: Generating all feasible schedule patterns of days off.
- Phase 2: Adding all feasible night shifts to the phase one schedule patterns.

The schedule pattern generating algorithm works by generating all possible day on and off patterns for the first week of the roster (this results in $2^7 = 128$ schedule patterns for each nurse). Then, by applying schedule constraints for each nurse, illegal patterns are eliminated. For the remaining feasible patterns, the pattern generating algorithm is recursively called, and patterns for the second week are added, and so on until the total number of weeks in the roster are reached (in this case two), and all feasible patterns for all nurses are generated. The schedule constraints defined in the current application require the following information:

- Maximum and minimum shifts to be worked per schedule and per week.
- Maximum and minimum *generally and exceptionally*[7] allowable unbroken stretches of days on and of days off.
- Size of unbroken stretch of days on worked at the end of the last roster.
- Size and type of unbroken stretch of the same shift type worked at the end of the last roster.
- List of any requests for each day of the roster (requests can either be *normal* or *fixed*, and for any shift type or for a day off: a normal request can be replaced by a day off, however a fixed request must be honoured).

The phase two allocation of night shifts takes each day off schedule pattern and calculates how many schedules containing night shifts can be generated from the given pattern. The night shift schedules are then generated and added to a linked list of schedules for each nurse. As previously mentioned, night shifts are constrained in

[7]The difference between a general and an exceptional constraint is that an exceptional constraint can be allowed once in a schedule and after this the general constraint holds.

having to be followed by another night shift or by a day off. The following night constraints are also defined, based on hospital policy and individual preferences :

- Maximum and minimum nights due for the schedule.
- Number of unbroken blocks of nights allowed and maximum size of blocks.
- Minimum generally and exceptionally allowable block of days off after a night shift.

When all feasible schedules for all nurses have been generated, a schedule score for each schedule is calculated according to the WeightedSchedule function defined in Section 3.4.3. The schedules are then stored in a three dimensional dynamic array ready for use in the main algorithm. Tailoring this system of schedule generation to individual nurses, it is possible to generate any schedule pattern currently employed at the Gold Coast Hospital. The result is that a typical full-time nurse working night shifts will have between 100-200 feasible schedules. However, some part-time nurses are able to work more flexible schedules and can still generate more than 30,000 feasible schedules in a particular roster.

4.1.4 Unconstrained Schedule Generation

For part-time nurses able to work single days on and not requiring a minimum of two consecutive days off, it becomes uneconomic to exhaustively generate all feasible schedules. In the normal operation of the cyclic descent algorithm, all feasible schedules for a particular nurse are tried out in the roster and the schedule having the lowest cost is selected. The cost evaluation function is fairly complex, and for nurses with thousands of schedules, a complete evaluation can take several seconds. In such circumstances it is easier to *calculate* the best schedule than to iterate through all feasible schedules. For a nurse working days only, the task is to evaluate the cost of the nurse working each day of the roster versus the cost of not working. If the nurse is due to work five days, then the five days that result in the lowest cost are selected. The problem becomes more complex if the nurse is to work nights, but the basic principles remain the same.

4.1.5 Evaluation of the Schedule Representation Techniques

The various techniques described above have the combined advantage of reducing the number of feasible schedules that need to be stored in memory in order to solve a given roster problem. At the same time optimality has been maintained, in that no feasible solution possibility has been discarded. This results in an approach that can execute faster and use less memory than the alternative approach of using an explicit enumeration of all feasible schedules.

However, the use of a special cost function that can change shift values as it evaluates schedules, and the use of an algorithm to generate schedules during the execution of the main rostering algorithm, means that the method of schedule representation becomes tied to a cyclic descent approach to rostering. In order to present a roster problem to an Integer Linear Programming (ILP) algorithm it is necessary to “turn off” some of the schedule generation features, ie all night shift possibilities and all part-time staff schedules must be explicitly generated. This can increase the problem size by a factor ranging from x2 to x500 depending on individual staff preferences and constraints.

4.2 Cost Evaluation Function

The cost evaluation function used in the study evaluates all *fixed* constraints for the problem. The objective of minimising nurse dissatisfaction with schedules is dealt with separately in the schedule selection process of the cyclic descent algorithm. The evaluation function counts any deviations away from the fixed problem constraints and, by summing these deviations, an overall roster score is produced. The constraints considered in the research problem are as follows (remembering that day shift constraints are an amalgamation of early shift and late shift constraints):

- Maximum and minimum total staff for each day shift and night shift of the roster.
- Maximum and minimum staff of level CN and above for each day shift and night shift of the roster.

- Maximum and minimum staff of level senior RN or above for each day shift and night shift of the roster.
- Maximum and minimum staff of level EN for each day shift and night shift of the roster.

The special feature of the cost evaluation function is that it can selectively change night shifts into day shifts. This occurs if the shift change is allowable and results in a reduced overall cost. As described previously, feasible schedules are generated which contain the longest allowable blocks of nights for each nurse. These night blocks can then be "trimmed" by the cost function in the following way:

- Night shift constraints are evaluated sequentially by day, starting with the earliest day in the roster.
- If, for a particular day and a particular constraint, there are too many night shifts being worked, then each night shift participating in the constraint is considered for conversion to a day shift. A night shift is converted to a day shift if:
    1. The night shift has not been specifically requested by the nurse concerned.
    2. The shift immediately preceding the night concerned is *not* a night shift.
- Individual night shift staffing level constraints are evaluated first, then the total night shift staff constraints. Finally, the day shift constraints are considered.

Appendix 5 illustrates the process of night shift conversion with a simplified example.

4.3 The Basic Cyclic Descent Algorithm

Using the design described by Miller *et al.* (1976), an algorithm was developed, and set within the framework of the schedule generation algorithms and cost evaluation function previously described. In order to obtain an overall measure of roster cost that includes schedule quality (as per Miller *et al*.) the following equation was used:

Roster Cost = Total summed fixed constraint deviations + (Total roster grade/1000)

where the total summed fixed constraint deviations is given by the return value of the cost evaluation function described earlier (see Section 4.2) and the total roster grade is given by summing the individual schedule grades of each schedule appearing in the roster (schedule grades are calculated using the formula given in Section 3.4.3).

As the total roster grade is integral and typically ranges from 0 to 700, and the summed deviations are also integral ranging from 0 upwards, the above cost equation ensures an algorithm will always accept a lower fixed constraint score *before* trying to minimise the total roster grade.

Given these definitions, the basic cyclic descent algorithm is described by the following pseudocode (see also Section 2.2.1.4) :

cyclic descent algorithm

```
{
    calculate a set of feasible schedules for each nurse
    calculate the schedule grade  of each schedule
    select best grade schedule for each nurse to create an initial roster solution
    best cost = cost of initial roster solution
    current nurse = first nurse on roster
    cycle = 0
    while cycle < total number of nurses on roster
    {
        cycle = cycle + 1
        new schedule = first schedule in feasible schedule set for current nurse
        while more feasible schedules for current nurse
              and current nurse not unconstrained
        {
            remove existing schedule in roster for current nurse
            insert new schedule into roster for current nurse
            new cost = cost of new roster solution
            if new cost  <  best cost
            {
                best cost = new cost
                cycle = 0
            }
            else
            {
                remove new schedule from roster
                return previous existing schedule to roster
            }
            new schedule = next feasible schedule for current nurse
        }
        if current nurse is unconstrained
        {
            calculate new schedule for current nurse
            remove existing schedule in roster for current nurse
            insert new schedule into roster for current nurse

        }
        if current nurse = last nurse on roster
            current nurse = first nurse on roster
        else
            current nurse = next nurse on roster
    }
}
```

Figure 5: Cyclic descent algorithm (based on Miller et al. 1976)

The principle of the algorithm is that *all* feasible schedules for a particular nurse are evaluated in the roster whilst holding constant the schedules for all other nurses. Each nurse is tried in turn until no further improvement in the roster score is possible.

4.4 The Enhanced Cyclic Descent Algorithm

Three strategies were developed to improve the performance of the basic cyclic descent algorithm. These are described in the following sections:

4.4.1 Multiple Starting Positions

It was found that the basic cyclic descent algorithm converges quickly on a roster solution, even in more complex problem situations (usually within 30 seconds). The first, and simplest strategy employed was to repeatedly run the algorithm over the same problem. With each run, the order of the nurses in the problem is changed, which in turn causes the algorithm to follow a different path of descent. At the end of each descent the roster solution is evaluated against the best solution found so far, and if an improvement is found, the solution is kept, else it is discarded. As there is no guarantee of convergence for such an algorithm, it is run until either a predefined minimum score is found, or until a maximum number of iterations have been completed.

Although increasing the execution time of the program, the use of multiple starting positions was found to be an effective strategy in obtaining better quality roster solutions. This is because the basic cyclic descent algorithm can converge on a variety of different roster solutions according to the starting position, rather than repeatedly finding the same solution. As a rule of thumb, it was found that little improvement in roster score can be expected after 20 to 30 trial iterations, but that within those trials roster scores can vary noticeably.

4.4.2 Schedule Grade Selection Bias

Whilst the basic cyclic descent algorithm was found to be effective in finding roster solutions within the bounds of the fixed constraints, the algorithm did not appear effective in finding solutions with good quality schedule grades. A way of rectifying this would be to increase the importance of the schedule grade component in the cost function. However, the primary objective of a rostering algorithm is to meet *all* the fixed constraints. To increase the weight of the schedule grade component in the cost function could cause minimum cost solutions to contain violated constraints, where a solution meeting all fixed constraints would have been possible.

In order to address this problem it was decided to remove the schedule grade component from the cost function and introduce a schedule grade selection bias into the cyclic descent algorithm. Firstly, during the normal execution of the algorithm, all schedules for a particular nurse that can cause *any* improvement in the deviations from the fixed constraints are selected and stored as candidate schedules. Once all schedules for the nurse have been evaluated, the candidate schedule with the lowest or best grade is selected and inserted into the roster. If several schedules with the same minimum grade exist then the schedule causing the greatest reduction in deviations from fixed constraints is chosen. If multiple candidates still exist, a schedule is chosen from this final group at random. This contrasts with the basic cyclic descent algorithm which automatically selects the schedule with the lowest combined deviation and schedule grade cost.

The schedule grade selection bias should therefore cause the selection of higher quality schedules, at the expense of descending more slowly towards a solution.

4.4.3 Hill Climbing Algorithm

The main weakness of the cyclic descent algorithm is that it has no facility to climb out of local non-optimal solutions and continue to search for possibly superior solutions. From a given starting position, each iteration of the algorithm is directed by the solution from the previous iteration, until the solution becomes “stuck”. A final solution is accepted after all feasible schedules for all nurses have been tried in the roster and no overall improvement in the roster score is found. However, by accepting a new schedule in the roster that either causes the roster score to remain the same, or even to deteriorate, the possibility exists that further iterations of the algorithm may find a superior solution. The ability to accept solutions that cause a deterioration in score distinguishes simulated annealing from a straightforward cyclic descent (see Section 2.2.5), and is often referred to as “hill climbing” (Lo and Bavarian, 1992, p. 324).

The third strategy for improving the cyclic descent algorithm was to develop a hill climbing algorithm that is invoked each time the basic algorithm becomes stuck in a local minima. From an observation of roster solutions it became apparent that the cyclic descent algorithm becomes stuck on certain columns or days of the roster. This means it becomes impossible to improve the deviation score for a stuck column without causing a deterioration in score for some other column of the roster. The initial approach of the hill climbing algorithm is to reduce the roster score for a stuck column by moving the problem to some other column in the roster. This is done with the following steps:

1. Identify the stuck column as the column having the highest deviation score.
2. Identify any schedules from the set of all feasible schedules that cause the deviation score on the stuck column to improve.
3. From this set of schedules, select the schedule that causes the smallest deterioration in the overall deviation score for the roster.
4. Insert the selected schedule into the roster and then continue with the normal operation of the cyclic descent algorithm.

The problem with this approach is that the roster solution will tend to “flip-flop” between a series of two or more solutions without effectively descending. For example, a new schedule is selected by the hill climbing algorithm that causes the roster score to deteriorate, but when control is returned to the cyclic descent algorithm the new schedule is immediately replaced with the schedule that *it* replaced and the roster is returned to the original stuck position. This problem is addressed by creating a one dimensional “stuck-shift” array with an element for each column or day of the roster. Each time the hill climbing algorithm is invoked, the new reduced deviation score for the stuck column in the roster is inserted into the stuck-shifts array. From then on, no schedule is accepted either by the cyclic descent algorithm or by the hill climbing algorithm that causes the score for the stuck column to exceed the deviation score recorded in the stuck-shifts array. The total algorithm continues until the hill-climbing algorithm is unable to select a schedule without violating the stuck-shifts array constraints.

All three strategies used to develop the enhanced cyclic descent algorithm are expressed in the following two pages of pseudocode (the hill climbing algorithm is referred to as “move roster from minima”):

```
enhanced cyclic descent algorithm        {

calculate a set of feasible schedules for each nurse
calculate the grade of each schedule for each nurse
select schedule with best grade for each nurse to create an initial current roster
solution found = FALSE; stuck counter = 0
while (solution found = FALSE) and (stuck counter < MAXIMUM STUCKS)   {
        current nurse = first nurse on roster
        while more nurses on the roster   {
                number of candidate schedules = 0
                previous grade = grade of schedule in roster for the current nurse
                best grade = a predefined constant indicating a very poor grade
                previous best cost = cost of current roster
                previous roster = current roster
                current schedule = first feasible schedule for current nurse
                while more feasible schedules for the current nurse and current nurse not unconstrained   {
                        remove existing schedule in current roster for current nurse
                        insert current schedule into current roster for current nurse
                        new cost = cost of current roster
                        if current roster solution does not repeat a previous stuck solution   {
                                new grade = current schedule grade
                                if (new cost = previous best cost) and (new grade <= previous grade)          {
                                        if new grade < previous grade    {
                                                number of candidate schedules = 0
                                                previous grade = new grade
                                        }
                                        add schedule to set of candidate schedules
                                        number of candidate schedules = number of candidate schedules + 1
                                }
                                else if  (new cost < previous best cost) and  (new grade <= previous best grade)
        {
                                        number of candidate schedules = 0
                                        previous best cost = new cost
                                        if new grade < previous best grade          {
                                                previous best grade = new grade;
                                                previous grade = new grade;
                                        }
                                        add schedule to set of candidate schedules
                                        number of candidate schedules =  number of candidate schedules + 1
                                }
                        }
                        current schedule = next feasible schedule for current nurse
                }
                if current nurse is unconstrained
                        calculate best schedule for current nurse
                else
                        randomly select a schedule from the set of candidate schedules for current nurse
                insert selected schedule into current roster for current nurse
                new cost = cost of current roster
                current nurse = next nurse on roster
        }
        if current roster = previous roster          {
                if  new cost <= predefined acceptable cost
                        solution found = TRUE
                else        {
                        move roster from minima
                        if current roster = previous roster {
                                stuck counter = stuck counter + 1
                                change order of nurses in roster
                                select schedule with best grade for each nurse to create a new current roster
                        }
                }
        }
}

}
```

Figure 6: Pseudocode for the enhanced cyclic descent algorithm

```
move roster from minima
{
worst shift = shift on roster with highest deviation from shift constraints
worst shift score = numeric measure of deviation for worst shift
previous best cost = predefined large cost
new cost = previous best cost
previous best grade = predefined poor grade
number of candidate schedules = 0
current nurse = first nurse on roster
while more nurses on the roster
{
	original schedule = existing schedule in roster for current nurse
	current schedule = first feasible schedule for current nurse
	while more feasible schedules for current nurse
	{
		if current schedule <> original schedule
		{
			previous cost = cost of current roster
			remove existing schedule in current roster for current nurse
			insert current schedule into current roster for current nurse
			new cost = cost of current roster
			if current roster does not exceed shift threshold score for a previous worst shift
			{
				current worst shift score = score for worst shift in current roster
				if current worst shift score < worst shift score
				{
					new grade = current schedule grade
					if (new cost = previous best cost) and (new grade <= previous best grade)
					{
						add schedule to set of candidate schedules
						if new grade < previous best grade
						{
							previous best grade = new grade;
							number of candidate schedules = 0
						}
						number of candidate schedules =
							number of candidate schedules + 1
					}
					else if new cost < previous best cost
					{
						number of candidate schedules = 0
						add schedule to set of candidate schedules
						number of candidate schedules =
							number of candidate schedules + 1
						previous best cost = new cost
						previous best grade = new grade
					}
				}
			}
			else
				new cost = previous cost
		}
		current schedule = next feasible schedule for current nurse
	}
	replace current schedule in roster for current nurse with original schedule
	current nurse = next nurse in roster
}
if candidate schedules exist
{
	randomly select a candidate schedule
	insert the randomly selected schedule into the current roster
}
}
```

Figure 7: Pseudocode for "move roster from minima" function

4.5 The Basic Simulated Annealing Algorithm

Due to the inherent similarities between simulated annealing and the cyclic descent approach to rostering, a simulated annealing algorithm can be easily inserted into the existing framework of the basic cyclic descent algorithm. Instead of sequentially cycling through each nurse and each schedule, the simulated annealing algorithm randomly selects a nurse and then randomly selects a schedule from that nurse's set of feasible schedules (this approach was found to perform better than randomly selecting schedules directly from the total set of feasible schedules). The selected schedule then replaces the existing schedule for the selected nurse, and a new roster cost is calculated as before. When the roster cost improves, a schedule is automatically accepted, otherwise it is accepted only if a randomly selected probability is less than the probability returned by a given probability function (see Section 2.2.5 and below). In the case of a nurse using the unconstrained schedule generation algorithm (see Section 4.1.4), no set of feasible schedules will exist. To handle this situation in the simulated annealing algorithm, an additional function is used which randomly generates feasible nurse schedules according to the number of shifts the nurse is to work, whether night shifts are allowed and any requests, etc. Pseudocode for the basic simulated annealing algorithm is shown in figure 8. During the actual running of the program to generate test data, the program constants were set at the following values:

minimum score = score obtained by the enhanced cyclic descent algorithm for the same problem, max iterations = 1,000,000, start temperature = 5, chain length = 2000, cooling rate = 0.6

As described in Section 2.2.5, the basic simulated annealing algorithm uses a geometric cooling schedule of the form: $\text{temperature}_n = \text{temperature}_{n-1} * \text{cooling rate}$, where the temperature is decremented after each chain length (i.e. 2000) iterations and also uses a Bolzman probability distribution of the form: $e^{\frac{-change\_in\_cost}{temperature}}$.

```
basic simulated annealing
{
	calculate a set of feasible schedules for each nurse
	calculate the grade of each schedule for each nurse
	select schedule with best grade for each nurse to create an initial current roster
	roster score = current roster deviation score + (current roster grade / 1000)
	temperature = start temperature
	markov chain length = chain length
	loop counter = 0
	while roster score > minimum score and loop counter < max iterations
	{
		previous roster score = roster score
		randomly select current nurse
		previous schedule = existing schedule for current nurse
		if current nurse is unconstrained
			randomly generate current schedule for current nurse
		else
			randomly select current schedule for current nurse
		increment loop counter;
		insert current schedule for current nurse into roster
		calculate new roster deviation score
		calculate new roster grade
		roster score = new roster deviation score + (new roster grade / 1000)
		score change = roster score - previous roster score
		if score change > 0
		{
			if loop counter modulus markov chain length = 0
				temperature = temperature * cooling rate
			acceptance probability = e**-(score change / temperature)
			generate random probability between 0 and 1
			if random probability > acceptance probability
			{
				insert previous schedule for current nurse into roster
				roster score = previous roster score
			}
		}
	}
}
```

Figure 8: Pseudocode for the basic simulated annealing algorithm

4.6 The Enhanced Simulated Annealing Algorithm

In initial trials, the basic simulated annealing algorithm was able to find roster solutions within the fixed constraints of a problem reasonably quickly. However the quality of schedule grade for these solutions was consistently poor in comparison to results from the cyclic descent algorithms. When left to search for comparable solutions, the basic simulated annealing algorithm became very slow and was often unable to find a solution even after several hours of execution.

Following the concept used for the enhanced cyclic descent algorithm, it was decided to introduce a schedule selection bias into the simulated annealing algorithm. Firstly, all schedule grades for each nurse are normalised so that each nurse's minimum schedule grade = 0 and maximum grade = 1. Then, within the annealing algorithm, an additional criterion is added that a schedule is only accepted if the randomly generated probability is also greater than or equal to the normalised schedule grade, or if the schedule has a lower grade than the schedule which it replaces.

In addition, once a zero deviation score is found, a simple cyclic descent algorithm is invoked which repeatedly tries all feasible schedules in the roster until no further improvement in the overall roster grade is possible (named "try all schedules" in the pseudocode below). Once a zero score is found, the algorithm automatically terminates.

The above ideas are expressed in the following pseudocode, which would appear within the main while loop of the basic simulated annealing algorithm (see figure 8):

```
if zero deviation score found
        try all schedules in roster
min = minimum schedule grade for current nurse
max = maximum schedule grade for current nurse
if min < max and current schedule grade > previous schedule grade
        grade probability = 1 - (min + current schedule grade)/(min + max);
else
        grade probability = 1;
if zero deviation score not found and
   (random probability > acceptance probability or
    random probability > grade probability)
{
        insert previous schedule for current nurse into roster
        roster score = previous roster score
}
```

Figure 9: Additional pseudocode for the enhanced simulated annealing algorithm

4.7 Integer Linear Programming Implementation

A detailed description of the mathematical model used in the ILP implementation is provided in Appendix 3. The objective function and constraints described in the model are generated for each roster and stored in a text file using a specially written program. This program takes the feasible schedules as they are represented to the other algorithms and expands them to form a *full* set of feasible schedules. Firstly, schedules containing night shifts are expanded in a reversal of the process described in Section 4.1.2. Then feasible schedules are generated for all nurses categorised as having unconstrained schedules. Using the full feasible schedule set in conjunction with the staffing level constraints and the WeightedSchedule scores for each roster, the system of ILP equations can be constructed.

For the purposes of the research, the ILP model is forced into being a single rather than a two phase process. This is done by setting the desired level for each overall staff constraint equal to the minimum level. Deviations below desired staff levels can therefore not occur and the phase one optimisation becomes redundant (see Appendix 3, Section A3.4). The objective of the model therefore becomes one of minimising overall schedule score subject to the maximum and minimum constraints defined in Section 4.2. This is the same objective set for the other algorithms. Attempts to minimise deviations from desired staffing levels were abandoned for the following reasons:

1. The use of a two phase solution doubles the necessary execution time for the ILP algorithm.
2. It was judged more important to obtain a better quality allocation of schedules within the upper and lower constraints, than to obtain a marginally better allocation of shifts at the expense of schedule quality.

4.8 Summary

A major objective of the research was to find an efficient form of problem representation. It was decided to formulate the rostering task as one of finding the best combination of feasible schedules. This makes the problem suitable for an optimising algorithmic approach. The drawback to such an approach is the large computer memory resource required to hold all feasible schedules for a realistic rostering scenario. Therefore several techniques were devised to reduce the number of feasible schedules needed to express the problem. Firstly, the allocation of late and early shifts was considered as a separate problem. Secondly, night shift representations were simplified by allowing the cost function to eliminate unnecessary nights. Finally an algorithm was developed to calculate schedules for part-time nurses able to work flexible schedules.

Two techniques suggested by the literature were implemented in the current research. Firstly, Miller *et al.'s* cyclic descent algorithm (1976) was adapted to run within feasible schedule representation system previously described. The basic algorithm was then enhanced by adding a hill-climbing heuristic, a schedule selection bias and repeatedly running the algorithm from different starting positions. Secondly, a basic simulated annealing algorithm was developed using standard techniques described in the literature. The algorithm was then enhanced by adding a probabilistic bias towards selecting better grade schedules, and by invoking a cyclic descent algorithm to terminate each run.

As a result of programming implementation work, four algorithms have been created:

1. Basic cyclic descent algorithm
2. Enhanced cyclic descent algorithm
3. Basic simulated annealing algorithm
4. Enhanced simulated annealing algorithm

In addition, a program was developed to convert the roster problem into a system of linear equations suitable for solution by an existing branch and bound ILP algorithm. The remainder of the research tests the results obtained from these algorithms using a set of manually generated rosters supplied by the Gold Coast Hospital.

Chapter 5: Results

This chapter presents the results of the statistical analysis outlined in Chapter 3. The various roster generation methods are evaluated using three factorial Multiple Analysis of Variance (MANOVA) designs. Any significant main effects are further investigated by univariate tests of significance and then by testing the significance of differences between individual means. A detailed listing of results for each MANOVA analysis is provided in Appendix 7.

5.1 Raw Data Analysis

The raw data was generated for six roster generation methods over two hospital wards. Each method was presented with 52 rosters, from which overall values of WeightedShift, WeightedSchedule were obtained for each roster solution. In addition ExecutionTime values were calculated for the computerised roster solutions. As expected, all methods except the integer linear programming algorithm (ILP Method 6) were able to process the full data set. The ILP algorithm was able to solve 34 of the 52 rosters, including all 26 rosters from Ward 1.

5.1.1 Data Transformations

An examination of the raw data indicated problems with outliers and non-normality for all dependent variable distributions. As approximately normal data is a requirement for MANOVA (Tabachnick and Fidell 1989), the following transformations were performed (see Section 5.1.4 for a further discussion):

- Transformed WeightedShift = $\log_e(2 + \text{WeightedShift})$
- Transformed WeightedSchedule = $\sqrt{\text{WeightedSchedule}}$
- Transformed ExecutionTime = $\sqrt{\log_e(2 + \text{ExecutionTime})}$

5.1.2 Platform Adjustment Factor

An additional data transformation used was to multiply the ExecutionTime values for the ILP algorithm by the platform adjustment factor (see Section 3.3.3). The evaluation of the adjustment factor was not straightforward. The Unix operating system provides three measures of execution time for a given program or process. These are:

1. Real Time : the actual time the program takes to run on the network.
2. User Time : the time spent in execution of the program.
3. Sys Time : the time spent in execution of system commands.

In calculating the platform adjustment factor, the user time was used. However, this is probably a conservative measure. Larger problems (in excess of 5,000 variables) tended to use proportionally more real time to execute than smaller problems on the Unix system. This may have been because the problems were too large to hold in memory at one time, resulting in frequent disk read and write operations. Given limited RAM resources, it would be expected that a similar program operating on an IBM® compatible machine running under Windows® would encounter the same lengthening of execution times.
Bearing this qualification in mind, a series of different sized problems were run on both the IBM® PC and Sun® systems using the same program. After averaging the results a platform adjustment factor of 4.171 was derived. This means, on average, a roster calculating program taking 1 second of user time to execute on the Sun® network will take 4.171 seconds to execute on the stand-alone IBM® PC used in the research. All ILP ExecutionTimes used in the subsequent analysis have been calculated using the following formula:

- ILP ExecutionTime = 4.171 * User time recorded by Unix operating system in seconds

5.1.3 Evaluation of MANOVA Assumptions

After transformation, the raw data was evaluated against the basic requirements for a MANOVA analysis (Tabachnick and Fidell 1989) :

Unequal Sample Sizes and Missing Data: No results were obtained for eighteen of the Ward 2 rosters from the ILP algorithm. For sixteen of these cases the problem size became too large for the available computer memory (> 20,500 variables), for one case the problem was rejected as infeasible and a final case remained unsolved after four days of processing and so was terminated. To avoid problems with missing data, a separate MANOVA 3 analysis was devised which considers only those rosters which the ILP algorithm was able to solve. This resulted in a 2 x 5 factorial model, representing the two wards and the five rostering algorithms used in the study (MANOVA 3 does not consider the manual roster data). The elimination of unsolved rosters from the analysis, means that cell sizes are no longer equal: the five cells relating to Ward 2 having eight cases each whilst Ward 1 cells have a full set of 26 cases. Unequal cell sizes are allowable in MANOVA, if there are more cases than dependent variables (DVs) in every cell (Tabachnick and Fidell 1989, p. 377). As MANOVA 3 uses three DVs, a minimum cell size of eight is sufficient. In addition, MANOVA requires homogeneity of variance-covariance matrices for each cell so that a reliable pooled estimate of error can be calculated (Tabachnick and Fidell 1989, pp. 378-379). If cell sizes are equal, homogeneity can be assumed. However, with unequal cell sizes an additional test is required. To this end, a Boxes *M* test was employed on the MANOVA 3 data and found to be *not* significant with $p > .001$. This indicates that the variance-covariance matrices for each cell can be considered homogeneous. It is therefore concluded that the MANOVA 3 model is robust with respect to unequal cell sizes.

Multicollinearity and Singularity: Tests for multicollinearity and singularity were made via an examination of the correlation matrices for each cell of each MANOVA design. As no correlations greater than 0.9 were found, and general correlations were less than 0.4, it is concluded that multicollinearity and singularity are not a problem for the current designs.

Multivariate Normality, Linearity and Outliers: MANOVA is considered "robust to modest violations of normality so long as the violations are not caused by outliers" (Tabachnick and Fidell 1989, p. 378). The data transformations described in Section 5.1.1 were used both to normalise the distributions of the dependent variables and to reduce the effects of outliers. In order to assess the transformations, a cell by cell analysis was performed for each design used in the research. Factors considered for each cell were:

- Multivariate outliers: these were measured using the Mahalanobis distance for each case, with the probability of a case being an outlier set at $p < .001$.
- Univariate outliers: these were defined as any case deviating by more than $\pm$ 3 standard deviations from a dependent variable cell mean.
- Normality: the normality of each cell dependent variable was assessed using the Kolmogoroff-Smirnoff goodness of fit test with $p < .05$. Skewness and kurtosis were also considered and graphical analysis performed on suspicious distributions.
- Linearity: any cell variables with suspected non-normal distributions were further analysed for linearity using within cell bi-variate scatter plots.

The cell by cell analysis revealed one outlier in the MANOVA 1 model. This was a low transformed WeightedShift score for a manually solved roster (Ward 2, 1/2/93, outlier score = 0, next highest score = 11, total score range 0 to 379). It was decided to rescore the untransformed WeightedShift score, rather than delete the case, so that equality of cell sizes could be maintained (as suggesting in Tabachnick and Fidell 1989, p. 70). Due to the logarithmic transformation of the WeightedShift scores, the effect of low scoring outliers is considerably magnified. Therefore a relatively small increase in the untransformed score from 0 to 9 was sufficient to cure the outlier problem in the transformed score (this increased from 0.6931 to 2.3979 in a range of 5.2497).

Of the 74 within cell distributions examined, 5 were found to deviate from normal (Kolmogoroff-Smirnoff $p < .05$). No outliers were found in these distributions, so the deviations were considered acceptable. In addition, bi-variate scatter plots for the non-normal variables did not reveal any significant non-linear relationships.

The generally normal univariate distribution of dependent variables and the robustness of MANOVA to violations of normality indicate that the assumption of multivariate normality can be accepted. Problems with outliers have been eliminated by data transformations and the rescoring of one case. Additionally, no evidence was found of non-linear relationships between variables. It is therefore concluded that the transformed data set is suitable for MANOVA analysis.

5.1.4 Raw and Transformed Data Values

The following tables give a comparison between the raw and transformed mean criteria scores for each ward and method, with their associated skewness and kurtosis measures. The transformations were developed on a trial and error basis and were based on transformation functions recommended in Tabachnick and Fidell (1989, pp. 83-87):

WeightedShift before and after Transformation ($\log_e(2+x)$)						
	Mean	→ after	Skewness	→ after	Kurtosis	→ after
Enhanced Cyclic	42.9423	3.2510	2.9754	-0.2485	11.6999	-0.0789
Basic Cyclic	87.6538	4.0318	1.4101	-0.1407	0.9826	-0.5526
Manual	109.8654	4.3482	1.2714	-0.1980	0.8975	-0.5629
Basic Annealing	43.5385	3.2481	2.8666	-0.2295	10.8324	-0.1865
Enhanced Annealing	44.5385	3.2585	2.8758	-0.0187	8.9139	-0.0293
ILP	53.4118	3.5328	2.5969	-0.0897	8.4961	-0.0643
Ward 1	82.8707	3.9121	1.6418	0.0662	1.8855	-0.8331
Ward 2	51.5288	3.3429	2.6292	-0.3675	8.0600	-0.3194

Table 13: Mean scores for WeightedShift

WeightedSchedule before and after Transformation (√x)						
	Mean	→ after	Skewness	→ after	Kurtosis	→ after
Enhanced Cyclic	13.4455	3.6403	0.1266	-0.2461	0.0700	0.1835
Basic Cyclic	13.6537	3.6688	0.2820	-0.1034	0.5088	0.1586
Manual	18.2310	4.2480	-0.0803	-0.4143	0.1540	0.4491
Basic Annealing	13.7953	3.6889	0.0596	-0.3565	0.4248	0.5471
Enhanced Annealing	13.6400	3.6680	0.0108	-0.4542	0.5020	0.9785
ILP	12.4877	3.5084	0.5499	0.1656	0.6766	0.2795
Ward 1	15.3887	3.8983	0.6996	0.3595	0.5773	0.2307
Ward 2	13.7175	3.6673	0.2183	-0.1371	-0.2639	-0.2845

Table 14: Mean scores for WeightedSchedule

ExecutionTime before and after Transformation ($\sqrt{\log_e(2+x)}$)						
	Mean	→ after	Skewness	→ after	Kurtosis	→ after
Enhanced Cyclic	452.6352	2.2122	3.6551	-0.1085	14.1130	-0.6863
Basic Cyclic	18.9381	1.7059	0.8533	-0.4527	0.4798	-0.0332
Basic Annealing	6419.776	2.8665	0.3378	-0.8261	-1.1919	-0.4477
Enhanced Annealing	1436.712	2.4239	2.9966	0.6193	9.0278	-0.4612
ILP	3381.498	2.4037	5.3868	0.3361	30.1000	0.0557
Ward 1	1971.534	2.2823	2.1201	0.0441	3.4505	-1.1133
Ward 2	2192.496	2.3219	2.0748	0.0123	3.2831	-1.2713

Table 15: Mean scores for ExecutionTime

5.2 MANOVA 1 Results

MANOVA 1 is designed to measure differences between all methods, except ILP, in the dimensions of WeightedSchedule and WeightedShift. A 5 x 2 factorial MANOVA design is used, representing the five methods and two wards included in the model. All five methods were able to find solutions to the sample of 52 rosters, resulting in 26 cases per cell, and 260 cases in total.

The following table and graphs show the mean schedule and shift score values for each method and ward. In all cases a lower mean score represents a better quality solution.

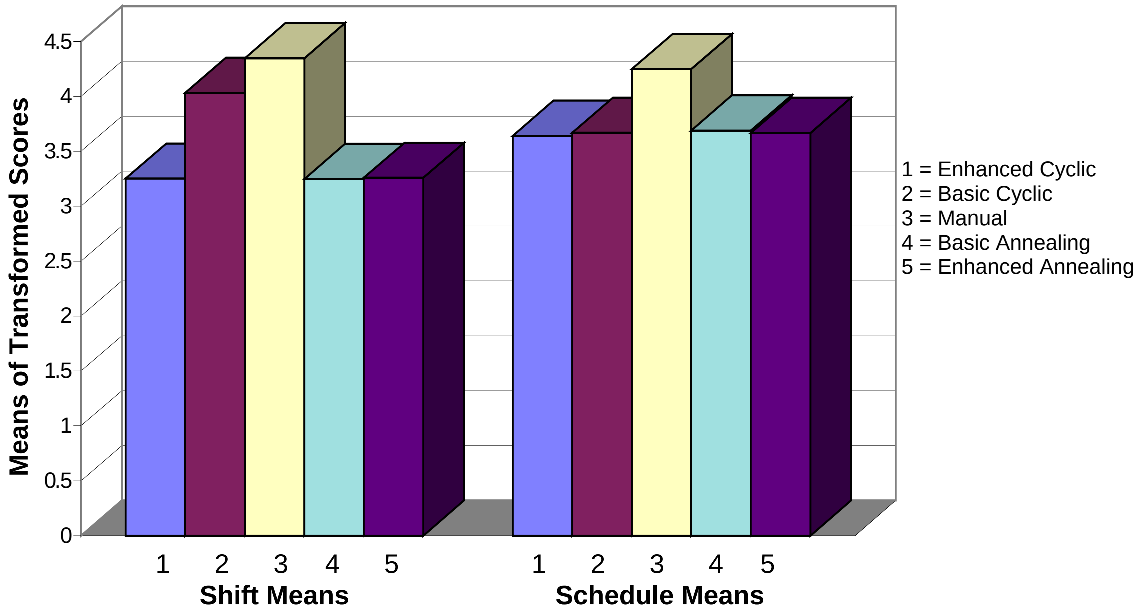


Figure 10: MANOVA 1 Comparison of method means

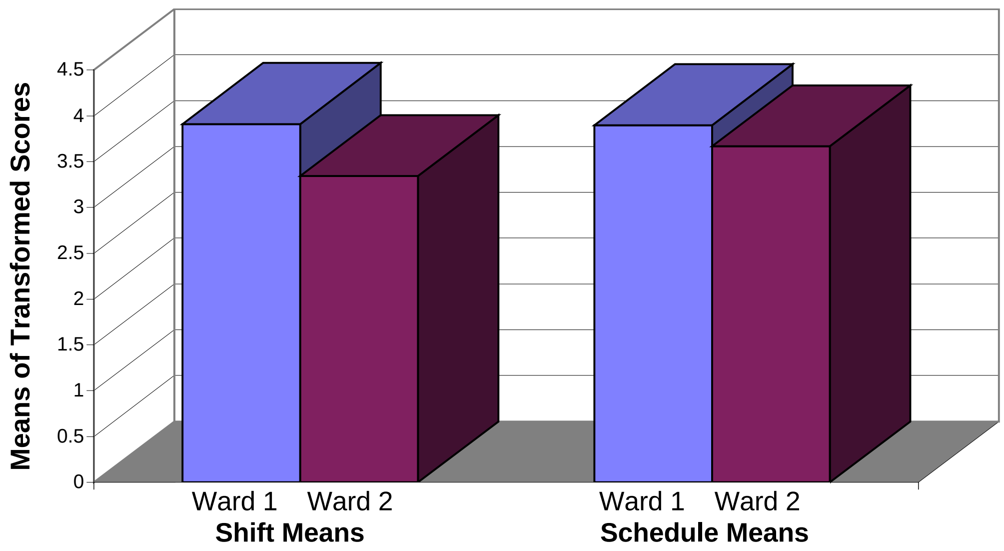


Figure 11: MANOVA 1 Comparison of ward means

	Shift Mean ± Standard Error ($\log_e$(2 + WeightedShift))	**Schedule Mean ± Standard Error** (√WeightedSchedule)
Enhanced Cyclic	3.25 ± 0.16	3.64 ± 0.06
Basic Cyclic	4.03 ± 0.14	3.67 ± 0.06
Manual	4.35 ± 0.13	4.25 ± 0.06
Basic Annealing	3.25 ± 0.16	3.69 ± 0.06
Enhanced Annealing	3.26 ± 0.15	3.67 ± 0.06
Ward 1	3.91 ± 0.09	3.90 ± 0.04
Ward 2	3.34 ± 0.10	3.67 ± 0.05

Table 16: MANOVA 1 Mean Data

The mean data results confirm the expectation that the manual method would score more highly than the computerised methods for both shift and schedule scores. Also, as expected, the basic cyclic descent algorithm has a generally higher shift score than the other algorithms. The remainder of the MANOVA 1 analysis investigates which of these differences between means are statistically significant. Table 17 gives the multivariate tests of significance for the model, showing that that both Method and Ward effects are significant and that there is no interaction effect between Method and Ward.

Effect	**Pillais** (sig of F)	**Hotellings** (sig of F)	**Wilks** (sig of F)
Method	0.35025 (0.000)	0.46463 (0.000)	0.66937 (0.000)
Ward	0.11410 (0.000)	0.12880 (0.000)	0.88590 (0.000)
Method by Ward	0.02038 (0.741)	0.02078 (0.741)	0.97963 (0.741)

Table 17: MANOVA 1 Multivariate tests of significance

Following from the multivariate tests of significance, univariate tests of significance were conducted to discover which dependent variables are responsible for the multivariate effect. These tests are shown in table 18, and indicate that shift and schedule scores differ between methods and that shift and schedule scores differ between wards ($p < .05$)

Effect	Variable	F-value	Significance of F
Method	WeightedShift	13.77009	0.000
Method	WeightedSchedule	19.32899	0.000
Ward	WeightedShift	18.96420	0.000
Ward	WeightedSchedule	20.18990	0.000

Table 18: MANOVA 1 Univariate tests of significance

To complete the MANOVA 1 analysis, a series of contrasts were made between the mean of the enhanced cyclic descent method and all other methods for both shift and schedule scores. A significant contrast is recognised if the joint univariate 95% Bonferroni confidence interval does *not* pass through zero. The Bonferroni confidence interval compensates for the inflated Type 1 error rate caused by making multiple contrasts (Tabachnick and Fidell 1989). This results in an overall α level of 0.05. The mean contrasts are shown in the following table (the final two columns showing the Bonferroni intervals) :

Variable	1st Mean	2nd Mean	t-value	significance of t	lower c-level	upper c-level
Weighted Shift	Enhanced Cyclic	Basic Cyclic	3.89853	0.00012	0.27691	1.28469
		Manual	5.47800	0.00000	0.59325	1.60103
		Basic Annealing	-0.01478	0.98822	-0.50685	0.50093
		Enhanced Annealing	0.03730	0.97027	-0.49642	0.51136
Weighted Schedule	Enhanced Cyclic	Basic Cyclic	0.33899	0.73490	-0.18252	0.23936
		Manual	7.24798	0.00000	0.39674	0.81862
		Basic Annealing	0.57856	0.56341	-0.16243	0.25945
		Enhanced Annealing	0.32962	0.74197	-0.18330	0.23857

Table 19: MANOVA 1 Contrasts with 95% Bonferroni confidence intervals

Using the Bonferroni confidence intervals we can conclude that the mean value of the enhanced cyclic descent algorithm for WeightedShift is significantly different from the mean values of WeightedShift for both the basic cyclic descent algorithm and the manual method. In addition, we can conclude that the mean value of the enhanced cyclic descent algorithm for WeightedSchedule is significantly different from the WeightedSchedule mean of the manual method. These differences are expressed in the following scatter plots:

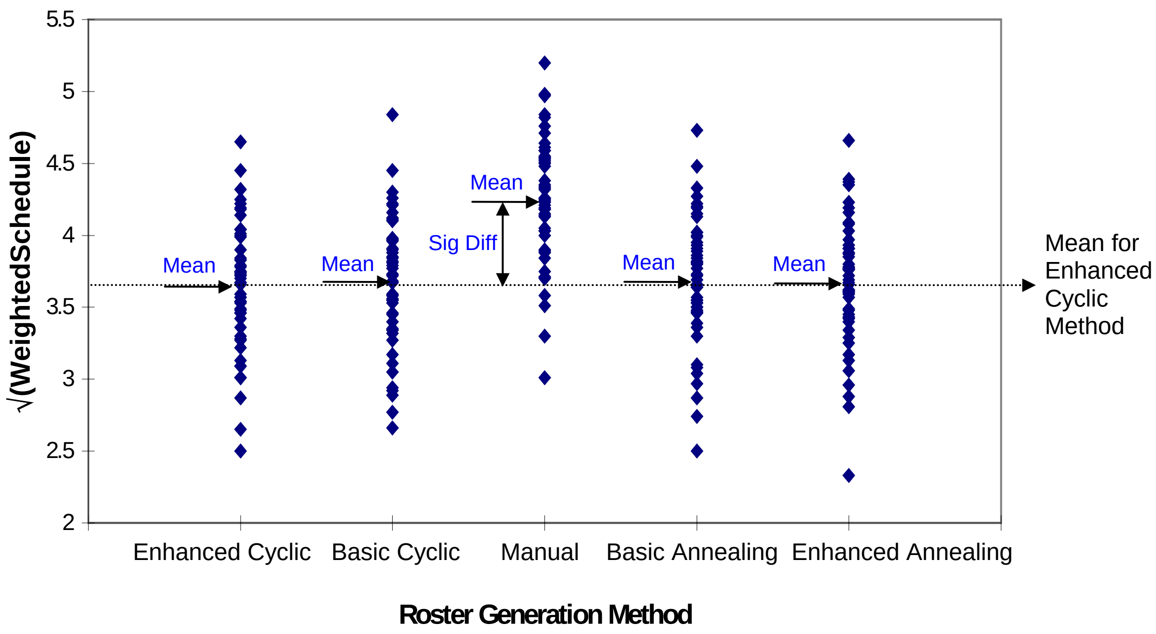


Figure 12: MANOVA 1 Scatter Plot of schedule scores for each method

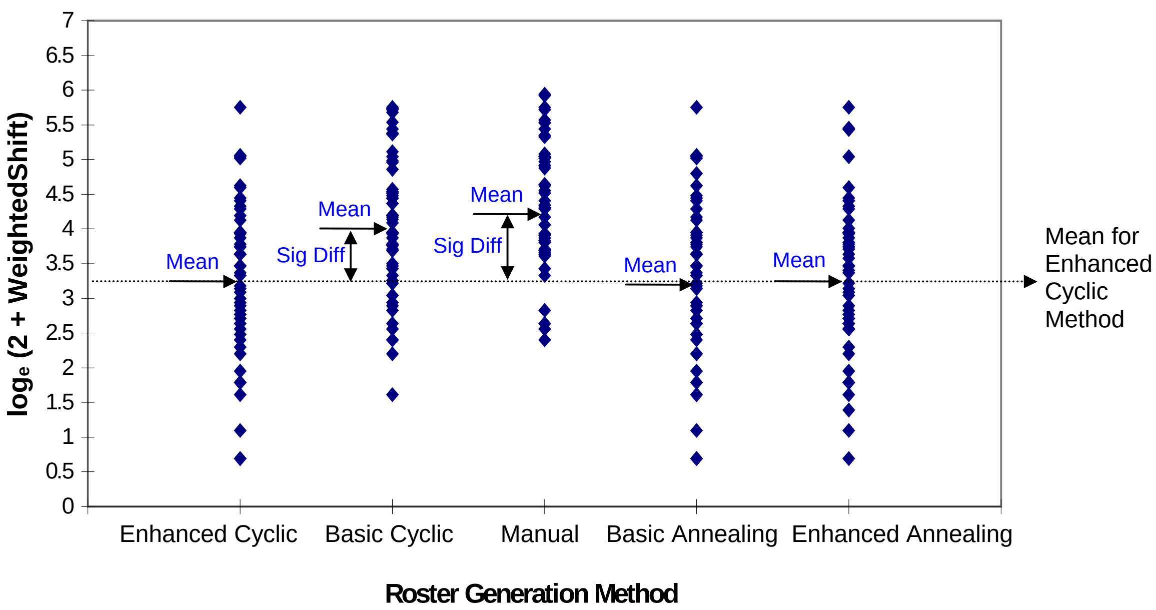


Figure 13: MANOVA 1 Scatter Plot of shift scores for each method

5.3 MANOVA 2 Results

MANOVA 2 is designed to look at differences in execution times between those algorithms that were able to solve all 52 rosters. Manual roster solutions were not included because execution time data was not available, and ILP roster solutions were not included because of missing data for the rosters the algorithm was unable to solve. MANOVA 2 therefore considers the four DOS-based algorithms (the enhanced cyclic descent algorithm, the basic cyclic descent algorithm, the basic simulated annealing algorithm and the enhanced simulated annealing algorithm). All three criteria measures are included in the model, but only differences in ExecutionTime are considered in detail. Differences in WeightedShift and WeightedSchedule have already been analysed in MANOVA 1. A 4 x 2 factorial MANOVA design is used, representing the four methods and two wards included in the model. Each cell in the design contains 26 cases, resulting in a total of 208 cases.

The following graphs and table summarise the mean criteria values for each ward and method (again a lower value indicates a better quality solution) :

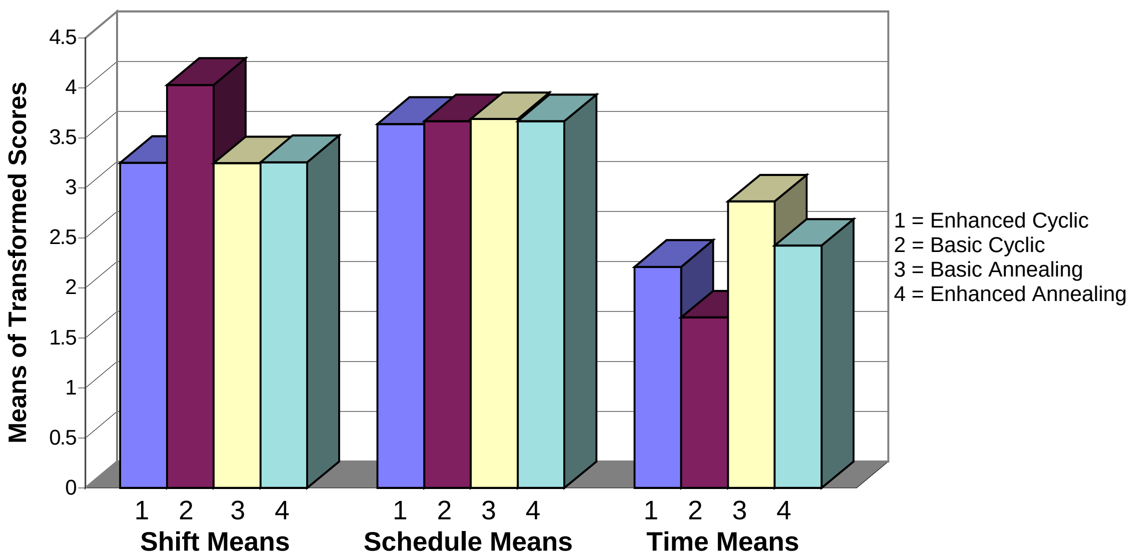


Figure 14: MANOVA 2 Comparison of method means

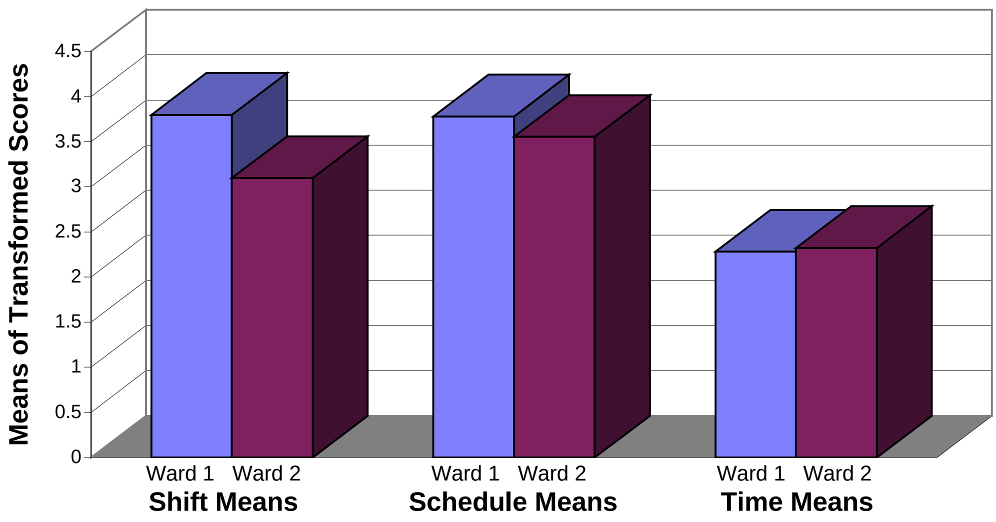


Figure 15: MANOVA 2 Comparison of ward means

	Shift Mean ± Standard Error $\log_e(2+\text{WeightedShift})$	**Schedule Mean ± Standard Error** $\sqrt{\text{WeightedSchedule}}$	**Time Mean ± Standard Error** $\sqrt{\log_e(2+\text{ExecutionTime})}$
Enhanced Cyclic	3.25 ± 0.16	3.64 ± 0.06	2.21 ± 0.05
Basic Cyclic	4.03 ± 0.14	3.67 ± 0.06	1.71 ± 0.02
Basic Annealing	3.25 ± 0.16	3.69 ± 0.06	2.87 ± 0.03
Enhanced Annealing	3.26 ± 0.15	3.67 ± 0.06	2.42 ± 0.04
Ward 1	3.80 ± 0.10	3.78 ± 0.04	2.28 ± 0.05
Ward 2	3.10 ± 0.11	3.28 ± 0.05	2.32 ± 0.05

Table 20: MANOVA 2 Mean Data

The shift and schedule mean results for each method are the same as those reported for MANOVA 1. New shift and schedule means have been obtained for each ward due to the removal of the manual method from the MANOVA 2 analysis. However, these ward mean values repeat the relative size and direction of difference observed in the previous analysis. Of more interest are the mean time values for each level of independent variable. As hypothesised, the basic cyclic descent algorithm has the fastest mean execution time. Also, as expected, both simulated annealing algorithms execute more slowly than the enhanced cyclic descent algorithm. Finally, a relatively small difference in mean execution time is recorded between wards.

The remainder of the MANOVA 2 analysis examines whether the differences between execution time means are statistically significant. Also, the new shift and schedule means for each ward are examined to see if the MANOVA 1 results are confirmed. Table 21 shows the multivariate tests of significance for the model. As with MANOVA 1, the tests show both Method and Ward effects are significant and that there is no interaction effect between Method and Ward.

Effect	**Pillais** (sig of F)	**Hotellings** (sig of F)	**Wilks** (sig of F)
Method	0.75945 (0.000)	2.69859 (0.000)	0.26403 (0.000)
Ward	0.14454 (0.000)	0.16896 (0.000)	0.85546 (0.000)
Method by Ward	0.00730 (0.997)	0.00733 (0.998)	0.99271 (0.997)

Table 21: MANOVA 2 Multivariate tests of significance

Univariate tests of significance were conducted to discover which dependent variables were responsible for the multivariate effects. These tests are shown in table 22. Firstly, WeightedShift and ExecutionTime are shown to differ between methods, whereas WeightedSchedule does not. Secondly, WeightedShift and WeightedSchedule are shown to differ between wards, whereas ExecutionTime does not (p > .05). Given these results we can conclude there is no significant difference between wards for ExecutionTime and that the significance of differences between wards for WeightedSchedule and WeightedShift are the same as those reported for MANOVA 1.

Effect	**Variable**	**F-value**	**Significance of F**
Method	WeightedShift	7.22767	0.000
Method	WeightedSchedule	0.11209	0.953
Method	ExecutionTime	172.77560	0.000
Ward	WeightedShift	23.25399	0.000
Ward	WeightedSchedule	14.63097	0.000
Ward	ExecutionTime	1.16362	0.282

Table 22: MANOVA 2 Univariate tests of significance

In comparing WeightedShift and WeightedSchedule means for each method, MANOVA 2 repeats the findings of MANOVA 1 (i.e. the Basic Cyclic Descent algorithm is found to be significantly different for the criteria of WeightedShift). To investigate the significance of differences in ExecutionTime between methods, further contrasts were made between the mean of the enhanced cyclic descent method and the means for the other three methods. These are shown in the following table (the final two columns showing the Bonferroni intervals) :

Variable	**1st Mean**	**2nd Mean**	**t-value**	**significance of t**	**lower c-level**	**upper c-level**
Execution Time	Enhanced Cyclic	Basic Cyclic	-9.76211	0.00000	-0.63143	-0.38103
		Basic Annealing	12.61874	0.00000	0.52917	0.77956
		Enhanced Annealing	4.08317	0.00006	0.08654	0.33694

Table 23: MANOVA 2 Contrasts with 95% Bonferroni confidence intervals

Using the 95% Bonferroni confidence intervals (overall $\alpha = .05$), we can conclude that the mean value of the enhanced cyclic descent algorithm for ExecutionTime is significantly different from the mean values of ExecutionTime for all methods tested. These differences and their direction are expressed in the following scatter plot:

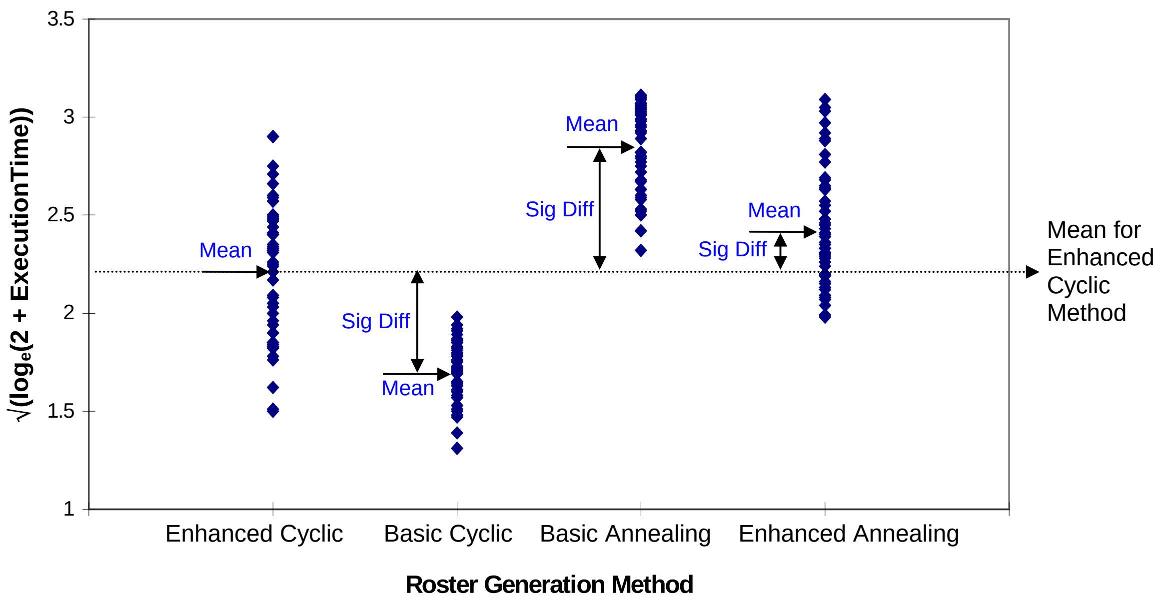


Figure 16: MANOVA 2 Scatter Plot of execution times for each method

5.4 MANOVA 3 Results

MANOVA 3 compares results for all five computerised methods, considering only those rosters which the ILP algorithm was able to solve. For these rosters, the ILP solutions provide an optimum measure of shift and schedule quality from which the other methods can be assessed. The speed of execution for the ILP algorithm is also of interest. Therefore all three criteria measures are included in the analysis.

The ILP algorithm found solutions for twenty-six Ward 1 rosters and eight Ward 2 rosters. The remaining eighteen Ward 2 rosters were unsolved for the following reasons:

1. Sixteen rosters were unable to be processed because the ILP matrices became too large to hold in memory. The largest problem to be successfully solved contained 20,325 variables (= 20,325 feasible schedules).
2. One roster problem was allowed to run for 4 days without finding a solution and was then terminated.
3. One roster problem was rejected as infeasible by the ILP software, although feasible solutions to this problem were found by the other algorithms.

A 5 x 2 factorial MANOVA design is used. Of the ten cells in the model, five contain 26 cases each and five contain 8 cases each, making 170 cases in total. Issues relating to unequal cell sizes are discussed in Section 5.1.3. Firstly, the relative mean criteria values for each ward and method are summarised in the following graphs and table:

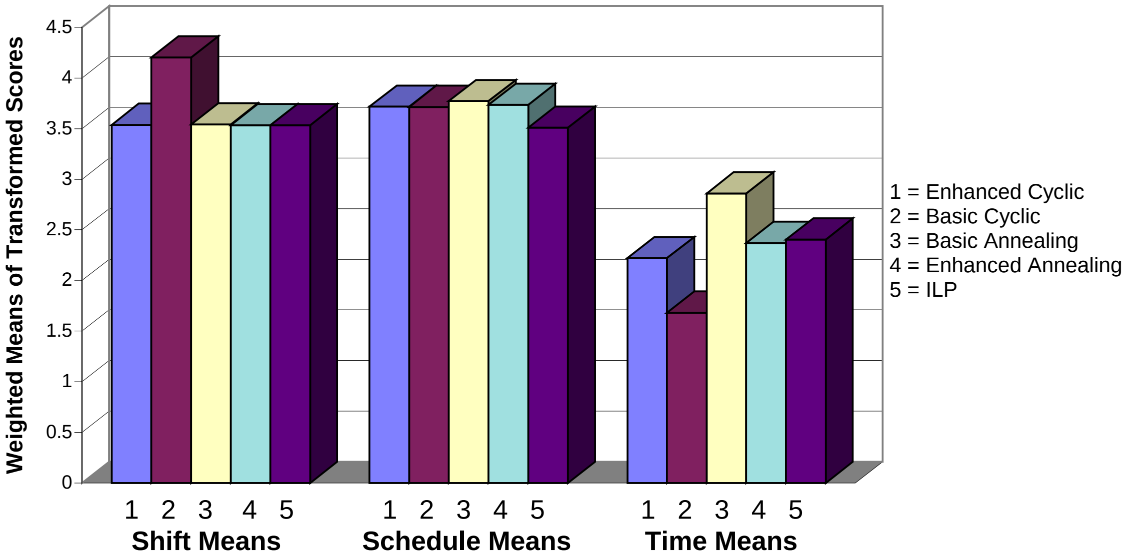


Figure 17: MANOVA 3 Comparison of method means

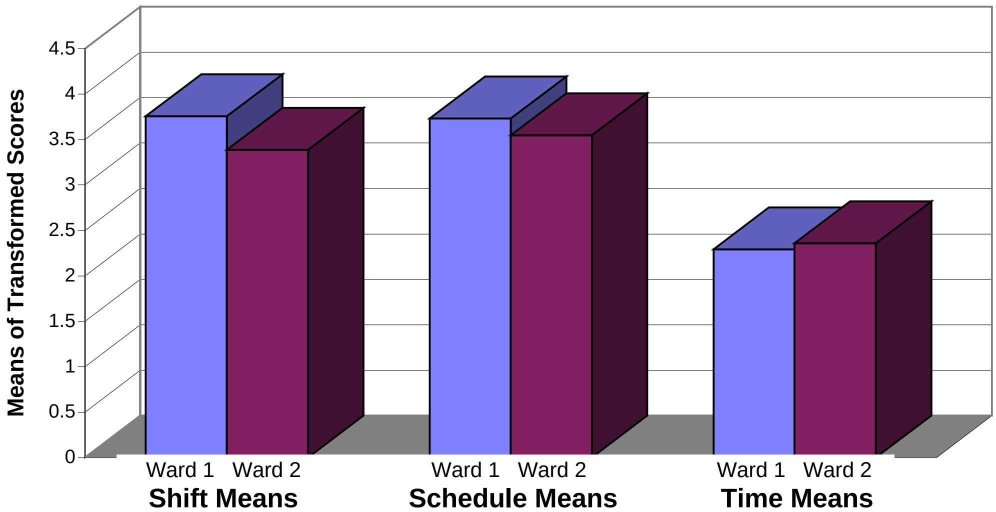


Figure 18: MANOVA 3 Comparison of ward means

	Shift Mean ± Standard Error $\log_e(2+\text{WeightedShift})$	**Schedule Mean ± Standard Error** $\sqrt{\text{WeightedSchedule}}$	**Time Mean ± Standard Error** $\sqrt{\log_e(2+\text{ExecutionTime})}$
Enhanced Cyclic	3.54 ± 0.17	3.72 ± 0.06	2.22 ± 0.06
Basic Cyclic	4.20 ± 0.17	3.71 ± 0.07	1.68 ± 0.03
Basic Annealing	3.54 ± 0.18	3.77 ± 0.06	2.86 ± 0.04
Enhanced Annealing	3.53 ± 0.18	3.73 ± 0.06	2.37 ± 0.05
ILP	3.53 ± 0.18	3.51 ± 0.07	2.40 ± 0.07
Ward 1	3.76 ± 0.09	3.73 ± 0.03	2.29 ± 0.04
Ward 2	3.39 ± 0.16	3.55 ± 0.06	2.36 ± 0.08

Table 24: MANOVA 3 Weighted Mean Data

An inspection of the mean data values indicates the differences between wards and methods are in line with results from MANOVA 1 and MANOVA 2. Of interest are the new results for the ILP method. As expected, the ILP algorithm has generated the lowest mean WeightedSchedule score, whilst matching the best WeightedShift score of the other methods. ILP ExecutionTime is second slowest being faster than the basic simulated annealing algorithm but slower than the enhanced simulated annealing algorithm.

The remainder of the MANOVA 3 analysis tests the observed differences between means to see which are statistically significant. Table 25 shows the multivariate tests of significance for the model. As with MANOVA 1 and MANOVA 2, the tests show both Method and Ward effects are significant and that there is no interaction effect between Method and Ward ($p > .05$).

Effect	**Pillais** (sig of F)	**Hotellings** (sig of F)	**Wilks** (sig of F)
Method	0.72649 (0.000)	1.90997 (0.000)	0.32592 (0.000)
Ward	0.05740 (0.025)	0.06090 (0.025)	0.94260 (0.025)
Method by Ward	0.03752 (0.911)	0.03876 (0.911)	0.96258 (0.911)

Table 25: MANOVA 3 Multivariate tests of significance

Univariate tests of significance were conducted to discover which dependent variables were responsible for the multivariate effects. These tests are shown in table 26. Firstly, WeightedShift and ExecutionTime are shown to differ between methods, whereas WeightedSchedule does not. Secondly, WeightedShift and WeightedSchedule are shown to differ between wards, whereas ExecutionTime does not ($p > .05$). Given these results we can conclude there is no significant difference between wards for ExecutionTime and that the differences between wards for WeightedSchedule and WeightedShift are the same as those reported for MANOVA 1.

Effect	Variable	F-value	Significance of F
Method	WeightedShift	2.90218	0.024
Method	WeightedSchedule	2.50178	0.045
Method	ExecutionTime	71.08434	0.000
Ward	WeightedShift	4.01229	0.047
Ward	WeightedSchedule	7.11016	0.008
Ward	ExecutionTime	1.51221	0.221

Table 26: MANOVA 3 Univariate tests of significance

Multiple univariate F-tests cause an inflated Type 1 error rate. In order to compensate for this error, a Bonferroni type adjustment was used (Tabachnick and Fidell 1989, p. 399). For an overall α of 0.05, this results in individual dependent variable α levels of 0.017. Using the adjusted α level, it can be seen from Table 26 that ExecutionTime means differ significantly between methods and that WeightedSchedule means differ significantly between wards. For all other means no significant difference is observed. It can therefore be concluded that (as before) the two wards have significantly different WeightedSchedule mean scores. However, with the smaller sample size, there is no longer a significant difference between each ward's WeightedShift mean score. In comparing each method, it is possible for no overall difference to be observed between means, but for a significant difference to exist between a pair of means. Therefore an additional contrast analysis was performed considering all dependent variable means for each method. As before, the enhanced cyclic descent algorithm mean is taken as the base from which the other means are contrasted. This is shown in the following table (the final two columns giving the Bonferroni intervals) :

Variable	1st Mean	2nd Mean	t-value	significance of t	lower c-level	upper c-level
Weighted Shift	Enhanced Cyclic	Basic Cyclic	2.13673	0.03414	-0.11368	1.36078
		Basic Annealing	-0.05175	0.95879	-0.75233	0.72213
		Enhanced Annealing	-0.30160	0.76335	-0.82524	0.64921
		ILP	-0.00880	0.99299	-0.73980	0.73466
Weighted Schedule	Enhanced Cyclic	Basic Cyclic	-0.28653	0.77485	-0.30690	0.24437
		Basic Annealing	0.30909	0.75765	-0.24191	0.30936
		Enhanced Annealing	-0.03162	0.97482	-0.27908	0.27218
		ILP	-1.90768	0.05822	-0.48377	0.06749
Execution Time	Enhanced Cyclic	Basic Cyclic	-6.45642	0.00000	-0.75252	-0.32925
		Basic Annealing	7.74756	0.00000	0.43741	0.86068
		Enhanced Annealing	1.55211	0.12261	-0.08161	0.34166
		ILP	3.05721	0.00262	0.04448	0.46775

Table 27: MANOVA 3 Contrasts with 95% Bonferroni confidence intervals

Looking at those 95% Bonferroni confidence intervals that pass through zero we can conclude, as indicated by the univariate F-tests, that there are no significant differences between methods for WeightedSchedule and WeightedShift mean scores. Conversely, for mean ExecutionTime scores we can conclude that all methods are significantly different from the enhanced cyclic descent algorithm *except* the enhanced simulated annealing algorithm. These results are summarised in the following two scatter plots:

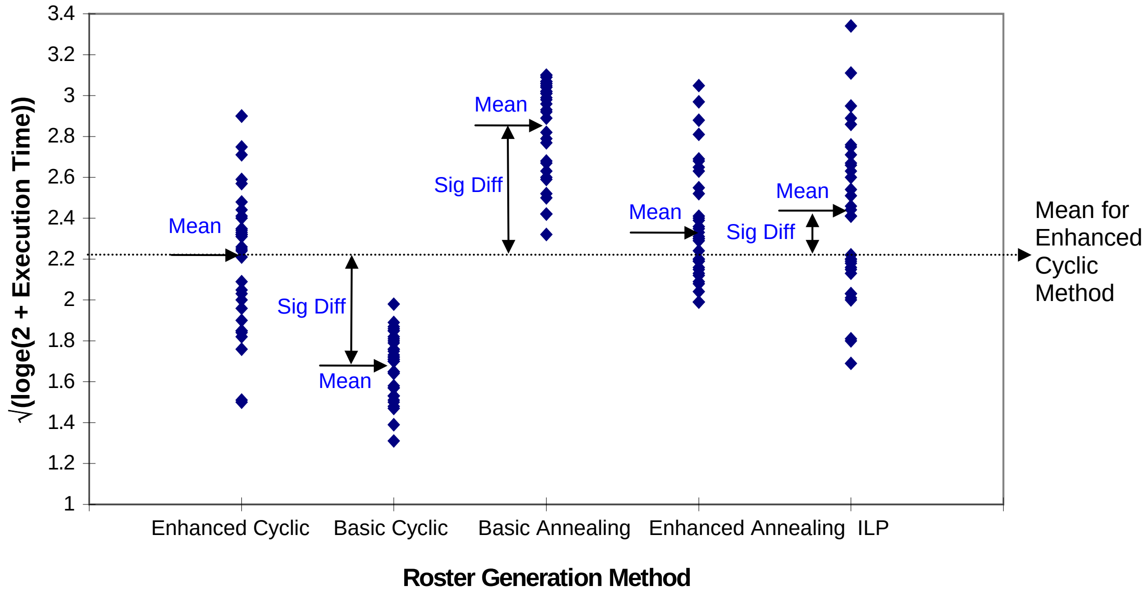


Figure 19: MANOVA 3 Scatter Plot of execution times for each method

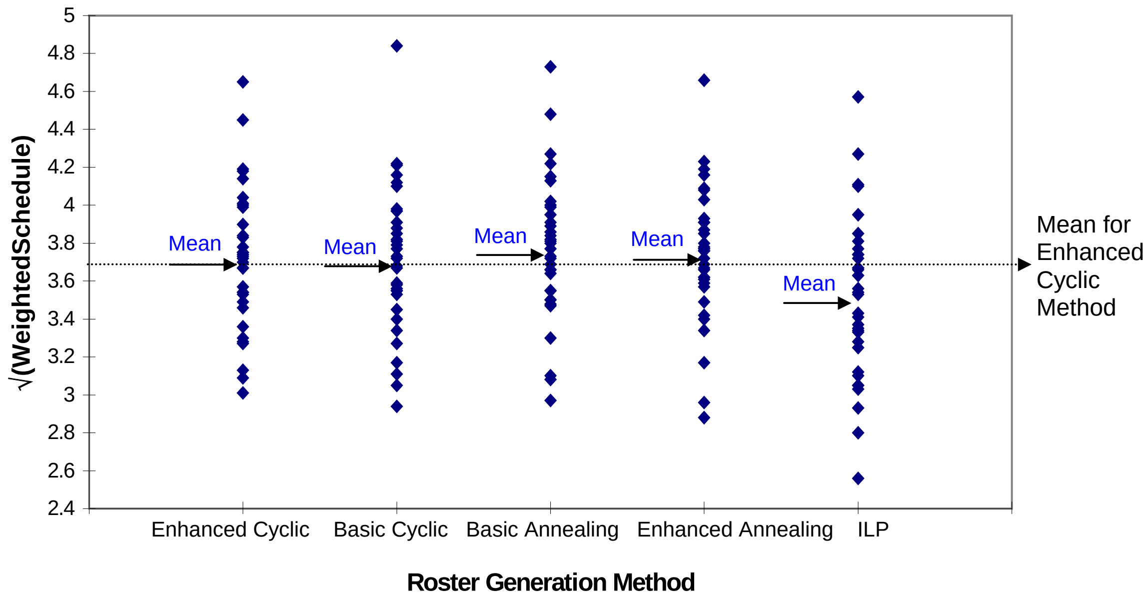


Figure 20: MANOVA 3 Scatter Plot of schedule scores for each method

5.5 Supplementary Results

5.5.1 Differences Between Wards

The previous sections have reported significant differences between wards for both WeightedSchedule and WeightedShift mean scores. To further explain these differences, the original problem specifications for each ward were re-examined. Firstly, the relative numbers of staff allocated to each ward were considered. A count was made of the number of shifts allocated to each ward for each roster. Then a similar count was made of the minimum allowable number of shifts required for each ward for each roster. By subtracting and averaging these two quantities a surplus staff measure was obtained, as shown in the following table:

	Mean Surplus Staff
Ward 1	9.7692
Ward 2	12.5769

Table 28: Mean surplus staff for each ward

Whilst Ward 1 was allocated less surplus staff than Ward 2, using a *t*-test, the difference was not found to be statistically significant ($p > 0.05$, samples approximately normally distributed). Secondly, the difference in problem size was examined. This was measured by taking the mean of the number of feasible schedules generated for each ILP roster problem for each ward. This is shown in the following table:

	Mean Feasible Schedules
Ward 1	2,637.54
Ward 2	26,212.04

Table 29: Mean feasible schedules for each ward

Using a *t*-test, a significant difference was found in problem size between wards ($p < .05$, samples approximately normally distributed).

5.5.2 Measurement of Full-Time Schedule Scores

The WeightedSchedule scoring criteria described in Chapter 3 was initially developed to measure full-time schedule quality. It was then generalised to include part-time schedules. Whilst being a good measure of full-time schedule quality, it provides a simplified measure of part-time schedule quality. The question arises whether human schedulers are applying more complex criteria when evaluating part-time schedules. If this is the case, then the WeightedSchedule comparison between computerised and manual methods may be biased against the manual method. To provide an additional test, the MANOVA 1 analysis was repeated, using only full-time WeightedSchedule scores. The mean values of the transformed full-time WeightedSchedule scores used in the design are shown in the following table:

Independent Variable	Level	Full-Time WeightedSchedule Mean ± Standard Error
Method	Enhanced Cyclic	2.71 ± 0.10
	Basic Cyclic	2.67 ± 0.10
	Manual	3.53 ± 0.09
	Basic Annealing	2.71 ± 0.09
	Enhanced Annealing	2.69 ± 0.09
Ward	Ward 1	2.83 ± 0.07
	Ward 2	2.89 ± 0.07

Table 30: Full-time WeightedSchedule means

Univariate tests of significance confirmed there was a significant difference between method means. Contrasting the manual method mean with the enhanced cyclic descent method mean again showed a significant difference, with a 95% Bonferroni confidence interval of 0.4752 to 1.16046. This demonstrates that the removal of part-time schedule scores from the model does not alter the relative positioning of method means. However, no significant difference was found between ward means when considering full-time schedules (previously Ward 1 had scored more highly). This indicates the difference in total WeightedSchedule scores between wards is primarily explained by differences in part-time schedule scores.

Chapter 6: Discussion

This chapter analyses and interprets the results presented in Chapter 5. Firstly an overview of the research findings is given and the experimental hypotheses are examined. The general findings are then qualified by a more detailed analysis of the results for selected methods and for each ward. Finally the limitations of the methods used in the study are examined and areas for further research are recommended.

6.1 Overview

The main finding of the empirical research is that of all the methods considered, the enhanced cyclic descent algorithm has the best overall performance. A secondary finding is that rosters generated for Ward 2 were of a generally higher quality than those generated for Ward 1.

Looking in turn at the hypotheses presented in Chapter 3:

Hypothesis 1: It was expected that the mean value of WeightedSchedule for the ILP algorithm would be less than the mean value of WeightedSchedule for the enhanced cyclic descent algorithm. The MANOVA 3 results showed no significant difference between these means and so hypothesis 1 can be rejected. The expectation of a lower WeightedSchedule score for the ILP algorithm was theoretically based on the algorithm being able to find an optimum solution. Whilst ILP scores for WeightedSchedule were without exception less than or equal to WeightedSchedule scores for all other computerised methods, this difference was not found to be significant at $\alpha = 0.05$.

Hypothesis 2: It was expected that the mean value of WeightedShift for the basic cyclic descent algorithm would be greater than the mean value of WeightedShift for the enhanced cyclic descent algorithm. In MANOVA 1 a significant difference in the expected direction was found and so hypothesis 2 can be accepted. This result confirms that the enhancements made to basic cyclic descent algorithm have produced a noticeable improvement in the quality of roster generated.

Hypothesis 3: It was expected that no significant difference would exist between the mean values of WeightedShift for all computerised methods *excluding* the basic cyclic descent algorithm. This hypothesis was confirmed in all three MANOVA analyses.

Hypothesis 4: It was expected that no significant difference would exist between the mean values of WeightedSchedule for all computerised methods *excluding* the ILP algorithm. This hypothesis was again confirmed in all three MANOVA analyses.

Hypothesis 5: In examining execution times it was expected that the basic cyclic descent algorithm would execute faster than the enhanced cyclic descent algorithm, whilst the enhanced cyclic descent algorithm would execute faster than both simulated annealing algorithms. Hypothesis 5 was confirmed by the significant differences found between execution times in MANOVA 2. However, with the reduced data set used in MANOVA 3 a significant difference was no longer found between the enhanced simulated annealing algorithm and the enhanced cyclic descent algorithm.

Hypotheses 6 and 7: These hypotheses expected the mean values of WeightedSchedule and WeightedShift for the manual method to be greater than the corresponding measures for the enhanced cyclic descent algorithm. Both hypotheses were confirmed by significant differences found in MANOVA 1.

In looking for the best overall method of roster generation, a predefined order of criteria importance is assumed:

1. WeightedShift (most important)
2. WeightedSchedule
3. ExecutionTime (least important)

Using this ordering of criteria, the manual method and the basic cyclic descent method can be eliminated from contention. This is because both methods scored poorly on the WeightedShift criteria. The manual method also scored poorly on the WeightedSchedule criteria. For the remaining methods, no significant differences in WeightedShift or WeightedSchedule scores were found. Therefore these methods can be separated entirely on the basis of ExecutionTime. As the enhanced cyclic descent algorithm had a significantly faster execution time than the ILP algorithm and both simulated annealing algorithms, it can be concluded that the enhanced cyclic descent algorithm is the best overall method.

The relative position of each method and ward in the dimensions of WeightedShift and WeightedSchedule from MANOVA 1 are shown in the following diagram:

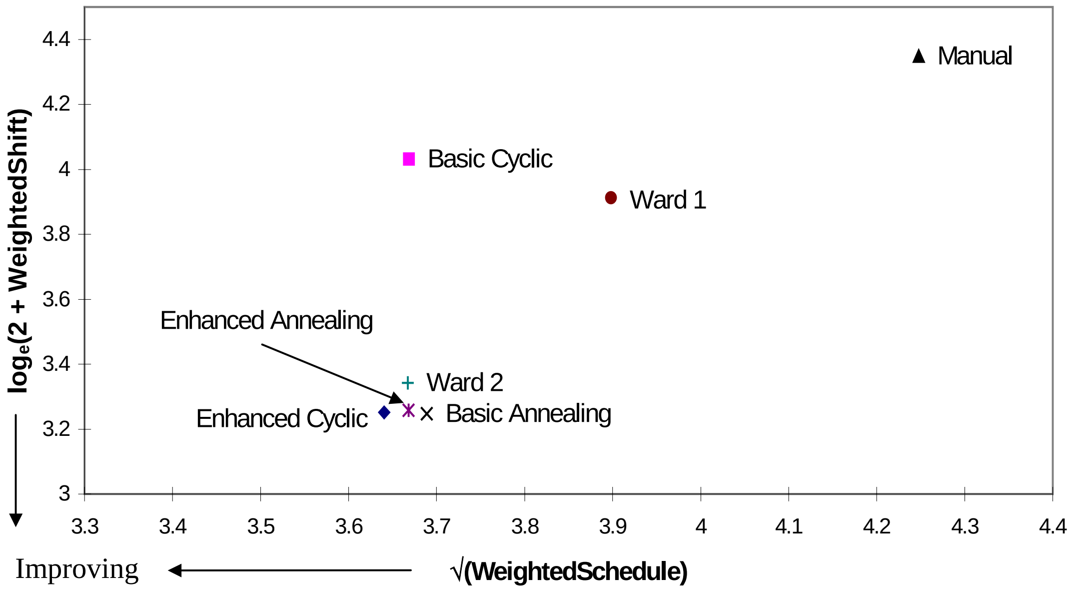


Figure 21: MANOVA 1 All means comparison

Figure 21 shows the separation of the basic cyclic descent method and the manual method from the other methods and explains how these methods were eliminated from contention (lower scores on both axes indicate a better quality solution). The significant difference between wards for both criteria is also illustrated. Figure 22 shows the relative positions of selected methods from MANOVA 3 for the criteria of WeightedSchedule and ExecutionTime:

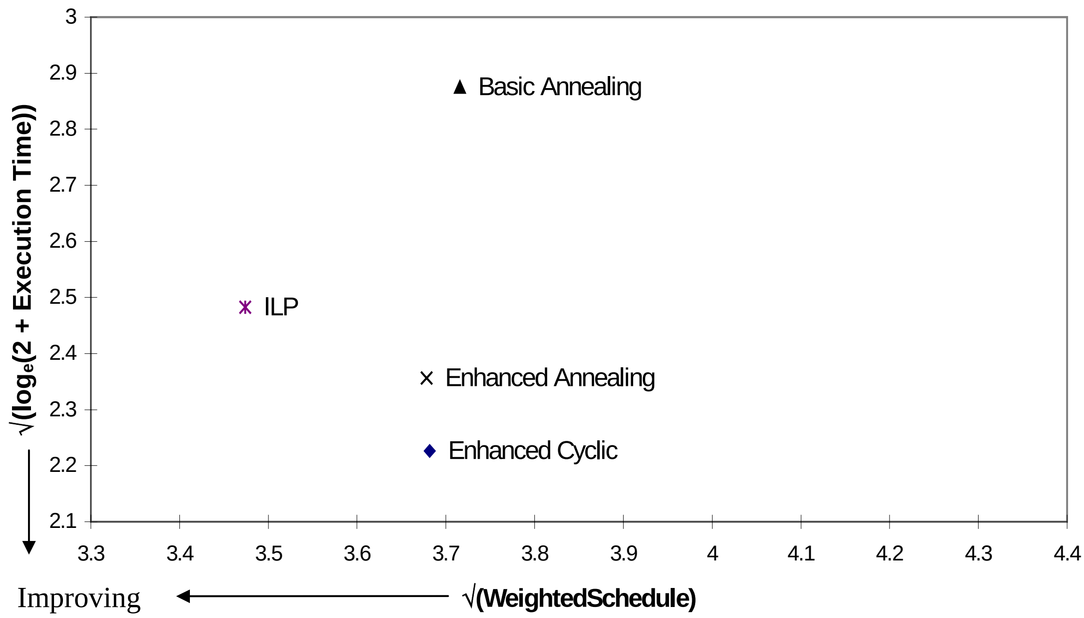


Figure 22: MANOVA 3 Selected means comparison

Given that all methods in figure 22 score equally on WeightedShift, it can be seen that the enhanced cyclic descent algorithm clearly exceeds the enhanced simulated annealing algorithm in the dimension of ExecutionTime. Therefore, the final decision over the best method lies between the enhanced cyclic descent algorithm and the ILP algorithm. The enhanced cyclic descent algorithm is finally chosen because it has a significantly faster ExecutionTime, whereas no significant difference in WeightedSchedule scores was found between the selected methods at $\alpha = 0.05$.

6.2 Evaluation of the ILP Algorithm

6.2.1 Sample Size

The study has found no significant difference between the ILP algorithm and the other enhanced algorithms for the WeightedSchedule and WeightedShift criteria. This result is unlikely to be upheld for larger sample sizes of rosters that the ILP algorithm is *able* to solve. This is because an ILP roster solution will always be optimal with respect the objective function and constraints defined for the problem. By definition, other methods used in the study might approach, but can never exceed, an optimal solution. Therefore it is to be expected that as sample size increases the distinction between ILP solutions and the other methods will grow.

6.2.2 Problem Size and Execution Times

It was expected that for larger rostering problems the ILP algorithm would be unable to find solutions within an acceptable period of time. However, the algorithm found solutions to most problems up to a size of 20,000 variables within 1 hour of Unix user time (approximately 4 hours of DOS time, see Section 5.1.2). Larger problems remained unsolved due to a shortage of computer memory and not because an upper limit to the capabilities of the ILP algorithm was found. A clear idea of the relationship between problem size and execution time could not be gauged. This was partly due to an uneven distribution of problem sizes, as illustrated in the following graph:

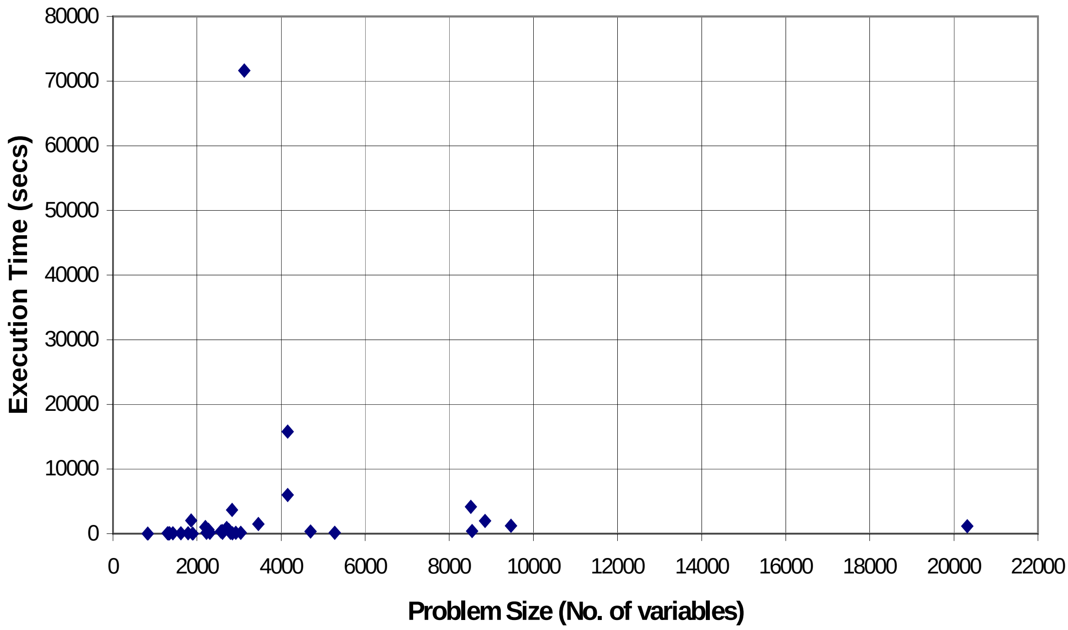


Figure 23: Scatter Plot of ILP DOS execution times against problem size

Figure 23 indicates that ILP solution times tend to be unpredictable for problem sizes over 3000 variables (noting that one 11,992 variable problem remained unsolved after four days of processing). However, for problems of under 3000 variables, the algorithm is able to consistently find optimal solutions within reasonable time.

6.2.3 Applicability of an ILP Approach

Three points arise from the above discussion:

1. An ILP solution will always be as good as or better than a solution produced by any other method in the study.
2. The ILP algorithm is consistently able to solve problems up to 3,000 variables.
3. ILP tests have only been conducted on a Sun®/Unix platform.

From this it is concluded that an ILP algorithm would be the preferred approach for the solution of smaller rostering problems, given the availability of a Sun®/Unix platform. With greater memory resources and/or a more powerful processor there is reason to expect an ILP algorithm would consistently solve larger problems. Conclusions as to the applicability of a DOS-based ILP algorithm must be left for further testing.

6.3 Evaluation of the Enhanced Simulated Annealing algorithm

Both simulated annealing algorithms were found to execute significantly more slowly than the enhanced cyclic descent algorithm considering the full set of rosters. Given the nature of a simulated annealing search, this result is to be expected (see Section 2.2.5). However, for the reduced set of rosters used in MANOVA 3, no significant difference in execution time was found between the enhanced simulated annealing algorithm and the enhanced cyclic descent algorithm (although on average the enhanced cyclic descent algorithm was still executing slightly faster). This indicates that for smaller problem sizes, there is little to choose between the enhanced cyclic and enhanced annealing approaches. The results also show that the simple addition of a schedule selection bias into the simulated annealing algorithm causes a noticeable speeding up of execution time. Further, it should be remembered that a general simulated annealing algorithm was applied to the problem. Faster simulated annealing algorithms have been reported in the literature (Lo and Bavarian 1992), and the application of such algorithms to the nurse rostering problem may further decrease execution times. Whether such a decreases in execution time would be paid for by poorer quality solutions must be left for further research to investigate.

6.4 Distinctions between Algorithms

From the discussion so far it can be concluded that the enhanced cyclic descent algorithm has the best overall performance of the methods considered, but that this difference is only significant for larger problem sizes. The distinguishing feature of a large roster problem is the presence of part-time staff with relatively unconstrained schedule requirements. As explained in Section 4.1.4, the enhanced cyclic descent algorithm is able to calculate the best schedule for each unconstrained staff member whilst the algorithm descends towards a solution. Without such a calculation, the algorithm would have to generate and test thousands of additional feasible schedules, meaning that larger problems would execute more slowly and finally run out of memory.

The ILP problem formulation used in the research requires that all feasible schedules for all nurses are specified in advance. It was the presence of staff with unconstrained schedules that caused the ILP method to run out of memory and fail to solve 16 roster problems. The relatively fixed nature of an ILP algorithm means that schedules cannot be calculated during the solution process. However, it may be possible to develop a new formulation of the problem constraints that would eliminate unconstrained nurse schedules from the model. If successful, such an approach could greatly expand the size of problem that an ILP algorithm is able to solve. This provides an interesting avenue for further research and is explored in more detail in Appendix 8.

The simulated annealing algorithm was able to use the reduced set of schedules generated for the enhanced cyclic descent algorithm. However instead of calculating the best schedule for an unconstrained nurse, in keeping with the simulated annealing search strategy, a schedule is randomly generated (see Section 4.5). Whilst this approach avoids the large memory overhead associated with the ILP algorithm, the size of the search space is not reduced (i.e. instead of immediately calculating the best schedule for a given roster, a simulated annealing algorithm must still ‘guess’ a solution from the thousands of possible unconstrained schedules). This could explain the slower execution times for the enhanced simulated annealing algorithm for those problems containing unconstrained nurse schedules. This finding suggests a hybrid solution may be effective: firstly a simulated annealing approach could be used to select from constrained schedule sets, whilst the schedule calculating algorithm from the cyclic descent algorithm could be used to generate unconstrained schedules.

6.5 Differences between Wards

The results from the MANOVA analyses indicate that Ward 2 has significantly better scores than Ward 1 for both WeightedShift and WeightedSchedule, whilst no difference was found in ExecutionTime. In looking for reasons for the generally better performance of Ward 2, measures were taken both of the mean size of problem and of the amount of surplus staff above minimum requirements. It was found that Ward 2 had significantly larger problem sizes, and slightly more surplus staff (see Section 5.5.1). As the main intent of the study is to investigate the differences between methods, further statistical analysis of differences between wards was not performed. However, it seems reasonable to suggest that the generally better solutions for Ward 2 were related to the greater number of feasible schedules and surplus staff available. A greater number of feasible schedules would produce a greater number of possible solutions to a problem. Given this enlarged pool of solutions, the possibility of a high quality solution must also increase. In addition, the presence of surplus staff means the chances of understaffing are reduced and the chances of overstaffing are increased. Referring to the WeightedShift criteria scores described in Section 3.3.1, it can be seen that understaffing is penalised more severely than overstaffing. Therefore a larger supply of surplus staff would tend to reduce the WeightedShift score for a roster.

6.6 Limitations and Further Research

6.6.1 Practical Application of a Rostering Algorithm

A discussion of the limitations of the research is provided in Section 3.5. However, a basic limitation of the computerised approaches used in the study must be discussed further, in order that the study's results are not misinterpreted. As mentioned in Section 3.5.2, all rosters presented to the computer algorithms have *already* been solved by human experts. A feasible problem solution is defined as any solution that is as good as or better than the manual solution. This means a feasible solution to every problem is guaranteed (i.e. the manual solution is always a feasible solution). If no manual solution exists, as would be the case in the normal implementation of a rostering algorithm, then there is no guarantee that a feasible solution exists for a given set of constraints. For instance, there may be insufficient staff available for a particular night shift. A human expert's response would be to find an extra member of staff to work the shift, either by finding extra staff from outside the ward or reassigning staff within the ward.

The area of resolving conflicting and infeasible constraints is not addressed in the current study. However, for the past year, the enhanced cyclic descent algorithm has been used to generate the rosters for one of the wards used in the study. Experience has shown that a problem often has to be run several times through the algorithm before a feasible solution is found. With each unsuccessful run, the best infeasible solution is examined and any unattainable constraints are suitably adjusted. This can be a time consuming process, lasting anywhere from 30 minutes to several hours.

6.6.2 An Expert System Approach to Constraint Resolution

The need for human intervention in the solution process tends to make rostering software unattractive. The inability of mathematical algorithmic approaches to resolve conflicting constraints has already been highlighted in the literature review (see Section 2.3.2). Artificial Intelligence researchers have therefore recommended the use of expert or knowledge-based systems for rostering (Chow and Hui 1993). However, expert system approaches, whilst providing flexibility, do not necessarily find the best quality solutions. The current research has demonstrated that mathematical algorithmic techniques are capable of finding optimal or near optimal solutions to realistic rostering problems. Practical experience with the enhanced cyclic descent algorithm has indicated there are difficulties in finding a feasible set of problem constraints. A logical next step for the research would be to look into the development of an expert system approach to the resolution of conflicting constraints, whilst still using a mathematical algorithmic technique for roster calculation.

Chapter 7: Conclusion

In conclusion, the contributions and findings of the current study are summarised. Firstly, due to a lack of empirical research in the literature, original criteria were developed with which to measure the performance of a set of roster generation methods. As previously stressed, these criteria were generated in relation to the hospital wards used in the study, and are not necessarily applicable to other rostering problems. In addition, based on previous research, two new roster generation algorithms were developed: the enhanced cyclic descent algorithm and the enhanced simulated annealing algorithm. The development of enhanced algorithms was necessary due to recognised deficiencies in existing rostering algorithms. The study also introduces several new techniques which reduce the size of the roster problem without reducing the set of feasible solutions to the problem.

Using the criteria developed in the study, the performance of the enhanced cyclic descent algorithm was measured against manually generated rosters and against four other computerised methods. These were a basic cyclic descent algorithm, basic and enhanced simulated annealing algorithms and an ILP algorithm. A series of MANOVA analyses produced the conclusion that the enhanced cyclic descent algorithm has the best overall performance of the methods considered. Additional significant findings that emerged from the research were:

- In almost all cases, the computerised methods produced better quality rosters than the manually generated solutions actually used at the hospital (as measured by the criteria developed in the study).

- Contrary to expectations, the ILP algorithm was able to solve a large proportion of the roster problems (34 from 52), and was consistently able to solve problems of up to 3,000 variables.

- The quality of solutions generated by the enhanced cyclic descent algorithm and the simulated annealing algorithms was not significantly different from the optimum quality rosters generated by the ILP algorithm.

Three methods, the enhanced cyclic descent algorithm, the enhanced simulated annealing algorithm and the ILP algorithm, produced the most promising results in the study. The enhanced cyclic descent algorithm was considered superior because it had the smallest average execution time and because the ILP algorithm was unable to solve larger problems. However, the study indicates that improvements to both the simulated annealing approach and to the ILP approach are possible. Therefore, a categorical conclusion as to the superiority of one method over another cannot be made. Of the algorithms used in the study, the enhanced cyclic descent algorithm would be preferred. However, given the increasing power of computer technology and the development of a better mathematical formulation of the problem, an ILP approach to nurse rostering looks increasingly promising.

The two main criticisms raised against mathematical optimising approaches to rostering are that they are inflexible to changing or conflicting constraints and are also unable to solve large realistic problems. The current study has shown that given a careful formulation of the problem, mathematical optimising techniques such as the branch and bound ILP algorithm can solve realistic rostering problems. If an ILP approach fails, an enhanced cyclic descent approach has been shown to generate near optimal solutions within an acceptable time. Given a set of feasible problem constraints, a systematic algorithmic optimising approach (e.g. ILP, cyclic descent or simulated annealing) must be the preferred approach over a heuristic or rule based search for a satisficing solution. The main weakness of the algorithmic approaches presented in the study is their inability to cope with or solve problems with inconsistent or unattainable constraints. It is in this area that a rule based or expert system approach could be used to resolve conflicting constraints and reformulate a problem so that a feasible/optimal solution can be found. The marriage of an expert system controller with an ILP or cyclic descent roster generating engine would bring together the strengths of both approaches, and is a recommended next step for nurse rostering research.

Appendix 1: Clarification of Rostering Terms

A1.1 Shifts

A shift is a predefined period of duty that is worked by an individual nurse. Each shift has a start time and an end time, with the maximum length of a shift being limited either by law or by agreement. A nurse is either on duty or off duty for a particular shift. If a nurse is off duty for all shift types on a particular day, it follows that the nurse has a day off. As an example, three types of shift considered in the current study are:

- Early Shift: 7.00am - 3.30pm
- Late Shift: 2.30pm - 11.00pm
- Night Shift: 10.45pm - 7.30am

Most hospitals operate on a similar three shift system. Other types of shift may be worked according to fluctuations in daily patient load (eg 12.30pm - 9.00pm). Additionally, some hospitals implement more flexible shifts, with variable starting times and lengths (Ozkarahan and Bailey 1988).

A1.2 Schedules

The term schedule has been used rather loosely in the nurse rostering literature, referring both to the shifts allocated to an individual nurse, and to describe the roster as a whole. For the purposes of this study, a schedule is defined as the pattern of shifts allocated to an individual nurse, lasting for the duration of the rostering period. In other words, a schedule defines the times during which a nurse is expected to be on duty during the rostering period. Table a shows an example one week schedule, using the three shift system previously introduced.

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Night	*Night*	*Off*	*Off*	*Late*	*Early*	*Early*

Table a: An example nurse schedule

A1.3 Feasible and Infeasible Schedules

In all rostering problems, there are explicit and/or implicit rules governing the type of schedules that employees are expected to work. From this arises the concept of an infeasible schedule. An infeasible schedule is a schedule that breaks one or more of the accepted scheduling rules, and therefore should not be considered in a final roster solution. By discarding all infeasible schedules, the set of feasible schedules for a given nurse and a given roster can be generated (Warner 1976).
As an example, consider the following situation:

- A ward operates one shift type, so a nurse is either on duty or off duty on any one day.
- No nurse is expected to work longer than six days or less than four days without a day off.
- All days off are given in consecutive pairs.
- A schedule lasts for fourteen days.
- A full-time nurse works exactly ten shifts in any one schedule.
- A selected full-time nurse has ended the preceding schedule with two days off.

Given these rules and conditions, the following table shows all feasible schedules for the selected nurse:

Mo	Tue	We	Th	Fri	Sat	Sun	Mo	Tue	We	Th	Fri	Sat	Sun
On	*On*	*On*	*On*	**Off**	**Off**	*On*	*On*	*On*	*On*	**Off**	**Off**	*On*	*On*
On	*On*	*On*	*On*	**Off**	**Off**	*On*	*On*	*On*	*On*	*On*	**Off**	**Off**	*On*
On	*On*	*On*	*On*	**Off**	**Off**	*On*	*On*	*On*	*On*	*On*	*On*	**Off**	**Off**
On	*On*	*On*	*On*	*On*	**Off**	**Off**	*On*	*On*	*On*	*On*	**Off**	**Off**	*On*
On	*On*	*On*	*On*	*On*	**Off**	**Off**	*On*	*On*	*On*	*On*	*On*	**Off**	**Off**
On	*On*	*On*	*On*	*On*	*On*	**Off**	**Off**	*On*	*On*	*On*	*On*	**Off**	**Off**

Table b: Set of feasible schedules for an individual nurse

Table b illustrates how the application of rules and constraints can dramatically reduce the total number of feasible schedules for a particular nurse. In a situation where no rules were to apply, the total number of feasible schedules would have been 2^{14} (16,384). Of these, 99.96% have been eliminated.

A1.4 Rosters

A nurse roster is a collection of nurse schedules. It defines all schedules that will actually be worked, for all nurses on a particular ward or unit, within a given time period. In doing this, the roster sets the number and identities of the staff working each shift. Therefore, a roster is not only concerned with providing nurses with feasible schedules, but also with providing adequate patient care on each shift, by supplying sufficient numbers of qualified staff.

As with the generation of feasible schedules, a roster is governed by rules and constraints that severely limit the number of possible solutions. Using the schedule constraints from the previous section, consider a ward with four nurses (nurses one, two, three and four), and the following additional constraints:

- Hospital policy states that there must be at least two nurses on duty each day
- The most desirable schedules are those which contain unbroken work stretches of exactly five shifts
- Each nurse worked the following number of days at the end of the previous roster:

- Nurse one: 0 days (ie the previous schedule ended in days off)
- Nurse two: 4 days
- Nurse three: 5 days
- Nurse four: 2 days

The number of days worked at the end of the previous roster is used to calculate the length of unbroken work stretches between rosters. The following table represents the best feasible roster solution to this problem:

Nurse	Mo	Tu	We	Th	Fri	Sat	Su	Mo	Tu	We	Th	Fri	Sat	Su
One	*On*	*On*	*On*	*On*	*On*	**Off**	**Off**	*On*	*On*	*On*	*On*	*On*	**Off**	**Off**
Two	*On*	**Off**	**Off**	*On*	*On*	*On*	*On*	*On*	**Off**	**Off**	*On*	*On*	*On*	*On*
Three	**Off**	**Off**	*On*	*On*	*On*	*On*	*On*	**Off**	**Off**	*On*	*On*	*On*	*On*	*On*
Four	*On*	*On*	*On*	**Off**	**Off**	*On*	*On*	*On*	*On*	*On*	**Off**	**Off**	*On*	*On*

Table c: Example two week roster for four nurses

Table c represents the best roster solution because it not only meets all the *hard* constraints defined, it also achieves the *soft* constraint of granting all nurses exact work stretches of five days. A soft constraint is a constraint whose attainment is desirable, but not mandatory. Conversely, hard constraints are constraints that must be met, or the solution becomes infeasible. A simple inspection of the problem reveals that no other configuration of shifts would provide all nurses with five day stretches. The solution is therefore *feasible, optimal* and *unique.*

Appendix 2: A Mathematical Introduction to the Problem

In the following discussion, a mathematical notation for the problem is put forward, based on the models of Warner (1976) and Miller *et al.* (1976). The distinguishing feature of these models, is that they are based on the use of feasible schedules.

Using the roster problem introduced in Appendix 1 :

- Let i denote the index of n nurses, such that $i = 1 \,..\, n$. In this case $n = 4$, as there are four nurses.
- Let j be the index of all feasible schedules for each nurse, and let J_i be the total number of feasible schedules for the ith nurse. Given the constraints defined in section A1.3 and A1.4, each nurse will have the following number of feasible schedules:
    - For nurse one, $J_1 = 6$, so $j = 1 \,..\, 6$
    - For nurse two, $J_2 = 9$, so $j = 1 \,..\, 9$
    - For nurse three, $J_3 = 6$, so $j = 1 \,..\, 6$
    - For nurse four, $J_4 = 9$, so $j = 1 \,..\, 9$
- Let $\mathbf{a}_{ij}$ be a vector of length 14 representing the jth schedule for the ith nurse. Each position in the vector represents a particular day in the schedule, and can have a value of 0 or 1. A value of 1 indicates the nurse is on duty for that day and a 0 value indicates a day off. The resulting set of vectors, for a particular nurse, represent all the feasible schedules for that nurse. In order to clarify this, the following table shows all the $\mathbf{a}_{ij}$ vectors for nurse one:

$\mathbf{a}_{1j}$	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su
$\mathbf{a}_{11}$	1	1	1	1	0	0	1	1	1	1	0	0	1	1
$\mathbf{a}_{12}$	1	1	1	1	0	0	1	1	1	1	1	0	0	1
$\mathbf{a}_{13}$	1	1	1	1	0	0	1	1	1	1	1	1	0	0
$\mathbf{a}_{14}$	1	1	1	1	1	0	0	1	1	1	1	0	0	1
$\mathbf{a}_{15}$	1	1	1	1	1	0	0	1	1	1	1	1	0	0
$\mathbf{a}_{16}$	1	1	1	1	1	1	0	0	1	1	1	1	0	0

Table d: Full set of schedule vectors for nurse one

Note that Table d and Table b (the set of feasible schedules for nurse one) represent the same information. Also, by reading Table d and referring back to the final roster solution in Table c it can be seen that nurse one was given the schedule represented by vector $\mathbf{a}_{15}$.

- Now let X_{ij} represent the variables in the problem, such that :

$$X_{ij} = \begin{cases} 1 \text{ if nurse } i \text{ is to work schedule } \mathbf{a}_{ij} \\ 0 \text{ if not} \end{cases}$$

  Again considering nurse one from the previous roster solution, it can be seen that $X_{15} = 1$, and all other $X_{1j} = 0$

- From the definition of X_{ij} it follows that for any nurse i, with a total number of feasible schedules J_i , $\sum_{j=1}^{j=J_i} X_{ij} = 1$, as a nurse can only work one schedule in any one roster.

- In the example problem, a constraint was defined such that there must be at least two nurses on duty during each shift. This can be expressed as a vector $\mathbf{b}_{tot}$ with 14 elements. Each element of $\mathbf{b}_{tot}$ then represents the minimum number of staff required for each day of the roster, such that $\mathbf{b}_{tot}$ = (2,2,2,2,2,2,2,2,2,2,2,2,2,2)
- Finally, a measure of the quality of each schedule needs to be introduced into the model. For the present example, let us assign a coefficient c_{ij} to each schedule *j*, belonging to nurse *i*, indicating the schedule's quality. A zero value of c_{ij} represents a perfect schedule, so as schedule quality decreases, the value of c_{ij} will increase. c_{ij} is defined such that:

$$c_{ij} = \begin{cases} 0 \text{ if a schedule contains only 5 day work stretches} \\ 1 \text{ if a schedule contains no more than one 6 or 4 day work stretch} \\ 2 \text{ otherwise} \end{cases}$$

Given these definitions, the complete problem can be expressed in the following form:

$$\text{minimise} \quad z = \sum_{i=1}^{i=n} \sum_{j=1}^{j=J_i} c_{ij} X_{ij} \qquad (1)$$

$$\text{subject to} \quad \sum_{j=1}^{j=J_i} X_{ij} = 1, \quad i = 1..n \qquad (2)$$

$$\sum_{i=1}^{i=n} \sum_{j=1}^{j=J_i} \mathbf{a}_{ij} X_{ij} \geq \mathbf{b}_{tot} \qquad (3)$$

$$X_{ij} = (0 \text{ or } 1), \quad i = 1..n; \quad j = 1..J_i$$

The objective of these equations is to find the combination of schedules with the minimum c_{ij} values (equation 1), whilst ensuring that each nurse works only one schedule (equation 2), and that there are sufficient staff working each shift (equation 3). The restriction that X_{ij} should be either 0 or 1 reflects that a nurse can only either work (X_{ij} = 1) or not work (X_{ij} = 0) a schedule (ie 0.25 of a schedule would have no meaning).

The equation system, as defined, can be solved using existing integer programming techniques. This is illustrated in the following worked example of the previously defined problem:

A2.1 Feasible Schedules

The following tables define the feasible schedules for each nurse as $\mathbf{a}_{ij}$ vectors, where i is an index of nurses, $i = 1 \,..\, n$, $n = 4$, and j is an index of schedules, $j = 1 \,..\, J_i$, $J_1 = 6$, $J_2 = 9$, $J_3 = 6$, $J_4 = 9$.

$\mathbf{a}_{1j}$	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su
$\mathbf{a}_{11}$	1	1	1	1	0	0	1	1	1	1	0	0	1	1
$\mathbf{a}_{12}$	1	1	1	1	0	0	1	1	1	1	1	0	0	1
$\mathbf{a}_{13}$	1	1	1	1	0	0	1	1	1	1	1	1	0	0
$\mathbf{a}_{14}$	1	1	1	1	1	0	0	1	1	1	1	0	0	1
$\mathbf{a}_{15}$	1	1	1	1	1	0	0	1	1	1	1	1	0	0
$\mathbf{a}_{16}$	1	1	1	1	1	1	0	0	1	1	1	1	0	0

Table e: Feasible schedules for nurse one (last roster ended in days off)

$\mathbf{a}_{2j}$	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su
$\mathbf{a}_{21}$	0	0	1	1	1	1	0	0	1	1	1	1	1	1
$\mathbf{a}_{22}$	0	0	1	1	1	1	1	0	0	1	1	1	1	1
$\mathbf{a}_{23}$	0	0	1	1	1	1	1	1	0	0	1	1	1	1
$\mathbf{a}_{24}$	1	0	0	1	1	1	1	0	0	1	1	1	1	1
$\mathbf{a}_{25}$	1	0	0	1	1	1	1	1	0	0	1	1	1	1
$\mathbf{a}_{26}$	1	0	0	1	1	1	1	1	1	0	0	1	1	1
$\mathbf{a}_{27}$	1	1	0	0	1	1	1	1	0	0	1	1	1	1
$\mathbf{a}_{28}$	1	1	0	0	1	1	1	1	1	0	0	1	1	1
$\mathbf{a}_{29}$	1	1	0	0	1	1	1	1	1	1	0	0	1	1

Table f: Feasible schedules for nurse two (last roster ended with 4 days on)

$\mathbf{a}_{3j}$	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su
$\mathbf{a}_{31}$	0	0	1	1	1	1	0	0	1	1	1	1	1	1
$\mathbf{a}_{32}$	0	0	1	1	1	1	1	0	0	1	1	1	1	1
$\mathbf{3}_{33}$	0	0	1	1	1	1	1	1	0	0	1	1	1	1
$\mathbf{a}_{34}$	1	0	0	1	1	1	1	0	0	1	1	1	1	1
$\mathbf{a}_{35}$	1	0	0	1	1	1	1	1	0	0	1	1	1	1
$\mathbf{a}_{36}$	1	0	0	1	1	1	1	1	1	0	0	1	1	1

Table g: Feasible schedules for nurse three (last roster ended with 5 days on)

$\mathbf{a}_{4j}$	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su
$\mathbf{a}_{41}$	1	1	0	0	1	1	1	1	0	0	1	1	1	1
$\mathbf{a}_{42}$	1	1	0	0	1	1	1	1	1	0	0	1	1	1
$\mathbf{a}_{43}$	1	1	0	0	1	1	1	1	1	1	0	0	1	1
$\mathbf{a}_{44}$	1	1	1	0	0	1	1	1	1	0	0	1	1	1
$\mathbf{a}_{45}$	1	1	1	0	0	1	1	1	1	1	0	0	1	1
$\mathbf{a}_{46}$	1	1	1	0	0	1	1	1	1	1	1	0	0	1
$\mathbf{a}_{47}$	1	1	1	1	0	0	1	1	1	1	0	0	1	1
$\mathbf{a}_{48}$	1	1	1	1	0	0	1	1	1	1	1	0	0	1
$\mathbf{a}_{49}$	1	1	1	1	0	0	1	1	1	1	1	1	0	0

Table h: Feasible schedules for nurse four (last roster ended with 2 days on)

A2.2 Daily Minimum Staff Constraints

The constraints for the minimum number of staff working each day were defined as:

$$\sum_{i=1}^{i=n}\sum_{j=1}^{j=J_i} \mathbf{a}_{ij} X_{ij} \geq \mathbf{b}_{tot}$$

where $\mathbf{b}_{tot} = (2,2,2,2,2,2,2,2,2,2,2,2,2,2)$

This results in 14 constraints, one for each element of the $\mathbf{b}_{tot}$ vector, representing each day of the week. Therefore, the first constraint is obtained by reading off all the $\mathbf{a}_{ij}$ coefficients from the Monday column of the above tables (a zero $\mathbf{a}_{ij}$ coefficient removes the corresponding X_{ij} variable from the equation, otherwise it is included):

First Monday Constraint:

$$X_{11} + X_{12} + X_{13} + X_{14} + X_{15} + X_{16} + X_{24} + X_{25} + X_{26} + X_{27} + X_{28} + X_{29} + X_{34} + X_{35} + X_{36} + X_{41} + X_{42} + X_{43} + X_{44} + X_{45} + X_{46} + X_{47} + X_{48} + X_{49} \geq 2$$

First Tuesday Constraint:

$$X_{11} + X_{12} + X_{13} + X_{14} + X_{15} + X_{16} + X_{27} + X_{28} + X_{29} + X_{41} + X_{42} + X_{43} + X_{44} + X_{45} + X_{46} + X_{47} + X_{48} + X_{49} \geq 2$$

First Wednesday Constraint:

$$X_{11} + X_{12} + X_{13} + X_{14} + X_{15} + X_{16} + X_{21} + X_{22} + X_{23} + X_{31} + X_{32} + X_{33} + X_{44} + X_{45} + X_{46} + X_{47} + X_{48} + X_{49} \geq 2$$

First Thursday Constraint:

$$X_{11} + X_{12} + X_{13} + X_{14} + X_{15} + X_{16} + X_{21} + X_{22} + X_{23} + X_{24} + X_{25} + X_{26} + X_{31} + X_{32} + X_{33} + X_{34} + X_{35} + X_{36} + X_{47} + X_{48} + X_{49} \geq 2$$

First Friday Constraint:

$$X_{14} + X_{15} + X_{16} + X_{21} + X_{22} + X_{23} + X_{24} + X_{25} + X_{26} + X_{27} + X_{28} + X_{29} + X_{31} + X_{32} + X_{33} + X_{34} + X_{35} + X_{36} + X_{41} + X_{42} + X_{43} \geq 2$$

First Saturday Constraint:

$$X_{16} + X_{21} + X_{22} + X_{23} + X_{24} + X_{25} + X_{26} + X_{27} + X_{28} + X_{29} + X_{31} + X_{32} + X_{33} + X_{34} + X_{35} + X_{36} + X_{41} + X_{42} + X_{43} + X_{44} + X_{45} + X_{46} \geq 2$$

First Sunday Constraint:

$$X_{11} + X_{12} + X_{13} + X_{14} + X_{22} + X_{23} + X_{24} + X_{25} + X_{26} + X_{27} + X_{28} + X_{29} + X_{32} + X_{33} + X_{34} + X_{35} + X_{36} + X_{41} + X_{42} + X_{43} + X_{44} + X_{45} + X_{46} + X_{47} + X_{48} + X_{49} \geq 2$$

Second Monday Constraint:

$$X_{11} + X_{12} + X_{13} + X_{14} + X_{15} + X_{23} + X_{25} + X_{26} + X_{27} + X_{28} + X_{29} + X_{33} + X_{35} + X_{36} + X_{41} + X_{42} + X_{43} + X_{44} + X_{45} + X_{46} + X_{47} + X_{48} + X_{49} \geq 2$$

Second Tuesday Constraint:

$$X_{11} + X_{12} + X_{13} + X_{14} + X_{15} + X_{16} + X_{21} + X_{26} + X_{28} + X_{29} + X_{31} + X_{36} + X_{42} + X_{43} + X_{44} + X_{45} + X_{46} + X_{47} + X_{48} + X_{49} \geq 2$$

Second Wednesday Constraint:

$$X_{11} + X_{12} + X_{13} + X_{14} + X_{15} + X_{16} + X_{21} + X_{22} + X_{24} + X_{29} + X_{31} + X_{32} + X_{34} + X_{43} + X_{45} + X_{46} + X_{47} + X_{48} + X_{49} \geq 2$$

Second Thursday Constraint:

$$X_{12} + X_{13} + X_{14} + X_{15} + X_{16} + X_{21} + X_{22} + X_{23} + X_{24} + X_{25} + X_{27} + X_{31} + X_{32} + X_{33} + X_{34} + X_{35} + X_{41} + X_{46} + X_{48} + X_{49} \geq 2$$

Second Friday Constraint:

$$X_{13} + X_{15} + X_{16} + X_{21} + X_{22} + X_{23} + X_{24} + X_{25} + X_{26} + X_{27} + X_{28} + X_{31} + X_{32} + X_{33} + X_{34} + X_{35} + X_{36} + X_{41} + X_{42} + X_{44} + X_{49} \geq 2$$

Second Saturday Constraint:

$$X_{11} + X_{21} + X_{22} + X_{23} + X_{24} + X_{25} + X_{26} + X_{27} + X_{28} + X_{29} + X_{31} + X_{32} + X_{33} + X_{34} + X_{35} + X_{36} + X_{41} + X_{42} + X_{43} + X_{44} + X_{45} + X_{47} \geq 2$$

Second Sunday Constraint:

$$X_{11} + X_{12} + X_{14} + X_{21} + X_{22} + X_{23} + X_{24} + X_{25} + X_{26} + X_{27} + X_{28} + X_{29} + X_{31} + X_{32} + X_{33} + X_{34} + X_{35} + X_{36} + X_{41} + X_{42} + X_{43} + X_{44} + X_{45} + X_{46} + X_{47} + X_{48} \geq 2$$

A2.3 One Schedule per Nurse Constraints

The constraints to limit the number of schedules allocated to each nurse were defined as:

$$\sum_{j=1}^{j=J_i} X_{ij} = 1, \quad i = 1..n$$

This leads to the following 4 constraints for each nurse:

Nurse one:

$X_{11} + X_{12} + X_{13} + X_{14} + X_{15} + X_{16} = 1$

Nurse two:

$X_{21} + X_{22} + X_{23} + X_{24} + X_{25} + X_{26} + X_{27} + X_{28} + X_{29} = 1$

Nurse three:

$X_{31} + X_{32} + X_{33} + X_{34} + X_{35} + X_{36} = 1$

Nurse four:

$X_{41} + X_{42} + X_{43} + X_{44} + X_{45} + X_{46} + X_{47} + X_{48} + X_{49} = 1$

A2.4 The Objective Function

The objective of the problem is to select the four schedules (one for each nurse), that are of the highest quality. This is expressed in the objective function:

$$\text{minimise} \quad z = \sum_{i=1}^{i=n} \sum_{j=1}^{j=J_i} c_{ij} X_{ij}$$

$$\text{where } c_{ij} = \begin{cases} 0 \text{ if a schedule contains only 5 day work stretches} \\ 1 \text{ if a schedule contains no more than one 6 or 4 day work stretch} \\ 2 \text{ otherwise} \end{cases}$$

this results in the following equation:

$$\begin{aligned} \text{minimise z} \quad = {} & 2X_{11} + 1X_{12} + 2X_{13} + 1X_{14} + 0X_{15} + 2X_{16} \\ & + 2X_{21} + 1X_{22} + 2X_{23} + 1X_{24} + 0X_{25} + 1X_{26} + 2X_{27} + 1X_{28} + 2X_{29} \\ & + 1X_{31} + 0X_{32} + 1X_{33} + 2X_{34} + 1X_{35} + 2X_{36} \\ & + 2X_{41} + 1X_{42} + 2X_{43} + 1X_{44} + 0X_{45} + 1X_{46} + 2X_{47} + 1X_{48} + 2X_{49} \end{aligned}$$

A2.5 The Solution

By entering the objective function and the above constraints into a computerised integer linear programming application, the following result can be obtained:

$X_{15} = 1$, $X_{25} = 1$, $X_{32} = 1$ and $X_{45} = 1$, all other $X_{ij} = 0$, and z = 0.

This result can be verified by a simple inspection of the equations: the minimum value of z (ie z = 0), given the constraints that each nurse should work exactly one schedule, will be given by those four X_{ij} variables associated with 0 c_{ij} coefficients, ie X_{15}, X_{25}, X_{32} and X_{45}. By constructing a roster with these four schedules, as in section A1.4, Table c, it can be shown that the additional minimum staffing constraints have been met.

Appendix 3: The Research Problem

The following material is based on data collected from a series of interviews with senior nursing personnel at the Gold Coast Hospital, in Southport, Queensland. The policies and practices laid out below refer specifically to two 30 bed medical wards within the hospital:

A3.1 General Features of the Problem

- **Rostering Period**: The rostering period lasts for 14 days, and rosters are prepared at least one week in advance of the period to be worked.
- **Hours of Duty**: Nursing care is provided 24 hours a day, and 7 days a week.
- **Shifts**: Nurses can work early, late and night shifts (see section A1.1).
- **Shift Combinations**: Nurses can work combinations of all three shift types without separating days off.
- **Full and Part-Time Staffing**: Nursing staff can be either full-time or part-time. Full-time staff work 10 shifts per roster, whilst part-time staff can work from 1 to 9 shifts per roster.
- **Days Off**: There are two types of days off :
    1. **Scheduled days off** : these are days off that normally occur within a roster. For instance, a full-time nurse working 10 shifts per roster will have 4 scheduled days off in a 14 day roster.
    2. **Arranged days off** : these days off occur in addition to scheduled days off and reduce the number of a shifts a nurse will work in a roster. Arranged days off can be due to annual leave, unpaid leave, sickness or reallocation of a nurse to other duties.

- **Staff Seniority Levels**: There are 4 official levels of staff, shown below in order of seniority:
    1. **Clinical Nurse Consultant** (CNC) : the CNC is the head nurse for a ward, and works Monday to Friday early shifts. There is one CNC per ward.
    2. **Clinical Nurse** (CN) : Usually there are 4 CN’s per ward. These nurses take charge of shifts when the CNC is off duty.
    3. **Registered Nurse** (RN) : An RN is a fully qualified nurse and is capable of performing the same nursing duties as a CNC or a CN. In practice, RN’s will occasionally be in charge of shifts in the absence of senior staff. Not all RN’s are considered to have sufficient expertise to perform the in-charge role, and so there is a further *unofficial* division of staff:
        - Senior RN: considered capable of performing an in-charge role.
        - Junior RN: not considered capable of performing an in-charge role.
    4. **Enrolled Nurse** (EN) : An EN receives less training than an RN, and is legally barred from performing certain essential nursing tasks, such as the administration of injections and drugs.

A3.2 Specific Features of the Problem

- **Requesting Policy**: Nurses are free to request any arrangement of shifts and scheduled days off within a particular roster. These requests are dealt with at ward level, and are accepted or denied at the discretion of the person responsible for rostering (usually a senior nurse). Requests for arranged days off are dealt with by nursing administration.
- **Fixed Schedules**: In agreement with the CNC, certain members of staff may arrange to work a fixed schedule, ie a pattern of shifts that repeats from roster to roster. Typically, nurses with child care responsibilities and nurses exclusively working night shifts will be granted fixed or restricted schedules.
- **Overall Staffing Levels**: The number of staff working on a ward can vary dramatically over time depending on hospital policy. For the ward considered in this study, staff numbers have varied from 24 to 30 nurses during any one roster.
- **Shift Staffing Levels**: Shift staffing levels can vary from week to week according to the patient load. Nevertheless, the following figures show some typical levels and are given to illustrate the complexity of the staffing level problem:
    - **General Rules for Early and Late Shifts**: There should be at least one CNC or CN in charge of each shift and at least one senior RN on duty. There should be no more than two EN’s on any one shift. In the event that no CNC or CN is available for a shift, a senior RN may “act up” or be temporarily promoted to the CN role for the duration of the shift.
    - **Early Shift Staffing Levels**: staffing levels can vary according to patient load, with the absolute minimum total number of staff for an early shift being 6. Often this minimum is raised to 7 during weekdays, with a desired total number of staff of 7 during the week and 6 at weekends. A maximum a level of staff may also be stipulated: this would range from 8-9 during the week and from 7-8 during the weekend (depending on the total staff available).
    - **Late Shift Staffing Levels**: For weekdays, the minimum total number of staff for a late shift is 5, with the desired level being 5 or 6 depending on patient load. At weekends, the minimum and desired level is 5. The maximum late shift staff level is one less than the total number of early shift staff during the week, and equal to the total number of early shift staff during the weekend.
    - **Night Shift Staffing Levels**: the night shift comprises of exactly 3 staff for each day of the week. One CN, one RN (senior or junior) and one EN are required to be on duty. An RN may replace the EN, and a senior RN may act as a CN.

- **Feasible Schedule Guidelines:**
    - **Full-Time Nurses, No Requests**: full-time nurses, making no requests, are not expected to work more than 7 days, or less than three days, without a day off. These nurses would also expect to receive two sets of two consecutive days off within the roster period (this is similar to the example problem illustrated in Table b).
    - **Full-Time Nurses, with Requests**: If a full-time nurse requests shift patterns that break the previous guidelines, these are allowed. However, no nurse is allowed to work more than 10 days without a day off. Also, if the constraints of a roster mean that certain guidelines cannot be met, then nurses making requests are the first to receive less desirable schedules.

- **Part-Time Nurses**: Part-time nurses are liable to work any combination of days on and off. In practice, however, long stretches without days off are avoided, whilst patterns of consecutive days off are encouraged. Typically, part-time nurses are those nurses with child care responsibilities, or who work exclusively on night shifts. In these cases, fixed or restricted schedules are often negotiated (see Fixed Schedules above).
- **Late and Early Shift Policy**: Generally, nurses will be rostered to work an early shift before days off and a late shift after days off, thereby extending the time off period. A nurse can elect not to work an early shift immediately after a late shift. In practice this is rarely done, because it means a nurse cannot reliably expect to receive the extended time off period. Also, most nurses prefer not to work long stretches of the same shift type, and like a balance of both late and earlies. Within these constraints, the number of consecutive late/early shift combinations is minimised.
- **Night Shift Policy**: Staff can elect to work exclusively on night shifts. The remaining night shift duties are shared out as evenly as possible between the other staff members. Generally, a nurse can expect to work a block of between one and four consecutive night shifts, followed by two days off. Only one such block of nights is worked in any one schedule, and a nurse can expect a period of between two to four weeks before working another night shift.
- **Schedule Quality**: Unless a nurse requests otherwise, the best quality schedule for a full-time nurse is considered to be one with 5 day consecutive work stretches, separated by two days off. As work stretches get longer or shorter then schedule quality decreases (as in section A.2.4).

A3.3 The Rostering Objectives

Given the policies and practices defined above, the person in charge of rostering will also have a set of prioritised objectives that are used when constructing a roster. An example ordered list of priorities is laid out below, reflecting current rostering practice on the ward being studied:

1. To meet the *minimum* total staffing requirements for each shift. If this is impossible to achieve, then additional staff can be brought into the ward from elsewhere in the hospital.
2. To meet the *minimum* staffing requirements for each level of staff. As discussed previously, some of these requirements can be met by temporarily promoting junior staff into more senior positions.
3. To meet the *desired* total staffing requirements for each shift.
4. To provide all non-requesting full-time staff with a minimum standard of schedule.
5. To grant as many requests as possible, given the previous objectives are satisfied.
6. Not to exceed the maximum staffing requirements for each shift.
7. To provide the fairest distribution of schedules between nurses. This means that shift types and requests should be evenly allocated.
8. To provide the best quality of schedule day on and day off pattern to each staff member, given the previous objectives have been satisfied.
9. To provide the best mix of late and early shifts to each staff member.

A3.4 A Mathematical Formulation of the Complete Problem

The basic mathematical model developed in Appendix 2 is now taken and expanded so that it can describe the full problem specifications. This is the formulation that is used for the integer linear programming algorithm used in the main body of the research.

A3.4.1 The Problem Constraints

The initial constraints developed in Appendix 2 still hold. Namely, each nurse still has to work one and only one schedule, and the minimum staffing levels still have to be met for each shift. However, in the original example there were 14 minimum staffing level constraints, as there was only one shift type per day, whereas in the current problem there will be 28 constraints of this type (14 days x 2 shifts). The two shift types are day shifts and night shifts. The allocation of late and early shifts is carried out in a separate heuristic procedure, as described in Chapter 4. Therefore, $\mathbf{b}_{tot}$ is redefined as a vector with 28 elements, such that:

- The first 14 elements are the minimum total staff requirements for each day shift.
- The second 14 elements are the minimum total staff requirements for each night shift.

Given the staffing levels defined in the previous section,
$\mathbf{b}_{tot}$ = (12,12,12,12,12,11,11,12,12,12,12,12,11,11,3,3,3,3,3,3,3,3,3,3,3,3,3,3)
Likewise, $\mathbf{a}_{ij}$ now becomes a vector of 28 elements, each element describing whether a particular shift is worked on a particular day. For example if the first element of $\mathbf{a}_{23} = 1$, this means that in the third feasible schedule, of the second nurse, a day shift is allocated on the first Monday of the schedule (assuming the rostering period starts on a Monday).

Given these redefinitions, and that n and J are no longer fixed, the following constraints still hold:

$$\sum_{j=1}^{j=J_i} X_{ij} = 1, \quad i = 1..n \qquad (1)$$

$$\sum_{i=1}^{i=n}\sum_{j=1}^{j=J_i} \mathbf{a}_{ij} X_{ij} \geq \mathbf{b}_{tot} \qquad (2)$$

$$X_{ij} = (0 \text{ or } 1), \quad i = 1..n; \quad j = 1..J_i \qquad (3)$$

However, there are now both minimum and maximum constraints for the total number of staff on each shift. Letting $\mathbf{b}_{mintot}$ and $\mathbf{b}_{maxtot}$ represent these minimum and maximum staff numbers, then $\mathbf{b}_{mintot} = \mathbf{b}_{tot}$, and the previous constraint (2) becomes:

$$\mathbf{b}_{maxtot} \geq \sum_{i=1}^{i=n}\sum_{j=1}^{j=J_i} \mathbf{a}_{ij} X_{ij} \geq \mathbf{b}_{mintot} \qquad (4)$$

In addition, there are constraints for each level of staff, remembering that staff can also be reassigned between levels during the rostering period. To express this, let $\mathbf{r}_i$ be a vector of length 28, with each element representing a shift on a particular day as before, such that:

$$\mathbf{r}_i = \begin{cases} 1 \text{ if nurse i is a CN or CNC} \\ 0 \text{ otherwise} \end{cases}$$

Similarly, let $\mathbf{s}_i$ be a vector of length 28, such that:

$$\mathbf{s}_i = \begin{cases} 1 \text{ if nurse i is a senior RN, CN or CNC} \\ 0 \text{ otherwise} \end{cases}$$

and let $\mathbf{e}_i$ be a vector of length 28, such that:

$$\mathbf{e}_i = \begin{cases} 1 \text{ if nurse i is an EN} \\ 0 \text{ otherwise} \end{cases}$$

Now let $\mathbf{b}_{\boldsymbol{mincharge}}$ and $\mathbf{b}_{\boldsymbol{mincharge}}$ be vectors of length 28, representing the minimum and maximum number of staff required to be in charge of a shift. In this case, the in charge constraint would be

$$\mathbf{b}_{\min charge} \geq \sum_{i=1}^{i=n} \sum_{i=1}^{i=J_i} \mathbf{r}_i \mathbf{a}_{ij} X_{ij} \leq \mathbf{b}_{\max charge} \qquad (5)$$

Now let $\mathbf{b}_{\boldsymbol{minsenior}}$ and $\mathbf{b}_{\boldsymbol{maxsenior}}$ be vectors of length 28, representing the minimum and maximum number of senior staff required to work each shift. In this case, the senior staff constraint would be

$$\mathbf{b}_{\min senior} \geq \sum_{i=1}^{i=n} \sum_{i=1}^{i=J_i} \mathbf{s}_i \mathbf{a}_{ij} X_{ij} \leq \mathbf{b}_{\max senior} \qquad (6)$$

Now let $\mathbf{b}_{\boldsymbol{minen}}$ and $\mathbf{b}_{\boldsymbol{maxen}}$ be vectors of length 28, representing the minimum and maximum number of enrolled nurses required to work each shift. In this case, the enrolled nurse constraint would be

$$\mathbf{b}_{\min en} \geq \sum_{i=1}^{i=n} \sum_{i=1}^{i=J_i} \mathbf{e}_i \mathbf{a}_{ij} X_{ij} \leq \mathbf{b}_{\max en} \qquad (7)$$

Finally, there is also a desired level of staff for each shift. Let this desired level of staff be defined as a 28 element vector $\mathbf{b}_{\boldsymbol{desired}}$ as before. Also let $\mathbf{d}^-$ be a vector of 28 elements representing the deviations below the desired level of staff for each shift. The desired level of staff constraint is then given by:

$$\sum_{i=1}^{i=n} \sum_{j=1}^{j=J_i} \mathbf{a}_{ij} X_{ij} + \mathbf{d}^- \geq \mathbf{b}_{\boldsymbol{desired}} \qquad (8)$$

where for each element d_k of $\mathbf{d}^-$, $k = 1 \,..\, 28$, $d_k \geq 0$

It follows that the closer the individual values of d_k are to zero, the closer the solution is to achieving the desired levels of staff for each shift.

A3.4.2 A Two-Phase Optimising Approach

The rostering objectives proposed in Section A3.3 fall into two categories

1. The primary objectives are to achieve the best staffing levels (objectives 1 - 3 and 6).
2. The secondary objectives are to provide staff with the best possible schedules (objectives 4, 5, 7, 8 and 9).

Using this division of objectives into categories, a two phase optimising approach can be developed:

Phase One

Firstly the problem is to meet all the fixed staffing constraints and to minimise the deviations below the desired staffing levels. This is expressed in the following model:

$$\begin{aligned}
&\textit{minimise} \quad z = \sum_{k=1}^{k=28} d_k \\
&\text{subject to} \quad \sum_{j=1}^{j=J_i} X_{ij} = 1, \quad i = 1..n \\
&\mathbf{b}_{mintot} \le \sum_{i=e}^{i=n} \sum_{j=1}^{j=J_i} \mathbf{a}_{ij} X_{ij} \le \mathbf{b}_{maxtot} \\
&\mathbf{b}_{mincharge} \le \sum_{i=e}^{i=n} \sum_{j=1}^{j=J_i} \mathbf{r}_i \mathbf{a}_{ij} X_{ij} \le \mathbf{b}_{maxcharge} \\
&\mathbf{b}_{minsenior} \le \sum_{i=e}^{i=n} \sum_{j=1}^{j=J_i} \mathbf{s}_i \mathbf{a}_{ij} X_{ij} \le \mathbf{b}_{maxsenior} \\
&\mathbf{b}_{minen} \le \sum_{i=e}^{i=n} \sum_{j=1}^{j=J_i} \mathbf{e}_i \mathbf{a}_{ij} X_{ij} \le \mathbf{b}_{maxen} \\
&\sum_{i=1}^{i=n} \sum_{j=1}^{j=J_i} \mathbf{a}_{ij} X_{ij} + \mathbf{d}^- \ge \mathbf{b}_{desired} \\
&X_{ij} = (0 \text{ or } 1), \; i = 1..n; \; j = 1..J_i \\
&\text{and for each element } d_k \text{ of } \mathbf{d}^-, \; k = 1 \,..\, 28, \; d_k \ge 0
\end{aligned}$$

From this the minimum possible level of $\sum_{k=1}^{k=28} d_k$ can be obtained. This minimum level represents the closest a solution can come to meeting the desired staffing levels without violating any of the other constraints. Letting this minimum level be d_{min}, a new model can be created with the additional constraint that $\sum_{k=1}^{k=28} d_k = d_{min}$

Phase Two

The objective of the second phase model is to minimise nurse dissatisfaction with schedules. As in Appendix 2, this can be expressed as:

$$\text{minimise} \quad z = \sum_{i=1}^{i=n} \sum_{j=1}^{j=J_i} c_{ij} X_{ij}$$

where c_{ij} represents the relative value nurse i places on schedule j, such that $c_{ij} \geq 0$ and the value of c_{ij} increases as nurse dissatisfaction with a schedule increases.

However, the definition of c_{ij} is now more complex than the (0,1,2) values used in Section 2.2.1. Each c_{ij} must reflect nurse preferences for requests, shift patterns and day off patterns. In addition, the model should also consider a nurse's previous schedule history in evaluating c_{ij}. For example, if a nurse has not worked a night shift for a long period, relative to other nurses, then schedules containing night shifts for that nurse should have a reduced c_{ij} value. In this way the night shifts can be evenly distributed over time. The evaluation of c_{ij} is therefore a complicated task. The current research retains a fairly simplistic method of calculating c_{ij}. More sophisticated methods, involving nurse questionnaires, are described in the literature (see Miller *et al*. 1976, Warner 1976, Kostreva and Jennings 1991).

The following equations show the full second phase model:

$$\text{minimise} \quad z = \sum_{i=1}^{i=n} \sum_{j=1}^{j=J_i} c_{ij} X_{ij}$$

$$\text{subject to} \quad \sum_{j=1}^{j=J_i} X_{ij} = 1, \quad i = 1..n$$

$$\mathbf{b}_{mintot} \leq \sum_{i=e}^{i=n} \sum_{j=1}^{j=J_i} \mathbf{a}_{ij} X_{ij} \leq \mathbf{b}_{maxtot}$$

$$\mathbf{b}_{mincharge} \leq \sum_{i=e}^{i=n} \sum_{j=1}^{j=J_i} \mathbf{r}_i \mathbf{a}_{ij} X_{ij} \leq \mathbf{b}_{maxcharge}$$

$$\mathbf{b}_{minsenior} \leq \sum_{i=e}^{i=n} \sum_{j=1}^{j=J_i} \mathbf{s}_i \mathbf{a}_{ij} X_{ij} \leq \mathbf{b}_{maxsenior}$$

$$\mathbf{b}_{minen} \leq \sum_{i=e}^{i=n} \sum_{j=1}^{j=J_i} \mathbf{e}_i \mathbf{a}_{ij} X_{ij} \leq \mathbf{b}_{maxen}$$

$$\sum_{i=1}^{i=n} \sum_{j=1}^{j=J_i} \mathbf{a}_{ij} X_{ij} + \mathbf{d}^{-} \geq \mathbf{b}_{desired}$$

$$\sum_{k=1}^{k=28} d_k = d_{min}$$

$$X_{ij} = (0 \text{ or } 1), \; i = 1..n; \; j = 1..J_i$$

and for each element d_k of $\mathbf{d}^-$, $k = 1 \,.\,.\, 28$, $d_k \geq 0$

Appendix 4: A Numerical Comparison of US and Australian Rostering Practices

A4.1 An Australian Example

The following table shows all the allowable patterns of days off for a particular nurse, abiding by the constraints laid out in Appendix 2. It is assumed the nurse has worked the last two days of a previous roster. A ‘0’ in the table represents a day off and a blank represents a day worked. The Total column shows the total allowable schedules for the given row of the table:

Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Total
	0	0				0	0							316
	0	0					0	0						316
	0	0						0	0					312
	0	0							0	0				312
	0	0								0	0			312
		0	0				0	0						348
		0	0					0	0					352
		0	0						0	0				344
		0	0							0	0			344
		0	0								0	0		344
			0	0				0	0					368
			0	0					0	0				368
			0	0						0	0			360
			0	0							0	0		360
			0	0								0	0	592
				0	0				0	0				376
				0	0					0	0			376
				0	0						0	0		368
				0	0							0	0	608
					0	0				0	0			376
					0	0					0	0		376
					0	0						0	0	608

Total Feasible Schedules :	8436

Table i: Total feasible schedules for a full-time nurse

As an example to illustrate how these figures were derived, consider the first row pattern:

`- 0 0 - - - 0 0 - - - - - -`

First, let an ‘E’ represent an early shift, and ‘L’ represent a late shift, an ‘N’ represent a night shift and a ‘-’ represent an undecided shift. The following pattern represents the situation where a nurse works early and late shifts only (remembering that, whenever possible, a nurse receives an early shift before days off and a late shift after days off) :

`E 0 0 L - E 0 0 L - - - - -`

In this case there are 6 undecided shifts that can be either a late shift or an early shift. This means there are $2^6 = 64$ possible non night shift schedules for this pattern of days off. The complete pattern set of allowable night shifts, for the given day off pattern, is shown by the following (remembering that night shifts must be followed by days off, and must occur in a single block of 4 shifts or less):

`N 0 0 L - E 0 0 L - - - - -` = 6 undecided shifts = 2^6 = 64 possible schedules
`E 0 0 L - N 0 0 L - - - - -` = 6 undecided shifts = 2^6 = 64 possible schedules
`E 0 0 L N N 0 0 L - - - - -` = 5 undecided shifts = 2^5 = 32 possible schedules
`E 0 0 N N N 0 0 L - - - - -` = 5 undecided shifts = 2^5 = 32 possible schedules
`E 0 0 L - E 0 0 L - - - - N` = 5 undecided shifts = 2^5 = 32 possible schedules
`E 0 0 L - E 0 0 L - - - N N` = 4 undecided shifts = 2^4 = 16 possible schedules
`E 0 0 L - E 0 0 L - - N N N` = 3 undecided shifts = 2^3 = 8 possible schedules
`E 0 0 L - E 0 0 L - N N N N` = 2 undecided shifts = 2^2 = 4 possible schedules

This makes the total allowable schedules that include night shifts to be:
64 + 64 + 32 + 32 + 32 + 32 + 16 + 8 + 4 = 252
Adding this to the 64 allowable non night shift schedules gives the total allowable schedules for the given day off pattern as 252 + 64 = 316. This corresponds to the total figure for row one of the above table. All the other rows in the table were calculated in a similar fashion.

A4.2 Adding US Constraints

In the case where every change of shift type must be separated by days off (relating to US rostering practices), then the shift type from the previous roster will already be known, and will fix the shift type at the beginning of the new roster. For example, if a nurse worked an early shift at the end of the last roster, then early shifts must be worked up to the first day off in the current roster. Using this. the following patterns show the total feasible schedules for the row one pattern from the above table:

```
E 0 0 E E E 0 0 E E E E E E
E 0 0 E E E 0 0 L L L L L L
E 0 0 E E E 0 0 N N N N N N
E 0 0 L L L 0 0 E E E E E E
E 0 0 L L L 0 0 L L L L L L
E 0 0 L L L 0 0 N N N N N N
E 0 0 N N N 0 0 E E E E E E
E 0 0 N N N 0 0 L L L L L L
E 0 0 N N N 0 0 N N N N N N
```

In the above case there are two blocks of shifts to be assigned, each separated by days off. As each block can have three shift types, the total number of patterns is $3^2 = 9$. The same will hold for the other day off patterns in the above table, except for those patterns ending or beginning with a day off. Those ending with a day off have only one block to assign, and so there are only three possibilities. Those beginning with a day off have three blocks to assign, so there are $3^3 = 27$ possibilities.
Referring to the original table above, there are zero patterns beginning with a day off, three ending with a day off, and 19 remaining. Therefore, the total number of feasible schedules is given by: (3 x 3) + (19 x 9) = 180.

Appendix 5: Selective Elimination of Night Shifts

To illustrate the process of night shift elimination used in the cyclic descent algorithm cost function, consider the following simplified rostering problem: a ward consists of 5 nurses, all of the same seniority level, and for each day there must be *at least* 2 and *no more than* 4 nurses on duty on a day shift and *exactly* one on a night shift. A possible combination of schedules for the five nurses is shown below:

	Mon	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**	**Sun**	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**	**Sun**
N1	*Day*	*Day*	*Day*	*Day*	*Day*	*Off*	*Off*	*Day*	*Night*	*Night*	*Night*	*Night*	*Off*	*Off*
N2	*Off*	*Off*	*Day*	*Day*	*Day*	*Day*	*Day*	*Day*	*Off*	*Off*	*Night*	*Night*	*Night*	*Night*
N3	*Day*	*Night*	*Night*	*Night*	*Night*	*Off*	*Off*	*Day*	*Day*	*Day*	*Day*	*Day*	*Off*	*Off*
N4	*Night*	*Off*	*Off*	*Day*	*Day*	*Day*	*Off*	*Off*	*Day*	*Day*	*Day*	*Day*	*Day*	*Day*
N5	*Day*	*Day*	*Day*	*Night*	*Night*	*Night*	*Night*	*Off*	*Off*	*Day*	*Day*	*Day*	*Off*	*Off*
Day	0	0	0	0	0	0	1	0	0	0	0	0	1	1
Night	0	0	0	1	1	0	0	1	0	0	1	1	0	0
Total	0	0	0	1	1	0	1	1	0	0	1	1	1	1

Table j: Possible combination of schedules for a five nurse roster problem

The scores for the roster in the above table are shown in the last three rows. Each score is a count of the deviation away for either the maximum or minimum constraint for each day or column. Therefore the day score for the final Sunday of the roster is one because only one nurse is working that day where a *minimum* of two are required (2 - 1 = 1). Similarly, the night score for the first Thursday of the roster is one because two nurses are working the night shift, where a *maximum* of one is required. The total row is obtained by summing each day and night column. The final roster score for the roster is obtained by summing the total row, ie 1+1+1+1+1+1+1+1 = 8. The next table shows how the same roster would appear after the application of the night swapping algorithm :

	Mon	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**	**Sun**	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**	**Sun**
N1	*Day*	*Day*	*Day*	*Day*	*Day*	*Off*	*Off*	*Day*	*Night*	*Night*	*Night*	*Night*	*Off*	*Off*
N2	*Off*	*Off*	*Day*	*Day*	*Day*	*Day*	*Day*	*Day*	*Off*	*Off*	*Day*	*Day*	*Night*	*Night*
N3	*Day*	*Night*	*Night*	*Night*	*Night*	*Off*	*Off*	*Day*	*Day*	*Day*	*Day*	*Day*	*Off*	*Off*
N4	*Night*	*Off*	*Off*	*Day*	*Day*	*Day*	*Off*	*Off*	*Day*	*Day*	*Day*	*Day*	*Day*	*Day*
N5	*Day*	*Day*	*Day*	*Day*	*Day*	*Night*	*Night*	*Off*	*Off*	*Day*	*Day*	*Day*	*Off*	*Off*
Day	0	0	0	0	0	0	1	0	0	0	0	0	1	1
Night	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Total	0	0	0	0	0	0	1	1	0	0	0	0	1	1

Table k: Table j Roster after Application of Night Swapping Algorithm

By comparing the previous two tables, it can be seen that nurse five's (N5) night shifts on the first Thursday and Friday of the roster have been changed to day shifts, and nurse two's (N2) night shifts on the second Thursday and Friday of the roster have also been changed to day shifts. Consequently, for the days where a shift exchange has been made there is no longer an excess of nights, and the overall roster score is reduced from 8 to 4.

Appendix 6: Schedule Grade Selection Bias

To illustrate the schedule grade selection bias introduced into the enhanced cyclic descent algorithm, consider the following situation: a nurse has 150 feasible schedules and at a particular iteration in the cyclic descent algorithm, 5 of the 150 schedules are found to cause an improvement in the deviation score for the overall roster. The schedules and their various scores are shown in the following table:

Schedule Number	Schedule Grade	Deviation Score Improvement	Formula Based Roster Cost
12	0	1	9.277
44	12	3	7.289
57	9	3	7.286
89	5	2	8.282
113	4	1	9.281

Table l: Example schedule selection scenario

The above table assumes the existing roster deviation score is 10 and the existing total roster grade is 277. Hence schedule 89 causes the deviation score to improve by 2 giving a new deviation of 10 - 2 = 8, and creates a new total roster grade of 277 + 5 = 282. Using the basic cyclic descent cost formula, this results in a new roster cost of :

$$8 + (282/1000) = 8.282$$

Using the cost formula, schedule 57 would be selected as it results in the lowest overall cost of 7.286. However, using the schedule grade selection bias method, schedule 12 would be selected, because out of the schedules that cause any improvement in deviation score (regardless of the size of improvement), schedule 12 has the best or lowest grade.

Appendix 7: SPSS® Output for Statistical Analysis

A7.1 MANOVA 1 Results

```
* * * * * * A n a l y s i s   o f   V a r i a n c e * * * * * *

       260 cases accepted.
       RTTSCHED = WeightedSchedule
       LN_SHIFT = WeightedShift
       METHOD 1 = Enhanced Cyclic Descent Algorithm
       METHOD 2 = Basic Cyclic Descent Algorithm
       METHOD 3 = Manual Method
       METHOD 4 = Basic Simulated Annealing Algorithm
       METHOD 5 = Enhanced Simulated Annealing Algorithm

 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
              CELL NUMBER
                1    2    3    4    5    6    7    8    9   10
 Variable
   METHOD       1    1    2    2    3    3    4    4    5    5
   WARD         1    2    1    2    1    2    1    2    1    2

 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 Cell Number .. 1 Correlation matrix with Standard Deviations on Diagonal

             RTTSCHED   LN_SHIFT
 RTTSCHED        .375
 LN_SHIFT        .512      1.011

 Cell Number .. 2 Correlation matrix with Standard Deviations on Diagonal

             RTTSCHED   LN_SHIFT
 RTTSCHED        .489
 LN_SHIFT        .175      1.129

 Cell Number .. 3 Correlation matrix with Standard Deviations on Diagonal

             RTTSCHED   LN_SHIFT
 RTTSCHED        .370
 LN_SHIFT        .126      1.022

 Cell Number .. 4 Correlation matrix with Standard Deviations on Diagonal

             RTTSCHED   LN_SHIFT
 RTTSCHED        .495
 LN_SHIFT       -.172       .963

 Cell Number .. 5 Correlation matrix with Standard Deviations on Diagonal

             RTTSCHED   LN_SHIFT
 RTTSCHED        .411
 LN_SHIFT        .505      1.010

 Cell Number .. 6 Correlation matrix with Standard Deviations on Diagonal

             RTTSCHED   LN_SHIFT
 RTTSCHED        .430
 LN_SHIFT        .259       .819

 Cell Number .. 7 Correlation matrix with Standard Deviations on Diagonal

             RTTSCHED   LN_SHIFT
 RTTSCHED        .376
 LN_SHIFT        .405      1.009

 Cell Number .. 8 Correlation matrix with Standard Deviations on Diagonal

             RTTSCHED   LN_SHIFT
 RTTSCHED        .461
 LN_SHIFT        .076      1.153

 Cell Number .. 9 Correlation matrix with Standard Deviations on Diagonal

             RTTSCHED   LN_SHIFT
 RTTSCHED        .345
 LN_SHIFT        .348      1.028

 Cell Number .. 10 Correlation matrix with Standard Deviations on Diagonal

             RTTSCHED   LN_SHIFT
 RTTSCHED        .489
 LN_SHIFT        .136      1.032
```

```
Multivariate test for Homogeneity of Dispersion matrices

Boxs M =                         26.64610
F WITH (27,166501) DF =            .95537, P =   .530 (Approx.)
Chi-Square with 27 DF =          25.79935, P =   .530 (Approx.)

Combined Observed Means for METHOD
Variable .. RTTSCHED
       METHOD
            1          3.64035
            2          3.66877
            3          4.24803
            4          3.68885
            5          3.66798

Variable .. LN_SHIFT
       METHOD
            1          3.25104
            2          4.03184
            3          4.34818
            4          3.24808
            5          3.25851

Combined Observed Means for WARD
Variable .. RTTSCHED
         WARD
            1          3.89826
            2          3.66734

Variable .. LN_SHIFT
         WARD
            1          3.91211
            2          3.34295

EFFECT .. METHOD BY WARD
Multivariate Tests of Significance (S = 2, M = 1/2, N = 123 1/2)
Test Name         Value  Approx. F Hypoth. DF   Error DF  Sig. of F

Pillais          .02038     .64356       8.00     500.00       .741
Hotellings       .02078     .64409       8.00     496.00       .741
Wilks            .97963     .64384       8.00     498.00       .741
Roys             .01961
Note.. F statistic for WILKS' Lambda is exact.


Univariate F-tests with (4,250) D. F.
Variable   Hypoth. SS   Error SS Hypoth. MS   Error MS          F  Sig. of F

RTTSCHED       .03561   45.69130     .00890     .18277     .04871       .996
LN_SHIFT      4.93297  260.73143    1.23324    1.04293    1.18248       .319


EFFECT .. WARD
Multivariate Tests of Significance (S = 1, M = 0, N = 123 1/2)
Test Name         Value    Exact F Hypoth. DF   Error DF  Sig. of F

Pillais          .11410   16.03572       2.00     249.00       .000
Hotellings       .12880   16.03572       2.00     249.00       .000
Wilks            .88590   16.03572       2.00     249.00       .000
Roys             .11410
Note.. F statistics are exact.

Univariate F-tests with (1,250) D. F.
Variable   Hypoth. SS   Error SS Hypoth. MS   Error MS          F  Sig. of F

RTTSCHED      3.46600   45.69130    3.46600     .18277   18.96420       .000
LN_SHIFT     21.05657  260.73143   21.05657    1.04293   20.18990       .000

EFFECT .. METHOD
Multivariate Tests of Significance (S = 2, M = 1/2, N = 123 1/2)

Test Name         Value  Approx. F Hypoth. DF   Error DF  Sig. of F

Pillais          .35025   13.26912       8.00     500.00       .000
Hotellings       .46463   14.40353       8.00     496.00       .000
Wilks            .66937   13.83627       8.00     498.00       .000
Roys             .28023
Note.. F statistic for WILKS' Lambda is exact.

EFFECT .. METHOD (Cont.)
Univariate F-tests with (4,250) D. F.
Variable   Hypoth. SS   Error SS Hypoth. MS   Error MS          F  Sig. of F

RTTSCHED     14.13067   45.69130    3.53267     .18277   19.32899       .000
LN_SHIFT     57.44474  260.73143   14.36119    1.04293   13.77009       .000
```

Estimates for RTTSCHED
--- Joint univariate .9500 BONFERRONI confidence intervals
--- two-tailed observed power taken at .0500 level

METHOD

Parameter	Coeff.	Std. Err.	t-Value	Sig. t	Lower -95%	CL- Upper
2	.028421878	.08384	.33899	.73490	-.18252	.23936
3	.607682903	.08384	7.24798	.00000	.39674	.81862
4	.048507393	.08384	.57856	.56341	-.16243	.25945
5	.027635523	.08384	.32962	.74197	-.18330	.23857

WARD

Parameter	Coeff.	Std. Err.	t-Value	Sig. t	Lower -95%	CL- Upper
6	-.23091775	.05303	-4.35479	.00002	-.33535	-.12648

METHOD BY WARD

Parameter	Coeff.	Std. Err.	t-Value	Sig. t	Lower -95%	CL- Upper
7	.006934232	.16768	.04135	.96705	-.41494	.42881
8	-.03503899	.16768	-.20896	.83465	-.45692	.38684
9	-.05725033	.16768	-.34142	.73307	-.47913	.36463
10	-.02429388	.16768	-.14488	.88492	-.44617	.39758

Estimates for LN_SHIFT
--- Joint univariate .9500 BONFERRONI confidence intervals
--- two-tailed observed power taken at .0500 level

METHOD

Parameter	Coeff.	Std. Err.	t-Value	Sig. t	Lower -95%	CL- Upper
2	.780801153	.20028	3.89853	.00012	.27691	1.28469
3	1.09714055	.20028	5.47800	.00000	.59325	1.60103
4	-.00295951	.20028	-.01478	.98822	-.50685	.50093
5	.007470681	.20028	.03730	.97027	-.49642	.51136

WARD

Parameter	Coeff.	Std. Err.	t-Value	Sig. t	Lower -95%	CL- Upper
6	-.56916362	.12667	-4.49332	.00001	-.81864	-.31969

METHOD BY WARD

Parameter	Coeff.	Std. Err.	t-Value	Sig. t	Lower -95%	CL- Upper
7	.154804834	.40056	.38647	.69948	-.85297	1.16258
8	.644202707	.40056	1.60825	.10904	-.36358	1.65198
9	-.04481936	.40056	-.11189	.91100	-1.05260	.96296
10	-.12903974	.40056	-.32215	.74761	-1.13682	.87874

A7.2 MANOVA 2 Results

```
* * * * * * A n a l y s i s   o f   V a r i a n c e * * * * * *

       208 cases accepted.
       RTTSCHED = WeightedSchedule
       LN_SHIFT = WeightedShift
       RTL_TIME = ExecutionTime
       METHOD 1 = Enhanced Cyclic Descent Algorithm
       METHOD 2 = Basic Cyclic Descent Algorithm
       METHOD 3 = Basic Simulated Annealing Algorithm
       METHOD 4 = Enhanced Simulated Annealing Algorithm

 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
              CELL NUMBER
                1    2    3    4    5    6    7    8
 Variable
   METHOD       1    1    2    2    3    3    4    4
   WARD         1    2    1    2    1    2    1    2


 Cell Number .. 1 Correlation matrix with Standard Deviations on Diagonal

             LN_SHIFT   RTTSCHED   RTL_TIME
 LN_SHIFT       1.011
 RTTSCHED        .512       .375
 RTL_TIME       -.205      -.022       .345

 Cell Number .. 2 Correlation matrix with Standard Deviations on Diagonal

             LN_SHIFT   RTTSCHED   RTL_TIME
 LN_SHIFT       1.129
 RTTSCHED        .175       .489
 RTL_TIME        .046      -.110       .351

 Cell Number .. 3 Correlation matrix with Standard Deviations on Diagonal

             LN_SHIFT   RTTSCHED   RTL_TIME
 LN_SHIFT       1.022
 RTTSCHED        .126       .370
 RTL_TIME       -.119      -.387       .157

 Cell Number .. 4 Correlation matrix with Standard Deviations on Diagonal

             LN_SHIFT   RTTSCHED   RTL_TIME
 LN_SHIFT        .963
 RTTSCHED       -.172       .495
 RTL_TIME       -.430       .352       .129

 Cell Number .. 5 Correlation matrix with Standard Deviations on Diagonal

             LN_SHIFT   RTTSCHED   RTL_TIME
 LN_SHIFT       1.009
 RTTSCHED        .405       .376
 RTL_TIME        .242       .269       .249

 Cell Number .. 6 Correlation matrix with Standard Deviations on Diagonal

             LN_SHIFT   RTTSCHED   RTL_TIME
 LN_SHIFT       1.153
 RTTSCHED        .076       .461
 RTL_TIME        .132      -.086       .181

 Cell Number .. 7 Correlation matrix with Standard Deviations on Diagonal

             LN_SHIFT   RTTSCHED   RTL_TIME
 LN_SHIFT       1.028
 RTTSCHED        .348       .345
 RTL_TIME        .121       .138       .281

 Cell Number .. 8 Correlation matrix with Standard Deviations on Diagonal

             LN_SHIFT   RTTSCHED   RTL_TIME

 LN_SHIFT       1.032
 RTTSCHED        .136       .489
 RTL_TIME       -.006       .213       .319
```

```
Combined Observed Means for METHOD
Variable .. LN_SHIFT
       METHOD
            1            3.25104
            2            4.03184
            3            3.24808
            4            3.25851
Variable .. RTTSCHED
       METHOD
            1            3.64035
            2            3.66877
            3            3.68885
            4            3.66798
Variable .. RTL_TIME
       METHOD
            1            2.21215
            2            1.70592
            3            2.86652
            4            2.42389
Combined Observed Means for WARD
Variable .. LN_SHIFT
         WARD
            1            3.79684
            2            3.09789
Variable .. RTTSCHED
         WARD
            1            3.78031
            2            3.55267
Variable .. RTL_TIME
         WARD
            1            2.28234
            2            2.32190

EFFECT .. METHOD BY WARD
Multivariate Tests of Significance (S = 3, M = -1/2, N = 98 )
Test Name         Value  Approx. F Hypoth. DF   Error DF  Sig. of F

Pillais          .00730     .16272       9.00     600.00       .997
Hotellings       .00733     .16017       9.00     590.00       .998
Wilks            .99271     .16127       9.00     482.03       .997
Roys             .00438

Univariate F-tests with (3,200) D. F.
Variable   Hypoth. SS   Error SS Hypoth. MS   Error MS          F  Sig. of F

LN_SHIFT       .55294  218.49253     .18431    1.09246     .16871       .917
RTTSCHED       .03281   36.83487     .01094     .18417     .05939       .981
RTL_TIME       .05731   13.98342     .01910     .06992     .27321       .845

EFFECT .. WARD
Multivariate Tests of Significance (S = 1, M = 1/2, N = 98 )
Test Name         Value    Exact F Hypoth. DF   Error DF  Sig. of F

Pillais          .14454   11.15140       3.00     198.00       .000
Hotellings       .16896   11.15140       3.00     198.00       .000
Wilks            .85546   11.15140       3.00     198.00       .000
Roys             .14454
Note.. F statistics are exact.

Univariate F-tests with (1,200) D. F.
Variable   Hypoth. SS   Error SS Hypoth. MS   Error MS          F  Sig. of F

LN_SHIFT     25.40412  218.49253   25.40412    1.09246   23.25399       .000
RTTSCHED      2.69465   36.83487    2.69465     .18417   14.63097       .000
RTL_TIME       .08136   13.98342     .08136     .06992    1.16362       .282

EFFECT .. METHOD
Multivariate Tests of Significance (S = 3, M = -1/2, N = 98 )
Test Name         Value  Approx. F Hypoth. DF   Error DF  Sig. of F

Pillais          .75945   22.59699       9.00     600.00       .000
Hotellings      2.69859   58.96929       9.00     590.00       .000
 Wilks            .26403   39.01128       9.00     482.03       .000
Roys             .72717

Univariate F-tests with (3,200) D. F.
Variable   Hypoth. SS   Error SS Hypoth. MS   Error MS          F  Sig. of F

LN_SHIFT     23.68788  218.49253    7.89596    1.09246    7.22767       .000
RTTSCHED       .06193   36.83487     .02064     .18417     .11209       .953
RTL_TIME     36.23990   13.98342   12.07997     .06992  172.77560       .000
```

Estimates for LN_SHIFT
--- Joint univariate .9500 BONFERRONI confidence intervals
--- two-tailed observed power taken at .0500 level

METHOD

Parameter	Coeff.	Std. Err.	t-Value	Sig. t	Lower -95%	CL- Upper
2	.780801153	.20498	3.80911	.00019	.28592	1.27569
3	-.00295951	.20498	-.01444	.98850	-.49785	.49193
4	.007470681	.20498	.03645	.97096	-.48742	.50236

WARD

Parameter	Coeff.	Std. Err.	t-Value	Sig. t	Lower -95%	CL- Upper
5	-.69895688	.14494	-4.82224	.00000	-.98477	-.41314

METHOD BY WARD

Parameter	Coeff.	Std. Err.	t-Value	Sig. t	Lower -95%	CL- Upper
6	.154804834	.40996	.37761	.70612	-.83497	1.14458
7	-.04481936	.40996	-.10932	.91305	-1.03459	.94495
8	-.12903974	.40996	-.31476	.75327	-1.11881	.86073

Estimates for RTTSCHED
--- Joint univariate .9500 BONFERRONI confidence intervals
--- two-tailed observed power taken at .0500 level

METHOD

Parameter	Coeff.	Std. Err.	t-Value	Sig. t	Lower -95%	CL- Upper
2	.028421878	.08416	.33770	.73595	-.17477	.23162
3	.048507393	.08416	.57634	.56503	-.15469	.25170
4	.027635523	.08416	.32835	.74299	-.17556	.23083

WARD

Parameter	Coeff.	Std. Err.	t-Value	Sig. t	Lower -95%	CL- Upper
5	-.22764045	.05951	-3.82504	.00017	-.34499	-.11029

METHOD BY WARD

Parameter	Coeff.	Std. Err.	t-Value	Sig. t	Lower -95%	CL- Upper
6	.006934232	.16833	.04119	.96718	-.39946	.41333
7	-.05725033	.16833	-.34011	.73413	-.46364	.34914
8	-.02429388	.16833	-.14432	.88539	-.43069	.38210

Estimates for RTL_TIME
--- Joint univariate .9500 BONFERRONI confidence intervals
--- two-tailed observed power taken at .0500 level

METHOD

Parameter	Coeff.	Std. Err.	t-Value	Sig. t	Lower -95%	CL- Upper
2	-.50623084	.05186	-9.76211	.00000	-.63143	-.38103
3	.654366296	.05186	12.61874	.00000	.52917	.77956
4	.211739743	.05186	4.08317	.00006	.08654	.33694

WARD

Parameter	Coeff.	Std. Err.	t-Value	Sig. t	Lower -95%	CL- Upper
5	.039554532	.03667	1.07871	.28201	-.03275	.11186

METHOD BY WARD

Parameter	Coeff.	Std. Err.	t-Value	Sig. t	Lower -95%	CL- Upper
6	.067905849	.10371	.65475	.51338	-.18249	.31830
7	.051207971	.10371	.49374	.62203	-.19919	.30160
8	.090090083	.10371	.86864	.38608	-.16030	.34048

A7.3 MANOVA 3 Results

```
* * * * * * A n a l y s i s   o f   V a r i a n c e * * * * * *

       170 cases accepted.
       RTTSCHED = WeightedSchedule
       LN_SHIFT = WeightedShift
       RTL_TIME = ExecutionTime
       METHOD 1 = Enhanced Cyclic Descent Algorithm
       METHOD 2 = Basic Cyclic Descent Algorithm
       METHOD 3 = Basic Simulated Annealing Algorithm
       METHOD 4 = Enhanced Simulated Annealing Algorithm
       METHOD 5 = ILP Algorithm

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
              CELL NUMBER
                1    2    3    4    5    6    7    8    9   10
 Variable
   METHOD       1    1    2    2    3    3    4    4    5    5
   WARD         1    2    1    2    1    2    1    2    1    2

 Cell Number .. 1 Correlation matrix with Standard Deviations on Diagonal

             LN_SHIFT   RTTSCHED   RTL_TIME
 LN_SHIFT       1.011
 RTTSCHED        .512       .375
 RTL_TIME       -.205      -.022       .345

 Cell Number .. 2 Correlation matrix with Standard Deviations on Diagonal

             LN_SHIFT   RTTSCHED   RTL_TIME
 LN_SHIFT       1.084
 RTTSCHED        .100       .380
 RTL_TIME       -.559      -.119       .396

 Cell Number .. 3 Correlation matrix with Standard Deviations on Diagonal

             LN_SHIFT   RTTSCHED   RTL_TIME
 LN_SHIFT       1.022
 RTTSCHED        .126       .370
 RTL_TIME       -.119      -.387       .157

 Cell Number .. 4 Correlation matrix with Standard Deviations on Diagonal

             LN_SHIFT   RTTSCHED   RTL_TIME
 LN_SHIFT        .812
 RTTSCHED        .258       .407
 RTL_TIME       -.826      -.010       .159

 Cell Number .. 5 Correlation matrix with Standard Deviations on Diagonal

             LN_SHIFT   RTTSCHED   RTL_TIME
 LN_SHIFT       1.009
 RTTSCHED        .405       .376
 RTL_TIME        .242       .269       .249

 Cell Number .. 6 Correlation matrix with Standard Deviations on Diagonal

             LN_SHIFT   RTTSCHED   RTL_TIME
 LN_SHIFT       1.162
 RTTSCHED       -.118       .348
 RTL_TIME        .320      -.123       .197

 Cell Number .. 7 Correlation matrix with Standard Deviations on Diagonal

             LN_SHIFT   RTTSCHED   RTL_TIME
 LN_SHIFT       1.028
 RTTSCHED        .348       .345
 RTL_TIME        .121       .138       .281

 Cell Number .. 8 Correlation matrix with Standard Deviations on Diagonal

             LN_SHIFT   RTTSCHED   RTL_TIME
 LN_SHIFT        .985
 RTTSCHED       -.008       .352
 RTL_TIME        .393       .454       .300

 Cell Number .. 9 Correlation matrix with Standard Deviations on Diagonal

             LN_SHIFT   RTTSCHED   RTL_TIME
 LN_SHIFT       1.016
 RTTSCHED        .498       .433
 RTL_TIME       -.090      -.305       .393
```

```
Cell Number .. 10 Correlation matrix with Standard Deviations on Diagonal

             LN_SHIFT   RTTSCHED   RTL_TIME
LN_SHIFT        1.084
RTTSCHED         .168       .429
RTL_TIME        -.118      -.099       .288

Multivariate test for Homogeneity of Dispersion matrices

Boxs M =                        64.92509
F WITH (54,6647) DF =            1.06114, P =   .354 (Approx.)
Chi-Square with 54 DF =         57.82874, P =   .336 (Approx.)


Combined Observed Means for METHOD
Variable .. LN_SHIFT
       METHOD
            1        WGT.     3.53673
                   UNWGT.     3.46765
            2        WGT.     4.20255
                   UNWGT.     4.09120
            3        WGT.     3.53992
                   UNWGT.     3.45255
            4        WGT.     3.53342
                   UNWGT.     3.37964
            5        WGT.     3.53280
                   UNWGT.     3.46508
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Variable .. RTTSCHED
       METHOD
            1        WGT.     3.71523
                   UNWGT.     3.68192
            2        WGT.     3.71373
                   UNWGT.     3.65065
            3        WGT.     3.77193
                   UNWGT.     3.71564
            4        WGT.     3.73467
                   UNWGT.     3.67847
            5        WGT.     3.50843
                   UNWGT.     3.47378
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Variable .. RTL_TIME
       METHOD
            1        WGT.     2.22220
                   UNWGT.     2.22634
            2        WGT.     1.68169
                   UNWGT.     1.68546
            3        WGT.     2.86051
                   UNWGT.     2.87539
            4        WGT.     2.37164
                   UNWGT.     2.35637
            5        WGT.     2.40373
                   UNWGT.     2.48246
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Combined Observed Means for WARD
Variable .. LN_SHIFT
         WARD
            1        WGT.     3.75607
                   UNWGT.     3.75607
            2        WGT.     3.38637
                   UNWGT.     3.38637
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Variable .. RTTSCHED
         WARD
            1        WGT.     3.73209
                   UNWGT.     3.73209
            2        WGT.     3.54809
                   UNWGT.     3.54809
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Variable .. RTL_TIME
         WARD
            1        WGT.     2.29263
                   UNWGT.     2.29263
            2        WGT.     2.35778
                   UNWGT.     2.35778
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

EFFECT .. METHOD BY WARD
Multivariate Tests of Significance (S = 3, M = 0, N = 78)

Test Name	Value	Approx. F	Hypoth. DF	Error DF	Sig. of F
Pillais	.03752	.50665	12.00	480.00	.911
Hotellings	.03876	.50606	12.00	470.00	.911
Wilks	.96258	.50611	12.00	418.32	.911
Roys	.03451				

Univariate F-tests with (4,160) D. F.

Variable	Hypoth. SS	Error SS	Hypoth. MS	Error MS	F	Sig. of F
LN_SHIFT	.45022	166.71743	.11256	1.04198	.10802	.980
RTTSCHED	.06589	23.30456	.01647	.14565	.11310	.978
RTL_TIME	.45366	13.73896	.11341	.08587	1.32079	.264

EFFECT .. WARD
Multivariate Tests of Significance (S = 1, M = 1/2, N = 78)

Test Name	Value	Exact F	Hypoth. DF	Error DF	Sig. of F
Pillais	.05740	3.20714	3.00	158.00	.025
Hotellings	.06090	3.20714	3.00	158.00	.025
Wilks	.94260	3.20714	3.00	158.00	.025
Roys	.05740				

Note.. F statistics are exact.

Univariate F-tests with (1,160) D. F.

Variable	Hypoth. SS	Error SS	Hypoth. MS	Error MS	F	Sig. of F
LN_SHIFT	4.18074	166.71743	4.18074	1.04198	4.01229	.047
RTTSCHED	1.03562	23.30456	1.03562	.14565	7.11016	.008
RTL_TIME	.12985	13.73896	.12985	.08587	1.51221	.221

EFFECT .. METHOD
Multivariate Tests of Significance (S = 3, M = 0, N = 78)

Test Name	Value	Approx. F	Hypoth. DF	Error DF	Sig. of F
Pillais	.72649	12.78182	12.00	480.00	.000
Hotellings	1.90997	24.93568	12.00	470.00	.000
Wilks	.32592	18.39440	12.00	418.32	.000
Roys	.64606				

Univariate F-tests with (4,160) D. F.

Variable	Hypoth. SS	Error SS	Hypoth. MS	Error MS	F	Sig. of F
LN_SHIFT	12.09608	166.71743	3.02402	1.04198	2.90218	.024
RTTSCHED	1.45757	23.30456	.36439	.14565	2.50178	.045
RTL_TIME	24.41562	13.73896	6.10391	.08587	71.08434	.000

Estimates for LN_SHIFT
--- Joint univariate .9500 BONFERRONI confidence intervals
--- two-tailed observed power taken at .0500 level

METHOD

Parameter	Coeff.	Std. Err.	t-Value	Sig. t	Lower -95%	CL- Upper
2	.623551973	.29183	2.13673	.03414	-.11368	1.36078
3	-.01510208	.29183	-.05175	.95879	-.75233	.72213
4	-.08801436	.29183	-.30160	.76335	-.82524	.64921
5	-.00256791	.29183	-.00880	.99299	-.73980	.73466

WARD

Parameter	Coeff.	Std. Err.	t-Value	Sig. t	Lower -95%	CL- Upper
6	-.36970008	.18457	-2.00307	.04686	-.73420	-.00520

METHOD BY WARD

Parameter	Coeff.	Std. Err.	t-Value	Sig. t	Lower -95%	CL- Upper
7	-.15969353	.58365	-.27361	.78474	-1.63415	1.31476
8	-.06910450	.58365	-.11840	.90590	-1.54356	1.40535
9	-.32000984	.58365	-.54829	.58426	-1.79446	1.15444
10	.005135823	.58365	.00880	.99299	-1.46932	1.47959

```
Estimates for RTTSCHED
 --- Joint univariate .9500 BONFERRONI confidence intervals
 --- two-tailed observed power taken at .0500 level

 METHOD

  Parameter      Coeff.  Std. Err.    t-Value     Sig. t Lower -95%  CL- Upper

        2   -.03126208     .10911    -.28653     .77485    -.30690     .24437
        3   .033724097     .10911     .30909     .75765    -.24191     .30936
        4   -.00344982     .10911    -.03162     .97482    -.27908     .27218
        5   -.20814124     .10911   -1.90768     .05822    -.48377     .06749

 WARD

  Parameter      Coeff.  Std. Err.    t-Value     Sig. t Lower -95%  CL- Upper

        6   -.18400219     .06901   -2.66649     .00845    -.32028    -.04772

 METHOD BY WARD

  Parameter      Coeff.  Std. Err.    t-Value     Sig. t Lower -95%  CL- Upper

        7   -.11243368     .21821    -.51524     .60709    -.66370     .43883
        8   -.08681693     .21821    -.39785     .69127    -.63808     .46445
        9   -.08646457     .21821    -.39624     .69246    -.63773     .46480
       10   -.00505600     .21821    -.02317     .98154    -.55632     .54621

 Estimates for RTL_TIME
 --- Joint univariate .9500 BONFERRONI confidence intervals
 --- two-tailed observed power taken at .0500 level

 METHOD

  Parameter      Coeff.  Std. Err.    t-Value     Sig. t Lower -95%  CL- Upper

        2   -.54088081     .08377   -6.45642     .00000    -.75252    -.32925
        3   .649045162     .08377    7.74756     .00000     .43741     .86068
        4   .130026779     .08377    1.55211     .12261    -.08161     .34166
        5   .256115151     .08377    3.05721     .00262     .04448     .46775

 WARD

  Parameter      Coeff.  Std. Err.    t-Value     Sig. t Lower -95%  CL- Upper

        6   .065154639     .05298    1.22972     .22061    -.03948     .16979


 METHOD BY WARD

  Parameter      Coeff.  Std. Err.    t-Value     Sig. t Lower -95%  CL- Upper

        7   -.00139409     .16755    -.00832     .99337    -.42466     .42188
        8   .040565703     .16755     .24211     .80900    -.38270     .46384
        9   -.07333584     .16755    -.43770     .66219    -.49661     .34993
       10   .281764559     .16755    1.68169     .09458    -.14151     .70503
```

Appendix 8: Unconstrained Schedules in an ILP Model

A promising approach to reducing the size of an ILP roster problem containing nurses with unconstrained schedules would be to model the unconstrained schedule shift by shift instead of modelling all feasible schedules. For instance, consider a nurse able to work any six shifts in a roster, up to three of which can be night shifts and requiring one day off at the end of each night shift block (this is an example taken from Ward 2 in the study). Instead of generating variables to represent all feasible schedules for such a nurse, a reasonably simple set of constraints can be developed to define the schedule:

Let the unconstrained schedule be defined by 28 variables x_1 .. x_{14} and y_1 .. y_{14}

where x_1 .. x_{14} represent the 14 day shifts in the roster,
x_1 = first Monday, x_2 = first Tuesday etc,
such that x_i = 1 or 0,
if x_i = 1 then a day shift is worked on day i,
if x_i = 0 then a day shift is not worked on day i, i = 1..14
similarly y_1 .. y_{14} represent the 14 night shifts in the roster.

The variables can be simply incorporated into the existing model, ie for each Monday day shift constraint x_1 is added to the left hand side of the equation, and so on for each day constraint, each night constraint and each seniority level constraint in which the unconstrained schedule participates.

Additional constraints can then be defined to limit the total shifts in the schedule to six, with no more than three nights and a day off after each night block:

Firstly, to ensure either a day or a night shift is worked, or neither, but not both, the following constraints can be defined:

$$x_i + y_i \leq 1 \text{ for } i = 1..14$$

Secondly to ensure the correct total number of shifts are worked:

$$\sum_{i=1}^{i=14} x_i + \sum_{j=1}^{j=14} y_j = 6$$

Thirdly to ensure the correct number of nights are worked

$$\sum_{i=1}^{i=14} y_i \leq 3$$

Finally to ensure that night shifts are not immediately followed by day shifts:

$$y_i + x_{i+1} \leq 1 \text{ for } i = 1..13$$

Whilst adding 29 new constraints to the model this is much less overhead than the 20,000 to 30,000 feasible schedules that would otherwise have been generated for this nurse.

BIBLIOGRAPHY

Abramson, D. 1992. A Very High Speed Architecture for Simulated Annealing. *IEEE Computer,* May: 27-36.

Abramson, D., Dang, H. and Krishnamoorthy, M. In press. Enhanced Simulated Annealing for 0-1 Problems with Linear Programming Preprocessing. *Annals of Operations Research.*

Arthur, J.L. and Ravindran, A. 1981. A Multiple Objective Nurse Scheduling Model. *AIIE Transactions* **13** (1): 55-60.

Australian Nursing Federation. 1992. Principles of Rostering in the Australian Nursing Industry. *The Australian Nurses Journal* **21** (8): 26.

Baker, K.R. 1974. *Introduction to Sequencing and Scheduling.* John Wiley and Sons, New York.

Baker, K.R. 1976. Workforce Allocation in Cyclical Scheduling Problems. *Operational Research Quarterly* **27** (1): 155-167.

Bell, P.C., Hay, G. and Liang, Y. 1986. A Visual Interactive Decision Support System for Workforce (Nurse) Scheduling. *Infor* **24** (2): 134-145.

Biegel, J. and Davern, J. 1990. Genetic Algorithms and Job Shop Scheduling. *Computers and Industrial Engineering* **19** (1-4): 81-91.

Blau, R.A. and Sear A.M. 1983. Nurse Scheduling with a Microcomputer. *Journal of Ambulatory Care Management* **6**: 1-13.

Carling, A. 1992. *Introducing Neural Networks.* Sigma Press, Wilmslow, UK.

Chow, K.P. and Hui, C.K. 1993. Knowledge-Based System for Rostering. *Expert Systems with Applications* **6**: 361-375.

Collins, N.E., Eglese, R.W. and Golden, B.L. 1988. Simulated Annealing - An Annotated Bibliography. *American Journal of Mathematical and Management Sciences* **8** (3): 209-307.

Colorni, A., Dorigo, M. and Maniezzo, V. 1991. *Using Genetic Algorithms to Solve the Timetable Problem*. Unpublished Working Paper, Dipartimento di Elettronica, Politecnico di Milano.

Connolly, D. 1992. General Purpose Simulated Annealing. *Journal of Operational Research Society* **43** (5): 495-505.

Dhar, V. and Ranganathan, N. 1990. Integer Programming vs. Expert Systems: An Experimental Comparison. *Communications of the ACM* **33** (3): 323-336.

Frances, M.A. 1966. Implementing a Program of Cyclical Scheduling of Nursing Personnel. *Hospitals, J.A.H.A.* **40**: 108-125.

Goldberg, D. 1989. *Genetic Algorithms in Search Optimization and Machine Learning.* Addison-Wesley, New York.

Graves, G.W., McBride, R.D., Gershkoff, D.A. and Mahidhara, D. 1993. Flight Crew Scheduling. *Management Science* **39** (6): 736-745.

Healy, W.C. 1964. Multiple Choice Programming. *Operations Research* **12**: 122-138.

Hillier, F.S. and Lieberman, G.J. 1990. *Introduction to Operations Research.* McGraw Hill, New York.

Hoffman, K.L. and Padberg, M. 1993. Solving Airline Crew Scheduling Problems by Branch-and-Cut. *Management Science* **39** (6): 657-682.

Howell, J.P. 1966. Cyclical Scheduling of Nursing Personnel. *Hospitals, J.A.H.A.* **40**: 77-85.

Hui, C.K. 1988. *Knowledge-based approach to roster scheduling problems*. Unpublished master's thesis, University of Hong Kong.

Hung, R. 1995. Hospital Nurse Scheduling. *Journal of Nursing Administration* **25** (7): 21-23.

Johnston, M. and Adorf, H. 1992. Scheduling with Neural Networks - The Case of the Hubble Space Telescope. *Computers and Operations Research* **19** (3-4): 209-240.

Kosho, B. 1992. *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence.* Prentice Hall, Englewood Cliffs, New Jersey.

Kostreva, M. and Genevier, P. 1989. Nurse Preferences vs. Circadian Rhythms in Scheduling. *Nursing Management* **20** (7): 50-62.

Kostreva, M. and Jennings, K. 1991. Nurse Scheduling on a Microcomputer. *Computers and Operations Research* **18** (8): 731-739.

Kostreva, M., Leszcynski, M. and Passini, F. 1978. The Nurse Scheduling Decision via Mixed-Integer Programming. pp. 281-305 in *Proceedings of the American Hospital Association Forum on Nurse Scheduling*.

Lo, Z.P. and Bavarian, B. 1992. Optimization of Job Scheduling on Parallel Machines by Simulated Annealing Algorithms. *Expert Systems with Applications* **4**: 323-328.

Lukman, D. 1986. *An Hierarchical Approach in Schedule Formulation and Maintenance under Uncertainty*. Unpublished doctoral thesis, University of Pittsburgh. Cited by Sitompul 1992.

Megeath, J.D. 1978. Successful Hospital Personnel Scheduling. *Interfaces* **8** (2): 55-59.

Miller, H.E., Pierskalla, W.P. and Rath, G.J. 1976. Nurse Scheduling Using Mathematical Programming. *Operations Research* **24** (5): 857-870.

Musa, A.A. and Saxena, U. 1984. Scheduling Nurses Using Goal-Programming Techniques. *IIE Transactions* **16** (3): 216-221.

Papadimitriou, C.H. and Steiglitz, K. 1982. *Combinatorial Optimisation: Algorithms and Complexity.* Prentice-Hall, Englewood Cliffs, New Jersey.

Ozkarahan, I. and Bailey, J.E. 1988. Goal Programming Model Subsystem of a Flexible Nurse Scheduling Support System. *IIE Transactions* **20** (3): 306-316.

Randhawa, S.U. and Sitompul, D. 1990. Nurse Scheduling: A State-of-the-Art Review. *Journal of the Society for Health Systems* **2** (1): 62-72.

Randhawa, S.U. and Sitompul, D. 1993. A Heuristic-Based Computerized Nurse Scheduling System. *Computers and Operations Research* **20** (8): 837-844.

Rosenbloom, E.S. and Goertzen, N.F. 1987. Cyclic Nurse Scheduling. *European Journal of Operational Research* **31**: 19-23.

Rothstein, M. 1973. Hospital Manpower Shift Scheduling by Mathematical Programming. *Health Services Research Journal* **8**: 60-66.

Sandhu, B., Kerouac, S. and Duquette, A. 1992. A Case Study of a Nursing Assignment Pattern. *Canadian Journal of Nursing Research* **24** (1): 51-64.

Sitompul, D. 1992. *Design and Implementation of a Heuristic-Based Decision Support System for Nurse Scheduling*. Unpublished doctoral thesis, Oregon State University.

Smith, L.D. and Wiggins, A. 1977. A Computer-Based Nurse Scheduling System. *Computers and Operations Research* **4**: 195-212.

Tabachnick, B.G. and Fidell, L.S. 1989. *Using Multivariate Statistics*. HarperCollins, New York.

Taha, H.A. 1992. *Operations Research: An Introduction.* 5th edn Maxwell Macmillan International, Singapore.

Turban, E. 1990. *Decision Support and Expert Systems : Management Support Systems.* Maxwell Macmillan International, Singapore.

Warner, D.M. 1976. Scheduling Nursing Personnel According to Nursing Preference: A Mathematical Programming Approach. *Operations Research* **24** (5): 842-856.