An Investigation Of Case-based Heuristic Selection For University Timetabling

by Adam Eckersley, BSc

Thesis submitted to the University of Nottingham
for the degree of Doctor of Philosophy,
October, 2004

Contents

1 Introduction **9**
- 1.1 Timetabling . . . 9
  - 1.1.1 The University Timetabling Problem . . . 11
  - 1.1.2 Examination Timetabling . . . 12
  - 1.1.3 Complexity Issues . . . 14
- 1.2 Local Search Meta-heuristics . . . 15
- 1.3 Aims and motivation . . . 18
- 1.4 Layout of the thesis . . . 19

2 Exam Timetabling survey **22**
- 2.1 Introduction . . . 22
- 2.2 Algorithmic techniques . . . 26
  - 2.2.1 Constraint based methods . . . 26
  - 2.2.2 Graph-based Initialisation and Construction techniques . 32
  - 2.2.3 Local Search Meta-heuristics . . . 39
  - 2.2.4 Evolutionary Methods . . . 56
  - 2.2.5 Multi-criteria approaches . . . 60
  - 2.2.6 Case-based reasoning and hyper-heuristic methods . . . 65
- 2.3 Results for Benchmark Problems . . . 73
- 2.4 Conclusions . . . 79

3 Case based reasoning (CBR) **82**
- 3.1 Introduction to CBR . . . 82
- 3.2 CBR and Scheduling . . . 85

4 Investigating Similarity Measures For Exam Timetabling Problems **90**
- 4.1 The need for a similarity measure . . . 91
- 4.2 Data sets . . . 93
- 4.3 Analysis of Data Sets . . . 95
  - 4.3.1 Removing redundancy from Data Sets . . . 95
  - 4.3.2 Examining subsets of the Data Sets . . . 98
- 4.4 Conclusions . . . 103

5 Using Simulated Annealing to Study the Behaviour of Exam Timetabling Data Sets **105**
- 5.1 Initialisation heuristics . . . 106

5.2 Experiments using a LD heuristic with randomisation 108
5.3 Experiments using a greedy LD heuristic 114
5.4 Conclusions 118

6 Analysing Features For Similarity In Examination Timetabling 119
6.1 The requirements of a CBR system 120
6.2 Qualitative analysis of features used within the CBR system . . 122
6.2.1 *Number of Students* 122
6.2.2 *Number of Events (Exams)* 123
6.2.3 *Number of Periods* 125
6.2.4 *Conflict Matrix Density* 126
6.2.5 *Largest Degree* 127
6.2.6 *Further Conflict Matrix Measures* 128
6.2.7 *Fluidity Analysis* 129
6.2.8 *Cliques* 133
6.2.9 *Side Constraints & the Objective function* 134
6.3 Conclusions 136

7 A Variable Neighbourhood Search (VNS) technique for Exam Timetabling 138
7.1 The Importance of the Neighbourhood 139
7.2 Variable Neighbourhood Search 141
7.3 VNS for Exam Timetabling 143
7.3.1 Neighbourhoods used within VNS 145
7.3.2 Variations of VNS for exam timetabling 149
7.4 Results 153
7.5 Conclusions 159

8 Combining a Genetic Algorithm with VNS to improve solution quality 161
8.1 Combining VNS with CBR - the requirements 162
8.2 The GA technique for neighbourhood selection 164
8.2.1 The neighbourhoods 167
8.3 Results 170
8.3.1 Notes on Results 173
8.4 Combining VNS-GA with CBR 176
8.4.1 More complex timetabling problems 179
8.5 Conclusions 183

9 Conclusions and Future Work 186
9.1 Measuring Similarity for CBR 187
9.2 Developing the meta-heuristic technique(s) 189
9.3 Future Work 192

List of Tables

2.1 Graph colouring benchmark data sets - minimum number of periods reported [45] 74
2.2 Characteristics of uncapacitated benchmark problems [45] . . . 75
2.3 Selected results from the literature on uncapacitated benchmark problems from [45] (best results given) 76
2.3 (cont.) Selected results from the literature on uncapacitated benchmark problems from [45] (best results given) 76
2.4 Results on Capacitated benchmark problems (set 1) with objective to minimise occurrences of two consecutive exams in a day (best results shown) 78
2.5 Results on Capacitated benchmark problems (set 2) with objective to minimise occurrences of two exams in consecutive time slots (best results shown) 78

4.1 Simple features for benchmark data sets 94
4.2 Features for benchmark data sets based on students and enrolments 95
4.3 Results of simulated annealing applied to three benchmark data sets with all single enrolment students removed 96
4.4 Percentages and numbers of students in the Base Set, Singleton Set and Weights Set for benchmark data sets 99

5.1 Results from SA initialised by the same solution each time, using the standard objective function 109
5.2 Results from SA initialised by the same solution each time, using the simplified objective function 110
5.3 Results from SA initialised by a different solution each time, using the standard objective function 110
5.4 Results from SA initialised by a different solution each time, using the simplified objective function 111
5.5 Results from SA initialised with a greedy LD heuristic and using the standard objective function 115
5.6 Results from SA initialised with a greedy LD heuristic, using the simplified objective function 115

6.1 Percentage of total number of exams which never move in x runs out of 100 of Simulated Annealing using the simple move neighbourhood 130

7.1 Results from the basic VNS meta-heuristic with random and greedy initialisations 155
7.2 Results from the VNS with biased neighbourhoods meta-heuristic with random and greedy initialisations 155
7.3 Ascent-descent Biased VNS compared to results from the literature (best results given) 158

8.1 Best results obtained from the VNS-GA algorithm with neighbourhoods given 172
8.2 Results from VNS comparing all neighbourhoods with the 'best' subset of neighbourhoods 173

List of Figures

7.1 The steps of the Basic VNS meta-heuristic 143
7.2 A Kempe chain move before execution 148
7.3 The result of a Kempe chain neighbourhood move 148

8.1 The one-point crossover operator for chromosomes of length 8 . 167
8.2 The steps of VNS-GA procedure 168

Abstract

The research presented in this thesis contributes to a larger project to develop a case-based reasoning (CBR) meta-heuristic selector for exam timetabling problems. The theory of the CBR system is that *similar* problems can be solved equally well by the same technique. Chapter 2 gives a detailed survey of current techniques applied to exam timetabling to give a good indication of the most successful current techniques. Chapters 4, 5 and 6 investigate how a measure of similarity between two exam timetabling problems can be developed so that this theory holds. The key features of a number of benchmark problems are examined and the behaviour of the problems when optimised is also considered to decide which problem features are important.

The other key aspect of the CBR system covered in this thesis is the techniques to be included within the case base. Chapter 7 introduces a variable neighbourhood search (VNS) technique which proves to be highly successful on a number of benchmark problems as well as providing a high degree of flexibility. This flexibility is further exploited in Chapter 8 where I propose a variant of VNS which can simplify the case base, changing the focus slightly. Instead of retrieving from a selection of different techniques to solve a new problem, the system will always retrieve VNS, but with a set of neighbourhoods which provided a successful solution to a similar problem within the case base. Chapter 9 concludes the thesis with a summary of the work and thoughts on how the VNS technique can be further developed to improve its generality.

Acknowledgements

I would like to thank a number of people for their continued support and assistance throughout the course of my PhD studies. Firstly my supervisors, Professor Edmund Burke and Dr Sanja Petrovic whose vision has contributed much to this work and whose support throughout the difficult times of my PhD has been invaluable. Also many thanks to Professor Peter Cowling (University of Bradford) and Dr Barry McCollum (Queen's University of Belfast) for their advice and input to this project. My work in this thesis has been just one part of a wider project which would not have been possible without the excellent work of Dr Rong Qu who carried out much of the remaining work on the project and whose advice and assistance throughout my work is greatly appreciated.

The ASAP research group at Nottingham has changed beyond recognition since I joined four years ago. It was a highly successful group when I joined, but has now developed, under the leadership of Professor Burke, into a thriving research community at the forefront of many innovative ideas in the field of optimisation. It has been a great pleasure to work with the ever growing number of members, both past and present, who have all helped to make the research group what it is.

I would also like to extend my thanks to the many researchers whose comments and ideas have helped to improve the quality of my work. The time spent by those referees who gave comments on my papers is much appreciated. In particular I would like to thank Pierre Hansen, who, without knowing it, inspired me to many of the ideas in the later chapters of this thesis with his seminar at the INTROS '03 workshop in Nottingham.

Finally, but by no means the least I would like to thank my family, the most important people in my life, for their constant support and advice during all my studies. My sister, Suzanne, who has always been on hand with helpful advice

when needed and to my parents, without whom none of this would have been possible.

Chapter 1

Introduction

1.1 Timetabling

The automated timetabling problem has been studied in various forms for the last 40 years with a large number of papers published and many applications developed to solve real life problems in that time.

Timetabling is a very large field covering many different types of problem, all with their own unique characteristics. Probably the most common type of timetable is a bus or train timetable. Without these, the whole transport network would be in chaos so it is easy to see the importance that timetables have in people's lives. In these cases, the timetable defines where and when buses, trains or other types of transport should be. Such a simple looking timetable can be very complicated to produce for the timetablers though. Train timetablers must take into account where engines start and finish each day, where and when drivers are available and how many hours they can work in one shift as well as the obvious task of trying to cater for the needs of the passengers who will be travelling on the trains. Similarly bus timetables, although usually less complicated must be planned so that there is as little dead mileage and lay-over[1] time for buses as

[1]Here *dead mileage* would be defined as the distance a bus must travel whilst not in service to return to a depot or outpost and *lay-over* is the amount of time a bus spends idle between routes

possible and that driver changes can be effected as efficiently as possible. Again driver hours must be taken into account with no driver being allowed to drive for more than a certain pre-defined time in any shift. If the driver timetable is inefficient then the company will have to employ a much larger pool of drivers than is really necessary meaning that their costs will be much higher. Therefore the problem of minimising the workforce whilst taking maximum allowed hours into consideration must be considered in the construction of a timetable.

The above example of transport timetabling shows the main elements of a generic timetabling problem which is defined in the next paragraph. Whilst this thesis is concerned with University timetabling, the same key ideas are involved in all other forms of timetabling. In simple terms, some *events* must be scheduled in a certain time and with a number of rules, known as constraints which must be obeyed either completely or as well as possible.

Timetabling is an example of the larger, more general problem of *scheduling* in which events must be scheduled either in time or place so that a variety of constraints are met and others are almost met. Wren [106] defines scheduling as ”the arrangement of objects into a pattern in time or space in such a way that some goals are achieved, or nearly achieved, and that constraints on the way objects may be arranged are satisfied or nearly satisfied”. He also makes the distinction that in *timetabling* problems there is not necessarily any allocation of resources to the events timetabled. In the minimum case, the timetable just states at what time a given event will happen. In reality however, it is almost always required to know that there are sufficient resources available for the given event to take place at a particular time and in some cases exactly which resources are allocated.

The overview of transport timetabling given above gives a good example of how the problem of timetabling has many different constraints to be considered. Some of these constraints, known as *hard constraints* must be satisfied whilst

others, known as *soft constraints* are required to be satisfied as well as possible. It is the violation of these soft constraints which determines how *good* a particular timetable is. The definition of *good* depends entirely on the individual problem. The evaluation of solution quality is usually done using an objective function which gives a weighting to each of the soft constraints depending on their importance. The higher the weight, the more important it is that the constraint is satisfied.

The main hard constraint in any timetabling problem is that no resource must be required to be in two different places simultaneously. In the case of bus timetabling this means that no one vehicle can be required to perform any part of two different routes simultaneously; for university timetabling, a student cannot have two exams at the same time and members of staff and students cannot be present at two different lectures at any one time. As well as these basic constraints, individual institutions will have their own particular hard and soft constraints which they require to be satisfied.

In the remainder of this chapter, the University timetabling problem will be defined with the more common constraints outlined and various models used to solve the problem shown. Having defined the problem, the most recent research conducted in this field will be summarised and discussed in Chapter 2.

1.1.1 The University Timetabling Problem

University timetabling can be divided into two main problem areas, these being *course (or lecture) timetabling* and *examination timetabling*. The types of problems these involve are fairly similar, but there are a number of points to note regarding the differences between timetabling lectures and timetabling exams.

In course timetabling, the purpose is to schedule lectures, seminars, laboratory sessions and any other student/teacher activities which may form part of a

given course into a weekly pattern. This can be an extremely complex problem and differs greatly in its requirements depending on the institution. Unlike exam timetabling, which will be covered in more detail in section 1.1.2, there is generally strict one to one mapping of events to rooms in course timetabling, although this is not always the case. This thesis will be focused on the exam timetabling problem, but a detailed overview of course timetabling and related work applying case based reasoning to the problem is given by Qu [97]

1.1.2 Examination Timetabling

Whereas in course timetabling the period in which events are to be scheduled is fixed (usually a week) and this schedule is repeated cyclically, in examination timetabling a set of exams must all be scheduled within a certain time period of a few weeks usually and the exact length of the timetable is in general not essential. The number of examinations to be scheduled varies greatly depending on the institution.

The examination timetabling problem is concerned with the assignment of a set $E = e_1, \ldots, e_n$, of exams into an ordered set $T = t_1, \ldots, t_m$ of timeslots subject to a number of hard and soft constraints. The hard constraints must be satisfied in order to produce a *feasible* timetable, whilst violation of the soft constraints should be minimised. For some problems, it is not possible to find a solution which satisfies all hard constraints so in such circumstances these must be relaxed to soft constraints with a high priority to minimise. The main hard constraints encountered in practical exam timetabling are:

- No student should be expected to be in two places at once
- There must be sufficient resources available for all exams scheduled in any given timeslot.

- Certain pairs of exams may be subject to precedence constraints or may be required to take place at the same time as one another

A variety of less common hard constraints may often be in effect at different institutions depending on the particular needs of the timetabling problem in question. Soft constraints vary far more than the hard constraints as these essentially define what is considered to be a good timetable to give a measure of comparison amongst feasible timetables. Using a single objective function to minimise, these soft constraints must all be given a weighting within the evaluation function to represent their relative importance to the final timetable produced. Using a multi-criteria approach, values are defined measuring the violation of constraints and all are considered simultaneously by an algorithm attempting to find a solution as close as possible to the ideal point in the preference space ([94]) at which all criteria are satisfied. Some examples of common soft constraints [35] are given below:

- *Time assignment* - it may be desired to schedule an exam in a particular timeslot. Also, many institutions favour exams to be scheduled as close to the beginning of the timetable as possible, especially the larger enrolment exams.

- *Time constraints between events* - an exam may need to be scheduled before/after another (depending on circumstances this could also be a hard constraint)

- *Spreading events out in time* - students should not have exams in consecutive periods and preferably not in the same day

- *Resource assignment* - it may be desired to schedule an exam in a particular room

Some institutions will also have constraints regarding multiple exams in a single room or exams split across more than one room as well as a variety of less common constraints. The number of periods in the timetable can either be fixed *a priori* or can often be included as an objective to minimise. This is of course, a directly conflicting constraint with that of wanting exams to be spread out across the timetable.

1.1.3 Complexity Issues

As with any scheduling or timetabling problem, there exist two distinct forms of question to be considered:

1. Is there an acceptable solution according to some particular criteria?

2. Which is the best solution?

Questions of type 1 are known as *decision* problems where the answer is always a simple 'yes' or 'no'. Question 2 gives the related *optimisation* problem in which the aim is not only to establish that there is a solution, but also to find the best possible solution. The definition of *best* depends on the problem and is measured by the objective function which must be minimised.

Timetabling problems of these types are known to be NP-complete and NP-hard respectively as shown by Cooper & Kingston [48]. In basic terms a problem is either NP-complete (decision problem) or NP-hard (optimisation problem) if it is very difficult to solve and would take far too long to find the best using simple exact methods such as evaluating every possible exam timetable. It is due to the NP-hard nature of this problem that local search meta-heuristics such as tabu search, simulated annealing and genetic algorithms have been applied to find good solutions. An in-depth analysis of the literature on these and other techniques applied to exam timetabling will be given in Chapter 2.

1.2 Local Search Meta-heuristics

Local search meta-heuristics are some of the most popular and successful methods for finding good solutions to NP-hard optimisation problems including that of examination timetabling. Many of the current best known techniques for solving exam timetabling benchmark problems fall into this category and are discussed in more detail in section 2.2.3. This thesis is concerned with the development of a case based reasoning system to intelligently select from a variety of local search techniques, the one best suited to a given problem. Whilst a technique will work very successfully on one problem, it may be far less successful on other problems, therefore when given a new problem it is not always obvious which technique will give best results. The CBR system developed in this project attempts to eliminate the need for a user to make the decision which technique to use. Instead the CBR system will match the new problem with one previously solved and will retrieve the technique which was most successful for that problem. Whilst the system is not required to be filled exclusively with local search techniques, these will form the bulk of techniques included. In this section I give a brief introduction to local search techniques and their application to exam timetabling problems, defining some of the many terms to be encountered later in the thesis.

Local search techniques can be considered as methods which iteratively search through the solution space of a problem, the *solution space* being defined as the set of all possible solutions to the problem. This is done by making small changes to the current solution at each iteration, with the aim being to minimise (or maximise) a given objective function. An initial solution must be constructed to seed the local search method which may be either feasible or infeasible. A *feasible* solution is one which does not violate any hard constraints whereas an *infeasible* one does. The local search itself may also be confined to the feasible solution

space or may be allowed to move into infeasible areas. One major disadvantage of allowing the search to be conducted only within the space of feasible solutions is that the search space may become *disconnected*, i.e. it will be impossible for the search to reach certain areas of the solution space. The obvious advantage however is that you are guaranteed a feasible solution (assuming the initial solution was feasible) and the search can concentrate on minimising the objective function rather than also eliminating hard constraint violations. Different local search techniques are defined by the criteria which they use for accepting or rejecting moves, a *move* being defined as the change from one solution to the next.

The *landscape* of an optimisation problem can be thought of as containing the solution space of the problem (in the case of exam timetabling this means every possible assignment of all exams into the given number of timeslots) on a horizontal plane, with the vertical component at each point being defined by the objective function evaluation for the given solution. Clearly this is an over-simplification of the situation since there are in some cases an infinite number of possible solutions (if we allow a variable number of timeslots) and there is no obvious way to order all solutions in a two-dimensional plane. However, when considering a local search meta-heuristic which searches the solution space, we need only consider those solutions in the neighbourhood of the current solution. The *neighbourhood* of an incumbent solution defines the set of new solutions to which the search can move with the remainder of the search space excluded. A neighbourhood is usually defined by some move operator which determines how the current solution is changed to arrive at a new solution within the neighbourhood. In the case of exam timetabling, the most simple move operator is that defined by moving a single exam from its current timeslot to a new one and thus the neighbourhood is defined as all solutions which can be arrived at from the current one as a result of such a move. Therefore, the landscape around the

current solution defined by a given neighbourhood can be more easily visualised in a three-dimensional space with the *distance* between any two solutions being defined by the difference in height given by the objective function.

This leads to two different aspects of local search meta-heuristics to be included within the CBR system. The first aspect is the type of algorithm, for example tabu search or simulated annealing. Different types of local search techniques are usually distinguished by their method of accepting moves which determines how they navigate around the search space and in particular how they can escape local minima. A strict descent algorithm only ever accepts moves with a negative cost (i.e. moves which lead to a better solution with a lower cost) and therefore terminates when it reaches a local minima where all solutions within the neighbourhood incur a cost increase. Many other techniques however include strategies to escape from these local minima to continue the search in other areas of the solution space.

The second aspect of local search techniques is the neighbourhood they employ with most using just a single neighbourhood throughout the search. Two different types of algorithm both using the same neighbourhood will of course be faced with the same landscape, however different areas of the problem landscape will be inaccessible depending on the move acceptance criteria. In the case of simulated annealing, the whole neighbourhood landscape is potentially accessible with a probabilistic move acceptance criteria making the steep uphill moves less favourable, whereas in tabu search a tabu list forbids certain solutions, rendering parts of the neighbourhood landscape as inaccessible, but any move to the accessible parts can potentially be accepted, albeit with the *best* move selected from the subset tested. On the other hand, two simulated annealing algorithms, each using a different neighbourhood will be confronted by completely different neighbourhood landscapes which may have a far larger effect on performance than the difference between two different local search methods both using the

same neighbourhood.

In the early chapters of this thesis, the focus is on the problem features which make up the problem definition and thus help to define the problem landscape for a given simple neighbourhood. In Chapters 7 and 8, I focus more on the effect of the neighbourhood definition by introducing a variable neighbourhood search (VNS) approach to exam timetabling.

1.3 Aims and motivation

The main aim of this thesis is to contribute to a case based approach to heuristic and meta-heuristic selection for timetabling. As part of a wider project, this thesis will concentrate on the low-level issues of a case based reasoning system. In particular, the complex issue of how to measure similarity between two exam timetabling problems is considered together with related issues concerning the structure of such problems and their behaviour when optimisation techniques are applied. As well as studying exam timetabling datasets themselves, a number of heuristic and meta-heuristic techniques are implemented to further understand the behaviour of the datasets and ultimately to build up the case base of techniques which can be applied to a wide range of exam timetabling problems. The case based reasoning (CBR) aspect will be covered within the thesis, but most elements of its implementation are beyond the scope of this thesis and form the rest of the project. The work presented in this thesis provides much of the basis for the low level system elements, working directly on the exam timetabling problems, whilst other work undertaken elsewhere on the project deals with the high level workings of the case-based reasoning (CBR) system and the knowledge discovery techniques used to train the case base for higher performance. Ultimately, the aim of the project as a whole is to develop a case-based reasoning system which, when presented with a new timetabling problem, will perform

a matching process within the case base to retrieve the most similar problem. The technique(s) which was most successfully applied to the retrieved problem will then be applied to the new problem.

The project to which this thesis contributes is motivated by the challenge of developing systems which are much more general than many of the current techniques applied to exam timetabling, which will often work very well on a specific problem, but not so well on a different problem. Increasing the generality of timetabling systems may well come at the cost of slightly lower solution quality, but the aim is to produce a system which can obtain results on a wide range of exam timetabling problems which are in the same *ball-park* as those produced by techniques designed specifically for individual problems. One of the major current research areas aiming towards this goal is that of hyper heuristics which work at a higher level of abstraction than more standard methods which operate directly on the problem. The aim of a hyper heuristic is to select from a list of heuristics the best for the current problem or problem state without using any domain knowledge, only the knowledge passed up to it from each of the low level heuristics regarding their evaluation when applied to the problem. Case based reasoning has been successfully applied to a wide variety of problems and has recently been applied directly to university timetabling problems, but is now being considered at hyper heuristic level also which is the focus for the work in this thesis. The work described here will be directed towards a case base for meta-heuristic selection, providing more complex algorithms than the more simple low-level heuristics which are often studied in hyper-heuristic systems.

1.4 Layout of the thesis

This chapter has presented an introduction to the general timetabling problem and to the more specific exam timetabling problem, giving two of the more pop-

ular models for the problem as well as a brief discussion on its complexity.

In Chapter 2, a detailed review of the state of the art techniques applied to exam timetabling is given with the emphasis on those techniques and paper published since the survey of Carter & Laporte [43] in 1996.

Chapter 3 gives an overview of the case based reasoning (CBR) methodology and its application to scheduling problems.

Chapter 4 describes an initial investigation into some of the major features of exam timetabling problems which may be important to a similarity measure together with a discussion of how any redundancy in the problem definition must be removed for the purpose of measuring similarity.

In Chapter 5, I investigate the effect of both the initial solution fed to a local search meta-heuristic and the objective function used to optimise a problem. In particular, the behaviour of a number of benchmark datasets with respect to two objective functions and two methods of initialisation are considered to examine how these factors affect the similarity between two problems.

Chapter 6 brings together earlier work from Chapters 4 & 5 to give a detailed analysis of the main features of exam timetabling problems which will be fed into a knowledge discovery process to determine those most important for measuring similarity. Simple features together with more complex features and ratios of pairs of features are all considered.

In Chapter 7, I introduce a variable neighbourhood search (VNS) approach to exam timetabling which proves to be a simple yet powerful method capable of matching the results of most state of the art techniques on benchmark problems. A number of variations to the basic VNS are considered, some of which improve performance still further whilst others may require more detailed research if they are to provide improvement.

Chapter 8 extends the work in Chapter 7 by using a genetic algorithm (GA) to evolve the *best* subset of neighbourhoods to use with VNS for a given problem.

The ability to easily add more and more neighbourhoods to VNS means that a method for choosing the subset which gives best performance on a problem is important. A discussion of how this approach can be successfully combined with CBR is also given.

Chapter 9 gives overall conclusions of the work presented in this thesis together with ideas for future work resulting from the ideas and results presented in this thesis.

Chapter 2

Exam Timetabling survey

2.1 Introduction

Over the last decade, the examination timetabling problem has been studied by researchers across the world, exploring a wide variety of techniques, many of which are very recent ideas or techniques which have been successfully applied in other areas of optimisation and are now being applied to timetabling. Exam timetabling is a difficult problem which all teaching institutions must contend with, often many times per year - many of these institutions still do their timetabling mostly by hand, others use a computer to aid the process and an increasing number now use some form of automated or semi-automated system.

In 1995, Burke et al [19] conducted a survey of exam timetabling in British universities with three main questions regarding the problem structure at different universities, how the problem is currently solved and what qualities make up a good timetable. Among the fifty six universities who replied to the survey, responses were very mixed indicating a wide range of different problems needing to be solved - many institutions had problems accommodating all their exams in the rooms available, some have large numbers of exams per students where others have smaller numbers and most had their own differing constraints on the

timetable and what is considered as *good*. One major finding from the survey was that just 21 per cent (eleven) of the institutions who responded to the survey use some form of automated exam timetabling system, some of which still require a certain amount of manual input and problem specific knowledge. 58 per cent of the universities use a computer at some stage of the timetabling process, leaving a surprisingly large 42 per cent who do not use a computer to aid the timetabling process at all. It was noted that those institutions who do not use either a computer or the previous year's timetable to produce the new timetable can take up to four weeks to produce a timetable. Of those institutions who did not use any form of automation, four stated that they were considering either a commercial package or developing their own customised system. With almost ten years passed since this survey it is likely that many more institutions now use either a fully or partially automated system for their exam timetabling, but the results presented in the survey do indicate many problem areas with the automation of exam timetabling together with the scope for much further research in the area.

Carter and Laporte [43] produced a survey of the 'Recent Developments in Practical Examination Timetabling' in 1996 which gives a detailed summary of the research into exam timetabling during the decade succeeding Carter's 1986 survey [41]. Carter and Laporte divide the papers they report on into four types - Cluster, Sequential, Generalised Search and Constraint Based methods, giving conclusions on the success of methods implemented in each of these areas together with thoughts on future research directions in these areas. In particular, regarding Generalised Search, encompassing Local Search techniques such as Tabu Search, Simulated Annealing and Genetic Algorithms, the authors noted that these techniques are 'relatively recent innovations in the area of heuristic search' and that they provide a very promising area for future research. It was also pointed out that 'It is difficult to draw more accurate conclusions without

comparisons with other approaches on identical problems to see if the additional effort produces better quality solutions.' In this survey I aim to update the findings of Carter and Laporte's 1996 survey with a strong emphasis on the above areas. Over the last decade, much of the focus of exam timetabling algorithms has been in the area of "Generalised Search" and with the availability of a number of benchmark test problems available [1][2] it is now far easier to compare the different techniques on a range of different problems to draw conclusions as to which are better on which type of problems and/or whether the extra time spent by some algorithms is worthwhile. Of course, some techniques will work very well on certain problems whilst not on others so there may be a large amount of problem dependency for many algorithms and there is no guarantee that they will efficiently solve a new and different problem just because they produce high quality results on the benchmark data sets.

In 1999, Schaerf [99] published a survey of automated timetabling with the wider brief of covering all types of educational timetabling. Schaerf gives a mathematical formulation of the basic search problem together with the related optimisation problem as well as discussing some of the variants of the problem. The paper also gives a description of some of the techniques applied to exam timetabling by different authors up to that time, concluding with comments on possible future research directions. Burke and Petrovic [35] give a detailed account of the recent research directions in automated timetabling which have been pursued by the ASAP research group at the University of Nottingham up to 2002. These include heuristic and evolutionary techniques, multi-criteria approaches and the application of case-based reasoning to utilise previous experience to solve new timetabling problems. The authors also consider the issue of diversity and quality in the initialisation process for local search techniques

[1] ftp://ftp.cs.nott.ac.uk/ttp/Data/

[2] http://www.or.ms.unimelb.edu.au/timetabling/ttframe.html?ttpublic.html

and conclude the paper by discussing promising directions for future research and work within the group. In 2004, Petrovic and Burke [91] provide a further survey of work undertaken by the authors and members of their research team, focusing on the multi-criteria and case-based reasoning approaches studied since their 2002 survey as well as other relatively recent research directions - meta-heuristic approaches which are less dependent on parameter settings or which use parameters useful to real world users out-with the research community and hyper-heuristics and self adaptive approaches which aim to raise the level of generality of existing methods. Also presented is an extended classification of exam timetabling techniques from that presented by Carter and Laporte [43] to reflect the new techniques developed since that survey was published. Petrovic and Burke [91] conclude that the development of more general approaches to timetabling problems is a very challenging goal, but one which could have significant impact, allowing systems to be successfully applied to a wider range of problems. Other recent surveys on automated timetabling include Bardadym [5] and Burke et al [24].

In this chapter, I aim to update Carter and Laporte's survey of 1996 [43], reporting on the various techniques which have been applied to exam timetabling in the decade since that survey whilst attempting to answer the questions which that survey raises, in particular with regard to the comparison of different techniques on the publicly available benchmark datasets. Whilst my aim is to provide as comprehensive a survey as possible on the papers published in recent years, I realise that some may have been overlooked since research into examination timetabling continues to move at pace with new techniques being developed and applied constantly. Having defined the exam timetabling problem in section 1.1.2, in section 2.2 I summarise the papers which have been published and the techniques applied to exam timetabling, giving conclusions in section 2.4

2.2 Algorithmic techniques

2.2.1 Constraint based methods

> "All of the papers that we considered for constraint based methods have been used to solve fairly small problems (around 300 examinations)...In constraint methods, the number of constraints grows quite quickly as problem size increases (at least as the square), so it would appear that they may have trouble with large examples in the range of 1,000 exams. Obviously, this limitation will be reduced over time."
>
> *Carter & Laporte, 1996 [43]*

In the years since Carter & Laporte's 1996 survey there have been a number of papers published focusing on constraint-based methods, including Constraint Logic Programming (CLP) and a number of other Constraint satisfaction techniques. Brailsford et al. [8] give an introduction to constraint satisfaction problems (CSP) with a definition stating that a (finite domain) CSP consists of:

- a set of *variables* $X = x_1, \ldots, x_n$;
- for each variable x_i, a finite set D_i of possible values (its *domain*);
- a set of *constraints* restricting the values that the variables can simultaneously take.

A CSP is solved to give a *feasible* solution by allocating to each variable a single value from its domain so that every constraint is satisfied. If no such assignment exists, the problem is *unsatisfiable* with the given constraints. Brailsford et al. [8] also comment that while the usual focus of methods to solve CSPs

is on finding a *feasible* solution, such techniques can be adapted to find an *optimal solution* by introducing an objective variable to represent the objective function. The constraint on the objective variable is tightened repeatedly until the problem becomes unsatisfiable, at which point the last solution found is an optimal solution. The constraints in a CSP are generally expressions based on the variables, thus reducing the feasible domains of these constrained variables. Brailsford et. al. go on to describe various search procedures applied by constraint programming algorithms as well as methods for formulating problems, while in Section 7.6 they consider applications to timetabling problems including that of Nuijten et al. [87] discussed by Carter & Laporte [43]. Also considered by Carter & Laporte was the Constraint Logic Programming (CLP) technique applied by Boizumault et al. [7] to solve their examination timetabling problem at l'Université Catholique de l'Ouest in France where the authors identify twelve types of constraints, eleven of which are considered to be hard constraints, with an added *soft* constraint regarding preferences for which room exams are scheduled into. They used the CHIP constraint logic programming language to formulate their problem with the twelve types of constraints reduced to five categories for implementation. Whilst their approach could produce solutions in under a minute for their problem at hand, the authors did note that when increasing the size of the problem only the "best-fit decreasing" labelling strategy with room constraints considered beforehand could produce results with their other techniques tested being unable to solve all instances. The authors conclude that the use of an intelligent labelling strategy is very important when using CLP.

Philippe David [54] has successfully applied constraint satisfaction techniques to generate exam timetables for the École des Mines de Nantes where one of the major constraints is that the computing time must be less than one minute since timetabling is required to be done "on-line". David's technique uses an incomplete assignment algorithm with local repair techniques which the author

notes can miss solutions, but to minimise this risk, the program is run a number of times and some constraints can be relaxed. Each candidate at École des Mines de Nantes must take a complete examination consisting of four parts: mathematics, physics & chemistry, foreign language and discussion about a text. Each of these subjects has its own set of examiners (typically seven examiners per subject) with foreign language having examiners for each different language and the exams take place during a day consisting of 15 consecutive periods. The aim of the problem is to schedule the exams for each day, for every subject assigning every candidate to an examiner for one period subject to a variety of constraints - at most 105 candidates (seven examiners $\times$ 15 periods) can be scheduled in a single day. David [54] presents a formulation of the problem as a CSP model followed by a description of his assignment algorithm. The algorithm used comprises two steps, a *pre-assignment* step which creates the domains for the examiners, thus taking into account some of the constraints during domain creation, and a final assignment step which assigns the candidates to a period for the examiner they were pre-assigned to. If the final assignment finishes with some domains empty, but not all candidates assigned, a set of local repair procedures are used to correct the problem - these involve swapping an already assigned student with one not assigned having established that both students can then be assigned successfully. There are five different local repair techniques which are called successively if the previous one failed to resolve the problem. The complexity of the total algorithm is calculated to be $O(n_c^2 p)$ where n_c is the number of candidates and p the number of periods. The technique was run successfully on 50 test problems with run times of between 0.2 and 7.5 seconds before being applied to the real-life problems for which a timetable was successfully generated for each of the 13 days with a runtime of under 0.3 seconds, all runs carried out on a SUN SPARC 5 with 32 MB memory. The author concludes that his method of an incomplete algorithm with local repair techniques rather than the more standard exhaustive

search could give interesting results applied to other timetabling problems as well as being sufficient for the problem at École des Mines de Nantes.

In [104], White classifies the timetabling problem into three classes: school timetabling, course timetabling and exam timetabling and focuses on the course timetabling problem, but notes that "the principles of course timetabling can be used in exactly the same way for other forms of timetabling" and gives a set of nineteen common constraints (12 hard, seven soft) for course timetabling, many of which are also applicable to or can be adapted for exam timetabling problems. White presents a model of the timetabling problem as a CSP with variables to represent courses (or exams) with room-time pairs forming the domains for these variables and the a constraint C formed as a conjunction of the sub-constraints. White considers in his model whether the timeslots should be of fixed or variable length, commenting that the former resembles a bipartite matching while the latter requires the day to be broken into units of say 0.5 hours which, combined with a room, form the resource units in a formulation akin to job shop scheduling. When formulating a problem as a CSP, it is important to know the specific question being asked - White gives four potential problem questions in increasing order of difficulty, adding that the second is most common in practical timetabling:

- Is there a solution?
- If there is a solution, what is it?
- How many solutions are there?
- What are all these solutions

The issue of which algorithm to use to solve CSPs is also covered by White [104] with seven different solvers described and considered for suitability. One of the

conclusions of the paper is that constraint satisfaction can provide satisfactory solutions to a wide spectrum of timetabling problems and is highly flexible.

A relatively recent idea, investigated by Terashima et al. [100] in 1999, is to combine constraint satisfaction techniques with evolutionary algorithms in a *non-direct* manner rather than applying directly to the problem. That is, a Genetic Algorithm (GA) using a *non-direct* chromosome representation can be used to evolve the configuration of Constraint Satisfaction methods which have a large variety of options. In their implementation, they use an array of encoded instructions and parameters for guiding a search algorithm as a chromosome representation rather than the more common direct representation consisting of an array of integers representing the position of an exam within a timetable, pointing out that the direct representation has been found to be quite restrictive. The idea behind the non-direct chromosome representation is that it represents a method for how to construct a timetable rather than an actual timetable - the chromosome consisting of ten elements, each representing a different part of the choice of constraint satisfaction technique. Three strategies are considered, Brelaz, Backtracking and Forward Checking, each with a variety of different methods for variable (exams) and value (timeslot) ordering. Two of the three strategies are potentially used during the solving process, both represented within the chromosome together with a condition which establishes when the timetable construction process switches from the first strategy to the second. Eight different variable orderings and nine different value orderings are included, each represented by an integer in the chromosome representation, together with four different conditions. A partial timetable is constructed using the method encoded for strategy one, then the timetable is completed using the information encoded for strategy two which consists of one of the three strategies, a variable ordering and a value ordering. Experiments were carried out on a number of benchmark problems with the three main constraints included - clashes, near-clashes and ca-

pacity constraints with the near-clashes constraint being represented by a penalty based on how close two clashing exams were timetabled. In all test problems, the GA technique found solutions with zero constraint violations regarding hard clashes and capacity whilst beating other reported strategies on minimising the near-clashes constraint. The authors propose that the use of non-direct representations for solving exam timetabling problems seems to be the 'right direction' when using GAs, but add that issues relating to the time taken to produce solutions with this method still require further research. A discussion on the use of direct chromosome representations is given in section 2.2.3.

Another hybrid approach involving constraint programming is that of Anh & Hoa [3] who use a constraint programming phase to create an initial solution followed by a simulated annealing (SA) phase to optimise this. The aim of the authors is to take advantage of the strong points of both techniques, with CP able to find initial solutions satisfying all hard constraints whilst the SA phase can work on optimising based on the soft constraints. It is noted that the initial solution fed to an SA algorithm can be critical to the success of the technique, especially if, as is common in exam timetabling, the search space is disconnected. Constraint Programming is seen as a good method to achieve a good initial solution in as little time as possible in order to allow more time for the SA phase. The method Anh & Hoa use is *backtracking with forward checking* (BC-FC), a technique which looks ahead when assigning a value to a variable and removes from the domains of "future" variables, any values which conflict with the assignment. Performance of the algorithm is improved by using a dynamic variable ordering known as *fail first*, which selects the next variable as the one with fewest values remaining in its domain. The second phase of the algorithm uses Simulated Annealing with a Kempe Chain neighbourhood, a technique which is discussed further in section 2.2.3. Experimental results on a real data set from HCMC University of Technology (324 exams, 30 timeslots, 64,000 enrolments)

show that the CP phase produces good results in less than two minutes runtime on a 450 MHz Pentium II PC, while the SA phase is run with a variety of stopping criteria based on the number of steps, with a runtime of around 50 minutes for 50,000 steps. Compared to a pure constraint programming approach or graph colouring method on the same data, they conclude that the hybrid technique takes much more run time, but yields a noticeably better solution for highly constrained problems. Merlot et al [82] also use Constraint Programming to create an initial feasible solution in a three-phase approach covered in more detail in section 2.2.3.

A Constraint-based approach has been applied to the high school timetabling (STT) problem by Meisels et. al [81], whilst Cheng et al [47] and Guéret et al. [62] apply constraint logic programming and Abbas & Tsang [1] constraint-satisfaction techniques to the course timetabling problem. Whilst applications on exam timetabling are still being used to solve relatively small problems of around 350 exams, Cheng et al [47] do test their course timetabling approach on data with over 2000 courses, indicating that the approach may be feasible for larger problems in exam timetabling also.

2.2.2 Graph-based Initialisation and Construction techniques

In the survey of Carter & Laporte [43] in 1996, Cluster Methods and Sequential Methods were two of the four major categories of techniques for solving exam timetabling problems. In the intervening years, these methods have become less prevalent as techniques in their own right, but have become very important in hybrid algorithms, often providing good quality initial solutions to seed a variety of local search techniques. Here I consider some of the papers recently published looking at such initialisation strategies, also encompassing the cluster and sequential methods.

Sequential methods use a sequencing strategy to select the next exam to add to an initially empty timetable and construct the full timetable by assigning the exams one at a time to a selected period. Among such construction techniques, two major aspects are the heuristic used to order the exams and the method for selecting the period for the chosen exam. Carter et al. [45] compare the following five heuristic methods for sequencing the exams:

- *Largest Degree (LD)*: Largest number of conflicting examinations - highly conflicting exams are difficult to schedule later in the construction process
- *Saturation Degree (SD)*: Number of periods in conflict - exams with few remaining feasible slots should be scheduled sooner to avoid having no remaining feasible slots for the exam
- *Largest Weighted Degree (LWD)*: Number of students in conflict - similar to Largest Degree, but where each conflict (edge of the graph) is weighted by the number of students involved
- *Largest Enrolment (LE)*: Largest number of students enrolled for exam - exams with high enrolment are often difficult to schedule as they create many conflicts
- *Random Ordering (RO)*: Select the next exam completely at random - largely used for comparison purposes with the above techniques.

They also investigate five more strategies defined by first finding the largest clique in the conflict graph and assigning those exams before continuing with one of the above procedures. The authors note that whilst this clique initialisation technique is not of any practical use when applied to random data sets, it does provide better results when applied to real life problems. This finding is easily explained by the fact that in a random problem, there is an equal probability of any two exams clashing with each other so there is no defined clique

structure within the graph. In real exam timetabling problems however this is not the case and major cliques do exist making the clique initialisation strategy very effective, especially at minimising the number of periods required to schedule all exams - clearly the number of periods required is at least the size of the largest clique in the problem. Carter et al. [45] also implement a backtracking strategy to deal with exams which, when selected to be scheduled, clash with at least one exam in every period of the timetable, this essentially involves undoing some assignments already made in order to schedule the new exam. It was found that the use of such a backtracking method vastly reduces the number of periods needed for the timetable when compared to a simple sequencing method without backtracking, thus the authors incorporate backtracking as well as clique initialisation in all their future tests. A major contribution made to the exam timetabling community by Carter et al. is the publication of 13 benchmark problems (available from ftp://ftp.cs.nott.ac.uk/ttp/Data/) on which to test different strategies. In [45], the authors first compare the five heuristic strategies with the aim of minimising the number of periods required to schedule all exams (graph colouring), then by adding proximity costs (weightings on the edges of the conflict graph) and fixing the number of periods they tested the five strategies based on minimising an objective function aimed at spreading clashing exams around the timetable (examination timetabling). As reported in [43], Carter, Laporte & Chinneck [44] implemented a system based on the above five heuristic orderings, also incorporating a "k-opt" improvement phase.

Burke, Newall & Weare [33] investigate issues of quality and diversity when seeding an evolutionary algorithm using heuristic sequencing strategies. Since a purely heuristic method would produce an initial population of identical solutions the authors consider two methods for adding a random element to the procedure - these being *tournament selection* and *bias selection*. Tournament selection for a given heuristic sequencing involves picking the next exam to sched-

ule as the best from a randomly generated subset whereas bias selection picks at random from the best n choices. When using a population based methods such as evolutionary algorithms, it is important to have a certain element of diversity in the initial population so as to be able to search a wider proportion of the solution space, whilst also retaining a certain level of quality since studies have shown that randomly initialised Genetic Algorithms (GAs) perform badly when compared to other methods. Burke, Newall & Weare [33] use three methods for measuring diversity of solutions produced when adding a random element to a graph colouring heuristic while testing six different heuristics and three different tournament sizes. The authors conclude that the use of dynamic orderings such as Colour Degree [3] gives an initial population which is both relatively diverse and of a high quality. It was also found that such initialisation of a population is superior to random initialisation both in terms of time and quality, allowing the evolutionary algorithm to focus on "fine tuning" solutions and optimising more general side constraints.

In 2001, Carter & Johnson [42] investigated extensions to clique initialisation techniques which are often used in conjunction with heuristic methods, as in [45]. The usual method for incorporating a clique initialisation phase is simply to identify a maximum clique of the problem and assign these exams to the timetable before using a heuristic sequencing technique to schedule the remaining exams. Carter & Johnson point out that in many real world exam timetabling problems there are many alternative maximum size cliques as well as a number of near-maximum sized cliques and "quasi-cliques". They provide an analysis of cliques on the eleven of the benchmark problems discussed earlier, reporting the size of maximum clique and the number of cliques of maximum size in each problem as well as the total number of exams included in all such cliques. Where

[3] Colour Degree: Events are ordered by the number of conflicting events already scheduled in descending order

the number of maximum-sized cliques is small and incorporates few exams more than a single max-size clique, cliques of size $max - 1$ are also considered to be useful for the initialisation phase. When also taking into account quasi-cliques, Q_k, of degree k which have at most k edges missing from a full clique, the number of examinations which can relatively easily be scheduled during a clique initialisation phase can be increased by up to 50%. Three different strategies are considered:

- Assign all examinations which fall in *any* maximum clique, rather than just assigning one maximum clique
- Assign all examinations which fall in either a maximum clique, or a clique one smaller than the maximum
- Assign all examinations which fall in the largest quasi-clique of degree one (or possibly two)

Carter & Johnson conclude that the employment of these methods can improve the clique initialisation phase of a construction heuristic by assigning more of the densely clashing exams before handing over to a sequential heuristic.

Burke and Newall [31] look at methods for adapting the standard heuristic ordering techniques with the aim of developing approaches with a higher level of generality. It is widely accepted that a specific technique which works well on one problem may not work as well as other techniques on a different problem, so the ability for a method to adapt itself to a given problem is an extremely useful one, making the choice of heuristic far less important than it currently is. The method proposed leads to an iterative procedure with a dynamic ordering arrived at by experience obtained from earlier iterations as to how *difficult* a particular exam is to schedule rather than simply using fixed heuristic-defined measures. They introduce a "Heuristic modifier" onto an ordering whose purpose

is to promote more difficult to schedule exams higher up the order than they would otherwise be with a given heuristic. The *perceived difficulty* of an exam e at iteration i is given by:

$$difficulty(e, i) = heuristic(e) + heurmod_{ei} \tag{2.1}$$

where $heuristic(e)$ is the standard heuristic measure of difficulty (e.g. largest degree) and $heurmod_{ei}$ is the heuristic modifier for exam e at iteration i. A number of different methods are considered for when and how to modify $heurmod_{ei}$ taking into account only hard constraints or also considering soft constraints. The authors run a range of tests on the real world timetabling problems used by Carter et al. [45] using two different measures of solution quality - the first involving proximity costs for clashing exams, but no limit on available seats/rooms, the second using a limit for the number of available seats in each period together with a penalty function penalising clashing exams in successive periods with a large penalty of 5000 for unscheduled exams. A total of 2000 iterations were used to run most test problems with run times reported and comparisons made with other reported techniques. The main conclusions of the paper are that experiments have shown the adaptive method to perform very efficiently and competitively on a wide range of problems and is quick and relatively easy to implement. Also that the technique demonstrates robustness shown by a narrow gap between best and worst results in a wide range of experiments. The adaptive method has shown to be capable, after a number of iterations, of improving a bad initial ordering of the exams thus lowering the dependency on the chosen heuristic in order to produce good results and increasing the level of generality of the approach.

Also concerned with increasing the level of generality of exam timetabling techniques to make them more widely applicable, Petrovic et al [95] propose a

case-based reasoning technique for initialisation of meta-heuristics. Their technique uses the Great Deluge (GD) meta-heuristic which will be discussed in more detail in Section 2.2.3 to perform the local search improvement when seeded with a good quality initial solution. They consider the Largest Degree, Largest Enrolment, Largest Colour Degree, Largest Weighted Degree and Largest Saturation Degree sequential heuristics for the initialisation with a number of methods for enriching these to form a hybrid technique:

- Maximum Clique Detection (MCD) - identify a maximum clique and schedule these exams first (as used by Carter et al [45]
- Adding Random Elements (ARE) - tournament based selection, as used by Burke et al. [33]
- Backtracking (BT) - rescheduling already scheduled exams when conflict arises

A Case based reasoning method is used to select amongst these different hybrid techniques which is most suitable for a given problem. This is done by maintaining a case-base of previously solved problems where each case is represented by a 2-tuple (G, H), G being a graph representation of the problem and H the sequential heuristic used to produce a good initial solution for GD. Graph isomorphism techniques are used to match a new problem with one from the case-base which can then be retrieved and the heuristic applied to the new problem to initialise Great Deluge. Case retrieval is effected by a two-stage tabu search approach as described in [93]. Experiments are carried out on a number of benchmark problems and variations thereof to seed the case-base. The problem variants were created by either adding or removing a percentage of exams and students with respect to the benchmark problems used. This resulted in 77 total cases for the large case base (seven benchmark problems and 70 variations) and

42 cases (seven benchmark problems and 35 variations) for the small case base. An exhaustive set of tests was carried out with all possible sequential heuristics on each problem with the one which led to the best final solution (after Great Deluge was applied to the initial solution with 20 million iterations) was stored in the case base. One of the aims of the paper is to determine whether the size of the case base has a significant impact on the performance of the case base reasoning system with results indicating that a larger case base can lead to better results at the cost of extra time. The authors demonstrate that the performance of their technique on benchmark problems is comparable to those of other approaches with which they compare. In conclusion they stated that their results demonstrate that knowledge gained in initialising one problem can be used for solving a *similar* timetabling problem, providing a good foundation for the development of a general CBR system for solving timetabling problems.

2.2.3 Local Search Meta-heuristics

> "All of these techniques are relatively recent innovations in the area of heuristic search...This looks a very promising area for future research. Of course, all of these search algorithms require considerable computer time and/or horse power. It is difficult to draw more accurate conclusions without comparisons with other approaches on identical problems to see if the additional effort produces better quality solutions"
>
> *Carter & Laporte, 1996 [43]*

Local Search techniques perform their search in the solution space of a problem, having been seeded with an initial solution which may be either feasible or infeasible. Typically, a local search method with iteratively make small changes

to an incumbent solution whilst retaining the majority of the features of the old solution. These techniques use an objective function to measure the quality of each solution and use a variety of different methods for deciding whether to accept a *move* or not. A large variety of different local search techniques have been applied to exam timetabling problems over the last decade with the criteria for accepting or rejecting moves, defining how the algorithm escapes from local minima, being the distinguishing factor between these techniques. The most common forms of local search include hill climbing (HC), simulated annealing (SA), tabu search (TS) and genetic algorithms (GAs), but a wide range of other techniques are also being studied. In section 2.2.4 I will consider genetic algorithms and other population based evolutionary methods in more detail, whilst in this section focusing on those techniques which work on a single incumbent solution.

Simulated Annealing

Simulated annealing [71] is a highly popular meta-heuristic which models itself on the physical process of annealing in which a substance is slowly cooled from an initial high temperature at which its molecules can move around freely (liquid state) to a low temperature at which it becomes solid and the molecules have stabilised. In simulated annealing, the temperature determines how likely it is that a *worsening* move will be accepted, with a high temperature giving a high probability of acceptance and a low temperature causing most worsening moves to be rejected. The initial temperature, *cooling schedule* and finishing temperature are the three main parameters which control the performance of an SA algorithm as it optimises an objective function. Numerous authors have successfully applied this approach to exam timetabling with some of the earlier examples reported by Carter & Laporte [43]. Here I concentrate mainly on those papers published post-1996.

Thompson & Dowsland [101, 102] considered the effects of different cooling schedules on the performance of a simulated annealing algorithm, using a two-phased approach, the first phase finding a feasible timetable, the second using a more complex neighbourhood structure to optimise based on the secondary objectives. Their ultimate aim was to identify an adaptive cooling schedule which could be efficiently used within an existing timetabling system. This technique has been successfully used by the authors in a timetabling package known as TISSUE at Swansea University. In 1998, Thompson & Dowsland [103] extend their work to investigate the robustness of the approach, focusing on the affect of different cooling schedules and a wider range neighbourhood structures than was used in [101]. Using eight data sets, exhibiting a range of different features, they focus on the second phase of their two-phased approach with the first phase finding feasible solutions relatively easily. A number of different second order conflicts were considered concerning the proximity of the exams taken by a student either within a single day or x exams in y consecutive timeslots. Three different types of neighbourhoods are investigated:

- Standard neighbourhood - A single exam is moved to a new feasible timeslot (as used in [101])
- Kempe chain neighbourhood - Two subsets of exams in periods i and j are exchanged in such a way that the new timetable is feasible with respect to the hard constraints. Selecting an exam in period i to move to period j will induce a Kempe Chain as described in [101]
- S-chains - An extension of Kempe chains in which S periods are chosen instead of just two as used in the Kempe chain neighbourhood - this is described further in [103]. For their experiments the authors restrict themselves to those chains produced with $S = 3$

Experiments using a very slow cooling rate for all neighbourhoods showed that the Kempe chain and S-Chain neighbourhoods were clearly superior to the Standard neighbourhood, but showed little difference between themselves. With this in mind, the authors restricted further tests to comparing only the Kempe Chain and Standard neighbourhoods. Further experiments were performed to test different geometric cooling rates ($t \rightarrow \alpha t, \alpha < 1$) with α ranging from 0.6 to 0.99 using a variety of different adaptive cooling schedules. Results indicated that a value of $\alpha = 0.99$ with the temperature being reduced after 5000 iterations at each level was the best strategy to adopt, with Kempe chain neighbourhoods again outperforming the Standard neighbourhood leading the authors to conclude that this neighbourhood combined with a slow cooling schedule would provide a "robust and flexible approach to the examination scheduling problem." Further analysis is included, looking into why Kempe chain neighbourhoods are so much better and considering the effect of different sampling techniques, concluding that the method of sampling is also an important factor in solution quality. Since their work in [101], the nature of the problem at Swansea University had changed considerably, becoming larger and more tightly constrained due to modularisation, yet the system described was able to deal effectively with the changes.

Bullnheimer [10] uses a quadratic assignment problem (QAP) formulation which is then transformed into a quadratic semi assignment problem as a method for solving small scale exam timetabling problems using simulated annealing. The main focus of Bullnheimer's work is to create a timetable without clashes which also maximises student study time, introducing a parameter, α which can be set by the timetabling institution to give more or less emphasis to the spreading of exams. Setting $\alpha = 0$ takes into account only "back-to-back" conflicts, whereas higher values of α put more focus on the spread of exams to maximise the student's study time. The exam timetabling problem at the Faculty of Eco-

nomics and Management, Otto-von-Guericke-University in Magdeburg consists of two distinct sets of courses, the largest having 27 exams with 419 students, the other having 15 exams with 391 students, with three slots on each of 15 days to schedule the 42 exams. A simulated annealing technique is applied using two classes of neighbourhood structure: a *slot move* re-arranges a number of slots without altering the exam assignments in each slot whilst an *exam move* randomly picks an exam to move to a randomly chosen slot. The algorithm was run for a total of 5000 iterations and a number of schedules produced using different values of α are displayed to demonstrate the ability of the technique and formulation to produce good solutions. The author concludes that the proposed QAP-based model combined with simulated annealing is suitable for small scale problems and can be used for larger problems, broken down into subproblems.

Tabu Search

Tabu Search is a technique similar to simulated annealing, but which uses a different criteria for move acceptance and neighbourhood search. Whereas in simulated annealing, moves are generally selected at random from the neighbourhood, tabu search performs an exhaustive search of the neighbourhood or a subset of the neighbourhood, selecting the best move (according the the objective function) from those considered. If this move improves on the best solution found so far during the search it is generally accepted unconditionally. However, to prevent the search from getting stuck in local optima a tabu list is maintained of previously visited solutions which are not to be revisited within a certain number of moves. The *tabu tenure* determines how long a given solution or move remains tabu. In general it is not feasible to store complete solutions on the tabu list, it is more usual that an element which was moved recently becomes tabu for the duration of the tabu tenure.

Di Gaspero & Schaerf [58] propose a family of tabu search methods which

they apply to a set of variants of the exam timetabling problem. In addition to the standard hard constraints that all exams must be scheduled into exactly one timeslot and no two exams with students in common should be assigned to the same time slot, the hard constraints they consider are capacity constraints and pre-assignments & unavailabities. Capacity constraints take into account the limitations of room sizes and number of rooms for each period whilst pre-assignments simply represent exams which are fixed into a certain slot and unavailabilities represent slots which are not possible for a given exam. Three types of soft constraints are also taken into account and are weighted to contribute to the objective function to be minimised:

- *Second-order conflicts*: A penalty is added to the objective evaluation for each occurrence of a student being assigned exams in consecutive periods
- *Higher-order conflicts*: A proximity cost, $pc(i)$, is included whenever a student is assigned to take two exams within i timeslots. This proximity cost is multiplied by the number of students involved in the two examinations and is defined as in [45]: $pc(1) = 16$, $pc(2) = 8$, $pc(3) = 4$, $pc(4) = 2$, $pc(5) = 1$.
- *Preferences*: A soft version of pre-assignments and unavailabilities, taking into account student and teacher preferences as to the scheduling of exams.

Di Gaspero & Schaerf define their search space to include all complete assignments of exams to slots including infeasible ones, with the exception that unavailabilities and pre-assignments are imposed from the start. The search is guided by a hierarchical objective function which penalises the violation of hard constraints far higher than that of the soft constraints in a linear combination. A mechanism for allowing these weights to change during the algorithm's running time is also included. The neighbourhood used within the search is the

standard neighbourhood as defined earlier, with a single exam being moved to a new timeslot whilst the inverse of a move, which is added to the tabu list, is defined to be any move involving that same exam. Only exams which violate either soft or hard constraints are considered to be moved during the search, with two different strategies employed for move selection - one being exhaustive, the other biased towards those exams adding highest penalty with all periods considered for the new assignment. Two different search techniques are considered, one focusing more on the hard constraint violations, the other searching for any form of improvement. Parameters were arrived at after extensive tests and consisted of a tabu tenure varying randomly between 15 and 25 and a stopping criterion based on the number of iterations since the last improvement. Results are presented for various benchmark problems and compared with the results of other authors. For a comparison of these results with those of other current techniques on the uncapacitated benchmarks of Carter at al. [45], please refer to table 2.3. Di Gaspero [57] discusses a number of improvements to this family of tabu search algorithms by employing a multi-neighbourhood local search using two neighbourhoods, *recolour* and *shake,* together with perturbations known as *kicks* which use compositions of neighbourhoods of relatively long length to make a single move.

Another tabu search algorithm using a four-phase system and known as OT-TABU has been implemented by White & Xie [105] to solve the exam timetabling problem at the University of Ottawa. They use a bin packing style of algorithm with largest enrolment first exam ordering to create an initial solution which may be feasible or infeasible. Similar to Di Gaspero & Schaerf [58], they use the standard move neighbourhood and keep two candidate lists: one containing all exams in the problem, the other containing only those involved in conflicts. A subset of the neighbourhood is iteratively explored and the best move is selected, irrespective of whether it leads to an improvement or not of the current solution. Two

forms of memory are used in the tabu search, these being a recency-based short-term memory and a frequency-based longer-term memory. The former uses a tabu tenure of nine and adds the move (x, i) to the tabu list whenever a move (x, i, j) is made where x is the exam moved, i is its original timeslot and j is the new timeslot. It is noted that when used in combination with the longer-term memory, the tabu tenure for the short-term memory is not critical. The longer-term memory is based on a frequency table which keeps track of whenever an exam is moved. The purpose of this longer-term memory is to forbid over-active nodes from moving constantly to help avoid cycling. A method of tabu relaxation is also included which empties all entries in both tabu lists if no improvement on the best solution so far has been found after a certain number of iterations. A four-pass intensification strategy is used to intensify the search in region of the search space containing the best solution so far obtained. Full details of the OT-TABU algorithm are given in [105] together with results when applied to the real life problem at the University of Ottawa which has 771 exams to be assigned into a timetable of 36 slots with 2200 seats available. In conclusion, the authors note that tabu search with both longer-term and short-term memory can generate better solutions on all data sets tested than TS using short-term memory alone and that the technique successfully avoids cycling during the search. Also they conclude that the OTTABU four-pass algorithm is an effective mechanism for controlling intensification and diversification in the search.

Paquete & Stützle [89] use tabu search to solve the examination timetabling problem using a lexicographic formulation similar to the multi-criteria approaches I will review in section 2.2.5 in which constraints are prioritised by the user rather than setting explicit weights within an objective function. Two district strategies are used, Lex-tie compares solutions based on the objective value of the higher priority objective whilst Lex-seq takes constraints into account one at a time in priority order to find a solution satisfying each in turn. The Tabu search algo-

rithm used is adapted from one successfully applied to graph colouring problems and uses a 1-opt neighbourhood (equivalent to the Standard neighbourhood of Thompson & Dowsland [101] with all pairs of exams and timeslots considered at each move and the one which maximally reduces constraint violations is chosen. The tabu tenure used is based on the number of constraint violations with a random integer added. The Lex-seq technique performed better on most datasets tested when measuring best performance, but using average performance as a measure, Lex-tie provided more consistent results. One other point of note from the authors was that as problem size increases, the value of their constant α, used to weight the contribution of constraint violations to the tabu tenure should also increase to obtain best performance when using the Lex-seq approach. Whilst Lex-tie performed less well when constraints became harder the satisfy they conclude that the results indicate that a future approach combining both strategies could prove promising.

Great Deluge

Of the variety of other local search techniques applied to exam timetabling, one of the most promising has been the Great Deluge (GD) algorithm applied by Burke & Newall [30] and Burke et al. [12] as part of two different techniques. In [30], the focus is on the affects of parameters on solution quality when Great Deluge is applied to a high quality initial solution with the results compared to those of a simulated annealing algorithm also tuned to give best performance, using desired initial average probabilities for acceptance of worse moves. The Great Deluge technique works similarly to simulated annealing, but uses a simpler mechanism for move acceptance with moves again generated at random from the neighbourhood. Rather than a probabilistic acceptance of moves based on a cooling schedule, Great Deluge uses a ceiling which starts off equal to the initial solution multiplied by some factor and is gradually lowered as the search

progresses until eventually it will fall below the current solution. All moves from an incumbent solution which result in a new solution whose valuation by the objective function falls below the ceiling are accepted and the search is terminated after 1 million iterations without improvement in Burke & Newall's implementation to allow for a hill climbing phase at the end of the search. As a parameter to the Great Deluge algorithm, the desired number of iterations, N, is input, from which the rate at which the ceiling is lowered at each iteration is calculated by dividing this into the initial ceiling to provide the increment. An index of improvement is used to measure the performance of the technique with different values for N over a range of benchmark problems and is calculated by summing the average percentage improvement (over five runs on each data set) from the initial solution across all problems. From the experiments performed, the authors conclude that the initial ceiling should be set as the initial solution multiplied by a factor of 1.3 and that launching the algorithm for more moves yields greater improvements. Great Deluge is found to perform better than simulated annealing and the parameters used for both techniques are observed to have a major effect on final solution quality. When compared to results from other techniques in the literature, Great Deluge performs very well, obtaining best known results on a number of benchmark problems supplied by Carter et al. [45]. These results are included in table 2.3

In [12], Burke et al. again use simulated annealing and the Great Deluge (GD) method for comparison as they investigate time-predefined versions of both techniques on exam timetabling problems. One of the key aims of their work is to eliminate parameters which are rather abstract (e.g. cooling schedule rate for simulated annealing or number of generations for a genetic algorithm) and not easily understandable to a real world user and replace these with just two parameters, both of which they feel are meaningful and easily understandable for a user. The two parameters employed by Burke et al. are the computational time

for the algorithm and an approximation of the objective function value that would be desirable. Computational time is clearly a very desirable input parameter as it allows the user to specify how much time they are willing to spend to find a solution, whilst an estimate of the desired final solution objective value can be easily acquired using a fast hill climbing technique. It is logical that the longer a search is run for, the greater the exploration of the search space and thus the probability of reaching a good solution is increased. The challenge which the authors consider is to make the most of all the available computation time so that the method does not converge too quickly and is allowed to use all the available time searching the solution space whilst creating a flexible method which can be used to produce average quality solutions very quickly or high quality solutions in more time. The authors present an approach which determines the rate at which the GD ceiling should be decreased based on the time allowed and the estimate of final solution quality and the value of the initial solution and ceiling. This approach allows the algorithm to spend all the allowed time effectively without the ceiling falling below the current solution too early in the search, thus preventing further exploration. A large number of experiments are reported with the overall stated aims as follows:

- To investigate the properties of the time-predefined techniques by generating “cost progress” diagrams for the search process - these are diagrams which track the evolution of the cost function.
- To explore the manner in which the prolongation of the search can increase the quality of solution.
- To evaluate the quality of the results produced by the time-predefined search in an acceptable time by comparison of its range with the outcomes of other techniques applied to the same datasets and published in the literature.

Brelaz' saturation degree graph colouring sequencing algorithm [9] with random timeslot assignment is used to create 20 initial solutions from which the best was chosen to seed the local search techniques. The neighbourhood structure used was the Standard Neighbourhood described earlier, consisting of all solutions which can be produced from the current one by reallocating a single exam to a new timeslot with moves selected at random from the neighbourhood for each iteration. A large number of observations are reported with respect to the speed of convergence with cost progress diagrams showing the improvements over time during the run of each algorithm with one of the major features noted being that time-predefined simulated annealing is much more uncertain and parameter dependent in its behaviour than the Great Deluge algorithm with more preliminary work needed to determine the parameters of the search. Regarding the trade-off between search time and quality of results, it is reported that whilst a prolongation of the search time does yield better results, the improvement becomes slower later in the search and that the trade-off holds mostly for "large" exam timetabling problems as expected. Results comparing performance of these algorithms to the current state of the art are recorded in table 2.3 and can be seen to compare very favourable with other techniques. Results of experiments on more advanced problems with more constraints are also included in [12] and again compared to those of other techniques in the literature. Improvements to the time-predefined technique are also discussed with conclusions that the neighbourhood used probably influences results notably and that the technique is open to different extensions and hybridisations, but overall results show the technique proposed to be very competitive as well as its numerous advantages regarding simpler parameters.

GRASP

Casey & Thompson [46] present a Greedy Randomised Adaptive Search Procedure (GRASP) technique for solving exam timetabling problems. The specific problem focused on is that of minimising the proximity of exams in students' timetables subject to the timetable being clash-free. The GRASP technique is a two-phase algorithm in which the first greedy phase is used to produce a feasible, clash-free timetable whilst the second phase concentrates on optimising the solution based on the objective function. Exams are ordered according to one of the criteria proposed by Carter et al. [45] for the construction phase with the next exam to be scheduled being chosen from the top n in the list using a roulette wheel selection and is assigned to the first available period. If there is no feasible period, a backtracking technique is employed using a tabu list to prevent cycling and ensure that a complete feasible timetable is produced. In phase two, exams are considered in descending order of their contribution to the total cost of the timetable, with all periods evaluated and the best feasible move is chosen if it causes a decrease in the cost. This process continues until no further improvement is found within a given cycle limit with the process then returning to phase 1 with a blank timetable. A number of improvements to the simple algorithm are outlined in an attempt to maximise the performance of the algorithm in a limited amount of time. These include the use of Kempe chain based neighbourhoods as used successfully with simulated annealing in [101], including a limited form of simulated annealing during phase two and the inclusion of a memory function to allow for diversification of the search. Experiments showed that this memory should only be included with the construction phase as its inclusion in the improvement phase prevented the search from attaining high quality solutions. Reported results on benchmark problems use ten GRASP iterations on a 1000 MHz Pentium computer and use the evaluation function also used by Carter et

al. [45]. The Saturation Degree (SD) ordering of exams for Phase 1 was found to give the best results compared to other strategies whilst Kempe chain based neighbourhoods were found to perform better than the standard move neighbourhood. Results compared to other reported techniques are again included in table 2.3.

Hybridised Local Search

One of the more recent research directions is to consider more hybridised local search algorithms, aiming to take the best ideas from a number of approaches and combine them into a single successful technique. Merlot et al. [82] present a three phase hybrid algorithm including constraint programming to produce an initial solution, simulated annealing to improve solution quality and finally a hill climbing phase to add further improvement. Their method is applied to the University of Melbourne (~ 650 exams, $\sim 20,000$ students) exam timetabling problem where it proves to be superior to their previous method and is also applied to the well known Carter's [45] benchmark data sets. The constraint programming phase used is similar to that of Boizumault et al. [7] discussed earlier and is designed to find a feasible solution very quickly with some exams allowed to remain unscheduled if needed. The simulated annealing phase uses a Kempe chain neighbourhood similar to that of Thompson & Dowsland [101, 103] with the difference that moves are selected by picking an exam and a new slot at random as oppose to Thompson & Dowsland who select two timeslots at random before selecting an exam from the first timeslot. A geometric cooling schedule is applied with the temperature being lowered by a factor of 0.999 after ten iterations at each temperature, from a starting temperature of 30,000. The hill climbing stage of the algorithm exhaustively searches a smaller neighbourhood to provide a final improvement phase, considering each exam in turn together with every other period. Results are included on a number of benchmark datasets

- graph colouring, uncapacitated and capacitated with the uncapacitated results being compared to those of other techniques in table 2.3. The conclusions of the authors regarding their method are that it performs well in comparison to other methods on benchmark problems as well as proving superior to their previous method at the University of Melbourne and also suggest that methods combining solution construction with local search will be dominant in the future for exam timetabling problems.

Caramia et al. [40] propose a family of timetabling algorithms based on local search considering both the minimisation of the number of time slots and the minimisation of an overall penalty using a fixed number of timeslots. Initially exams are ordered to be scheduled by their degree in the conflict graph with a greedy scheduler used to assign exams in turn to the lowest available timeslot which does not produce a conflict. Following this, a *penalty decreaser* is used to try to decrease the overall penalty of the timetable without increasing the number of slots used, considering each exam in order of non-increasing penalty. Once no further penalty reduction can be obtained, the greedy scheduler is restarted. A *penalty trader* is invoked when no improvement can be found by the greedy scheduler and the penalty decreaser. This part of the algorithm checks whether the penalty of the current schedule can be decreased by incrementing the number of timeslots by one with exams chosen to be moved to the new slot which will decrease this penalty by the largest amount. When the penalty trader finds a larger number of timeslots for which the penalty is decreased, the exam priorities, p_i (used for the greedy scheduler) for each exam i are re-assigned as $p_i = 1/t_i$ where t_i is the current slot of exam i. Then the greedy scheduler is restarted with the new exam priorities. A check-pointing scheme is incorporated to avoid the search getting stuck for long periods of time in particular areas of the search space. Best results presented on the benchmark data are given in table 2.3 for comparison with other techniques on problems with a fixed number of timeslots.

Results were also reported giving the minimum number of timeslots required to find a feasible solution with five different strategies evaluated for both sets of results. Overall the results are extremely competitive on the benchmark data considered, although no averages across many runs are included.

2003 International Timetabling Competition

One of the notable recent innovations was the 2003 International Timetabling competition [4] organised by the Metaheuristics Network to promote research into automated methods for timetabling. 20 different problem instances were supplied to competitors with the aim being to develop an algorithm to minimise the total penalty on each of these problems within a given time limit. Three soft constraints were included, with any violations of these adding one penalty point to the total:

- no student has only one event per day
- there are no students that attend an event in the last slot of a day
- no student has to attend more than two events consecutively on one day

Hard constraints which must not be violated were as follows:

- no student attends more than one event at the same time
- the room is big enough for all the attending students and satisfies all the features required by the event
- only one event is in each room at any timeslot

The problems themselves were reductions from typical course timetabling problems, but share many features in common with exam timetabling and as such

[4] http://www.idsia.ch/Files/ttcomp2002/

was considered worthy of mention here. The techniques of all the most successful competitors were based on variants of local search, Kostuch [73] favouring simulated annealing, Cordeau et al. [49] and Arntzen & Løkkentangen implemented a tabu search approach, Bykov [39] used the Great Deluge algorithm whilst Di Gaspero & Schaerf [59] opted for a three-stage local search technique including hill climbing and tabu search.

Local Search Conclusions

In conclusion, it is easy to see that a large amount of research has been carried out in this area since Carter & Laporte's 1996 survey and much work is still ongoing. Local search meta-heuristics are clearly some of the most successful and popular approaches to exam timetabling problems with a huge variety of different techniques applied and many hybrid methods now being developed which draw from the best elements of their components to produce an even better algorithm. Many of these techniques do have a lot of parameters, many of which are not easily understandable to a non-expert and they do unquestionably take more computational time and/or effort than sequential construction techniques, but in the area of exam timetabling this increase in runtime is considered acceptable provided it is within reason since the exam timetabling problem is generally only solved two to three times per year at most institutions. Some more recent methods such as those described in [12] now incorporate run time as a parameter to the algorithm so that the user can decide for themself how long they are willing to spend producing a solution and the algorithm uses this to make sure the search makes the best use of the time allowed. As with most local search techniques, there is a trade-off between extra solution time and solution quality.

2.2.4 Evolutionary Methods

The umbrella term of evolutionary algorithms (EAs) includes a variety of techniques inspired by nature and which make an attempt to simulate naturally occurring processes, usually involving populations of "solutions". Amongst the most common EAs are genetic algorithms and ant systems, although ant systems are yet to be thoroughly investigated as a technique for exam timetabling although Dowsland et al. [60] have proposed a method for using ant systems to find feasible solutions based on a graph colouring model. Genetic algorithms and similar evolutionary algorithms have been applied more widely with Corne et al. [51] giving an overview of techniques applied up to 1994 and Burke et al. [21, 22] propose a genetic algorithm for solving university timetabling problems using a population of always feasible timetables.

Burke et al. [32] use a hybrid method combining an evolutionary algorithm with local search to produce what is known as a *memetic* algorithm. This allows the possible solution space to be reduced to the subspace of local optima since every member of the population is optimised by the local search, but also increases the computational time significantly. The problem is represented by a population of memes, each of which contains information on which exams are in which room in each period of the timetable. Members of the initial population are generated using a weighted roulette wheel technique to choose the period in which to place each exam in order to produce a higher quality yet still diverse starting population. Light and heavy mutation operators are included, the light operator moving a number of exams to new feasible periods, the heavy operator disrupting entire periods. Hill climbing is then applied, taking the exams in each period in turn and checking all other periods to move an exam to the period of lowest penalty. The evaluation function penalises unscheduled exams heavily as well as taking into account the number of conflicts in the timetable

between two periods on the same day. Roulette wheel selection is again applied during the selection process in order to keep a specified population size. The algorithm was tested on a range of real data including the Nottingham University dataset of 1994/5 which includes 805 exams with 10,034 student conflicts between them and 7,896 students with 34,265 different enrolments, all exams must be scheduled into 32 periods with a maximum capacity of 1550 per period. The memetic algorithm is found to compare favourably to a multi-start random descent method, finding a solution which does not violate any constraints whilst the descent method fails to do so even when given a longer time. The technique performs less well on highly constrained problems as some methods, but shows promise overall.

In [98], Ross, Hart & Corne carry out an extensive investigation of GAs using a simple direct representation, showing their weaknesses as well as considering a number of positive research directions for GAs. In the representation they use, a timetable is represented as an array of E exams, with the i^{th} number indicating the timeslot of the i^{th} event. A simple penalty function based on clash violations is used to evaluate the fitness of a chromosome. Some of the weaknesses of the simple direct representation approach for a GA are exposed with two specific examples on which the GA performs very badly or fails to find a feasible solution, even on subproblems of more highly-constrained problems which it can solve, although it is also pointed out that other methods also struggle on some of these problems, in particular the 'sequence of cliques' class. It is also found that these GAs perform relatively badly on Carter's benchmark data sets [45] due to a failure to co-ordinate different parts of the solution until it is too late. The authors conclude by suggesting that GAs would be put to better use searching for a good algorithm to solve a problem rather than acting on the problem itself, an approach which has been applied in other domains as well as to timetabling by Terashima et al. [100] who used a GA to evolve constraint

satisfaction strategies as discussed in section 2.2.1.

A decomposition approach combined with an evolutionary algorithm is used by Burke & Newall [29]. They consider the problem with seating capacities and taking into account both first and second order conflicts, using a penalty of 5000 for any unscheduled exams to discourage incomplete timetables. The set of exams is decomposed into smaller sets and scheduled in different phases with earlier sets being fixed in their positions for the later phases. Two options are considered to get around the problem of this fixing of events making it impossible to schedule later events: firstly, exam subsets can be created using heuristic sequencing methods, secondly a form of look ahead can be implemented where two subsets are optimised at a time, but only one is fixed at the end of the stage. A population of size 50 is used for all reported experiments with mutation operators applied followed by hill climbing, using the same technique as discussed earlier in [32]. In the implementation, the already fixed parts of the timetable were considered as *super events*, i.e. a single event in each period of the table whose enrolment and clashes is equal to the sum of all events fixed in that period. Results are presented on 4 benchmark datasets using a variety of different sized subsets and for each size tests were carried out both with and without lookahead. Largest degree, colour degree and saturation degree were all tested for the exam ordering with very varied results produced across the range of datasets tested. Saturation degree was found to be the most reliable heuristic overall, though not the best for every problem, whilst a subset size of 50 was found to be most effective for smaller problems and 100 for larger problems with lookahead used in both cases. Results are produced in much faster run times than the previous memetic algorithm approach without decomposition and indicate that combining heuristic sequencing with evolutionary methods can utilise knowledge of the problem to produce better solutions than either technique on its own.

Erben [61] uses a *grouping genetic algorithm* (GGA) based on the grouping

character of graph colouring problems to avoid many of the problems of previous unsuitable encodings for GAs. In this formulation, a chromosome is made up of groups as genes, with each group representing all the elements of a given colour in the graph colouring representation. It is on these groups of nodes that the mutation and crossover operators work with numerous mutation operators tested and a crossover operator resulting in children containing groups from both parents as well as some new groups in general. The fitness function used first calculates the total degree, D_j of each group in the chromosome, then the fitness (to be maximised) of the chromosome P with k colour groups is defined as:

$$f(P) = \frac{1}{k}.\sum_{j=1}^{k} D_j^2$$

Results are presented for the troublesome "pyramidal chain" and "sequences of cliques" problems which more standard directly encoded GAs struggle with. The phase transitions in the latter set of problems and also for "equipartite graphs" are also located and discussed in detail. The technique is easily adapted to exam timetabling by considering the solution as a *sequence* of groups rather than a *set* of groups when second order conflicts are to be considered and the fitness function changed to also include these in its evaluation. Results reported on exam timetabling datasets are unimpressive compared to the best results of other authors, but it is noted that since the algorithm has simply been adapted from one developed for graph-colouring problems these results are still quite promising and the computational expense is relatively low. Regarding parameters for the GA, it is reported that a population size of greater than 20 does not seem to improve performance significantly whilst the GGA is quite robust with regards to crossover and mutation rate settings.

2.2.5 Multi-criteria approaches

The methods discussed so far generally combine all soft-constraints into a linear weighted objective function to be minimised (or maximised) during the search. In multi-criteria approaches, each soft constraint is represented by a criterion and the search method must deal with a vector of these criteria. The main advantage of multi-criteria approaches is that they do not require explicit weights for each constraint violation to be balanced against each other in a single objective function and are thus very flexible.

Burke et al. [11] use a model with just a single hard constraint, common to all exam timetabling problems, to define a feasible timetable, that being that exams which are in conflict (have students in common) must not be scheduled in the same period. Criteria are defined with respect to all other constraints present in a timetabling problem, each measuring the level of violation of the corresponding constraint. Using this approach, the list of criteria can easily be added to and subtracted from for any given problem and the authors present nine of these criteria, split into three groups:

1. *Room capacities*

    - C_1 represents the number of times that room capacities are exceeded.

2. *Proximity of exams* - criteria concerning the spread of exams across the timetable

    - C_2 represents the number of conflicts where students have exams in adjacent periods on the same day
    - C_3 represents the number of conflicts where students have two or more exams in the same day
    - C_4 represents the number of conflicts where students have exams in adjacent days

- C_5 represents the number of conflicts where students have exams in overnight periods

3. *Time and order of exams* - criteria concerning inappropriate times or order of exams for students

    - C_6 represents the number of times that a student has an exam that is not scheduled in a time period of the proper duration
    - C_7 represents the number of times that a student has an exam that is not scheduled in the required time period
    - C_8 represents the number of times that a student has an exam that is not scheduled before/after another specified exam
    - C_9 represents the number of times that a student has an exam that is not scheduled immediately before/after another specified exam

The overall aim is to minimise each of the nine criteria, but clearly this is not strictly possible as some are conflicting and many are of a very different nature. Weights can be assigned to each of the criteria to show their relative importance with *hard* constraints being given a much higher weighting than *soft* constraints. A mathematical formulation of the multi-criteria problem is given by the authors and the criteria space is defined of dimension equal to the number of criteria with each timetable being represented by a point in the criteria space. An *ideal point* is defined in the criteria space which optimises all criteria simultaneously, but which in reality does not exist in general, therefore the concept of an *anti-ideal point* is used. The criteria spaced is then mapped into a preference space in which the worst value for each criterion, C_k, is mapped onto zero and the best mapped to w_k, the weight assigned to criterion k.

An algorithm for heuristic search of the preference space is presented ([11]), based on compromise programming with solutions measured by their “distance”

from the ideal point in the preference space. The proposed algorithm consists of two phases, the first aims to find a set of high-quality timetables in terms of each criterion separately which form the initial timetables for the second which aims to improve the other criteria values in each solution using hill climbing and heavy mutation operators to explore the neighbourhood of timetables. The final solution is chosen from the set of timetables produced, with each initial solution yielding one timetable, as the one closest to the ideal point. Results are given for the University of Nottingham exam timetabling problem (800 exams, 7896 students, 33997 enrolments) whose characteristics are defined. It is noted that in most cases the final solution significantly improves the other criteria not taken into consideration initially at the expense of a slight degradation in the criteria from that initial solution. The timetabling officer is able to vary the weights on each criterion to produce different timetables from which the most suitable can be chosen. In conclusion, their approach is described as giving an insight into timetabling problems that is not provided by existing approaches, but results cannot be compared with those of single-cost functions since the evaluations are incomparable. The multi-criteria approach gives great flexibility for handling constraints, far more so than is possible with a single objective function.

Multi-objective evolutionary algorithms (EAs) are considered by Pacquete & Fonseca [88] with an EA based on a direct encoding of the mapping between exams and timeslots used to minimise violations of each type of constraint as separate objectives. They use a Pareto-ranking of the population to assign the fitness having evaluated each objective individually. The mutation operators considered are applied with probability based on the level of constraint violation of each exam, with a constant, β, used to control the level of bias towards those exams involved in constraint violations. The algorithm is tested on the exam timetabling problem at the former Unit of Exact and Human Sciences (UCEH) of the University of Algarve (249 exams, 30 time slots) where exams are in groups, typically

of 8-10 exams, with rules applied to these. Experiments were run to test a variety of objectives concerning the comparison of Pareto-ranking to linear-ranking, independent mutation compared to single-position mutation and the value of the parameter β. 10 runs, each of 5000 generations were performed for each test with a variety of statistical tests performed on the data produced. When assessed based on solution quality, the Pareto-ranking approach was found to give better performance than the linear sum of objectives, but little difference was noted between the two mutation operators whilst significant performance differences were observed for different values of β. When time was considered as an additional objective it was found that the linear aggregation of objectives was more effective at minimising constraint violations whilst the Pareto-ranking method provided better covering of the objective space. Versions of this algorithm have been in use at UCEH since 1999.

In 2002, Petrovic & Bykov [92] adapted the Great Deluge local search algorithm [12] to apply to multi-objective optimisation with the aim being to find a solution which dominates a reference solution provided by a user, improving the values of all objectives by following a search trajectory. A solution is said to *dominate* another solution if the values of all its criteria are superior to those of the second solution with the set of non-dominated solutions forming a Pareto-front from which one solution is usually chosen. The trajectory-based approach presented by Petrovic & Bykov places the reference solution provided by the user into the criteria space with the trajectory being drawn from there to the origin and the search algorithm should gradually improve the solution keeping close to the trajectory line. For the search itself, a random initial solution is used with the reference solution being used only to define the trajectory for the search. Initially the search guides the solution towards the trajectory line which it then follows until it reaches the reference solution and continues until convergence, whereupon any further solution will clearly dominate the reference solution. Us-

ing the Great Deluge algorithm, a weighted sum cost function is applied, but with weights which vary dynamically as the search progresses. In the example given of a bi-criteria space with criteria c_1 and c_2, solutions are accepted if they fall below the borderline, B, defined by:

$$B = c_1 w_1 + c_2 w_2$$

where w_1 and w_2 are the weights of the respective criteria. Increasing either w_1 or w_2 causes the borderline to rotate, directing the search to focus more on one criteria. Rather than reducing a level at each step as in the standard Great Deluge algorithm, their method increases a single weight, leading to a multi-objective extension of the Great Deluge algorithm. In the bi-criteria case, the trajectory from the reference point to the origin splits the criteria space into two. If the current solution is above the line then w_2 will be increased to guide the search back towards the trajectory, whereas w_1 would be increased if the current solution is below the trajectory line. Thus the dynamic altering of weights guides the search along the trajectory line in the direction of the origin. The method is also extended to a nine-criteria case, the nine criteria being the same as discussed earlier [11]. Experiments were run with a runtime of 20-25 minutes which is considered quite acceptable for exam timetabling with the initial reference solutions provided by those from [11]. Results show that the new solutions produced dominate the reference solutions on all criteria thus demonstrating the effectiveness of the variable weight approach to multi-objective timetabling. The approaches of Burke et al. [11] and Petrovic & Bykov [92] are further discussed in [35], [75] and [91] together with ideas for further research in the area. [75] also discusses numerous other applications of multi-objective meta-heuristics in scheduling and timetabling.

2.2.6 Case-based reasoning and hyper-heuristic methods

The case-based reasoning (CBR) methodology [72] has only very recently been applied to timetabling problems, but provides an interesting research area in the search for techniques which offer a higher level of generality than many current exam timetabling methods. Case-based reasoning approaches rely on past experiences rather than a set of rules to help solve new problems. A case-base of these past experiences is maintained with cases generally indexed by their key features to enable them to be retrieved when a similar problem is encountered. A case traditionally consists of a list of feature-value pairs which can easily be compared across problems to give a measure of similarity with each feature given a weighting in the similarity measure. When presented with a new problem, a case-based system will match it with the most similar case in the case-base, whose solution can then be retrieved and applied to the new problem, the assumption in CBR being that similar problems have similar solutions. In most cases there will also need to be an adaptation phase in which the solution to the retrieved problem must be altered in order for it to be applied to the new problem. Petrovic & Burke [91] note that CBR can have a twofold role in solving timetabling problems: either as a *solution reuse* technique or as a *methodology reuse* technique. Here I consider literature on both of these techniques.

Burke et al. [27] consider that the feature-value pair case representation is not always sufficient for complex problems such as timetabling and present an alternative approach using structured cases and attribute graphs. The approach presented is applied to course timetabling, but can also be adapted to exam timetabling with the attribute graphs consisting of nodes representing the events, edges representing conflicts and attributes on both the nodes and the edges which give further information on the problem structure. This approach allows different cases to have different structures unlike the feature-value pairs approach in

which all cases must have the same list of features to be compared. The attribute graph representation has many advantages, but also makes the matching process more complicated, equivalent to a graph isomorphism or sub-graph isomorphism problem which is known to be NP-Complete. Adjacency matrices are used to represent the attribute graphs for the matching process and two attribute graphs are regarded as similar if they differ by less than a specified threshold with penalties for any differences between the two graphs. A decision tree stores cases hierarchically with each case being classified to a node and all nodes below that node are retrieved as candidates. Once a case has been retrieved, the solution to the retrieved problem will generally need to be adapted to suit the new problem and this is done using a graph heuristic method which attempts to minimise constraint violations. Burke et al. [28] and Qu [97] discuss this approach for solution reuse in course timetabling in more detail with an overview of the system produced given in [91].

As well as being applied directly to problems, CBR can also be used at a higher level of abstraction as a method to select a heuristic to apply to a new problem as a hyper-heuristic. A hyper-heuristic can be considered as an automated approach for choosing heuristics to apply to a problem. Burke et al. [36] explore this approach for exam timetabling problems. A heuristic developed to work well on one problem will very often be far less successful when applied to a different problem, but may be successful on problems or subproblems which can be regarded as similar to the original problem, thus employing a variation of the main CBR assumption: *similar problems may be solved equally successfully by the same technique*. A case-based hyper-heuristic is presented which constructs solutions step by step, at each step using the case-base to retrieve the most similar case to the current partial solution. The case-base contains a number of partial solutions obtained during problem solving on previous problems with each case represented this time by a list of feature-value pairs describing the problem

characteristics. Four well known heuristics, largest degree, largest degree with tournament selection, colour degree and saturation degree are included in the system. The similarity measure used calculates the sum of differences of values between each pair of features in two cases being compared. The best heuristic for the retrieved case is applied in the next step of the new solution construction with the process terminating once a complete solution has been constructed.

Knowledge discovery techniques are applied to discover the most important features, which can contribute to a good choice of heuristic, to be used within the case-base. Tabu search is used to carry out the search in the search space of all possible combinations of features. A number of randomly generated data sets were used, ranging from 100 to 300 exams, to train and test the performance of the system. Having identified an initial set of cases for the case-base, these were then pruned systematically by removing any cases which were not contributing positively to the performance of the system. Tests were carried out to measure system performance on both training and test cases with the number of features, as suggested by the knowledge discovery process, varied between two and 10 and with differing sizes of case base. It was found that the system performs best with between three and seven features used, with more features detracting from the performance by diluting the impact of more important features. With these selected features, approximately nine out of every 10 testing cases obtained the expected heuristics as pre-calculated to give best performance. Results are also presented on 100 exam timetabling test problems, randomly generated with densities ranging from 0.1 to 0.6 in the conflict matrix, and compared to the performance of each heuristic when applied for the full construction process. The quality of solutions produced by the CBR heuristic selector was better in almost all cases, with only the saturation Degree technique providing solutions by itself which were comparable, but still worse if the right number of features is used in the CBR system. Petrovic & Burke [91] discuss this method further and give

comparisons with a similar method by the same authors which selects a single heuristic to applied throughout the construction process rather than dynamically changing heuristics.

Burke et al [37] present a CBR system for heuristic selection based on the work from [36], for both selecting a single heuristic for course timetabling and constructive heuristics during the problem solving for exam timetabling problems. A two-stage knowledge discovery process is used. The first phase of knowledge discovery finds the best feature vector, the second phase pruning the source cases by removing those which do not add to the performance of the system. Two sets of features are considered for both course and exam timetabling together with combinations of all these features with results presented on the different feature lists obtained by the knowledge discovery showing system performance (as defined above). The feature lists used generally employ combinations of the eleven features which are not immediately intuitive to the user, but which were found to give best performance when used together. As reported in [36], between three and seven features gives best performance of the system. The system is tested on both randomly generated and real-world exam timetabling problems, as presented by Carter et al [45]. The authors conclude that the CBR heuristic selector is highly flexible for use on a wide range of problems and whilst it cannot compete with techniques devised specifically for these problems, the aim is to provide a higher level of generality by producing competitive results across a wide range of problems. Further work is ongoing with CBR selection of meta-heuristic techniques for exam timetabling problems also being investigated.

Hyper-heuristics are a relatively recent area of research with respect to timetabling problems, but are now becoming more widely used in the search for more generality in solution finding techniques. Hyper-heuristics are essentially a heuristic to choose a heuristic and operate at a higher level of abstraction than many of the techniques discussed so far which are applied directly to the problem. The

use of a genetic algorithm (GA) to evolve constraint satisfaction strategies [100] was discussed in section 2.2.1, whilst the CBR technique for selecting heuristics discussed in this section is another hyper-heuristic method. Burke et al [25] have also applied a tabu search hyper-heuristic to nurse-rostering and course timetabling problems. Here I discuss three further approaches to the hyper-heuristic idea, the first two using tabu search, the second also considering hybrid graph heuristics using CBR whilst the third utilises variable neighbourhood search (VNS).

Kendall & Hussin [69] present an investigation of a tabu search based hyper-heuristic for examination timetabling with the aim being to design a generic system which can select the most appropriate algorithm for the current instance of a timetabling problem. The hyper-heuristic is a generic module which works with a lower level module containing a set of problem-specific heuristics so that the hyper-heuristic itself has no specific domain knowledge. The hyper-heuristic module simply selects from a set of low-level heuristics (known as $H_1, H_2, \ldots, H_n$) one to apply to the current problem state, the low level heuristic will then return an evaluation of its modification to the solution which is all information that the hyper-heuristic module receives to make its decisions from by which to guide the search. An initial solution, produced by a constructive heuristic, is fed to the system followed by a randomisation step which moves some exams around to allow for different initialisations. The hyper-heuristic module uses a tabu list of fixed length n equal to the number of low-level heuristics with each tabu entry containing information about each heuristic including recent change in evaluation function and CPU time taken to run the heuristic as well as a tabu status to determine how long the heuristic remains tabu. A number of strategies are identified for considering which heuristic to apply next:

- consider all heuristics

- consider all non-tabu heuristics
- consider only heuristics which lead to an improvement

The chosen implementation is a hyper-heuristic with fixed tabu duration (HH-FTD) where all heuristics are considered and only the heuristics which are non-tabu and lead to the best improvement are applied. The algorithm iterates for a fixed time or until no further improvement is reported for a given number of heuristic calls. The low-level heuristics module is the domain-specific part of the system, built up from problem-specific heuristics. Each low-level heuristic changes the current state of the problem to a new state and returns the move and its evaluation. Four categories of low-level heuristic are considered initially:

- Select and schedule exam - includes heuristics to select and assign an unscheduled exam.
- Move exam i from location x to y - includes heuristics to select which exam to move and where to move it to
- Swap - includes heuristics to choose two exams whose timeslots will be exchanged
- Remove - selects an exam to be unscheduled

Preliminary results reported are unable to beat the best results from the literature (see table 2.3) on well known benchmark problems, but can produce good solutions across the range of problems.

As discussed in section 2.2.2, graph colouring heuristics can be used to produce fast, good quality solutions to some exam timetabling problems, yet by themselves they struggle to compete when measured on solution quality with the local search techniques considered in section 2.2.3. Burke et al [15] aim to overcome this problem by hybridising two graph colouring techniques, namely

saturation degree and largest degree (both defined in section 2.2.2, first using a tabu-search based hyper-heuristic in a similar way to the technique of [37] which applied CBR. As in [37], solutions are built up by applying one of the heuristics at each point of the construction phase to schedule the exams. The tabu search method searches through the possible permutations of largest degree and saturation degree with a number of mechanisms added to reduce the size of this search space. Tabu search stores parts of heuristic lists which lead to infeasible solutions and any heuristic list which includes these combinations is automatically ignored. At each step of solution construction, five exams are scheduled using the selected heuristic which significantly reduces the size of the search space. Finally, the initial heuristic list of tabu search is set as a list of saturation degree only since it is observed that this heuristic appears far more often than largest degree. Experiments on random data sets with sizes of 200 or 400 exams and conflict matrix densities of 0.05, 0.15 and 0.25 showed that the tabu search method outperforms both saturation degree and largest degree when used purely on their own throughout the whole construction process. Based on the knowledge discovered from this, another technique is presented, randomly injecting a fixed percentage (23% here) of largest degree into the heuristic list of entirely saturation degree. The logic for this is that this is the density of largest degree found within solutions produced using the tabu search approach. This new approach operates much faster than the tabu search method as it does not involve the heuristic search and is more of a pure construction technique combining saturation degree and largest degree in an intelligent manner. Results are, as expected, not quite as good as using the tabu search approach, but are better than either saturation degree alone or largest degree alone and with run times comparable to those methods.

Another method presented in [15] makes use of case-based reasoning (CBR) to again inject largest degree into a heuristic list of initially only saturation de-

gree. The case base stores the heuristics used in different problem solving situations previously encountered and the new problem is presented to the case-base at each stage of its construction with the most similar case being retrieved and the suggested heuristic being applied to the next stage of construction. The knowledge discovery process is carried out by using the best heuristic lists obtained from the authors' tabu search approach discussed above. The objective of this is to discover the most relevant features by which to index cases in the case base with a feature-value representation used to represent the cases. A list of simple features, some of which describe the problem itself and others which describe the current state of the problem solving, together with combinations of these features are all considered initially. Tabu search is used for discovering the best feature lists by searching in the space of all possible feature lists with a move defined by the change of a feature and its weight. Training of the case-base is carried out in a similar two phase method to that used in [37] with the set of cases pruned after features have been identified. When tested on four real world benchmark problems, the CBR approach described shows promising results when compared to saturation degree alone, but is outperformed in all cases by the tabu search method for combining largest degree and saturation degree described earlier in the paper (and in the above paragraph). Results from the tabu search technique produce good results close to those of special purpose methods developed specifically for exam timetabling. As with other hyper-heuristic and case-based reasoning approaches, the aim of the methods presented in this paper is not to match the performance of the best techniques, but to increase the level of generality in the problem solving to allow wider applicability.

Ahmadi et al. [2] propose a perturbation based algorithm to search the space of parameterised heuristics to select the best heuristic for the current problem. They present an approach using a weighted decision function for dealing with violations of all hard and soft constraints which they point out has the draw-

back of being highly parametrised. To counter this problem they introduce their perturbation based heuristic search algorithm to explore the heuristic space and determine the best parameters. A number of different types of constraints are taken into account in the weighted decision function including clashing exams, resource constraints and time windows. Seven types of heuristics for exam, period and room selection are considered with variations on each giving a total of 90,750 different combinations of heuristics even before weights and other parameters are included. The heuristic search algorithm proposed defines a neighbourhood of a heuristic h by means of perturbing its parameters which induces a neighbourhood for the solution produced by the heuristic, in the solution space. Each heuristic is encoded with a set of parameters which can be perturbed in a number of ways including altering the heuristic or its weights defining two types of neighbourhoods which are switched between during the search process in an approach similar to the variable neighbourhood search (VNS) approach of Mladenović and Hansen [80]. A descent local search is used within a variable neighbourhood search framework to explore these neighbourhoods. The aim of the search process is to find near optimal sequences of heuristics in the heuristic space rather than solutions in the solution space. A variety of experiments are reported to test the performance of the approach which is found to be robust in terms of searching the heuristic space from random starts.

2.3 Results for Benchmark Problems

Since Carter & Laporte's survey of 1996 [43], there has been a wide range of different techniques applied to a number of variations of the exam timetabling problem. With a number of benchmark datasets publicly available [5] it is now possible for many of these different techniques to be compared to give a better

[5] Benchmark data sets used here are available on the web at http://www.or.ms.unimelb.edu.au/timetabling/ttframe.html?ttpublic.html

Data Set	Number of exams	Carter et al. [45]	Caramia et al. [40]	Merlot et al. [82]
CAR-S-91	682	**28**	**28**	30
CAR-F-92	543	**28**	**28**	31
EAR-F-83	190	**22**	**22**	24
HEC-S-92	81	**17**	**17**	18
KFU-S-93	461	**19**	**19**	21
LSE-F-91	381	**17**	**17**	18
PUR-S-93	2419	**35**	36	-
RYE-F-92	486	**21**	**21**	22
STA-F-83	139	**13**	**13**	**13**
TRE-S-92	261	**20**	**20**	21
UTA-S-92	622	32	**30**	32
UTE-S-92	184	**10**	**10**	11
YOR-F-83	181	**19**	**19**	23
MEL-F-01	521	-	-	**28**
MEL-S-01	562	-	-	**31**
NOT-F-94	800	-	-	**23**

Table 2.1: Graph colouring benchmark data sets - minimum number of periods reported [45]

idea of which are the most promising on the given problems.

The most basic timetabling sub-problem is that of graph colouring, where the objective is simply to minimise the number of periods in a timetable with the only constraints being that clashing exams must not be scheduled to the same time slot and all exams must be scheduled. Results from the literature on a set of graph colouring benchmarks are given in table 2.1. The results obtained by Carter et al [45], using a variety of heuristic construction techniques with backtracking, provide best known solutions to the majority of problems, with Caramia et al [40] matching these results on all problems tested except PUR-S-93 and obtaining a lower minimum number of periods for the UTA-S-92 data set.

The most commonly used of these benchmark problems are those of Carter et al. [45] (see table 2.2) with many researchers applying their techniques to these

Data Set	No. of exams	No. of students	No. of enrolments	Graph Density	No. of periods
CAR-S-91	682	16925	56877	0.13	35
CAR-F-92	543	18419	55522	0.14	32
EAR-F-83	190	1125	8109	0.27	24
HEC-S-92	81	2823	10632	0.42	18
KFU-S-93	461	5349	25113	0.06	20
LSE-F-91	381	2726	10918	0.06	18
PUR-S-93	2419	30032	120690	0.03	42
RYE-F-92	486	11483	45052	0.07	23
STA-F-83	139	611	5751	0.14	13
TRE-S-92	261	4360	14901	0.18	23
UTA-S-92	622	21267	58979	0.13	35
UTE-S-92	184	2750	11793	0.08	10
YOR-F-83	181	941	6034	0.29	21

Table 2.2: Characteristics of uncapacitated benchmark problems [45]

uncapacitated problems with the aim of minimising the penalty cost per student for the given objective function which considers proximity costs w_s for students having to sit two exams s periods apart: $w_1 = 16, w_2 = 8, w_3 = 4, w_4 = 2$ and $w_5 = 1$. Published results on these benchmark problems are included in tables 2.3.

Across the range of problems, best results come from five different techniques, three of which give best known results to just a single problem whilst the other two provide best known results on the rest of the problems. The hybrid approach of Caramia et al. [40] proves to be very successful on the majority of problems, but struggles to compete on the larger data sets where the adaptive great deluge technique of Burke & Newall [30] provides the best known results. Almost all the high quality results reported come from local search techniques with a variety of different initialisations which can prove equally important as the local search technique to the solution quality.

Whilst the uncapacitated benchmark problems provide an excellent test bed for new methods to compare against more established methods on the core problem of scheduling all exams feasibly, real life problems generally include a num-

Data Set	Carter et al. [45]	Caramia et al. [40]	Burke & Newall [30]	Di Gaspero [57]	Casey & Thompson [46]	Merlot et al. [82]
CAR-S-91	7.1	6.6	**4.6**	5.7	5.4	5.1
CAR-F-92	6.2	6.0	**4.0**	-	4.4	4.3
EAR-F-83	36.4	**29.3**	36.1	39.4	34.8	35.1
HEC-S-92	10.8	**9.2**	11.3	10.9	10.8	10.6
KFU-S-93	14.0	13.8	13.7	-	14.1	**13.5**
LSE-F-91	10.5	**9.6**	10.6	12.6	14.7	10.5
PUR-S-93	3.9	**3.7**	-	-	-	-
RYE-F-92	7.3	**6.8**	-	-	-	8.4
STA-F-83	161.5	158.2	168.3	157.4	**134.9**	157.3
TRE-S-92	9.6	9.4	8.2	-	8.7	8.4
UTA-S-92	3.5	3.5	**3.2**	4.1	-	3.5
UTE-S-92	25.8	**24.4**	25.5	-	25.4	25.1
YOR-F-83	41.7	**36.2**	36.8	39.7	37.5	37.4

Table 2.3: Selected results from the literature on uncapacitated benchmark problems from [45] (best results given)

Data Set	Petrovic et al. [93]	White & Xie [105]	Burke & Newall [31]	Di Gaspero & Schaerf [58]	Burke et al. [12]	Paquete & Stützle [89]
CAR-S-91	-	-	5.0	6.2	4.8	-
CAR-F-92	-	4.7	4.3	5.2	4.2	-
EAR-F-83	34.5	-	36.2	45.7	35.0	38.9
HEC-S-92	10.9	-	11.6	12.4	10.6	11.2
KFU-S-93	14.8	-	15.0	18.0	13.7	16.5
LSE-F-91	10.6	-	11.0	15.5	10.4	13.2
STA-F-83	159.9	-	161.9	160.8	159.1	158.1
TRE-S-92	**8.0**	-	8.4	10.0	8.3	9.3
UTA-S-92	-	4.0	3.4	4.2	3.4	-
UTE-S-92	-	-	27.4	29.0	25.7	27.8
YOR-F-83	36.7	-	40.8	41.0	36.7	38.9

Table 2.3: (cont.) Selected results from the literature on uncapacitated benchmark problems from [45] (best results given)

ber of further constraints. In particular, capacity constraints on the number of students who can sit an exam at any one time are usually defined by the number of available rooms. Two different objective functions have been used to develop two sets of capacitated benchmark problems. The first of these concentrates on minimising the number of students who have two consecutive exams in the same day, with results from a number of techniques presented in table 2.4. The second attempts to minimise the number of students who have exams in consecutive sessions with a weight of three applied to same-day clashes and a weight of one applied to overnight consecutive exams, results for these are given in table 2.5.

The capacitated problem has been less well studied at this point in time than the simpler uncapacitated problem, but with data sets now available which add constraints to the uncapacitated problems, a number of authors have tested their techniques on these. From table 2.4, it can be seen that the hybrid technique of Merlot et al [82], combining constraint programming, simulated annealing and hill climbing outperforms all other reported techniques on the first set of capacitated data with much better solutions all round as well as being the first technique to be fully applied to all available problems.

The second set of capacitated problems has been more widely studied, with five different techniques from table 2.5 providing best known solutions to at least one problem. Again, the technique of Merlot et al [82] is the only one to be applied to all problems and thus results in the majority of best known solutions, but the multi-criteria approach of Petrovic & Bykov [92] proves very competitive on the three data sets tested and three other techniques by Burke & Newall [29] and Burke et al. [12, 34] each provide a best known solution. The multi-stage evolutionary algorithm of Burke & Newall is particularly effective when applied to the very large PUR-S-93 data set, largely due to its decomposition method. With target solutions provided by Merlot et al [82], it is likely that a number of new techniques will be applied to these data sets in the near future.

Data Set	Time Slots	Total Capacity	Burke et al. [32]	Di Gaspero & Schaerf [58]	Caramia et al. [40]	Merlot et al. [82]
CAR-S-91	51	1550	81	88	74	**31**
CAR-F-92	40	2000	331	424	268	**158**
EAR-F-83	24	350	-	-	-	**564**
HEC-S-92	19	650	-	-	-	**184**
KFU-S-93	20	1955	974	512	912	**247**
LSE-F-91	18	635	-	-	-	**259**
STA-F-83	13	465	-	-	-	**2063**
TRE-S-92	35	655	3	4	2	**0**
UTA-S-92	38	2800	772	554	680	**334**
UTE-S-92	10	1240	-	-	-	**377**
YOR-F-83	22	300	-	-	-	**418**
MEL-F-01	28	3024	-	-	-	**279**
MEL-S-01	31	3024	-	-	-	**67**
NOT-F-94	23	1550	269	123	-	**88**
NOT-F-94	26	1550	53	11	44	**2**

Table 2.4: Results on Capacitated benchmark problems (set 1) with objective to minimise occurrences of two consecutive exams in a day (best results shown)

Data Set	Time Slots	Total Capacity	Burke & Newall [29]	Burke et al. [34]	Burke et al. [31]	Burke et al. [12]	Merlot et al. [82]	Petrovic & Bykov [92]
CAR-S-91	51	1550	-	-	-	-	**812**	-
CAR-F-92	36	2000	1665	2218	1775	**1506**	1744	1522
EAR-F-83	24	350	-	-	-	-	**2116**	-
HEC-S-92	19	650	-	-	-	-	**929**	-
KFU-S-93	20	1955	-	**3256**	-	-	-	-
KFU-S-93	21	1955	1510	-	1422	1321	**1082**	1262
LSE-F-91	18	635	-	-	-	-	**1192**	-
PUR-S-93	30	5000	**63824**	-	97237	-	-	-
STA-F-83	13	465	-	-	-	-	**7688**	-
TRE-S-92	35	655	-	-	-	-	**143**	-
UTA-S-92	37	2800	-	2440	-	-	**2387**	-
UTE-S-92	10	1240	-	-	-	-	**2024**	-
YOR-F-83	22	300	-	-	-	-	**1496**	-
MEL-F-01	28	3024	-	-	-	-	**1665**	-
MEL-S-01	31	3024	-	-	-	-	**1104**	-
NOT-F-94	23	1550	519	-	545	384	401	**326**

Table 2.5: Results on Capacitated benchmark problems (set 2) with objective to minimise occurrences of two exams in consecutive time slots (best results shown)

2.4 Conclusions

The public availability of more and more exam timetabling datasets is allowing the research community to test the efficiency of different techniques against each other far more easily now, allowing the development of better and better techniques. However, whilst the successful application of a technique to benchmark problems provides a strong basis for its further use, it must always be borne in mind that the ultimate test of any technique is whether it can be implemented successfully in a robust system capable of solving the real world problem at hand to a high level of quality. Further to the growing number of benchmark instances, Kingston [70] has also developed the STTL language for modelling timetabling problems. The three main aims of format presented are to be *general*, *complete* and *accessible* which it successfully achieves with an STTL interpreter also freely available. Whilst relatively few papers appear to have mentioned use of STTL, it has been used by the author to model number of real world problems and represents an important contribution to the research community, allowing far more standardisation of problem definitions.

The most popular methods over the last eight years appear to be those based on local search techniques with numerous different methods also considered for initialising these. Many of the more successful techniques now being applied to the benchmark problems involve hybridisations of a number approaches, aiming to take the best points from each. This is now one of the major areas of current research with a large variety of different techniques available to be combined in numerous different ways to provide improved algorithms. The most successful technique often depends heavily on the specific problem instance in question however so an algorithm which performs well on some problems may not perform as well on others. With this in mind, there has been a growing amount of research in recent years into systems with a higher level of generality such as

hyper-heuristic, case-based reasoning and adaptive techniques which can adapt themselves to the particular problem being solved or make use of past knowledge which may be of use for solving the new problem. These approaches often cannot compete on individual problems with specifically designed approaches, but they do provide a much greater level of generality to produce good quality solutions across a greater range of problems. This is a relatively recent area of research, but is moving forward at pace given its attractiveness for real world problem solving. Techniques which not only can be successfully applied across a wide range of problem instances, but also involve relatively few parameters or parameters which a non-expert can easily understand and set are now becoming very popular.

There are still only a relatively small number of benchmark datasets available which incorporate more complex constraints, making it difficult to accurately compare the effectiveness of different techniques when more side-constraints are included than the benchmark problems considered in table 2.3. However, tables 2.4 and 2.5 show that techniques are now being applied to those benchmark problems which have been established. Multi-criteria approaches are becoming increasingly popular as methods to deal with a larger number of constraints rather than the more common method of a linear weighted sum in a single evaluation function to be minimised. Due to their different nature of evaluating solutions, multi-criteria approaches cannot be easily compared with those which seek to minimise a single objective function, but have been shown to be successful on real life problems as well as allowing more flexibility for the user. Many of the techniques described in this chapter have been developed with a specific problem in mind whilst others are aimed more at comparing the methodology to others on benchmark problems. Two further papers not discussed above specifically detail their implementation and application of techniques to a real world system. Lim et al [76] present an automated timetabling system for examination scheduling at

the National University of Singapore by modelling their problem as a constraint satisfaction problem (CSP). Dimopoulou and Miliotis apply mathematical programming to their system which is in use at the Athens University of Economics and Business. Both these systems have proved to be very successful on the problems for which they were developed which is ultimately the most important test of any technique.

As in Carter & Laporte's survey [43], I have restricted the scope of this survey to examination timetabling with a small number of course timetabling papers included which I consider to demonstrate techniques applicable to exam timetabling. It is likely that techniques applied in other related areas may be easily adaptable to exam timetabling, but these are beyond the scope of this survey. Ho, Lim and Oon [68] give evidence of this fact by applying a technique more commonly employed on vehicle routing problems to find timetable solutions which maximise paper spread by converting the exam timetabling problem into a vehicle routing one. Their results indicate that the technique produces very promising results.

Chapter 3

Case based reasoning (CBR)

3.1 Introduction to CBR

Case-based reasoning (CBR) is a methodology which people use in everyday life without knowing it as such. In simple terms case-based reasoning involves learning from experiences and keeping these in a case-base so that when presented with a new problem the case-based reasoner can refer to these past cases and develop a solution to the new case. In doing so, the case-based reasoner is looking for the most similar case to compare with the new one. The idea behind this is that a solution used for a previous similar problem can be used again or adapted for the current problem. The main assumption underpinning the CBR methodology is that similar problems have similar solutions. The definition of *similarity* forms a key component of any CBR system and is one of the main issues which can determine the success or failure of the system.

For example, doctors will receive a large number of patients with similar symptoms, many of whom will have the same medical problem. Rather than starting from the beginning every time, doctors will remember their previous cases and will have an advantage in knowing what to look for and how to go about diagnosing and treating the current patient.

Kolodner [72] gives a very in-depth explanation of the key ideas involved in case-based reasoning together with some of its many applications. Here I will present a summary of the main concepts of CBR given by Kolodner.

Case-based reasoning can mean adapting old solutions to meet new demands, using old cases to explain new situations, using old cases to critique new solutions or reasoning from precedents to interpret a new situation or create an equitable solution to a new problem. A case-based reasoner also learns as part of its activity. It becomes more efficient and competent as a result of storing experiences and referring to them in later reasoning.

CBR views reasoning as a process of remembering one or more concrete instances or cases and basing decisions on comparisons between the new situation and the old one. The use of general knowledge, decomposition and recomposition are, therefore, not so important. Instead, emphasis is on the manipulation of knowledge in the form of specific instances. These large "chunks of composed knowledge" are used as the starting point for the reasoning process.

A case is a *contextualised* piece of knowledge which represents an experience. Cases in the case library can each teach a lesson which helps to achieve the goals of the reasoner using it. Cases are *indexed* by a combination of their descriptors that predict when the case is most likely to be useful in the future.

Because case-based reasoning involves both reasoning and learning, it is not enough for a case-based reasoner to end the process once a solution has been found. Feedback from the solution must be collected in order for the case-based reasoner to index the new case and learn about how best the used case can be indexed. Without this learning process, CBR would be too unreliable and bad solutions may be repeated.

The quality of a case-based reasoner's reasoning obviously depends on the experiences it has had previously, its ability to understand new situations in terms of those old experiences and its skill at adaptation and evaluation. The major

processes which a case-based reasoner uses are:

- *case storage* - The initial case base must be set up with cases indexed by their key features to facilitate easy case retrieval
- *retrieval* - When a new problem is presented to the CBR system a matching process takes place for all cases in the case base with a similarity measure determining which case(s) in the case base are most similar to the new problem. The most similar case(s) will then be retrieved to be applied to the new problem, usually after some form of adaptation
- *adaptation* - Once a similar case has been retrieved, it is usual that the solution of the retrieved case must be adapted before it can be applied to the new problem. Differences will usually exist between the old and new problems which must be reconciled by some form of repair technique before the retrieved solution can be applied to the new problem
- *criticism* - One of the most important parts of any CBR system is the feedback or criticism from the choices made in the retrieval process. If the match was good and the retrieved solution successful on the new problem the system is successful. However, if the retrieved solution was unsuitable for the new problem, this information must be fed back into the system and the retrieved case should be re-indexed accordingly so that it would not be matched again in the same circumstances.

Case-based reasoning can be applied to a variety of situations ranging from those in which there is a lot of knowledge to those where very little knowledge is available. In the latter situation, previous cases are often all that a reasoner has to work with in solving a new case. There are many advantages to the CBR approach as it allows the reasoner to propose a solution to a new problem quickly,

reason in subject areas which are not well understood and avoid previous problems. The major disadvantages are all related to bad indexing and use of cases. If the reasoner just relies on the result of a previous case rather than evaluating its applicability to a new case, it could provide a very poor solution. Feedback on the performance of the system is crucial for the learning process in order to continually improve the way a CBR system works.

CBR can be used on a number of different levels from a fully automated system down to a machine which just stores cases for a human to refer to in a similar way to books in a library. It is then up to the human to evaluate the solution.

3.2 CBR and Scheduling

Applying case based reasoning to timetabling problems is a relatively new idea and as such there has been relatively little research carried out in this area so far. However, Burke et al. [27] put forward the case for using CBR in connection with timetabling. Whilst there has been very little work on CBR in timetabling, the method has been applied to other scheduling problems in a few cases as discussed in [27]. In [79], MacCarthy and Jou discuss CBR in connection with general scheduling problems and review CBR systems which currently deal with these problems. They point out that so far expert scheduling systems have been fairly unsuccessful, partly because domain knowledge is not easy to acquire in scheduling problems and also because it is difficult to find good heuristic approximation techniques which work well on a range of different scheduling problem types.

MacCarthy and Jou describe the problems encountered in attempting to develop a CBR system for planning and scheduling of large and complex airlift operations by Koton [74]: “Problems arise due to the failure of the matching

process to produce a single good match." He also mentions Bezirgan's [6] discussion on the CBS-1 system for dynamic job-shop scheduling using CBR. Bezirgan comments that the dispatching rules provided are only elementary and this may result in infeasibility and the storage of the case base requires a large amount of memory.

Miyashita and Sycara [84] used CBR for "interactive schedule repair" for a job shop scheduling approach using constraint directed research. MacCarthy and Jou [79] give a summary of the system which aims to help the human scheduler in finding good solutions. The case base has a number of purposes:

- solution generation
- evaluation
- failure avoidance and recovery
- failure explanation

The indexing of cases is done based on abstractions of domain relations and constraints. Matching of cases is done using a nearest-neighbour method with each matched case being given a score and the best one used. Evaluations carried out show that the system works well when tested against a pure constraint-based approach.

MacCarthy and Jou [79] believe that CBR has a lot to offer in expert scheduling systems and cite a number of reasons:

- Knowledge of relaxation and prioritisation strategies in previous cases helps to identify conflicts early in the decision making process.
- The case base can store the degree of satisfaction of goals from the user's perspective and this knowledge can aid current scheduling practice.
- Context dependent features in dynamic scheduling can be handled by CBR.

In a general scheduling problem, there are a number of different areas to which CBR can be applied. As well as being applied directly to the scheduling problem, CBR can be used to choose an algorithm, model or heuristic for the given problem. The main body of this thesis will be concerned with choosing the best algorithm for a given timetabling problem by comparing the problem with those in a case base and finding the best match. Once this has been done, the heuristic or meta-heuristic used to successfully solve the matched problem from the case base can then be used either directly or after adaptation for the current problem. The idea being that an algorithm used for one problem should work well on a *similar* problem. Once the algorithm has been chosen and implemented, the other important part of the process is to provide the feedback on how successful the method was so that the case base can be updated, especially if the method was very unsuccessful so that the indexing can be changed so the same match will not be chosen the next time. The case base can also be used for the purpose of relaxing constraints, prioritising goals and generally re-shaping the problem. This may be done as a result of the earlier matching process which provided an algorithm for a similar problem.

In [79], the authors give details concerning problem structure and case structure for applying CBR to scheduling problems which can be outlined as follows: A case must contain three attributes - *problem instance, action and outcome*. The problem instance is a description of the problem, the action is the method used to solve the problem and generate a schedule and the outcome covers the expected and actual results for the schedule produced. The problem instance may include a lot more detail about allocation of resources, orderings and various other constraints as well as the required information such as data, constraints and goals.

Miyashita [83] also comments that CBR “appears to be a natural method for knowledge acquisition” in scheduling problems due to the interest in capturing user preferences and situation sensitive knowledge. However he also notes that

applying CBR to schedule improvement is a very challenging task. One of the main issues concerns how a case is defined in the domain of schedule optimisation and how these cases should be indexed. One answer he proposes to this is to use the whole schedule as a case with the advantage that the more information is transferred to the new case, the easier it should be to solve.

In [53], Cunningham and Smyth explore the reuse of cases in job scheduling problems. They give descriptions of the two main approaches to case-based scheduling which they have identified in the literature. The first approach is used by Koton's SMARTplan of 1989 [74] in which cases are used to propose preliminary schedules. These are then adapted in order to satisfy the schedule requirements. The second approach uses cases to adapt schedules which have been proposed by other methods.

Cunningham and Smyth consider the first of these approaches in their paper and look at two different modes of reuse. The first being to use single cases as skeletal solutions, the second being to reuse multiple cases as building blocks for a desired solution. The authors feel that the building block approach is more suitable for scheduling because it uses parts of already optimised structures in the new solution. For the two different methods of CBR, there are similarly two different approaches to retrieval. The first works to select the single best case which is to be used as a template for the target solution, the second approach gathers a collection of solution sequences from various cases, each of which can be used to produce part of the target schedule.

The system was evaluated with respect to two main issues. One of these concerned the solution time and solution quality, the other was to see how the CBR techniques perform as the size of the target problem is increased. The key points of note were that simulated annealing takes around 30 times longer than the CBR methods on average, but achieves better results. However, in the experiments conducted, the CBR solutions where within 102.5% of the SA so-

lutions and considerably better than those produced by a Myopic method. The authors conclude that CBR techniques can produce good quality schedules, but that schedule adaptation appears to be very problem dependent.

Recent applications of case based reasoning to exam timetabling, which is the focus of this thesis are covered in section 2.2.6, whilst Qu [97] provides a detailed discussion of the application of case based reasoning to timetabling problems. Since this thesis is part of a wider project of applying case based reasoning to meta-heuristic selection for timetabling problems, the focus of this work will be largely on the lower level aspects of the system concerning how to measure similarity between to exam timetabling problems and the development of the meta-heuristic techniques which will form the basis of the case base. Higher level CBR issues regarding the structure of the case base, methods of retrieval and feedback and the knowledge discovery techniques used to train the case base are beyond the scope of this thesis.

Chapter 4

Investigating Similarity Measures For Exam Timetabling Problems

The work presented in this chapter forms the basis of a paper published in the Proceedings of The 1st Multidisciplinary International Conference on Scheduling: Theory and Applications, entitled *Similarity Measures for Exam Timetabling Problems* [16]. The aim of this work is to conduct an initial investigation of some of the main features of benchmark exam timetabling datasets which may be important for measuring similarity between problems. As well as identifying those problem features which are important, a key element of this work is to discover those features which may at first seem important, but are in fact misleading as indicators of problem difficulty and structure. This work will feed into more detailed research into similarity measures between exam timetabling problems presented in Chapter 6 which will form an important element of the final case based reasoning (CBR) system. The underlying assumption in the CBR system for this project is that *similar* problems can be solved equally successfully using the same method therefore it is crucial to conduct an in-depth study of the problem structure to identify the key features which will define the term *similar*.

4.1 The need for a similarity measure

One of the next major areas for CBR is to work on the level of a hyper-heuristic. This would select, from a range of previously used heuristics or meta-heuristics, the *best* one to solve a new problem given to the system. The key issue for such a system is how we define the *best* algorithm to be used for the new problem. Applying the general theory behind CBR, we assume that an algorithm which performs well on one problem will also perform well on a *similar* problem. Therefore the main area of consideration is how two problems can be measured as *similar* in such a way that this reasoning holds.

Each case in our case base will consist of a problem definition together with one or more algorithms used to successfully solve the problem and an indication of their level of success. A standard format for defining exam timetabling problems has been developed for use in the system to enable matching of any two problems to measure their similarity. Once a new problem is presented to the system, the matching process will retrieve the most similar problem from the case base along with the most successful algorithm(s) used to solve that problem. Based on how similar the retrieved case is to the new one, the retrieved algorithm may be adapted by tuning its parameters in some way or by using a hybrid of more than one retrieved algorithm.

The development of such a system provides a large number of research areas. Of these, the biggest is the definition of *similarity* which is the one considered here. In this chapter I consider some of the key elements in the definition of an exam timetabling problem and work towards a definition for *similarity* based on which features seem to have the biggest effect on how successful an optimisation algorithm is. Of course, for our purposes, two *similar* timetabling problems would mean that the same algorithm would be suitable for solving both problems. Two problems would be dissimilar if a particular algorithm/heuristic worked very

well on one problem but not on the other. Essentially this means that two similar problems should have a similar problem landscape as seen from the point of view of the algorithm operating on these landscapes.

As discussed in section 1.2, the two main factors which define how a given local search technique traverses the solution landscape are the neighbourhood definition and the move acceptance criteria. For the work presented in this chapter, the simple move neighbourhood defined by relocating a single exam to a new feasible timeslot is considered with a simple simulated annealing (SA) algorithm used for testing purposes.

Given the nature of the domain in question, there are a countably infinite number of simple and more complex statistical measures which could be used to compare the features of two given exam timetabling problems. Many of these will actually have very little impact on the success of a given algorithm on the problem, whereas others could be major factors in how well the algorithm navigates the search space to find a good solution. My aim here is to study the structure of a number of benchmark data sets to examine the contribution of some of the more likely problem descriptors to problem *difficulty*

The main purpose of the experiments reported in this chapter is to eliminate the features in the problem definition which have no effect on the results of the SA algorithm and to see how stable the technique is regarding other features - for instance, how large a change in a particular feature is needed to have a significant effect on the quality of solution produced.

Due to the complex nature of the problem landscapes involved it is not possible to produce any concrete, quantitative results regarding exactly which parts of the problem definition have what effect on the algorithm performance. Instead, I aim to acquire as much qualitative information as possible to enable value judgements on which features to include in a similarity measure and what type of tolerance to allow for each individual feature in order for two cases to be

considered similar based on that feature. It is expected that whilst a large difference between problems with respect to some features will have relatively little impact on their similarity, other features will be far less robust with even a small difference causing a big change in the structure of the problem landscape.

A largest degree graph colouring heuristic with backtracking is used to provide an initial feasible solution for the SA algorithm which will then only explore the space of feasible solutions. The SA algorithm used for these experiments selects both exam, e, and period, p, at random, checking whether moving exam e to period p is a legal move[1]. If not, a maximum of nine more periods are tested to find a feasible move, otherwise a new exam is randomly chosen and the process repeated. This move is then accepted or rejected using the standard probabilistic acceptance criteria of simulated annealing (see e.g. [67]) with improving moves always accepted and worse moves accepted with decreasing probability based on the geometric cooling schedule. The starting temperature and cooling schedule are initially chosen arbitrarily and tuned based on results.

4.2 Data sets

All tests have been carried out on Carter et al's benchmark data sets [45] without any side constraints other than to spread conflicting exams around the timetable. The objective function used by Carter et al is based only on the sum of proximity costs as defined below:

$$w_s := \frac{32}{2^s},\ s \in \{1, \ldots, 5\}$$

where w_s is the penalty cost for a student taking two exams scheduled s periods apart.

[1] A move which retains the feasibility of the overall timetable

Data Set	No. of exams	No. of students	No. of enrolments	Graph Density	No. of periods	Exams per period
CAR-S-91	682	16925	56877	0.13	35	19.5
CAR-F-92	543	18419	55522	0.14	32	17.0
EAR-F-83	190	1125	8109	0.27	24	7.5
HEC-S-92	81	2823	10632	0.42	18	4.5
KFU-S-93	461	5349	25113	0.06	20	24.3
LSE-F-91	381	2726	10918	0.06	18	21.2
STA-F-83	139	611	5751	0.14	13	10.7
TRE-S-92	261	4360	14901	0.18	23	11.4
UTA-S-92	622	21267	58979	0.13	35	17.8
UTE-S-92	184	2750	11793	0.08	10	18.4
YOR-F-83	181	941	6034	0.29	21	9.0

Table 4.1: Simple features for benchmark data sets

The most obvious problem features are presented in table 4.1, although many of these are likely to have only a small effect on the performance of optimisation algorithms. As well as those shown, there are a large number of statistical measures which can be applied to the data sets in order to determine the distributions of exams and enrolments amongst students. Table 4.2 shows that there is a wide variation in the average and maximum number of enrolments per student and per exam between the different data sets. A statistical analysis of these enrolments should give an idea as to how well spread they are away from the average and whether there are any significant anomalies within any of the data sets. For the purposes of this research, only the mean and modal averages and standard deviation were calculated for enrolments per student and are included in table 4.2.

In the rest of this chapter I look in more detail at the student enrolments themselves and examine the effect they have on the structure of the problem. I also consider how big an impact the objective function has on these measures of similarity.

Data Set	enrolments per student (number of students at max/mode)			% students with modal enrolment	Standard Deviation	enrolments per exam	
	mean	max (num)	mode (num)			average	max
CAR-S-91	3.36	9 (1)	5 (4569)	27%	1.56	83.40	1385
CAR-S-92	3.01	7 (29)	4 (4168)	23%	1.46	102.25	1566
EAR-F-83	7.21	10 (9)	8 (409)	36%	1.20	44.80	232
HEC-S-92	3.77	7 (1)	5 (1071)	38%	1.43	131.26	634
KFU-S-93	4.70	8 (11)	5 (2515)	47%	1.36	51.67	1280
LSE-F-91	4.01	8 (3)	4 (1638)	60%	0.99	28.66	382
STA-F-83	9.41	11 (209)	9 (239)	39%	1.22	41.37	237
TRE-S-92	3.42	6 (20)	5 (1214)	28%	1.41	57.09	407
UTA-S-92	2.77	7 (23)	4 (4026)	19%	1.50	94.82	1314
UTE-S-92	4.29	6 (20)	5 (1503)	55%	1.01	64.09	482
YOR-F-83	6.41	14 (1)	8 (372)	40%	1.80	31.76	175

Table 4.2: Features for benchmark data sets based on students and enrolments

4.3 Analysis of Data Sets

Initial plans for starting the analysis of the data sets centred around making some small controlled changes to the existing data sets and running the same algorithm on the original set and the new set to examine the effects these small changes have on the running of the algorithm. From this I hoped to draw some conclusions as to whether the change in question had a major effect on the algorithm and if so, how big a change from the original data set was needed to observe this change in algorithm performance. In order to do this though, it is required to examine the impact of these changes on all the various problem descriptors – for instance, what effect would removing 10 students from a given data set have on enrolments and the conflict matrix[2]?

4.3.1 Removing redundancy from Data Sets

The input files for the data sets consist of groups of $(student, exam)$ pairs representing one enrolment for a given student. Further enrolments for the same student are recorded in the same fashion on following lines. It was observed that in many of the data sets there were a significant number of students who

[2]The matrix of exams denoting which exams have students in common and therefore clash with each other

Data Set	Students	Total enrolment	Average enrolment	Standard Deviation	SA Result
CAR-S-91	16925	56877	3.36	1.57	6.80
CAR-S-91_minus	13516	53468	3.96	1.15	6.80
TRE-S-92	4360	14901	3.42	1.41	11.35
TRE-S-92_minus	3693	14234	3.85	1.15	11.35
HEC-S-92	2823	10632	3.77	1.44	15.45
HEC-S-92_minus	2502	10311	4.12	1.11	15.45

Table 4.3: Results of simulated annealing applied to three benchmark data sets with all single enrolment students removed

had only one enrolment. These students have no impact on the difficulty of the exam timetabling problem or its landscape since they are involved in no clashes and therefore they do not have any effect on the end result – they can sit their one exam equally easily whenever it is scheduled since capacity constraints are not included. The first *new* data sets were then produced by removing all those students with a single enrolment from the problem and the simulated annealing algorithm run on this reduced student set. Table 4.3 shows some of the key statistics of these data sets for three of the benchmark problems and the results confirm that removing these students has no impact on the final result, as calculated by dividing the total penalty for the timetable by the number of students[3]. The reduced data sets are denoted by adding '_minus' to the original data set in the table.

The main points to note from the results in Table 4.3 are that, whilst removing a relatively large number of students from the problem (20% in the case of the CAR-S-91 data set) there is no effect at all on the final timetable produced by SA. [4] However, it does have a fairly noticeable effect on many of the other measures which are put forward as being possible factors in a similarity measure – many of which would be statistical measures based on the number of students

[3]The results for the reduced sets are calculated by dividing by the number of students in the equivalent full set to demonstrate that they can be removed from the data set without changing the problem

[4]This result is independent of the type of algorithm used

or enrolments and ratios involving these two factors. From the point of view of the algorithm operating on the problems, the CAR-S-91 data set and its reduced CAR-S-91_minus data set are identical and should therefore be regarded as *similar* from a case based reasoning 'heuristic selector' perspective. This result shows that a fairly detailed analysis of the data sets and what the students in these data sets actually add to the overall problem is necessary before considering any measures of similarity between two data sets.

On the face of it, using the measures shown in Table 4.3, the reduced data sets are not very similar to their equivalent complete sets at all, when comparing the students, enrolments, average enrolment and the standard deviation of the enrolments, yet as has been shown, the reduced sets should be considered identical to their complete sets from the point of view of selecting an algorithm to solve the problems. Therefore, if these factors are still to be considered when investigating measures of similarity, the data sets must be reduced to their minimum definition by removing any redundancy before comparing them for similarity. For example, if CAR-S-91_minus formed part of a case[5] within our case base and the CAR-S-91 data set were input to the system to find a match, it would be pre-processed before the matching process to remove the single enrolment students, then the CAR-S-91_minus case would be retrieved as an exact match and its corresponding algorithm used to solve the CAR-S-91 set.

This result prompted a variety of further research questions which needed to be examined further. Amongst these were the questions of how a 'good' timetable is defined and whether there is any more redundancy which can be removed from the data sets before the matching process.

One measure for reporting results for the benchmark data sets is *average penalty per student* which is calculated by taking the overall penalty for the whole timetable, defined by the objective function, and dividing by the number

[5]The best algorithm found to solve this problem forming the other part of the case

of students in the data set. However, it can be argued that since 3409 students in the CAR-S-91 data set are only taking one exam and therefore add no penalty to the overall timetable, that these students should not be included in the *average penalty per student* measure. In an extreme case, a data set could contain 50 per cent of students who take only one exam - in this case, dividing the total penalty for the timetable by all the students would give a result twice as good as if you ignore half of the students who add nothing to the "difficulty" of the problem - yet the timetable itself would be identical in both cases. In itself this is not a major concern since the results produced for these data sets are only used for the purposes of comparing different techniques against each other rather than comparing absolute results across data sets. These observations did, however lead to further research into the issue of more redundancy in the data sets.

4.3.2 Examining subsets of the Data Sets

Having removed all the single enrolment students from the data sets and witnessed the impact this had on the results and potential similarity measures, I looked for any further redundancy within the problem definition for these sets. The next obvious area to investigate was the issue of repeat students, i.e. two or more students with the exact same enrolments. These are very common in real life problems given that students on the same course tend to have many or all exams in common. In addition to exact repeat students, there are also students whose enrolments form a subset of one or more other students. All such duplicate students were removed from the data set including any students whose clashes had already been recorded by earlier students. For example, if a student with enrolments a, b and c is followed by three students with enrolments (a, b), (a, c) and (b, c) respectively, the last three students can all be discarded from the data set since their clashes were recorded by the first student.

Data Set	Total Students	Base Set	Singleton Set	Weights Set
CAR-S-91	16925	8194 (48%)	3409 (20%)	5322 (31%)
CAR-F-92	18419	6195 (34%)	3969 (22%)	8255 (45%)
EAR-F-83	1125	754 (67%)	1 (0%)	370 (33%)
HEC-S-92	2823	444 (16%)	321 (11%)	2058 (73%)
KFU-S-93	5349	1367 (26%)	276 (5%)	3706 (69%)
LSE-F-91	2726	1253 (46%)	99 (4%)	1374 (50%)
STA-F-83	611	150 (25%)	0 (0%)	461 (75%)
TRE-S-92	4360	1924 (44%)	667 (15%)	1769 (41%)
UTA-S-92	21267	7946 (37%)	6181 (29%)	7140 (34%)
UTE-S-92	2750	392 (14%)	79 (3%)	2279 (83%)
YOR-F-83	941	670 (71%)	1 (0%)	270 (29%)

Table 4.4: Percentages and numbers of students in the Base Set, Singleton Set and Weights Set for benchmark data sets

In removing these *duplicate* students, it was noted that their absence would have an impact on the final penalty for the timetable, unlike the removal of the single enrolment students. The reason for this being that the objective function used weights all clashes by the number of students involved in the clash so that clashes with a large number of students are a higher priority to spread well apart than those with only a few students. Despite this, it was still considered worthwhile to remove them to examine the remaining set. The justification for this being that while duplicate students do contribute to the problem definition, they only do so relative to the objective function used to define a good timetable. In terms of the hard constraints they do not alter the problem and the effect of the objective function on measuring the similarity of problems can also be considered now. The results obtained are shown in Table 4.4.

The *singleton set* is the set of single enrolment students removed initially, the *weights set* is the set of duplicate students (as defined above) whilst the *base set* is the set of students remaining after all students in both the singleton set and the weights set are removed. The base set is a (smaller) set of students which define the conflict matrix since all students removed to the other two sets did not add to the conflict matrix. Hence, this set of students and their conflicts defines the

set of feasible solutions to the larger problem whilst the weights set is so named because the students in that set simply add weights to the already existing clashes thus shifting the focus of the optimisation.

One major point to note from the three sets is that whilst the singleton set can be taken on its own, the base set and the weights set are less easy to split in a meaningful way since the base set does still include a number of weights on various edges. This is due to the fact that only students whose entire set of clashes had already been noted were removed to the weights set, leaving those who have only some of their clashes already noted to be included in the base set as their remaining clashes add new information to the set. If any single student is removed from the base set, the hard constraints on the problem will change because one or more clashes will be lost. Removing students from the weights set, whilst keeping the rest for the problem would retain the same basic problem definition, but will change the weightings and therefore the bias of the clashes.

Using the objective function given by Carter et al [45] it is difficult to use the information discovered by splitting the data sets up in this way due to the above mentioned weights included in the base set. However, this splitting up of the data sets into subsets looked promising as a potentially important factor in comparing different data sets since the percentage of students forming the base set varies greatly between data sets - for instance, the fact that only 14 per cent of the 2750 students in the UTE-S-92 data set are required to define the problem in terms of feasibility compared to 71 per cent of the 941 YOR-F-83 students is a significant difference.

As a result of this I considered another objective function to define a *good* timetable. From the point of view of a student taking the exams, their criteria for how important particular clashes are could include a variety of individual reasons based on how difficult they find certain exams to be, but none of these can be taken into account in the overall timetable since they are individual pref-

erences. However, the number of other students involved in a particular clash is completely unimportant to any individual student. Whether they are the only student doing two particular exams or whether there are 100 other students also taking those two exams does not matter to them and therefore weighting clashes by the number of students involved in the clash gives a bias in the timetable towards those pairs of exams with many students in common. These exams may often be easier ones from the point of view of the student having to revise for and sit the exam. Conversely, exams with relatively few students may be more specialised and difficult from a student's perspective - students would need more intensive revision for such exams if they are scheduled close together. Weighting the former far more than the latter would result in the 'easier' exams being spread well apart for lower overall penalty whilst the 'harder' exams for the students may be relatively close together since their total penalty in the timetable is small.

Using this justification I decided that removing all student weightings from the objective function and weighting clashes purely by the number of periods apart in the timetable would produce what could be considered a "fair" timetable from an individual student's point of view and would vastly simplify the problem from the point of view of the data sets, meaning that the weights set could be discarded to concentrate purely on the base sets of students which define the conflict matrix[6].

Experiments carried out on the whole data set and on just the base set of each data set using such an objective function with no student weightings showed, as expected, that the results produced (the total penalty for the timetable as defined by the new objective function) were identical for the full set of students and for the base set. The conclusion to be drawn from this is that the definition of a

[6]The weightings included in the base set would also be removed implicitly by the new objective function

good timetable can have a huge impact on the relevant statistics of a given data set. Using the new definition of *good*, the 2750 student UTE-S-92 problem is identical to the 392 student UTE-S-92 base set problem and therefore, should be matched as such by the similarity measure.

Further thoughts on the issue of base sets containing weights on various edges led me to consider further how useful these could actually be in measuring similarity between problems. Whilst each student enrolment set produces exactly one conflict matrix, the reverse is not the case. In fact, if we allow for single enrolment students, there are an infinite number of student enrolment sets which will each produce the same conflict matrix (and therefore define the same core problem, when room capacity constraints are not included). Even without single enrolment students, there are many different student enrolment sets of vastly differing sizes which still produce the same conflict matrix. Of particular note, the maximum base set to define a given conflict matrix could be that obtained by creating a new student for every clash in the conflict matrix (if student weights are included then multiple identical students would be generated for each clash) - these students would all have exactly two enrolments and uniquely define one clash each. Clearly this set of students forms a base set since no student could be removed without altering the conflict matrix.

Of more interest is the concept of *minimum* base set to define a conflict matrix. Theoretically, the original set of students could be removed and replaced by the smallest possible set of students to define the conflict matrix. This could be attempted by finding the largest clique in the conflict matrix, assigning all exams in the clique to a new student and removing all those edges from the graph. Then continue in this way until all clashes have been assigned to a student.[7] It is likely that an investigation of the sizes and number of cliques within each data set as

[7] In terms of student numbers, I have not proved that this method would create the absolute minimum base set, but it would produce at least a close approximation to the minimum, if not actually the minimum

well as their interconnectivity could form an important area of further research into similarity between problems. Carter and Johnson [42] examine the cliques and near-cliques of the data sets considered in this chapter in some detail. The results they present give some important indications regarding aspects of graph density and frequency of cliques which could prove extremely useful for our consideration of a similarity measure.

It should be noted, however, that if constraints of the type "no student should have x exams in y periods" are included to be minimised, the set of actual student enrolments becomes far more important than to just define the conflict matrix. As pointed out by Carter & Laporte [43], simply using the student enrolments to define a conflict matrix loses the potentially important information required to deal with constraints on the actual students themselves rather than on the exams. This does not affect the splitting of the actual student sets into base set, weights set and singleton set as all the student enrolment information is still recorded by the students in the base set. Considering different student sets which define the same conflict matrix is purely for the purposes of considering students as a feature in a similarity measure. For the benchmark problems considered in this chapter and throughout this thesis, constraints of this type are not included, only the constraint of spreading clashing exams around the timetable.

4.4 Conclusions

To summarise the main conclusions from the work in this chapter, in order to use any of the statistical measures produced for a given data set as a measure of similarity within our CBR system, we must first strip away any redundancy from the data based on the particular objective function used. Depending on the objective function, some features may be relatively unimportant and including these in a similarity measure could result in two problems which are in fact

very similar being regarded as totally different. In the case of student numbers, it is considered that only the minimum base set to define the conflict matrix would be a worthwhile similarity measure when room capacity constraints are not included. Experiments have shown that, depending on how you define a *good* timetable, the same data set can be split up and stripped down into very different looking data sets, but which are actually identical for the purposes of running an algorithm to find the best results.

In Chapter 6, a more in-depth analysis is presented of the various features to be considered for a similarity measure, although the aim of my investigations into problem features is to suggest potentially important features, rather than to decide on the exact features to include in a similarity measure. It is most likely that ratios of some of the simple features presented here, some of which may not be immediately intuitive will be most important in a similarity measure. The features suggested in Chapter 6 together with all ratios between pairs will be fed into knowledge discovery techniques to determine the actual set of features to be used within the case base. In Chapter 5, the impact of the objective function on the performance of the simulated annealing technique will be examined further.

Chapter 5

Using Simulated Annealing to Study the Behaviour of Exam Timetabling Data Sets

The work presented in this chapter is included in a paper published in the Proceedings of the Fifth Meta-heuristics International Conference (MIC 2003), entitled *Using Simulated Annealing to Study Behaviour of Exam Timetabling Data Sets* [17]. This work builds upon the results from Chapter 4 by further considering the impact of the objective function on the behaviour of a set of benchmark data sets when optimised using simulated annealing. This is done by considering two different objective functions based on the same constraints, but with differing emphasis with the performance of the SA method compared across datasets and across objective functions. The importance of the initial solution used to seed the simulated annealing algorithm is also investigated.

5.1 Initialisation heuristics

In section 2.2.2, I reviewed a variety of graph-colouring heuristics for solution construction which can easily be adapted to exam timetabling problems. Many of these techniques are used for constructing an initial solution to seed a local search method as discussed in section 2.2.3 with the main criteria for this being that the construction method used is fast so that computational time can be spent on the local search. By themselves, fast graph heuristic construction techniques cannot produce solutions which match those of more sophisticated meta-heuristics on a majority of problems, therefore in this thesis the focus is on developing a case base consisting of mainly local search meta-heuristics, from which the best will be selected for a given problem. However, this does leave the question of how to initialise each of these different techniques. A range of different construction techniques to be selected from could be developed with the best combination of construction technique and local search method being stored for each problem. Alternatively, a standard initial solution generator could be used to seed all local search meta-heuristics equally. Also the question of whether these initial solutions must be feasible or whether the local search techniques should search for feasibility must be considered.

In this work, I choose to use the largest degree (LD) heuristic (defined in section 2.2.2) to order exams as this is a static ordering which is capable of producing high quality results. Carter et al [45] found that the dynamic ordering, saturation degree (SD), was the most robust of the heuristic orderings tested, however largest degree often outperformed it based on solution quality. Also, being a dynamic heuristic, saturation degree tends to take more computational effort than largest degree as the exam ordering must be recalculated after each step of the solution construction. Both techniques require some backtracking in order to produce feasible solutions to some problems, although saturation degree

tends to require fewer backtracks due to the fact that it is calculated to schedule exams next which have the fewest available timeslots. For the work in this chapter and indeed, for ensuing work, I decided that it would be desirable for the local search techniques to be seeded with a feasible solution and only explore the search space of feasible solutions to enable them to concentrate on minimising the violation of soft constraints. Therefore a backtracking method with randomisation is employed with the largest degree heuristic in order to produce feasible solutions. If the next exam in the ordering cannot be scheduled, some exams already assigned are moved to different timeslots to create a feasible slot for the new exam. This technique generally takes fewer than 10 seconds to produce a feasible solution for any of the benchmark data sets considered.

The other main decision area for a construction heuristic is the choice of time slot to assign the next exam in the ordering to. In this chapter I consider two different approaches to this to compare their effect on the results produced by simulated annealing. The first is a *greedy* method which chooses the slot with lowest cost as measured by the objective function, the second method chooses a feasible slot at random to assign the next exam to. The greedy approach, as expected, produces higher quality initial solutions and with a lower variation in quality, whereas the random slot assignment technique creates far more diverse solutions with a lower average quality. The randomness in the greedy technique to produce different solutions each run comes from the backtracking algorithm which chooses exams to re-assign with a degree of randomness.

It is well known that some local search techniques are far more reliant on a good initial solution than others, therefore it is likely the results of the experiments in this chapter will provide valuable analysis as to which initialisation technique is favoured by simulated annealing. Clearly the greedy approach is most likely to yield better results as the initialisation already takes into account the objective function in the initial exam placings, unlike the random initialisa-

tion. However, this may not be the case for all local search techniques. If the neighbourhood of the local search method ensures that the whole solution space is collected, then the initial bad placing of a number of exams can be overcome. However, the simple move neighbourhood used initially with the simulated annealing algorithm does not have this property since it only allows a single exam to move to a new feasible slot - any exams which clash with exams in every other period are likely to remain immobile throughout the search process meaning that their initial position in the timetable and relative to each other is crucial. A neighbourhood which keeps the whole search space connected is therefore a clear advantage, although such neighbourhoods tend to be more complex and computationally expensive.

5.2 Experiments using a LD heuristic with randomisation

The first set of experiments conducted uses the LD heuristic with random time slot assignment. For each data set, 10 runs of simulated annealing were conducted on each of two objective functions for each data set. The two objective functions used are those considered in Chapter 4, first, the standard objective function for the benchmark data as given by Carter et al [45] with clashes between exams weighted by the number of students involved multiplied by the proximity cost, secondly the simplified objective function which assigned a weight of 1 to all clashes equally with only the proximity costs defining the solution quality. The simulated annealing (SA) algorithm is also the same as that used in Chapter 4.

In Tables 5.1 & 5.2, the same initial solution is used for all 20 runs of SA for each data set. The first 10 runs (Table 5.1) were optimised using the original ob-

Data Set	Initial Solution	Percentage Improvement	Standard Deviation	Average Final Solution
CAR-S-91	160224	22.2%	1.02%	124648
CAR-F-92	141506	24.8%	1.43%	106367
EAR-F-83	63933	11.1%	1.49%	56846
HEC-S-92	53598	22.9%	1.81%	41308
KFU-S-93	190237	47.0%	2.22%	100790
LSE-F-91	56393	20.0%	0.95%	45125
STA-F-83	101100	3.4%	0.68%	97701
TRE-S-92	59256	15.7%	1.54%	49944
UTA-S-92	141809	18.3%	1.02%	115827
UTE-S-92	116735	31.8%	2.44%	79587
YOR-F-83	49698	9.3%	1.04%	45073

Table 5.1: Results from SA initialised by the same solution each time, using the standard objective function

jective function including student weights for clashes (the Weighted Set), whilst the second set of 10 runs (Table 5.2) was optimised using the simplified function. The values given for Initial Solution in Tables 5.1 & 5.2 for a given data set represent the exact same solution, but measured by the two different functions.

Tables 5.3 & 5.4 show results when a different initial solution is fed to SA for each of the 20 runs across the two objective functions. Table 5.3 using the standard objective function, Table 5.4 using the simplified function.

Due to space constraints, some words in Tables 5.3 & 5.4 were abbreviated as follows: Avg. = Average, Init. = Initial, Soln. = Solution, Imp. = Improvement, Std Dev = Standard Deviation.

Of course, 10 runs for each data set and optimisation function is not statistically representative. However, in this initial phase of the research work the aim was to provide a guideline for further research with more runs in further experiments - on certain data sets a separate 100 runs were conducted to check that the results from the initial 10 runs give a relatively accurate representation. Also, it was felt that someone using the CBR system to find a solution to their problem may not wish to perform 100 runs of the retrieved technique, therefore 10 runs

Data Set	Initial Solution	Percentage Improvement	Standard Deviation	Average Final Solution
CAR-S-91	51091	20.0%	0.70%	40873
CAR-F-92	38634	21.6%	0.90%	30289
EAR-F-83	11316	12.3%	1.20%	9924
HEC-S-92	4329	6.5%	1.10%	4048
KFU-S-93	17616	23.8%	0.70%	13423
LSE-F-91	14613	29.5%	1.00%	10302
STA-F-83	6193	25.3%	*6.80*%	4626
TRE-S-92	16194	19.9%	1.30%	12971
UTA-S-92	41331	25.2%	0.90%	30916
UTE-S-92	8366	22.0%	2.60%	6525
YOR-F-83	13392	10.7%	1.20%	11959

Table 5.2: Results from SA initialised by the same solution each time, using the simplified objective function

Data Set	Avg. Init. Soln.	Std Dev	% Imp.	Std Dev	Avg. Final Soln.	Std Dev
CAR-S-91	152278	3.33%	24.1%	2.22%	115633	3.94%
CAR-F-92	141332	2.95%	23.5%	3.66%	108023	3.09%
EAR-F-83	65586	4.57%	15.0%	3.32%	55676	3.58%
HEC-S-92	55412	*8.27*%	20.5%	*10.13*%	43730	3.21%
KFU-S-93	161601	*10.77*%	36.1%	*11.85*%	102115	3.89%
LSE-F-91	57986	3.60%	25.2%	4.31%	43362	4.87%
STA-F-83	106717	2.89%	6.40%	1.45%	99887	2.71%
TRE-S-92	58615	3.95%	13.1%	1.71%	50953	3.77%
UTA-S-92	128946	4.06%	20.9%	2.60%	102074	5.61%
UTE-S-92	116636	*10.50*%	24.1%	*6.29*%	88041	4.64%
YOR-F-83	50142	2.03%	12.0%	2.98%	44109	3.12%

Table 5.3: Results from SA initialised by a different solution each time, using the standard objective function

Data Set	Avg. Init. Soln.	Std Dev	% Imp.	Std Dev	Avg. Final Soln.	Std Dev
CAR-S-91	51589	1.30%	20.7%	1.25%	40908	1.23%
CAR-F-92	38339	1.32%	21.5%	0.93%	30094	1.21%
EAR-F-83	11794	2.21%	14.5%	2.20%	10084	1.32%
HEC-S-92	4425	3.19%	7.0%	3.34%	4114	2.94%
KFU-S-93	17355	1.98%	25.4%	1.62%	12939	1.63%
LSE-F-91	14959	2.02%	30.9%	2.30%	10332	1.50%
STA-F-83	6025	4.18%	18.3%	*8.55*%	4917	*8.95*%
TRE-S-92	15749	1.87%	18.9%	3.02%	12775	1.73%
UTA-S-92	41770	1.25%	25.4%	1.81%	31145	0.90%
UTE-S-92	8039	5.19%	25.4%	5.08%	5997	*7.02*%
YOR-F-83	13270	1.80%	10.8%	2.68%	11832	1.92%

Table 5.4: Results from SA initialised by a different solution each time, using the simplified objective function

gives a more realistic idea of behaviour. The obtained results do provide some interesting points for deeper study since the standard deviations will still be of the same order for a larger number of runs. The results presented for standard deviation are calculated as a percentage of the average for a given set of results.

It is also worth noting that the initial solution used for each data set in the first set of experiments may be either good or bad so any absolute analysis of the percentage improvement from this solution is not worthwhile. This was a deliberate side effect of using random slot assignment for the initial solution heuristic. The second set of experiments from 10 different initial solutions each time aims to overcome any bias from a bad or good initial solution and give some indication as to the variation of initial solution quality also.

From the results presented in Tables 5.1 & 5.2, it can be seen that in all but one case, the SA meta-heuristic improves from the same initial solution 10 times to within a standard deviation of just 3% from the average, indicating that the data sets are relatively stable with respect to SA, i.e. when seeded with the same initial solution, the results produced are consistent over 10 runs without

huge variations. Further tests on two of the data sets, each run 100 times from the same initial solution show the same level of consistency, indicating a large degree of dependency of the SA meta-heuristic on the initial solution. The one exception to this is the STA-F-83 data set when optimised using the simplified function eliminating student weightings which gives a standard deviation of over six percent. Referring to Table 5.4, it can be seen that this behaviour is still apparent when 10 different initial solutions are used, whereas with the standard objective function, Table 5.3 confirms that this same data set is at least as stable as the rest.

This suggests that the objective function used to optimise the data set (i.e. the definition of what a good timetable is) can have a major effect on the consistency of results produced by the algorithm, indicating that this particular data set is similar in behaviour to others when using one function, but very different when using a different function. It can also be noted from Table 5.4 that this data set shows very different behaviour from the rest when considering the variation of initial solutions vs final solutions. In most cases the standard deviation of the initial solutions and final solutions are fairly similar, yet for the STA-F-83 data set, the SA meta-heuristic introduces a large amount of further variation into the final solution than was present in the 10 initial solutions. This is largely due to the unusual structure of this data set compared to others, each student takes an average of 9-10 exams each and there are a large number of maximum size cliques in the STA-F-83 data set as will be discussed further in Chapter 6.

From Table 5.3, it can be seen that when a variety of random initial solutions are used to seed SA, the HEC-S-92 and KFU-S-93 data sets suddenly start to produce a much wider range of final solutions with standard deviations of the order of 10 per cent. In the case of the KFU-S-93 data set, it can be seen that the initial solution used in Table 5.1 was very bad relative to the average initial solution used in Table 5.3. This, together with the high deviation in initial solutions

for this data set helps to explain why the percentage improvements are so varied. Likewise the HEC-S-92 and UTE-S-92 data sets show a wider range of initial solutions leading to a wider range of percentage improvements. A further set of 100 runs on the KFU-S-93 and HEC-S-92 data sets confirm this variation with KFU-S-93 producing both initial solutions and percentage improvements with standard deviations of over 10 per cent from the average. Despite this though, the average final solutions produced in Table 5.3 show similar deviations over all data sets. This indicates that although the initial solutions for some data sets show a higher variation, simulated annealing flattens this out when producing the final solutions by improving the worse initial solutions notably more than the better ones.

Analysis of the individual runs for these data sets shows that in many cases some of the best final solutions come from the worst initial solutions. This implies that how much the initial solution affects the quality of final solutions produced by SA depends more on the initial placing of certain *key* solution elements rather than necessarily giving a lower cost by the objective function. For other data sets though this is markedly not the case and the better final solutions generally come from the better initial solutions indicating a much stronger dependence on having a good initial solution for these data sets in order for SA to perform well.

Finally, when comparing results across the two objective functions it can be seen that there are some very striking differences between certain data sets. Most notably, the HEC-S-92 and KFU-S-93 data sets are improved a great deal more from all the initial solutions when optimised using the standard objective function than when optimised using the simplified function. On the other hand, the LSE-F-91, STA-F-83 and TRE-S-92 data sets show completely the opposite behaviour and are improved more from an initial solution using the simplified objective function than the standard objective function. This again indicates that

when measuring similarity between two data sets, the objective function used has a major impact.

5.3 Experiments using a greedy LD heuristic

The results presented in Tables 5.5 & 5.6 were obtained over 100 runs using the greedy initialisation technique which produces far better initial solutions as measured by the objective function than the above method, but in doing so takes away a lot of diversity in initial solutions. As can be seen from the final solutions produced when using this greedy technique to initialise SA, the results on all data sets bar one are notably improved from those presented earlier. This confirms my conclusions from Tables 5.1 & 5.2 that SA relies on being seeded with a good initial solution in order to produce better results. The main reason for this being that, using only the simple move neighbourhood which selects an exam and moves it to a new feasible timeslot, many of the exams in the timetable will never move at all due to the fact that they clash with at least one exam in every other period. In the case of the STA-F-83 data set, there are in fact over 37 per cent of the total exams which never move from their initial positions. In other cases a far smaller percentage of exams have this total lack of fluidity, but these exams are still highly significant since they are also the ones adding the highest penalty to the timetable normally.

For this set of experiments, I used a much higher initial temperature than previously which meant that to start, the solutions got much worse, but as the temperature reduced so the cost function dropped back below its initial cost. This produced more competitive results than when I started with a lower temperature which did not accept as many uphill moves at the start. One interesting point of note was that the greedy largest degree heuristic with backtracking (up to two levels deep) failed to obtain feasible solutions to the HEC-S-92 data set so I

Data Set	Avg. Init. Soln.	Std Dev	Avg. % Imp.	Std Dev	Avg. Final Soln.	Std Dev
CAR-S-91	102272	2.05%	10.74%	2.37%	91251	1.57%
CAR-F-92	95882	2.97%	10.69%	2.55%	85590	2.36%
EAR-F-83	50922	2.25%	8.35%	2.97%	46705	1.99%
KFU-S-93	93933	6.6%	8.51%	6.04%	85623	2.30%
LSE-F-91	40236	4.39%	9.59%	3.54%	36355	4.40%
STA-F-83	108598	0.03%	5.71%	0.12%	102397	0.12%
TRE-S-92	46005	3.13%	13.63%	3.40%	39702	2.12%
UTA-S-92	83834	3.47%	6.83%	3.62%	78035	2.39%
UTE-S-92	85960	1.05%	7.08%	2.01%	79863	1.67%
YOR-F-83	43043	1.74%	14.18%	2.36%	36931	1.73%

Table 5.5: Results from SA initialised with a greedy LD heuristic and using the standard objective function

Data Set	Avg. Init. Soln.	Std Dev	Avg. % Imp.	Std Dev	Avg. Final Soln.	Std Dev
CAR-S-91	44171	1.20%	18.22%	1.55%	36120	0.81%
CAR-F-92	32981	0.95%	17.51%	1.07%	27205	0.70%
EAR-F-83	11319	1.48%	21.33%	2.84%	8903	2.48%
KFU-S-93	14560	1.32%	15.12%	0.64%	12358	1.34%
LSE-F-91	11681	1.62%	18.96%	2.14%	9465	1.79%
STA-F-83	6420	1.02%	15.91%	0.74%	5398	1.06%
TRE-S-92	14012	1.23%	19.48%	2.02%	11281	1.40%
UTA-S-92	34173	0.68%	17.11%	1.16%	28325	0.99%
UTE-S-92	6246	1.64%	25.65%	7.28%	4641	6.54%
YOR-F-83	12812	1.27%	20.03%	2.25%	10245	1.83%

Table 5.6: Results from SA initialised with a greedy LD heuristic, using the simplified objective function

had to use the random assignment LD heuristic to initialise SA on that data set. However, despite this, the results produced with the high starting temperature of these experiments were of a high quality comparable to many current state of the art techniques. Results from other data sets were far better than when initialised randomly, but where slightly less competitive with the state of the art techniques than the HEC-S-92 data set. One of the main reasons for this is that, despite having a very high conflict matrix density causing the greedy heuristic to fail, every single exam in the HEC-S-92 data set moved at some point over 100 runs of SA, unlike every other data set, all of which had a number of static exams. This means that the initial solution is less important for this data set as the *fluidity* [1] of exams is higher.

As expected, due to the better initial solution, the overall percentage improvement by SA is generally lower than in the previous experiments. However, in some cases (e.g. YOR-F-83), the better initial solution actually allows the SA meta-heuristic to improve more on average than it did from the worse initial solutions used earlier. For this set of experiments I used a different initial solution in each of the 100 runs since the greedy element of the LD heuristic causes smaller variation in solutions produced over different runs. In particular, for the UTE-S-92 data set, where the random heuristic was producing initial solutions with a standard deviation of 10% from the average, the greedy heuristic produces more consistent results than on most other data sets. Again the STA-F-83 data set shows slightly unusual behaviour and in fact on this data set, the greedy heuristic performs very badly compared to the random one - it produces relatively bad initial solutions and with almost no variation. This also leads to the SA meta-heuristic performing worse when initialised this way than randomly because of the lack of fluidity in this data set. Of all the data sets studied here,

[1] fluidity being the level of movement of the exams within the timetable when simulated annealing is run

STA-F-83 is by far the most dependent on its initial solution with at least 37 per cent of its exams not moving from their initial positions during Simulated Annealing. The KFU-S-93 data set again gives the widest variation in initial solutions, despite these all being far higher quality than those given by the random LD heuristic in Table 5.3. Again, this variation tends to be carried through into the SA phase with 6 per cent standard deviation from average percentage improvement for KFU-S-93 compared to around half this on many other data sets.

The average final solution produced in these 100 runs has a lower deviation than when SA was initialised with the random LD heuristic, but the quality of solutions in all but one case is much higher. This indicates that when initialised with a good solution, SA consistently produces good quality solutions on these data sets, whereas when worse initial solutions are used, not only does the final solution quality suffer, but the consistency of SA is also lessened. However it should be noted that, as discussed in section 1.2 this may well have more to do with the neighbourhood used that the simulated annealing method. If a neighbourhood which allows the search space to remain connected is used then simulated annealing would perform better from a bad solution.

When optimised using the simpler objective function (Table 5.6) excluding any student weights from the edges of the graph, the behaviour of data sets is more similar to that shown in the previous set of experiments. Initial solutions are again better and also slightly less varied, but both the percentage improvement and the final solution of data sets follows the same trend as when SA is initialised randomly. The UTE-S-92 data set again has a far higher deviation than the other data sets. The STA-F-83 data set however behaves completely differently to earlier with very low deviation across final solutions and percentage improvements.

5.4 Conclusions

To summarise the findings of experiments reported in this chapter, it can be seen from the results that there are a number of interesting differences in the behaviour of some of the data sets when compared against each other for the same objective function and also when compared with the same data set optimised using a different objective function. The implications of this are that the objective function can play a major role in whether two problems should be considered similar or not. Clearly it is not possible to measure similarity between two problems with different objective functions, however results indicate that whilst two data sets may behave similarly when optimised using one objective function, they behave completely differently when optimised using a different objective function, even when the constraints are still the same.

When considering the impact of the initial solution on the performance of the simulated annealing algorithm, it was found that seeding SA with a bad initial solution (random initialisation) leads to a bad final solution also when compared to solutions produced by the greedy initialisation approach. This indicates that whilst there may be significant difference between the performance of certain meta-heuristics within a case based reasoning system, it is imperative that these are well initialised otherwise results will be bad irrespective of the algorithm when using the simple move neighbourhood. The issue of how different neighbourhoods affect the reliance of a technique on its initial solution is considered in Chapter 7.

Chapter 6

Analysing Features For Similarity In Examination Timetabling

The work presented in this chapter is based on a paper accepted for the PATAT 2004 (Practice and Theory of Automated Timetabling V) conference in Pittsburgh, August 2004 which is to appear in the conference proceedings, entitled *Analysing Similarity in Examination Timetabling* [18]. This Chapter aims to draw together work from the previous chapters together with further work to give an analysis of main features of exam timetabling problems which will be fed into a knowledge discovery process to determine the features to be used in our case based reasoning heuristic and meta-heuristic selector. The actual features to be used within the system will most likely be ratios of some of the main features presented here and by themselves are relatively unimportant since they will represent just one of many combinations of features each of which give close to optimal performance for the CBR system. What is important here is that the initial features to be fed into the knowledge discovery process are carefully considered and analysed which is the purpose of this work.

6.1 The requirements of a CBR system

As discussed earlier in the thesis, the overall aim of this project is to produce a case-based reasoning (CBR) heuristic/meta-heuristic selector which will intelligently choose, from a variety of techniques, the one best suited to a new problem given to the system. This is done by a matching process which employs a similarity measure between exam timetabling problems. Within the case base, each case is made up of a set of feature-value pairs representing a given problem, together with the heuristic or meta-heuristic technique(s) which give the best results for that problem. When presented with a new problem, its set of feature-value pairs is matched with those of all cases within the case-base and the most similar case(s) will be retrieved. The similarity measure employed will calculate a weighted sum of the difference in value for each feature between the two problems under comparison and is shown in equation 6.1:

$$S(C_x, C_y) = g(\sum_{i=1}^{n} h_i(|fx_i - fy_i|)) \tag{6.1}$$

where:

- n is the number of features in the similarity measure
- fx_i and fy_i are the values of the ith feature of cases C_x and C_y respectively
- $h_i(a) := w_i a^2$; w_i representing the weight of the ith feature
- $g(b) := 1/\sqrt{b+1}$

The similarity, $S(C_x, C_y)$, between two cases will be in the interval $(0, 1]$, with 1 representing two identical cases and results closer to 0 indicating cases with a low degree of similarity.

Clearly the key elements of this similarity function are the features fx_i, fy_i of the two problems being compared together with their weights representing

the importance of each feature to the overall similarity measure. These features will each be represented by a numerical value which can easily be compared to give a difference in value between the same feature (e.g. number of exams) across the two problems. To decide exactly which features should be used in the case-base and how to weight them is a near impossible task to perform purely by hand. Instead, we use knowledge discovery techniques in two stages, as reported by Burke et al. [37]. An initially large set of features and cases is systematically trimmed leaving only those cases which contribute positively to the knowledge of the system and identifying which combinations of features give the best system performance. In [37], the system performance is measured as being the number of successful retrievals of one of the best two heuristics as pre-calculated for a set of training cases and a set of test cases.

Using these knowledge discovery techniques the system can tune the weightings on the features used as well as selecting the best subset of features themselves to improve the performance of the case base. Tabu Search and Hill Climbing techniques are both used within the knowledge discovery in [37] to obtain the best feature vector from the search space of all possible feature vectors. It was found that the best system performance comes from having a relatively small number of features within the case-base (usually between three and seven). Fewer features give too little information from which to accurately measure similarity, whilst too many features reduce system performance by diluting the impact of the most important features. Clearly, identifying those problem features which should be fed into the knowledge discovery process is a key element of this research. The more features included, the larger the search space becomes (when including all ratios of pairs of features), hence it is important that features which are misleading, as discussed in Chapter 4 are excluded and the focus is put onto far stronger features.

In the following section I examine the initial list of features with which we

begin the knowledge discovery process. Ratios between all pairs of features are also included within the features search space as many of these will provide far more meaningful measures of similarity than single features. The knowledge discovery process will return a relatively small list of features considered to be most important, however in this work the aim is to suggest a wide range of features covering as many facets of the problems as possible so as not to miss out potentially key features.

6.2 Qualitative analysis of features used within the CBR system

6.2.1 *Number of Students*

By itself, the number of students within an exam timetabling problem can be extremely misleading as a measure of problem difficulty and as such it needs to be carefully considered before being used as a feature. In Chapter 4, I examined the role of the 'students' in the definition of an exam timetabling problem. It is evident that for real life problems with tight room capacity constraints resulting in a capacity constraint on each period of the timetable, the number of students is a crucial factor. However, it was noted that for the problems considered (no capacity constraints), the number of students was irrelevant. In such cases, the only role of the 'students' is to define the conflict matrix via their enrolments.

Having set up the conflict matrix, an $exams$ x $exams$ size matrix in which pairs of clashing exams are noted by a 1 with all non-clashing pairs being denoted by 0, the students play no further part in the problem and their number is unimportant. Indeed, there are a huge number of student enrolment sets which could define exactly the same conflict matrix - of particular note are the set of students in which every student has just two enrolments and contributes to ex-

actly one edge on the conflict graph[1]. In this case the number of students would be equal to the number of edges in the conflict graph (with weights on edges counting as multiple edges). At the other end of the spectrum would be the minimum set of 'students' which would define the same conflict matrix. This set would be obtained by assigning the largest clique of exams in the conflict graph to a student and then continuing, assigning the remaining largest clique to a new student until all exams and edges are assigned to a student. The relevance of this as a measure for similarity is considered in section 6.2.8.

In those problems where capacity constraints are included, the number of students becomes more important, but only as a ratio to the total capacity available over the period of the timetable and in other ratios concerning enrolments for individual exams. Purely by itself, the number of students in a problem does not represent a feature which can contribute to a similarity measure. In problems where constraints on the students themselves rather than on the exams are concerned, clearly the student set becomes far more important, but again, not simply the number of students.

6.2.2 *Number of Events (Exams)*

This is usually reported together with the number of students to give an idea of the size of a given data set. The exams form the core of the problem, being far more influential than the student numbers in defining the problem structure. The exam timetabling problem is concerned, of course, with assigning these exams to timeslots within a timetable with the main constraint being that two clashing exams are not scheduled in the same slot. In a graph colouring model of exam timetabling (see [26]), the exams form the nodes of the graph with the edges, defined by the student enrolments, representing the conflicts. It is the structure

[1]The conflict graph is made up of vertices representing the exams in the problem, with an edge between any two exams which have students in common - the weight on the edge represents the number of students enrolled to both exams

of this graph representation which is one of the key aspects in how well a given heuristic or meta-heuristic technique will perform when addressing the problem - the number of nodes in the graph is one of the important aspects of this structure. The other main aspect is how these nodes are joined together by edges with some areas forming dense cliques and other areas being relatively sparse in terms of number of edges. These aspects will be considered further in the section 6.2.8.

As a similarity feature by itself, results from the literature suggest that 'no. of exams' is a simple, yet effective indicator as to the potential best technique. The Great Deluge technique with adaptive initial solution generation presented by Burke and Newall [30] provides best known results at the time of writing on the three largest problems[2] from the Carter benchmark data sets [45] (see table 4.1), each with more than 500 exams. On some of the smaller problems, however, this technique proves less effective when compared to those of Caramia [40], Merlot et al. [82] and Casey and Thompson [46] (see table 2.3).

One major area which can be affected by the number of exams in the problem is the run time of a particular technique. In a majority of cases, more exams in the problem leads to longer running time of the algorithm or fewer iterations than would be possible in the same time on a smaller problem. As such, techniques which can converge to a good result in fewer iterations will have an advantage on larger problems. Other noteworthy features based on the number of exams include enrolments per exam and exams per period averages. The number of enrolments per student also gives a measure of how many exams an average student must take. However, as pointed out in Chapter 4 and section 6.2.1, measures involving the number of students and enrolments can be highly misleading.

In the case of the Carter benchmarks, the STA-F-83 data set has a very large number of enrolments per student and exhibits very different behaviour from other problems (see table 4.2). The GRASP technique of Casey & Thomp-

[2]measured by number of exams

son [46] gives a best known solution to this problem which is notably better than results reported from most other techniques. Clearly there are other factors which make this problem fairly anomalous amongst the 11 data sets considered however, as EAR-F-83 and YOR-F-83 also have a relatively large enrolments/student ratio whilst not exhibiting the same behaviour when addressed.

6.2.3 *Number of Periods*

The number of periods in a timetable can be either fixed *a priori* or it may be a variable to be minimised (as part of the objective function or separately). In this thesis, I consider the case where the number of periods is fixed. For those problems in which the number of periods is not fixed, similarity can only be measured against other problems in which this is also the case. As a feature by itself, the number of periods assigned to a problem tells us next to nothing since it is only relevant in conjunction with other features to determine how highly constrained the problem is. In particular, the average number of exams per period and the ratio of number of clashes to number of periods may be of importance. Certainly, the number of exams per period gives a simple measure of one aspect of the *difficulty* of the problem and when combined with the conflict matrix density[3] this measure is potentially a very important one. The number of periods used for the benchmark data sets is slightly higher in each case than the minimum number of periods found to schedule all the exams in. However, the constraints are still very high since the objective function used for these problems is concerned with spreading clashing exams as far apart as possible in the timetable.

Clearly, the more periods available in the timetable over the minimum required to obtain a feasible solution, the more the exams can be spread out. As can be seen from table 4.1, the ratio of exams to periods varies hugely between problems. The main reason for this is the difference in the conflict graphs for the

[3] discussed in more detail in section 6.2.4

problems. More highly conflicting problems will result in a lower exams per period average since the large number of conflicts often makes it hard to schedule all the exams in fewer periods. One of the key strengths of the knowledge discovery techniques we use to select which are the important features is that they can combine pairs of simple features which we select and chose amongst the resulting large number of features the most promising feature vector. This may include combinations of two simple features which we would not have otherwise considered, especially in the case of ratios involving the number of periods in the timetable. Reducing the number of periods in a timetable by one could create a very different problem for which becomes more similar to other problems than to the original problem with an extra period available. In problems where the number of rooms forms a tight constraint on the problem, the ratio between exams per period and the number of available rooms per period will also form an important characteristic of the problem.

6.2.4 *Conflict Matrix Density*

The conflict matrix, as defined earlier is an $exams$ x $exams$ size matrix in which pairs of clashing exams are noted by a 1 with all non-clashing pairs being denoted by 0. The conflict matrix density gives the ratio of 1's as a fraction of the total matrix. Therefore, a high conflict matrix density represents a high probability of conflict between any two exams. For example, a density of 0.5 implies that on average each exam will clash with half of the other exams in the problem. From table 4.1, it can be seen that there is a strong correlation between exams per period and conflict matrix (graph) density. As mentioned above, this is due to the fact that a densely conflicting exam graph tends to mean that a higher number of periods are needed than sparsely conflicting graphs of the same size, therefore the average number of exams per period is correspondingly lower.

The conflict matrix is one of the most important aspects of any exam timetabling problem, representing both the hard constraints and some major soft constraints. Problems with a very high conflict matrix density, such as HEC-S-92 tend to be more difficult to find feasible solutions to in the first place. Also, for meta-heuristics which work only in the space of feasible solutions, this can lead to the search space being very disconnected with respect to certain move neighbourhoods. This can have a major impact on how successfully a particular meta-heuristic can traverse the search space to find a high quality solution. Also, the more disconnected the search space is for a given local search technique, the more focus is placed on the initial solution that is fed into the local search. While some techniques are relatively independent of initial solution and can connect the majority of the search space very effectively, others rely heavily on a good initialisation. One of the key factors in this is the neighbourhood used within the local search technique. In section 6.2.7 I examine the issue of a disconnected search space further.

Conflict matrix density is one of the simplest metrics to be taken from the conflict matrix and also one of the most effective in measuring problem *difficulty*. However, it only gives an average on the percentage of other exams that each exam will clash with. Two problems with the same conflict matrix density can still be very different in structure. I discuss below some of the other measures, which when combined with conflict matrix density, should give a better indication of problem structure for the purposes of measuring similarity.

6.2.5 *Largest Degree*

The *degree* of an exam is defined as the number of other exams in the problem with which it conflicts through having students in common. One of the most common graph heuristics for constructing initial solutions for local search tech-

niques uses a largest degree ordering of the exams to assign sequentially to the timetable. In this way, the most conflicting exams are assigned first as these are deemed to be the most difficult to schedule. The largest degree of an exam timetabling problem is the largest degree that occurs in any of the exams in the problem. As another measure to be obtained from the conflict matrix, this gives us some more information on the problem structure, but by itself of course this is not enough to be of use. The number of exams of largest degree could provide useful information, but is in most cases equal to one.

6.2.6 *Further Conflict Matrix Measures*

Statistical measures resulting from the conflict matrix are of more interest to us than largest degree as features of the timetabling problem. The density mentioned earlier provides an average degree, with the largest degree giving us a maximum. In order to distinguish between two very different problems of equal conflict matrix density we need to consider statistical measures such as the variation in degree from the average and the degree of the exam at different percentiles[4]. We could also consider, as a variation, the number of exams or the percentage of the total exams whose degree is within a given percentage of the largest degree. As with all areas of statistical analysis, there are a large number of different statistical measures we could take based on the structure of the conflict matrix which could each provide some useful information relevant to measuring the similarity between two problems. For our work, we will concentrate on just a small number of these to be fed into the knowledge discovery process.

All of the above statistical measures can, of course, be combined as a ratio with other features and of particular note would be those ratios to the total number of periods in the timetable. Also, a relatively simple measure which could provide useful knowledge is the percentage of the total exams whose degree is

[4]with exams ordered in decreasing order of degree

strictly less than the number of periods. Depending on the neighbourhood used, exams which clash with another exam in every period of the timetable may be unable to move within the local search process if the search is conducted within the set of feasible solutions only. For instance, many techniques successfully utilise the most simple move neighbourhood which selects a single exam and moves it to a new period of the timetable, selected either at random or by some deterministic method. Using this neighbourhood, only those exams which do not have a clash in every other period of the timetable can be moved which can lead to a large amount of dis-connectivity in the search space. On the other hand, using a neighbourhood such as the Kempe chain neighbourhood employed by Thompson & Dowsland [101] and Casey & Thompson [46] ensures that every exam within the timetable can move to any other timeslot. In doing so, a series of other exams will also often have to be exchanged between the two periods in question. This is investigated further in section 6.2.7.

6.2.7 *Fluidity Analysis*

In table 6.1 is presented an analysis of the fluidity of the benchmark problems studied when optimised using simulated annealing with the standard move neighbourhood[5] over 100 runs, each with a different initial solution. The initial temperature and cooling schedule were set to be extremely high and slow respectively for these experiments since the aim was to examine how many exams within the timetable never moved from their initial position as given by the largest degree construction heuristic presented in Chapter 5. With such a high temperature and slow cooling schedule, I can say with a relatively high degree of certainty that any exam which is capable of moving within the neighbourhood used would do so at some point over the course of the 100 separate runs. Of course, there will be exams which may have a small window of opportunity to

[5] as defined in section 2.2.3

Data Set	No. of runs in which x% of exams never moved					
	100	75-99	50-74	25-49	1-24	0
CAR-S-91	1.17%	3.96%	3.96%	3.81%	10.12%	76.98%
CAR-F-92	3.31%	3.31%	2.76%	3.50%	9.39%	77.72%
EAR-F-83	0.55%	6.63%	0.55%	2.21%	10.50%	79.56%
HEC-S-92	0.00%	1.23%	3.70%	2.47%	**50.62%**	41.98%
KFU-S-93	1.65%	2.88%	1.44%	3.70%	4.53%	85.80%
LSE-F-91	1.57%	4.99%	1.57%	1.31%	3.15%	87.40%
STA-F-83	**37.41%**	0.00%	0.00%	3.60%	1.44%	57.55%
TRE-S-92	0.77%	2.68%	0.38%	1.15%	7.66%	87.36%
UTA-S-92	3.05%	5.31%	3.05%	1.93%	7.07%	79.58%
UTE-S-92	6.52%	0.54%	1.63%	1.09%	5.43%	84.78%
YOR-F-83	0.53%	1.58%	0.53%	0.00%	8.42%	88.95%

Table 6.1: Percentage of total number of exams which never move in x runs out of 100 of Simulated Annealing using the simple move neighbourhood

move in this neighbourhood, when a period briefly becomes available that they can move to, but given the random nature of the move selection, the exam was not selected to be moved during this window. However, such exams will be very few across 100 runs of the algorithm. My main objective was to examine how different the data sets are with respect to fluidity for this commonly used neighbourhood.

It is clear that in the case of exams which never move throughout the local search process, their positioning in the initial solution is crucial to the quality of the final solution. For the majority of data sets in table 6.1, this percentage of exams is relatively small and generally below around three per cent. There are also a relatively small percentage of exams which fail to move in > 75 of the 100 runs. A higher percentage are immobile in < 25 runs, but in most cases around 80 per cent of the exams in the data sets move at least once in every one of the 100 runs. Of course, this analysis does not reveal whether many of the exams moved just once during the local search or whether they moved hundreds of times, but in this analysis I am mostly interested in the boolean variable of whether an exam moved at all or not. Whilst nine out of the 11 benchmark data

sets presented share fairly similar fluidity analysis which does not add much to the similarity measurement, two of the data sets exhibit very different behaviour.

Standing out most obviously are the ~ 37 per cent of exams in the STA-F-83 data set which never move across any of the 100 runs. Also notable is the fact that there are no exams in the 50-99 group and very few in the 1-49 group. This indicates that if an exam can move at all in the STA-F-83 data set within this simple move neighbourhood, it will tend to do so in the vast majority of runs. Having over one third of its exams immobile relative to this simple move neighbourhood provides some important clues as to the anomalous nature often displayed by this data set. The reliance on the initial solution becomes massive with so many exams being set in the positions that they are originally placed in. Coupled with the fact that each student takes an average of 9-10 exams and these are spread across just 13 time slots, it is easy to see why this data set yields a very high penalty cost for all feasible solutions (see table 2.3).

Those exams which are immobile will, in general, be the ones with the most clashes and which therefore add the most penalty to the timetable. With this in mind, one of the reasons I believe Casey & Thompson's GRASP technique [46] is so successful on this problem relative to other techniques is the implementation of the Kempe chain based neighbourhoods. As discussed earlier, the Kempe chain neighbourhood allows any exam in the timetable to be moved to any other timeslot whilst always yielding a feasible solution. Using the standard neighbourhood, if the exam, e in timeslot $t_1$to be moved clashes with an exam in the chosen slot, t_2, it cannot be moved there and a new move must be selected. Kempe chains get around this problem by moving all those clashing exams from t_2 across to t_1. Any further clashes induced by this are resolved by moving the clashing exams across with the original exam e to timeslot t_2, with this process continuing until the two periods are conflict-free. Due to the fact that all periods are conflict-free before the first exam is moved, there will always exist a feasible

resolution to the Kempe chains. In the worst case scenario this would involve swapping all exams in t_1 with all exams in t_2. A further discussion of the Kempe chain neighbourhood is given in Chapter 7.

The other data set of note is HEC-S-92 whose behaviour is perhaps even more interesting than that of STA-F-83 and also less easily understandable. Contrary to the other 10 data sets, HEC-S-92 does not have a single exam which never moves across 100 runs of simulated annealing from a different initialisation each time. The percentage of exams which fail to move in 25-99 runs is also very low, yet over half the exams (41) in the data set fail to move in 1-25 of the 100 runs. A deeper analysis of this behaviour concerning how much overlap there is in the runs during which these $> 50\%$ of exams do not move would be required to draw any firm conclusions about this behaviour, but it is worthy of note as being significantly different from all other data sets.

This data set is also the only one which the greedy largest degree with backtracking initialisation technique fails to find a feasible initial solution to. This is due to the fact that the backtracking module only searches two levels deep before giving up and restarting, but on this data set it always reaches the same irresolvable point. This is probably caused by a combination of the high conflict matrix density and also the unusual behaviour indicated by our fluidity analysis. The LD heuristic with randomisation, used in section 5.2 avoids this problem, but tends to need a number of restarts. What this fluidity analysis seems to show is that whilst all exams in the data set can move around over the course of 100 runs from random initialisations, the actual fluidity of the data set from any given initialisation is not so high with a certain number of exams being fixed by their relative positions to other exams. Again, this behaviour can be attributed to the fact that the conflict graph is very dense. An analysis of cliques within the problem as discussed in the following section may also shed some light on this behaviour.

From this analysis it would seem that the fluidity of a given data set with

respect to the neighbourhood used in a meta-heuristic can prove crucial to how successful the meta-heuristic will be. From the point of view of our similarity measure, this is an area which could prove very important.

6.2.8 *Cliques*

One of the most significant features of exam timetabling problems when modelled as a graph is that large cliques and near-cliques tend to exist, unlike the structure of a typical random graph in which any two nodes have an equal probability of being connected by an edge. In exam timetabling, many of the edges which contribute to the conflict matrix are clustered together representing exams which form part of a particular discipline. Students taking science-based subjects will generally take very few, if any, humanities exams, but will take a large number of science exams meaning that the density of clashes between science exams will be far greater than between science exams and humanities exams. Carter & Johnson [42] investigate the cliques to be found within the benchmark problems considered in this thesis as well as looking at near-maximum cliques. An investigation of cliques gives another angle to measuring similarity (based upon the conflict matrix). As well as considering the size of the maximum clique in each problem graph, Carter & Johnson investigate how many cliques there are of max size and also (max-1) size. As a problem feature, the number of maximum size cliques could provide invaluable information for measuring similarity between problems by giving a much more in-depth view of the structure of the conflict matrix.

The two largest data sets (CAR-S-91 and UTA-S-92) are found to have over 100 maximum size cliques whilst the majority of data sets have fewer than five. Again the STA-F-83 data set exhibits very different behaviour to the other data sets, being the second smallest measured by exam size, but having 60 cliques

of size 13, which is also the number of periods used to construct the timetable. Carter & Johnson also calculate the number of nodes occurring in all max cliques and in any max clique together with an analysis of the complement graphs. When considering cliques of size (max-1), the number of these is significantly larger than max-size cliques in the majority of problems. The authors move on to consider Quasi-cliques, where all nodes in a quasi-clique, Q_k, have at most k missing edges from a true clique. Again these represent very dense areas of the conflict graph which have a far larger impact on the difficulty of the problem than the much less dense areas which balance these out to give the overall conflict matrix density.

There is a large amount of analysis which can potentially be carried out on cliques, with Carter & Johnson's work [42] providing a crucial backbone for this. How much of the clique analysis could be used as part of our similarity measure remains to be seen, however, since finding the largest clique in a graph of size n is in itself an NP-hard problem. As such, it may not be feasible for our CBR system to calculate the required feature data for a new problem in order to compare with those in the case base. Having said that, cliques clearly form the basis of the core problem definition in most exam timetabling problems so cannot be ignored.

6.2.9 *Side Constraints & the Objective function*

So far I have considered features which are common to the core exam timetabling problem where the only hard constraints are that every exam must be scheduled to exactly one timeslot within the timetable and no two exams with students in common can be scheduled in the same time slot. The only soft constraint so far considered is that of spreading clashing exams around the timetable using the proximity cost given by Carter et. al [45]. In reality, real world problems will

have a number of other constraints, both hard and soft. The hard constraints will determine the feasible solutions space, whilst the soft constraints will give a measure of how good the timetable is, either by being combined in a weighted single objective function or by forming a Pareto front in a multi-objective optimisation (e.g. [11, 92]).

When attempting to measure similarity between exam timetabling problems, it is required that the problems being compared have the same constraints. Attempting to compare two problems, one in which the number of periods in the timetable is fixed and one in which it is a variable to be minimised is clearly not sensible. For this reason, we need within the case-base a large variety of problems with different hard and soft constraints to give the system as wide applicability as possible. This can be done potentially by adding in constraints to the core problems and forming new cases with these additional constraints.

The work presented in Chapter 5 investigating the effect of the objective function on potential similarity measures showed that, even when the same soft constraints are employed within the problem, using a different set of weights on the constraints can have an effect on the performance of a given heuristic applied to the problem. This is to be expected since the objective function defines the height of the problem landscape at every point, therefore using different weights on the same set of constraints could change the structure of the landscape significantly causing a meta-heuristic which may previously have traversed the landscape very effectively to now get stuck more in local optima and provide a less high quality result.

From this analysis, it would seem that the matching process employed within our CBR system can only compare two problems whose definitions are the same regarding the hard and soft constraints included as well as the weightings applied to those soft constraints. It may be possible to adapt problems within the case base to use the weightings of a new problem given to the system with the same

constraints, but this would require further research.

6.3 Conclusions

In this chapter, I presented a discussion of the key features of exam timetabling problems. The motivation for this work is the aim of creating a similarity measure between timetabling problems, which can be used within a case-based reasoning (CBR) system to intelligently select a heuristic or meta-heuristic technique to solve a new problem. I selected a number of simple features which I expect to form a major part of this similarity measure and analysed their contribution and importance to the problem solving method. From the outset I knew that the contribution to a similarity measure of the simple features listed would by itself be quite small, but that the contribution of combinations of these features would form the key to the similarity measure.

Of the simple features themselves, the number of exams is considered to be a very good basic indicator of problem difficulty and also of which techniques are likely to be successful. In particular, it was noted that on the larger problems (500+ exams), the hybrid great deluge meta-heuristic [30] always outperforms the other techniques considered. For problems of smaller size, other features become more important to distinguish between different problems. The number of students in the problem and the number of periods in the problem are not considered to be of any value as features by themselves, but combined in ratios with other features, number of periods is an important factor. Of the other features studied, the conflict matrix density is thought to be an important element of a problem definition, providing a basic measurement of how highly constrained the problem is. However, it was also noted that other features of the conflict matrix are also required to get a better indication of the problem structure. Amongst these are the variation in degree of exams from the average and the cliques found

within the conflict matrix.

A knowledge discovery process will be applied (similar to [37]), which will select from a large number of features (including ratios of all pairs of features included in the system), those which provide the best measure of similarity between exam timetabling problems. It has been shown [37] that between three and seven features tend to give the best performance for a similarity measure and it is thought likely that most of these features will be ratios of the more simple features presented in this chapter, some of which may not have been considered to be important, but which may be found to provide crucial knowledge of the problems studied. Ultimately however, it will be the training within the knowledge discovery process which will conclude which combinations of features give best system performance based on our test cases. This work is currently ongoing as a part of the wider project.

Chapter 7

A Variable Neighbourhood Search (VNS) technique for Exam Timetabling

The main aim of the work presented in this chapter is to develop a competitive meta-heuristic technique for exam timetabling problems, to be tested on the benchmark data sets used throughout this thesis. Simulated annealing, tabu search, ant colony optimisation and great deluge have all been implemented for potential use within the CBR meta-heuristic selector, but only great deluge (as presented by Burke and Newall [30]) proves to be competitive with current state of the art techniques, mostly on larger benchmark problems. The success of our CBR system will ultimately depend on the quality of the meta-heuristic techniques in the case base therefore there is a clear need for more competitive techniques to include. In this chapter I present a successful Variable Neighbourhood Search (VNS) technique together with discussion on its many variations. Chapter 8 looks at how to improve the performance further and combine this technique successfully with case based reasoning.

7.1 The Importance of the Neighbourhood

In section 1.2, I discussed two of the major factors which determine the success of a local search technique when applied to a given problem, these being the technique itself and the neighbourhood employed during the search. It was pointed out that techniques such as simulated annealing and tabu search generally use a single neighbourhood throughout the entire search and the focus is more on the parameters affecting the acceptance of moves than on the neighbourhood. Clearly the type of technique is an important factor in local search, with simulated annealing and tabu search methods using the same neighbourhood definition providing differing results on benchmark problems.

Burke and Newall [30] and Burke et al. [12] perform a series of experiments comparing simulated annealing (SA) to great deluge (GD) with the focus being on the parameters for the two techniques, both using the same standard move neighbourhood. In [30], the authors demonstrate how big an effect the parameter selection can have on performance of the two techniques and also how much they rely on a good initial solution in order to produce the highest quality results. This is particularly noticeable when fewer iterations of the local search techniques are performed. Also discussed is the possibility for including these approaches in a hyper-heuristic framework which would decide on the parameters.

One of the main focuses in [12] is on modifying SA and GD to simplify the parameters so that run time and an estimate of desired solution quality are all that is required to be set, thus removing the need to tune less easily understandable parameters. Again the two techniques are compared with great deluge generally outperforming simulated annealing in both papers, but not by a significant amount. The biggest advantage of GD is that it is less reliant on its parameters and is more consistent than SA.

The issue of how much the choice of neighbourhood affects the quality of

solutions produced is considered by Thompson & Dowsland [101, 103], as discussed in section 2.2.3. The conclusion they arrive at is that the utilisation of a more complicated neighbourhood [1] than the standard neighbourhood can yield significant improvements in solution quality. The ability of the variable neighbourhood search (VNS) technique to incorporate a large number of neighbourhoods in the search makes it a potentially very interesting method with this in mind.

From the perspective of a CBR meta-heuristic selector, these two major factors can be considered separately or as one. If a number of local search techniques are to be used, with the most suitable for each problem in the case base being stored, it is important that a range of neighbourhoods are tested with each technique to get the best combination. On the other hand, if the technique itself is a far smaller factor than the neighbourhood selection, it is possible that just a single technique could be used with the matching process determining the best neighbourhood and parameters to apply within a given framework rather than choosing a technique itself.

In section 7.2 I present the basic variable neighbourhood search approach which uses a very simple search mechanism with the whole focus being on the neighbourhoods and in particular the ability to search a number of varied neighbourhoods. A huge number of variations of this basic VNS approach exist, making it potentially a very useful technique to combine with CBR to select components for the VNS algorithm for specific problems. The biggest motivation behind implementing a VNS approach to timetabling is that its utilisation of many different neighbourhoods may allow it to perform competitively across a whole range of problems, which is the ultimate aim of our CBR approach.

[1] in their case, Kempe chains which will be covered in more detail in section 7.3.1

7.2 Variable Neighbourhood Search

In the late 1990s, Pierre Hansen and Nenad Mladenović [80] proposed the Variable Neighbourhood (VNS) meta-heuristic for solving difficult combinatorial optimisation problems. It was felt that the reasons for the effectiveness of many local search techniques were difficult to pinpoint, so the authors chose to examine the effects of a relatively unexplored reason, changing the neighbourhood during the search. The resulting meta-heuristic is both very versatile and very successful compared to other local search techniques when applied to a range of different problem domains, in particular for the travelling salesman problem as demonstrated by the authors [80].

More standard local search techniques perform their search by selecting the next move from a fixed and constant neighbourhood at each point during the search with the aim of finding local optima, followed by some form of diversification technique to escape the local optima to search for a new one nearby. The most basic of these methods is the steepest descent (or ascent for maximisation problems) meta-heuristic which has no method for escaping local minimum and simply takes the steepest route down to a nearby local minimum and terminates there since no move within the same neighbourhood which was used to reach the local minimum can improve on the current solution.

More sophisticated techniques such as tabu search (TS) and simulated annealing (SA) include techniques for escaping these local minima to continue the search, either by probabilistically accepting moves with a worse objective cost (SA) or by simply selecting the best move from a neighbourhood or subset of a neighbourhood even if it leads to a worse solution (TS), the tabu list then prevents the search falling straight back into the same local minimum. Such techniques have proved extremely successful when applied to combinatorial optimisation problems. Section 2.2.3 gives a review of many of these local search techniques

applied to exam timetabling.

One point of note regarding such techniques is that a local optima in one neighbourhood is not necessarily a local optima in another neighbourhood, therefore changing neighbourhoods within the search can act as another method for escaping local minima within a particular neighbourhood. This is the basis of the VNS meta-heuristic and can be applied with any underlying local search technique, improving its versatility hugely. As described by Hansen and Mladenović [66], "Contrary to other meta-heuristics based on local search methods, VNS does not follow a trajectory but explores increasingly distant neighbourhoods of the current incumbent solution...".

The basic VNS meta-heuristic is a descent method, moving to a new solution if and only if it is better than the current solution. Since the neighbourhoods are varied regularly, there is no need to accept worsening solutions to escape local minima, although variations of VNS do exist in which such moves are accepted (as will be discussed in section 7.3.2). The VNS meta-heuristic essentially samples a large number of local minima by using a local search technique to bring the solution selected from the neighbourhood to its nearest local optima. Potentially any local search method can be used in this part of the search and the solution arrived at will of course only be a local minimum with respect to the neighbourhood used in the local search. The most basic VNS method uses a simple steepest descent local search with a single neighbourhood, but this can still yield very competitive results.

The steps of the Basic VNS meta-heuristic (refer to Hansen and Mladenović [80, 66] for a more detailed description) are presented in figure 7.1.

Any finite number, k_{max}, of pre-defined neighbourhoods may be used within VNS and the neighbourhoods are usually in some sense nested with k_{max} being the most diverse neighbourhood, although this is not a strict requirement. Stopping criteria may be selected as for any local search technique, total number of

- *Initialisation*: Select the set of neighbourhood structures N_k, $k = 1, \ldots, k_{max}$, to be used in the search; find an initial solution x; choose stopping criteria;
- *Repeat* until stopping criteria is satisfied:
    1. Set $k := 1$;
    2. Until $k = k_{max}$, repeat:
        (a) *Shaking*: Generate a point x' at random from the k^{th} neighbourhood of x ($x' \in N_k(x)$);
        (b) *Local search*: Apply a local search method with x' as initial solution, until a local optimum, x'' is obtained;
        (c) *Move or not*: If x'' is better than the incumbent solution x' then move there ($x \leftarrow x''$), and continue the search with N_1 ($k \leftarrow 1$); otherwise, set $k = k + 1$;

Figure 7.1: The steps of the Basic VNS meta-heuristic

iterations, number of iterations without improvement or CPU time being three of the most commonly used. In the basic VNS, the move selected from the neighbourhood at step 2 (a) is generated at random, avoiding any issues of cycling; also the local search is performed using a single neighbourhood. There are a large number of variations of the basic VNS, with changes possible to each step of the algorithm given in figure 7.1 - many of these are discussed in [66] and will be considered in context for exam timetabling in section 7.3.2.

7.3 VNS for Exam Timetabling

In this section I discuss my application of variable neighbourhood search to the exam timetabling problem, which has seen a large number of single neighbourhood local search techniques successfully applied in recent years. The initial implementation was based on the basic VNS presented in figure 7.1 to which a number of variations were sequentially added to further improve performance.

The one major change from the basic VNS presented is that instead of con-

tinuing the search with neighbourhood N_1 each time an improvement is found in step 2 (c) the search continues using the current neighbourhood which yielded the improvement. The main reason for this is that it places slightly less reliance on the ordering of the neighbourhoods and focuses the search on each neighbourhood for as long as it yields an improvement before moving to the next neighbourhood in the list. Experiments with the basic VNS from figure 7.1, always restarting with neighbourhood N_1 after an improvement have also been performed, but results so far have not been as successful as the first method. Hence, further experiments are all performed with the search continuing in neighbourhood N_k rather than N_1 at step 2 (c). This allows further neighbourhoods to easily be incorporated without the need to consider the ordering since, with this method the order only matters for determining which neighbourhood the search begins in.

To initialise VNS I use the greedy and randomised largest degree graph-colouring heuristics introduced in section 5.1, both of which produce a feasible initial solution. The greedy approach gives higher initial solution quality at the expense of diversification whilst the randomised approach provides the opposite. VNS performs its search only within the search space of feasible solutions for all experiments reported. The two different initialisation techniques were tested for a number of reasons. Firstly, the greedy method produces much better initial solutions as measured by the objective function and so may result in better, or at least faster solutions from VNS. However on one data set tested, this deterministic technique with backtracking fails to find a feasible solution so the random method had to be used which does find feasible solutions. Also, I was interested to note how much of an effect the quality of the initial solution has on the quality of solution produced after VNS is applied. Many local search techniques, using a standard trajectory search, are very dependent on their initial solution for certain problems where the search space is very disconnected, but it was felt that VNS

might be able avoid this dependence on initial solution by still being capable of reaching all areas of the search space using different neighbourhoods.

The stopping condition used requires a minimum of 10,000 iterations with 2,500 idle iterations (iterations without improvement) before the algorithm terminates. The local search part of the method uses a simple steepest descent approach which is fast and yields very good results. Other more complex local search techniques may be considered in future work. However, introducing a more complicated local search technique also involves introducing a number of new parameters which have to be carefully tuned and will also notably increase the running time of the algorithm.

7.3.1 Neighbourhoods used within VNS

For the exam timetabling problem, neighbourhoods used in local search techniques generally consist of moving some subset of exams from their current time slot to a new time slot, the number and identity of exams to move forming the definition of each neighbourhood. My initial implementation of VNS used the following eight neighbourhoods, although one of these was later replaced for further experiments for reasons explained below:

1. *Single move*: The simplest move neighbourhood for exam timetabling and the one most commonly used in single-neighbourhood local search techniques - this neighbourhood consists of all moves obtained by selecting a single exam and moving it to a new feasible time slot. This neighbourhood can be quite limiting as many exams in a timetable have no other feasible slots to move to so these exams will never be moved. (Also referred to as the Standard neighbourhood).

2. *Swap*: The swap neighbourhood contains all feasible moves involving swapping the time slots of a pair of exams, e_i and e_j. By itself this is a

very limited neighbourhood since it requires that the two exams selected can both feasibly move to each other's time slot. In some circumstances this neighbourhood can solve the main problem of the single move neighbourhood by moving the clashing exam from a given slot which prevented the first exam from moving there otherwise, however this is only the case if there is just a single clash in the time slot. Despite its limitations as a single neighbourhood, the Swap neighbourhood can prove very useful within a VNS framework.

3. *Move 2 exams randomly*: This neighbourhood is formed from all pairs of Single moves. Instead of picking a single exam to move to a new time slot, 2 exams are chosen at random and moved to a new feasible time slot. This allows for a slightly more diverse change to the current solution than the single move neighbourhood.

4. *Move 3 exams randomly*: As above

5. *Move 4 exams randomly*: As above

6. *Move 5 exams randomly*: As above

7. *Move a whole time slot*: In an exam timetable, the n time slots are ordered from 1 to n with clashing exams in nearby time slots adding a high penalty to the overall timetable cost. Rather than moving individual exams between timeslots, this neighbourhood moves an entire timeslot to a new position in the ordering, with the other periods being shuffled along accordingly. Moving entire timeslots allows exams which would otherwise be unable to move (due to clashing with exams in all other timeslots) to move around the timetable relative to all exams in other periods. This can be extremely useful, especially if VNS is seeded with a bad initial solution since it allows for most components of the timetable to move relative to

each other - the only components it does not allow to move relative to each other are exams in the same time slot.

8. *Swap timeslots*: Similar to the previous neighbourhood, instead of moving a single timeslot to a new position in the ordering and as a result shuffling other periods down the ordering, this neighbourhood only affects 2 time slots, simply swapping all exams in one with all exams in the other. Again, this allows for previously immobile exams to move around the timetable.

Neighbourhood 1 is the neighbourhood used in the steepest descent local search part of the algorithm and as such did not figure very strongly in the VNS part of the algorithm since each solution is optimised relative to that neighbourhood using steepest descent therefore picking a random move from neighbourhood 1 invariably led to the same local minimum being found instantly. A tabu list could be used to prevent the search from dropping straight back into the same local minimum, however I decided to replace neighbourhood 1 with the more successful Kempe chain neighbourhood used by Thompson and Dowsland in their simulated annealing technique [101].

The Kempe chain neighbourhood involves swapping a subset of exams in 2 distinct time slots (colours in the graph colouring model) - in my implementation, an exam e, in slot s_1 and a new timeslot, s_2, are selected at random in the same way as for the single move neighbourhood above, with the exams in the two timeslots forming a bi-partite graph since only feasible solutions are allowed. The Kempe chain is defined from the initial chosen exam as the connected components of this bi-partite graph, these are exams which clash with each other and therefore must be moved across to the other period as their clashing exams are moved to their current period. The single move neighbourhood is a subset of the Kempe chain neighbourhood consisting of all disconnected vertices (exams) in the bi-partite graph - these are the exams which can be moved across to the new

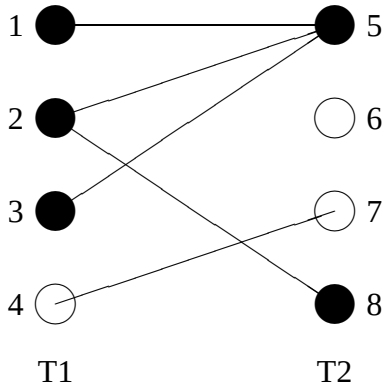

Figure 7.2: A Kempe chain move before execution

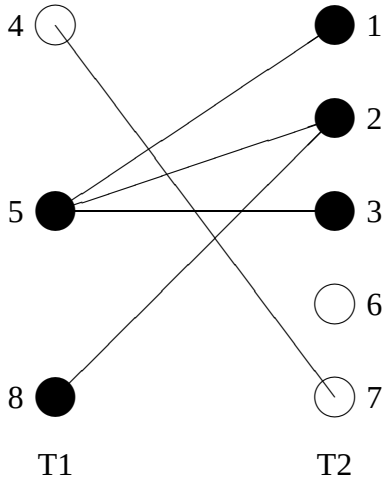

Figure 7.3: The result of a Kempe chain neighbourhood move

period without inducing a clash.

Figure 7.2 shows a simple Kempe chain move involving 2 timeslots, T1 and T2 each containing 4 exams. Exam 6 is a disconnected node which could move from slot T2 to T1 in the single move sub-neighbourhood of the Kempe chain neighbourhood. All other exams have clashes (represented by edges of the graph) with exam(s) in the other time slot and so could not be moved in the single move neighbourhood. If exam 1 is selected in the Kempe chain neighbourhood to be moved to slot T2, the Kempe chain represented by the black circles in figure 7.2 is constructed resulting in exams 1, 2 & 3 moving to T2 and exams 5 & 6 moving

across to T1 to maintain feasibility of the solution and the bi-partite nature of the graph defined by time slots T1 and T2 as shown in figure 7.3.

The Kempe chain neighbourhood eliminates the main failing of the single move neighbourhood by allowing any exam within the timetable to be moved to a new time slot since every pair of time slots forms a bi-partite graph meaning that there will always be a Kempe chain move starting with any exam. The largest Kempe chain move would result in every exam being exchanged between two periods - whilst in graph colouring this would not have any effect on the solution, in exam timetabling it does since the periods are all ordered. However, unlike the simple move neighbourhood, the swap timeslots neighbourhood is *not* wholly contained within the Kempe chain neighbourhood since in the example of Figure 7.2, exam 6 would never move across to T1 with the other exams because it is disconnected.

7.3.2 Variations of VNS for exam timetabling

The initial aim was to test out the most basic version of VNS on exam timetabling to investigate how well it performs compared to other local search techniques on benchmark problems with the intention to add more complexity afterwards so I could keep track of which aspect of the method is providing the improvement. Results (presented in section 7.4) from the basic VNS detailed in the previous sub-section were extremely encouraging and in fact, without any further improvements have produced results which are highly competitive with those of other state of the art techniques. It was felt that some of the many variations of VNS could produce even better results.

One of the significant advantages of VNS is that it is a very modular technique which can easily be added to in almost any of the steps given in figure 7.1 to produce potentially better results. A number of variations under consideration

are listed below, some of which are suggested by Hansen and Mladenović [66]:

- *Descent-ascent*: Basic VNS is a 'descent, first improvement method with randomization' [66]. A very simple change to the algorithm would be to make it a descent-ascent method by accepting worsening moves with some probability in a similar way to that used by simulated annealing.

- *Best improvement*: Instead of taking the first random move from a single neighbourhood, make a move to the best neighbourhood $k*$ among all k_{max} of them.

- *Variable neighbourhood descent (VND)*: VND uses many neighbourhoods during the local search phase of the VNS method as oppose to the more standard use of just a single neighbourhood. Hansen and Mladenović [66] report that this technique is crucial to obtain good results in certain problem domains.

- *Biased VNS*: In step 2 (a) of the basic VNS, it is possible to select the solution x' in a number of different methods rather than purely at random. One such method could be to choose the best move from a random selection from the k^{th} neighbourhood or to select at random from the n exams adding most penalty to the timetable for a neighbourhood such as the Kempe chain neighbourhood above.

- *More complex local search*: In place of the steepest descent local search technique to bring solutions to their local optimum, any other local search technique, such as simulated annealing, great deluge or tabu search could also be used, with a sufficient stopping criteria for the local search. One of the main disadvantages of the current implementation however is the run time so a more complex local search technique would be likely to increase

this still further unless it allows for significantly fewer iterations to be run to find a high quality solution.

- *Problem-specific neighbourhoods*: Since the structure of different exam timetabling problems can vary hugely, it may be the case that some neighbourhoods work far better on one problem than another and that discarding certain neighbourhoods from the original set will not detract from the quality of solutions produced and may in fact improve solution quality by allowing the search to spend longer in more promising neighbourhoods. In my implementation, the more neighbourhoods I include, the fewer iterations are performed in each neighbourhood before the stopping criteria is met - reducing the number of neighbourhoods to the *best* subset for the specific problem may allow for further improvements to be found or for good solutions to be found faster.

- *Different initialisation strategies*: Currently I use two different initialisation strategies to seed VNS, but there are a number of other methods which can be used. In section 7.4, I present results from VNS when initialised by a greedy construction technique and by a random construction technique - one of the aims of this comparison is to examine the importance of the initial solution to the quality of the final solution produced by VNS since, for most datasets the greedy initialisation technique produces far better starting solutions.

Of the above variations, the first one to be considered to improve on the results from basic VNS is the *biased VNS* method. Rather than apply this equally to all neighbourhoods I add in two new neighbourhoods based on the successful Kempe chain neighbourhood. The first of these samples a random 5% of the total exams and selects the one adding the highest penalty to the current timetable. This exam is then used to form a Kempe chain move in exactly the same way as

when a random exam is chosen. The second variation selects an exam at random from the 20% of exams adding the highest penalty to the timetable and continues with the Kempe Chain move as before.

The aim of adding these two neighbourhoods is to better utilise the strengths of the Kempe chain neighbourhood which has the ability to move any exam in the timetable to any other slot, unlike many simpler neighbourhoods. Selecting the first exam for the Kempe chain purely at random will quite often lead to a worse solution if the exam in question is already adding very little penalty to the timetable. It is hoped that by biasing selection towards the most troublesome exams, which are also the exams which can rarely move in the simpler neighbourhoods, better improvements can be found. Results from this variation of VNS are presented in section 7.4 together with those of basic VNS for comparison.

Also considered is the *descent-ascent* approach which includes an acceptance criteria combining aspects from both simulated annealing and great deluge. In order to maintain a mostly descent method, only solutions which are less than 1% worse than the current solution are considered, with on average 10 per cent of these being accepted. This variant of the basic VNS adds in further parameters, but can yield some improvement without these parameters having been tuned to each individual problem. Further improvement may be possible with better tuning, but I did not wish to introduce parameter tuning into the algorithm since that would take away one of the major advantages of VNS.

The *best improvement* and *variable neighbourhood descent* methods are not considered here because they involve far more significant changes to the algorithm and also increase the run time of each iteration. Another variation involving multiple Kempe chain move neighbourhoods was tested with up to five Kempe chain moves forming a neighbourhood, but this variation yielded no improvement in the tests performed. However, these neighbourhoods are consid-

ered again in Chapter 8. Also tested were variants of VNS utilising great deluge or simulated annealing in place of the simple steepest descent local search method, but again these introduced a large number of parameters and far longer run times without yielding any improvement.

The other variation which I looked into is the *problem-specific neighbourhood* selection. In the implementation in this chapter, when I consider a new neighbourhood which may add something to the performance of the algorithm I simply add it to the existing set of neighbourhoods to run the algorithm. If the stopping criteria are extended to take account of the extra neighbourhoods, this should not detract from the ability of the VNS meta-heuristic to find high quality solutions, however it will increase the run time of the algorithm. If the stopping criteria are kept the same, adding more and more neighbourhoods may cause performance to drop due to the search spending less time in each neighbourhood. Statistics collated from many runs of VNS on 11 benchmark problems show that for some problems a certain neighbourhood results in an improvement very often whilst in other problems the same neighbourhood rarely results in an improvement whilst a different neighbourhood is most effective.

Rather than just use these fairly crude statistics to decide which neighbourhoods to discard for each problem, in Chapter 8, I investigate the use of a genetic algorithm (GA) technique to intelligently select the best neighbourhoods to include in the search for a given problem. I also examine how this can fit into our case based reasoning framework.

7.4 Results

Experiments were carried out on the 11 benchmark data sets used in the earlier chapters of this thesis. Results for these problems are reported as average penalty per student with only the soft constraint of spreading clashing exams around

the timetable taken into account via the use of proximity costs, as detailed in section 4.2. Results from the literature for a variety of methods applied to these data sets are presented in table 2.3.

Table 7.1 compares the results produced by the Basic VNS algorithm with the best reported solution taken from Table 2.3. Figures in italics represent the best solution found between the two initialisation techniques. Basic VNS uses a Kempe Chain neighbourhood together with neighbourhoods 2-8 from section 7.3.1, all using random move selection in step 2 (a) of the algorithm in figure 7.1.

Table 7.2 gives the results of the Biased VNS approach when initialised by the two different methods. Biased VNS-RI and Biased VNS-GI use the eight neighbourhoods used for Basic VNS with random move selection, but also include the two additional biased Kempe Chain neighbourhoods described in section 7.3.2, one selecting the exam currently adding the largest penalty to the timetable from a random 5% sample, the other selects a random exam from the 20% of exams adding the largest penalty.

Average and best results from 100 runs on a 750MHz Athlon are presented for both Basic and Biased variants. Run times vary considerably between problems as well as across the 100 runs on a single problem due to the stopping criteria of 2,500 iterations without improvement after a minimum of 10,000 and the random move selection from neighbourhoods - some runs terminate after 10,000 iteration whilst others continue to improve past 40,000 iterations. On the smaller datasets, the algorithm terminates in the order of 1-2 minutes, whereas run times for larger data sets range from 30 minutes to 90 minutes. A further discussion of the run times on individual problems is provided in Chapter 8 for the most successful VNS variant. Run times for results reported from the literature in Table 2.3 vary between around 0.5 seconds for smaller datasets up to 15 minutes on the larger datasets.

Data Set	Best reported solution	Basic VNS-RI		Basic VNS-GI	
		Best	Average	Best	Average
CAR-S-91	4.6	5.10	5.29	*5.07*	5.24
CAR-F-92	4.0	4.20	4.39	*4.17*	4.30
EAR-F-83	29.3	*33.56*	36.33	33.70	36.35
HEC-S-92	9.2	*10.41*	11.08	-	-
KFU-S-93	13.5	*13.72*	14.40	13.85	14.54
LSE-F-91	9.6	*11.13*	11.70	11.18	11.74
STA-F-83	134.9	*156.86*	157.12	*156.86*	157.15
TRE-S-92	8.0	*8.48*	8.88	8.49	8.84
UTA-S-92	3.2	3.49	3.59	*3.40*	3.49
UTE-S-92	24.4	*25.10*	25.94	25.18	26.01
YOR-F-83	36.2	36.80	38.70	*36.77*	38.47

Table 7.1: Results from the basic VNS meta-heuristic with random and greedy initialisations

Data Set	Best reported solution	Biased VNS-RI		Biased VNS-GI	
		Best	Average	Best	Average
CAR-S-91	4.6	*5.02*	5.28	5.07	5.12
CAR-F-92	4.0	4.17	4.34	*4.12*	4.23
EAR-F-83	29.3	*33.10*	36.00	33.46	35.78
HEC-S-92	9.2	*10.26*	11.02	-	-
KFU-S-93	13.5	***13.38***	13.87	***13.38***	14.03
LSE-F-91	9.6	*10.66*	11.33	10.93	11.58
STA-F-83	134.9	*156.86*	157.04	*156.86*	157.06
TRE-S-92	8.0	*8.35*	8.76	8.39	8.77
UTA-S-92	3.2	3.47	3.55	*3.39*	3.50
UTE-S-92	24.4	*24.86*	25.41	*24.86*	25.43
YOR-F-83	36.2	36.48	38.33	*36.43*	38.03

Table 7.2: Results from the VNS with biased neighbourhoods meta-heuristic with random and greedy initialisations

From Table 7.1, it can be seen that the basic implementation of VNS fails to match the best reported solutions from the literature on the 11 benchmark problems, but does still provide high quality results across all problems. When compared with other techniques from Table 2.3 it can be seen that the results from Basic VNS are highly competitive and beat the results of each other technique on at least one problem, indicating that whilst it is unable to produce any best known solutions, it is highly consistent across the range of problems which was one of the my main aims for the technique. Methods from Table 2.3 which produce the best solution on one problem are often outperformed by a few techniques on other data sets, showing that they are more suited to some problems than others.

Comparing the results of Tables 7.1 & 7.2 shows the clear improvement in solution quality of introducing the two biased neighbourhoods. Apart from the anomalous STA-F-83 problem, Biased VNS outperforms Basic VNS for all problems and significantly also manages to produce a best known solution to the KFU-S-93 data set of 13.38 compared to 13.5 obtained by Merlot et al [82]. Improvements in some problems over the basic VNS are relatively small, whilst other problems yield a significant improvement, in particular the KFU-S-93 data set. Results on LSE-S-91 dataset also improve markedly when using the Biased VNS, but despite this improvement this is still the worst result for VNS when compared to the other methods in Table 2.3, providing only the 5th best result out of 7.

When comparing the two initialisation strategies, results are mixed with neither initialisation approach outperforming the other across all problems. For the Biased VNS method, the random initialisation outperformed greedy initialisation on five problems, whilst leading to inferior results on three problems and equal performance on the other three. This in itself is quite significant however, implying that VNS has the capability to overcome a seemingly bad initialisation

to still produce equally high quality results. Unsurprisingly, the average performance of the VNS initialised with the greedy technique was higher in a majority of cases, but not by a significant amount and not on all problems. Again, this shows that the VNS technique can take a range of highly diverse solutions (for certain problems the random technique yields some initial solutions which have twice the cost of others also produced by the technique) and optimise all to a relatively high standard.

Further research would need to be carried out to determine exactly how much the initial solution is changed to produce the final solution using the two initialisation techniques, but it may be the case that the increased diversity of the random initialisation approach allows VNS to more easily find higher quality solutions without being forced into a smaller area of the search space by the less diverse greedy approach whose initial solution may place certain exams in unfavourable timeslots which then have to be corrected by the VNS algorithm.

Also worthy of note is that the Biased VNS invariably leads to a higher level of consistency in the quality of solutions produced with the best average performance across 100 runs coming from the Biased VNS variant for 10 of the 11 problems. The average result from Biased VNS on some data sets also outperforms the best result of some techniques, showing that the method is capable of producing good results consistently across a number of runs as well as individual good results from a selection.

In table 7.3, I present the best results obtained by the descent-ascent variation of the Biased VNS approach [2] compared to those of other techniques in the literature. On the majority of datasets, this variation outperforms the pure descent Biased VNS variant with only the EAR-F-83 data set proving to be an exception when optimised using descent only Biased VNS-RI. However, further analysis

[2]These results are the best across both initialisation techniques with some best solutions coming from each method

Data Set	Carter et al. [45]	Caramia et al. [40]	Burke & Newall [30]	Casey & Thompson [46]	Merlot et al. [82]	Descent-ascent Biased VNS
CAR-S-91	7.1	6.6	**4.6**	5.4	5.1	4.9
CAR-F-92	6.2	6.0	**4.0**	4.4	4.3	4.1
EAR-F-83	36.4	**29.3**	36.1	34.8	35.1	33.2
HEC-S-92	10.8	**9.2**	11.3	10.8	10.6	10.3
KFU-S-93	14.0	13.8	13.7	14.1	13.5	**13.2**
LSE-F-91	10.5	**9.6**	10.6	14.7	10.5	10.4
STA-F-83	161.5	158.2	168.3	**134.9**	157.3	156.9
TRE-S-92	9.6	9.4	**8.2**	8.7	8.4	8.3
UTA-S-92	3.5	3.5	**3.2**	-	3.5	3.3
UTE-S-92	25.8	**24.4**	25.5	25.4	25.1	24.9
YOR-F-83	41.7	**36.2**	36.8	37.5	37.4	36.3
Total Pen	327.1	306.2	322.3	-	310.8	**305.8**

Table 7.3: Ascent-descent Biased VNS compared to results from the literature (best results given)

of the individual runs shows that the 33.10 result obtained by this technique was significantly better than all other results produced across 100 runs with the second best being only 33.88. In a method such as VNS, which involves a large random element there is always the possibility of any variant producing a very high quality solution on a single run, but not consistently across many runs.

It can be seen from table 7.3 that the descent-ascent Biased VNS method performs very favourably compared to the current state of the art techniques, improving the best known solution for the KFU-S-93 data set to 13.2. Perhaps more significantly however is the consistency of performance across the range of all 11 problems, where the approach is ranked 2nd out of the 6 presented on the other 10 data sets, giving a total penalty across all data sets lower than the four approaches whose techniques were applied to all data sets. This can be misleading as it may be biased heavily by one problem, especially since the penalty for the STA-F-83 dataset dwarves that of most others. However, there are clear indications that VNS is very capable of producing high quality results across all data sets tested upon.

7.5 Conclusions

The Variable Neighbourhood Search (VNS) approach presented in this chapter has been shown to produce solutions of a high quality across a range of benchmark data sets, producing a best reported result on one medium sized data set and performing consistently on most other data sets. In this chapter I have presented results from a basic version of VNS with the slight variation from that given by Hansen and Mladenović [80] and presented in Figure 7.1 that the search continues in the current neighbourhood if an improvement is found in step 2 (c) rather than returning to neighbourhood 1 as this yielded better results here without needing to carefully tune the ordering of the neighbourhoods.

A number of more complex variations to the basic VNS were detailed in Section 7.3.2 from which a Biased VNS has been tested with two new neighbourhoods. Results from this Biased VNS have shown significant improvement over the Basic VNS on many of the data sets tested and are very competitive with current state of the art techniques. Further improvements were yielded by the addition of an ascent mechanism similar to that of simulated annealing. The best results from this approach compare very favourably with techniques reported in the literature and it proves robustly competitive across all data sets.

One significant advantage of the Basic VNS approach is that it involves very few parameters out-with the selection of the neighbourhoods, all of which are easily implemented since only a random move is required rather than an exhaustive search of the neighbourhood. With a high degree of modularisation, neighbourhoods can be added and taken away easily, any local search technique can be used and the method of move acceptance altered. Basic VNS also has a large number of potential improvements, which, whilst adding more parameters can be used to improve performance. The one notable disadvantage to my VNS implementation so far is the time taken on large problems which can be as

much as 90 minutes for a single run. Whilst run time is less crucial for exam timetabling than many other combinatorial optimisation problems, since exams are generally only taken a few times a year with a fair degree of planning time, it is still an area which needs improvement.

When considered in the context of our case based reasoning meta-heuristic selector, VNS currently only gives a best known solution to one benchmark problem and thus would not be selected by a *perfect* CBR system for use on a new problem similar to any of the other data sets if all other reported techniques were also implemented. However, as the second best technique on all other data sets, it provides a clear benchmark against which to measure the performance of a CBR meta-heuristic selector. In Chapter 8, I propose another variant of VNS, utilising a genetic algorithm to intelligently select the neighbourhoods for use with VNS from an increased selection. This technique is specially designed to work with our CBR system and yields still better results than those presented in this chapter.

Chapter 8

Combining a Genetic Algorithm with VNS to improve solution quality

In Chapter 7, I presented a variable neighbourhood search approach to exam timetabling together with a number of variations, some of which were successfully implemented to improve the performance of the technique. In this chapter, I investigate another of those variations which has particular importance to combining my work on VNS with the wider case based reasoning project of which this thesis is a part. The main focus of VNS is of course on the neighbourhoods with the ability for the search to pick solutions out of a variety of neighbourhoods giving it a high degree of flexibility. In Chapter 7 I considered initially just eight neighbourhoods with a further two added for the Biased VNS technique. Here I increase that number to 23 neighbourhoods with the potential for many more. In order to make this viable I introduce a genetic algorithm based technique for intelligently selecting neighbourhoods for a given problem and show how this can be built into our CBR system.

8.1 Combining VNS with CBR - the requirements

The results presented in Chapter 7 show that the VNS approach presented is highly successful when applied to the 11 benchmark problems tested on, producing one best known solution and producing the second best solution amongst all techniques considered on the other 10 problems. For the purposes of creating a general technique to apply to exam timetabling, VNS is clearly a success, with a number of further improvements to the technique possible. However, for the purposes of our project, the CBR system is element which provides the generality across a range of problems with the techniques contained within the case base being more specific to certain sets of problems. For instance, if all the techniques included in table 7.3 were part of a *perfect* CBR system, the great deluge approach of Burke & Newall [30] would be selected as the best technique for the CAR-S-91, CAR-F-92, TRE-S-92 and UTA-S-92 problems whilst the technique of Caramia et al [40] would be selected as the best for five of the 11 problems and VNS for just the KFU-S-93 problem.

Of course, we do not have all the techniques from table 7.3, although many could be implemented for use within the system. However, this does highlight one of the major research aspects of the CBR system: for it to be successful at selecting from a range of meta-heuristics, we must implement many different techniques, each of which is sufficiently distinguishable from the rest so that we can say that a certain method works best on a set of *similar* problems. For this to be true, we ideally need to know not only which technique works best on which problems, but also some indication of *why* that technique is better on those problems and why some other technique is better on a different set of problems. In the case of many meta-heuristic techniques developed for exam timetabling, it is very difficult to pinpoint *why* the technique is successful. For instance, why is the great deluge of Burke & Newall [30] successful on the three largest data sets,

but is noticeably outperformed by the technique of Caramia et al [40] on many of the smaller data sets?

As discussed earlier in this thesis (see sections 1.2 & 7.1), distinguishing the methods in the case base by their techniques (e.g. great deluge, simulated annealing) is one way in which CBR could be used to select one of these to apply to a new problem after a matching process, but this still leaves the rather important question of parameters and neighbourhoods for use within these techniques. Burke & Newall [12] focus solely on the parameter aspect of time pre-defined great deluge and simulated annealing approaches in order to investigate the effective use of the time allowed for the algorithms, using the same neighbourhood structure for both. They do however note that the utilisation of a more advanced neighbourhood such as Kempe chains could produce better results. This would imply that ideally all techniques used would have to be tested with numerous neighbourhood structures to obtain the absolute best algorithm for each problem in the case base.

An alternative to this would be to use a single algorithm type for the majority (or the entirety) of the case base and tune this technique to the individual problems so that the CBR system would retrieve this technique with the parameters and neighbourhood used for the matched problem when a new problem is given to the system. Given the results presented in Chapter 7 and the possibilities for easily extending and altering the VNS method in many different ways, this technique seemed like a very good one to use for such a purpose. In order to do this however, it would need to be shown that VNS is capable of competing to obtain best known results on more than just a single benchmark problem. If it can be shown that different variations of VNS work well on different problems, this can be neatly combined with CBR to select a variation of VNS to apply to a new problem rather than a type of meta-heuristic. This would also eliminate the need to implement a wide variety of techniques. Instead the focus would be on

implementing a range of neighbourhoods, some of which will be more suited to certain problems than others.

In this chapter I present a genetic algorithm (GA) approach to intelligently selecting a subset of neighbourhoods to use within VNS for a given problem. The idea of using a GA at a higher level of abstraction rather than being applied directly to the problem itself has been successfully implemented by Terashima et al [100] to evolve the configuration of constraint satisfaction methods as discussed in section 2.2.1 while Ross et al [98] also comment that GAs may be better suited to searching for good algorithms rather than acting on the problem itself. Han et al [52, 63, 64, 65] successfully utilise a GA within a hyper-heuristic framework to evolve an ordering of low-level heuristics applied to the trainer scheduling problem. The key difference between the work presented here and the work of Han et al is that in their hyper-heuristic framework, low-level heuristics are being ordered by the GA to be applied sequentially to the problem, whereas in my implementation of VNS, all neighbourhoods used within the technique are searched, but a move is only made within a given neighbourhood if it fulfils the criteria for move acceptance.

8.2 The GA technique for neighbourhood selection

The implementation presented in this chapter uses a GA (referred to hereafter as VNS-GA) to evolve a subset of neighbourhoods from a large pool for use within the VNS framework on the 11 benchmark problems considered throughout this thesis, with further possibilities for extension also considered. A very simple chromosome representation is used with fixed length equal to the total number of neighbourhoods to select from. Each neighbourhood is represented by a number, but their ordering is unimportant since the VNS method presented in Chapter 7 cycles through all neighbourhoods, moving to the next when the

move selected from the current neighbourhood is not accepted rather than returning to the first neighbourhood whenever a move is accepted. Neighbourhoods can be repeated within the chromosome representation, but duplicates are removed when the chromosome is translated to the actual set of neighbourhoods to be used within VNS. This essentially means that repeat neighbourhoods in a chromosome represent the loss of a neighbourhood from the set, thus creating many possible distinct subsets of the full neighbourhood set. A chromosome in which all elements are the same would represent just that single neighbourhood supplied to VNS.

Crossover and mutation operators are both implemented in a very simple manner. A percentage of the population of each generation is chosen to produce an equal number of offspring with the rest of the next generation being selected directly from the chromosomes of the current generation. In my implementation, 70 per cent of the chromosomes are selected for crossover using a roulette wheel style selection based on their fitness evaluation. Most of the 30 per cent of chromosomes to survive to the next generation are also selected using the same roulette wheel method. The probability, $P(x_i)$, of a chromosome x_i being selected from the population X_g of chromosomes in generation g is given in equation 8.1 with the fitness function given in equation 8.2

$$P(x_i) = fitness(x_i)/(\sum_{\forall x_j \in X_g} fitness(x_j)) \tag{8.1}$$

$$fitness(x_i) = \max\{(\max_{\forall x_j \in X_1} \{VNS(x_j)\} \times f) - VNS(x_i), 0\} \tag{8.2}$$

where f is the fitness modifier, in my experiments ranging between 1.01 and 1.05. A lower value causes the better chromosomes to have a much larger chance of roulette wheel selection than the worse chromosomes, whereas a higher value

gives worse chromosomes a better chance of being selected by making the difference between fitness values of the best and worse chromosomes relatively much smaller. The fitness normalising value is taken from the worst chromosome of the first generation in order to retain consistency throughout the generations. Any chromosomes in future generations whose VNS evaluation is greater than the normalised value will have a zero per cent chance of selection, although due to the consistency of the VNS algorithm, this is rarely the case. $VNS(x_j)$ gives the result of applying the VNS algorithm with the neighbourhoods specified by chromosome x_j. If more than one VNS run per chromosome is used then this value can be either the average or best across the runs for a given chromosome.

A standard one-point crossover technique is used to produce the two *offspring* from the two selected *parents* with the crossover point selected randomly. Figure 8.1 shows an example of the crossover procedure for chromosomes of length 8 with the crossover point indicated by the dashed line. Child 1 takes the front portion of the chromosome from Parent 1 and the back portion from Parent 2 and Child 2 the opposite. Since multiple copies of a neighbourhood are permitted in this representation there is no need to perform any repair operator after the crossover as all chromosomes are equally feasible. The interpretation of each chromosome as a set of neighbourhoods is given to the right of figure 8.1.

The mutation operator acts after selection and crossover, changing an element (gene) of a chromosome to a random neighbourhood with a given probability which can be increased or decreased to alter the random element of the evolution.

The basic steps of the VNS-GA procedure are given in figure 8.2.

As discussed earlier, the fitness calculation in Step 1 of figure 8.2 can be calculated either from the average result or the best result across the runs of VNS if $v > 1$. In the experiments reported in this chapter, I always use the best result. A variety of different methods are possible for generating the next population using crossover and selection, for instance $s = n$ would cause all

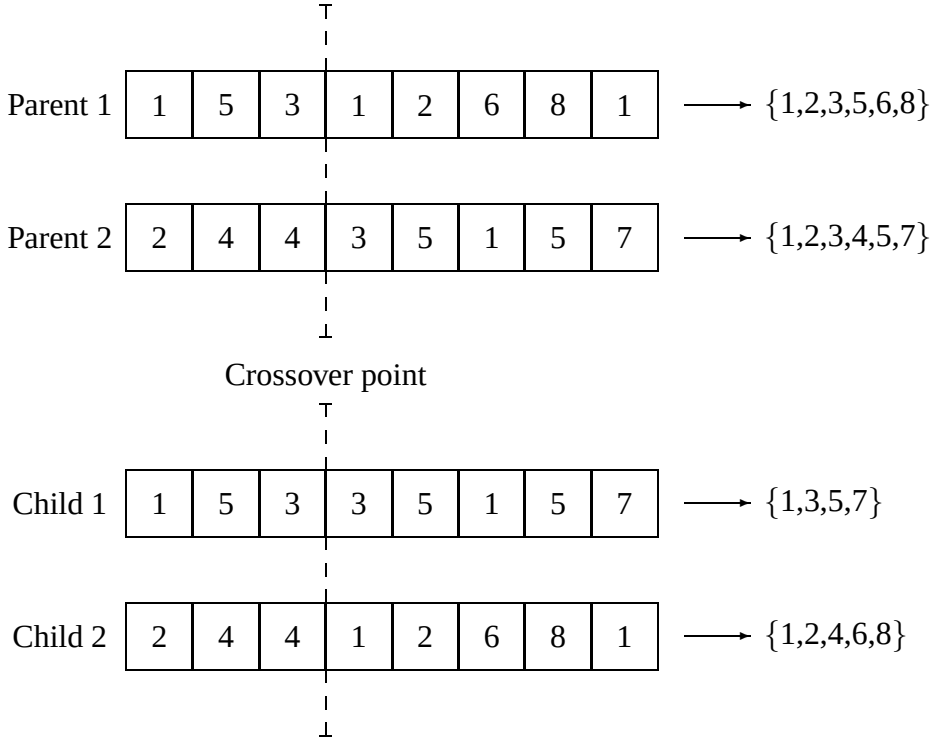

Figure 8.1: The one-point crossover operator for chromosomes of length 8

members of the new population to be formed by one-point crossover between two parents with no chromosomes carried through unchanged. Step 5 causes the most promising chromosome from the last population to be carried through to the new population automatically.

8.2.1 The neighbourhoods

For the experiments reported in this chapter, a total of 23 neighbourhoods were used to select subsets from for each problem. The random Kempe chain move neighbourhood and neighbourhoods 2-8 as detailed in section 7.3.1 are included together with the two "biased" neighbourhoods to select the first exam of a Kempe chain move based on the exam's contribution to the current overall penalty. To these were added 4 more Kempe chain move neighbourhoods, making from two to five random Kempe chain moves as well as a further 8 biased Kempe chain

- *Initialisation*: Set the following parameters:
  - Total number of generations, G
  - Population size, n
  - Number of runs of VNS per chromosome, v
  - Number of members of the population selected as parents for crossover, s
  - Fitness modifier, f
  - Mutation rate, m

  The initial population X of chromosomes $(x_1, \ldots, x_n)$ must also be created, either at random or by including some selection method.

- *Repeat* for G generations:
  1. Calculate fitness of each chromosome in X using v runs of VNS
  2. Select $s/2$ pairs of chromosomes from X using roulette wheel selection and a random crossover point for each pair
  3. Perform crossover between all $s/2$ pairs of parents to produce s children, $(c_1, \ldots, c_s)$
  4. Select $n - s - 1$ chromosomes, $(c_{s+1}, \ldots, c_{n-1})$, from the current population, X, using roulette wheel selection
  5. Set c_n to be the best chromosome from population X
  6. Perform mutation on all new chromosomes, $(c_1, \ldots, c_n)$ according to the mutation rate, m
  7. Set $X = (c_1, \ldots, c_n)$

Figure 8.2: The steps of VNS-GA procedure

neighbourhoods, 4 for each method of biased selection again making from two to five Kempe chain moves. The final neighbourhood included randomly permutes the ordering of all timeslots. These neighbourhoods were numbered as follows for the chromosome representation with 'Type A' representing the Kempe chain move with first exam selected being the one which gives the highest penalty from a random selection of 5% of the total exams and 'Type B' representing the Kempe chain move with first exam selected at random from the 20% giving the highest penalty:

1. Single random Kempe chain move
2. Swap two exams
3. Move two exams at random
4. Make two random Kempe chain moves
5. Move three exams at random
6. Make three random Kempe chain moves
7. Move four exams at random
8. Make four random Kempe chain moves
9. Move five exams at random
10. Make five random Kempe chain moves
11. Make one selective Kempe chain move (Type A)
12. Make one selective Kempe chain move (Type B)
13. Make two selective Kempe chain moves (Type A)
14. Make two selective Kempe chain moves (Type B)
15. Make three selective Kempe chain moves (Type A)
16. Make three selective Kempe chain moves (Type B)
17. Make four selective Kempe chain moves (Type A)
18. Make four selective Kempe chain moves (Type B)
19. Make five selective Kempe chain moves (Type A)
20. Make five selective Kempe chain moves (Type B)

21. Move a whole time slot

22. Swap time slots

23. Randomise time slots

8.3 Results

VNS-GA was tested on the 11 benchmark problems used throughout this thesis with certain parameters varied between data sets. These parameters were those which affect the time taken by the technique since my aim was to run VNS-GA on all problems for a similar amount of time rather than for the exact same number of runs of VNS. The total number of neighbourhood combinations evaluated [1] by running VNS at step 1 of figure 8.2 is given by $G \times n \times v$ with v of course determining how many times the same neighbourhood set is tested for a given chromosome.

In order to run all problems for roughly the same amount of time, I calculated an approximation of the time taken for a single VNS run to give a value for $G \times n \times v$ for the given amount of time. This was very approximate as run times can vary greatly for a single problem, but it was not important that all problems had exactly the same time since their results are not being compared against each other. For the larger data sets, v was always set to 1, with G and n either balanced equally or in a 2:1 or 1:2 ratio. Mutation rates of 0.002 (0.2%) and 0.01 (1%) were both tested with fitness multiplier, f, set at either 1.01 or 1.05.

The main aim of this work is to discover the potential performance of the VNS algorithm under favourable conditions (selected neighbourhoods) rather than to evaluate the GA itself. For this reason, I deliberately chose not to carefully tune the many GA parameters for my experiments. Instead I simply selected

[1] including multiple evaluations of the same neighbourhood set

a few sets of values which looked to give a balanced setup to run my experiments. If the method is to be used by exam timetablers who may not be experts in genetic algorithms, it is important to examine the performance with a set of default parameters rather than after careful tuning. The experiments reported here are based on a variety of different parameters, but these were not *tuned*, just selected in advance with no knowledge of how any would perform. The version of VNS used is the descent-ascent Biased VNS with the LD heuristic with randomisation used to give an initial solution as this proved to be the best combination overall from Chapter 7. The greedy initialisation heuristic was not used here as it is less reliable at finding feasible initial solutions and gives a lower diversity of solution.

In section 8.3.1 I will discuss future work aimed at improving the performance of the GA itself, but here the focus is on the performance of the VNS technique with different neighbourhoods in order to show that results can be improved by selecting subsets of the large set of neighbourhoods which may be more suitable to a problem so that more time can be spent searching these neighbourhoods rather than those which are not contributing. From the results obtained in these experiments, it should be possible to determine how effective this technique is so that the GA itself can then be more carefully considered to improve performance even further.

Table 8.1 gives the best results found by the VNS-GA algorithm on each data set and gives a comparison with the previous best result from the VNS variations presented in Chapter 7 and the best known solutions reported in table 2.3. The neighbourhoods which were used to produce the best solutions are also given. It can easily be seen that the VNS-GA method improves on the performance of the descent-ascent Biased VNS with the 10 neighbourhoods used previously. Experiments were also carried out using the full set of 23 neighbourhoods with VNS for all problems. In all cases these results were at least as good as the previous best results with the 10 neighbourhoods, but were not as good as the

Data Set	Best Reported Solution	Previous Best VNS Solution	Best VNS-GA Solution	Neighbourhood Subset For Best Solution
CAR-S-91	**4.6**	4.9	**4.6**	{1,4-8,11-13,16,17,19-23}
CAR-F-92	4.0	4.1	**3.9**	{1,3-6,8-11,13-17,19-23}
EAR-F-83	**29.3**	33.1	32.8	{1,3,4,7,11,13-15,17,21-23}
HEC-S-92	**9.2**	10.3	10.0	{1-4,6,8,10-12,14,16,17,19-22}
KFU-S-93	13.5	13.2	**13.0**	{2,4,6,8-10,12-15,17-20,22}
LSE-F-91	**9.6**	10.4	10.0	{2,3,5-8,10,13,15-17,19,20,22,23}
STA-F-83	**134.9**	156.9	156.9	Many
TRE-S-92	8.0	8.3	**7.9**	{2,4,7-12,15,19,21-23}
UTA-S-92	**3.2**	3.3	**3.2**	{1-9,13,18-22}
UTE-S-92	**24.4**	24.9	24.8	{1-3,5-10,13-17,19,20,22,23}
YOR-F-83	36.2	36.3	**34.9**	{1,5,6,9,10,12-14,16,17,19,21,22}

Table 8.1: Best results obtained from the VNS-GA algorithm with neighbourhoods given

results from the VNS-GA. This indicates that some (or all) of the additional 13 neighbourhoods are adding something useful to the ability of VNS, but that selecting a subset of these 23 to focus more on gives still better results.

In order to test whether the neighbourhoods which provided the best solutions given in table 8.1 are consistently better than using all 23 neighbourhoods or whether they just happened to be the neighbourhoods which were being used in VNS when the random element of the algorithm output that best result, I carried out a further series of tests. In the VNS-GA, in most cases a set of neighbourhoods was only tested for use within VNS a single time ($v = 1$) with HEC-S-92 and UTE-S-92 being the only problems for which $v > 1$ was tested. Table 8.2 shows the results of running VNS 100 times with the neighbourhood sets suggested from my previous results. Results are compared to those produced by 100 runs of VNS with the full set of 23 neighbourhoods. Average run times for the technique are also given in both cases with experiments carried out on a P4 1.8 GHz Athlon PC.

Results from table 8.2 are fairly inconclusive with regard to the merits of using selected neighbourhoods rather than just using all 23. Using selected neigh-

Data Set	VNS with all neighbourhoods			VNS with selected neighbourhoods		
	Best	Average	Time (s)	Best	Average	Time (s)
CAR-S-91	4.7	**4.9**	2751	4.6	**4.9**	3084
CAR-F-92	4.0	4.2	1605	3.9	**4.1**	1686
EAR-F-83	32.9	34.2	175	32.8	**34.1**	162
HEC-S-92	10.2	**10.6**	28	10.0	**10.6**	28
KFU-S-93	13.2	13.6	633	13.0	**13.4**	673
LSE-F-91	10.1	**10.6**	359	10.0	10.8	345
TRE-S-92	8.3	8.4	244	7.9	**8.2**	218
UTA-S-92	3.3	**3.4**	2358	3.2	**3.4**	2040
UTE-S-92	24.9	25.1	67	24.8	**25.0**	73
YOR-F-83	35.2	**36.4**	124	34.9	36.6	126

Table 8.2: Results from VNS comparing all neighbourhoods with the 'best' subset of neighbourhoods

bourhoods, the average result across 100 runs is better in five of the problems whilst using all 23 neighbourhoods provides a better average for two of the problems. In all cases the difference between the two sets of results is small. The STA-F-83 data set was excluded from further experiments because the best solution found by VNS is always 156.86 irrespective of the technique or neighbourhood selection and this solution is found regularly across 100 runs. Further analysis would be needed to investigate the reasons for this strange behaviour, but for the purposes of this research, VNS produces a competitive result on the data set with any selection of neighbourhoods.

8.3.1 Notes on Results

There are a number of points of note from the results presented in tables 8.1 & 8.2 together with the raw data produced by the experiments. It is clear that the VNS-GA is capable of producing the best results of all the VNS variants tested on all problems, although on many problems the difference between this method and supplying the VNS algorithm with all 23 neighbourhoods is relatively small. Also the use of selected neighbourhoods with the same stopping criteria, based on the number of iterations without improvement, gives no advantage in terms of

the time taken by the algorithm to find solutions. However, it is the case that the selected neighbourhoods presented in table 8.1 allowed VNS to obtain the best results whereas no other combination of neighbourhoods is proven to be able to produce results of that quality. That said, due to the very random nature of the way VNS works, selecting moves at random (or with some element of bias) from each neighbourhood in turn and bringing the resulting solution to a local minimum, it is unrealistic to say that any combination of neighbourhoods will categorically outperform every other set.

A major aim of this work was to show that VNS can provide highly competitive results in the right circumstances and the results shown prove that this is the case, irrespective of whether the GA was successful at selecting neighbourhoods or not. Having proved that VNS is capable of such high quality results, the next step is to give VNS the best chance of repeating that quality of results on a consistent basis. Comparing the average results from table 8.2 with the best results reported in the literature (table 2.3) confirms that even the average performance over 100 runs of VNS can outperform the best results of a number of specially designed meta-heuristic techniques.

As regards the effectiveness of the GA for selecting neighbourhoods, clearly more research will need to be carried out to determine whether the method is successful at evolving the 'best' subsets of neighbourhoods for a given problem, but results from the relatively untuned GA presented here give a lot of promise. One obvious drawback of the method compared to a more conventional GA is that the fitness function is calculated using VNS which involves a very high random element meaning that the same chromosome will be evaluated with a different fitness every time, unlike a standard GA fitness function which returns the same fitness for identical chromosomes. This can be significantly improved by increasing the value of v to 10 or more and using the average VNS result across the v runs to calculate chromosome fitness from. This would also allow

the GA to *evolve* the neighbourhoods more obviously than it does currently.

In the current GA implementation, with $v = 1$, the total fitness of each successive generation is very varied for most problems with only the larger data sets tested showing an obvious initial evolution. This is largely because of this variable fitness calculation causing a chromosome which may have been given a high fitness in one generation to have a relatively low fitness in the next generation. The higher the value of v, the more similar the fitness evaluation will be for two identical chromosomes meaning that when the fitter chromosomes are selected they represent sets of neighbourhoods which consistently give high quality results rather than just being able to produce a single good result. Further tuning of the other parameters of the GA, especially the population size, fitness multiplier and the mutation rate could also significantly improve the consistency of its performance. Combined with increasing the number of VNS runs per chromosome, v, tuning the fitness multiplier will give a much more consistent selection of the best chromosomes resulting in a convergence of the GA to a set of neighbourhoods which should be capable of giving highly consistent performance both in terms of best and average results.

My implementation contains just 23 neighbourhoods, many of which are very similar in terms of what they do. It is possible however to implement a huge variety of far more complex neighbourhoods which can easily be added into VNS to give a much larger pool of neighbourhoods. Some of these could be specifically designed with particular problems in mind. My work in Chapters 4-6 of this thesis was largely concerned with studying the features of 11 benchmark datasets and their behaviour when optimised using simulated annealing. Whilst that research was mostly aimed at defining a similarity measure, it can also be of use in developing new neighbourhoods for use in VNS. The more that is known about the structure of a given problem, the easier it may be to develop a neighbourhood specifically for that problem which takes into account its main features and how

they affect the search process.

In this way, the similarity measure we develop for comparing exam timetabling problems with the aim of retrieving a suitable technique to solve a new problem is more likely to be accurate. Without knowledge of exactly which problem features cause what behaviour in an algorithm, the theory behind our CBR system becomes less certain. It may be the case that two problems are matched as similar, but that the key features which cause a given technique to be successful on the first problem are not actually similar in the second problem whilst features which are less important to the performance of the technique in question are similar. If VNS is used with a set of problem specific neighbourhoods, or to be more precise, problem-feature specific neighbourhoods as well as a group of more general neighbourhoods, there is a much higher degree of certainty that VNS with those problem-feature specific neighbourhoods will successfully solve two problems which are similar based on that feature. Of course, developing neighbourhoods that can specifically combat certain difficult problem areas is not necessarily possible, but the ability to incorporate as many neighbourhoods as the user likes into VNS does make it highly versatile. In section 8.4 I consider how this VNS-GA implementation can be combined with CBR for our meta-heuristic selector.

8.4 Combining VNS-GA with CBR

One of the major reasons for investigating the intelligent selection of neighbourhoods for specific problems was to be able to use VNS as the major technique within our case based reasoning meta-heuristic selector. Rather than match two problems for similarity and retrieve the *technique* used for the problem in the case base to apply to the new problem, the CBR system would retrieve VNS automatically and use the set(s) of neighbourhoods stored with the problem in the case base to apply to the new problem. This would eliminate the need to include a

large variety of techniques within the case base whilst also considering variations of those techniques and which neighbourhood structure to use within a single-neighbourhood search algorithm. As discussed in section 7.5, a pre-requisite for this was to improve the performance of VNS so that it could produce best known solutions for more problems and be as competitive as possible on other problems.

The results presented in this chapter indicate that VNS is capable of producing (or matching) best known solutions for a number of benchmark data sets and providing results on others which are very close to the best known. These results come from selecting the *best* subset of neighbourhoods for the individual problem resulting in different sets of neighbourhoods being used for each of the problems considered. From a CBR perspective this is exactly what is required, especially if the improvements to the GA suggested in section 8.3.1 are capable of improving its performance and consistency.

To further improve the chances of the CBR system retrieving a successful set of neighbourhoods to apply with a new problem, the best 5 or 10 subsets of neighbourhoods for each problem in the case-base can be stored giving the user a choice of neighbourhood sets to use if the performance of the first set retrieved is unsatisfactory. Since no single set of neighbourhoods can be proved to outperform all others on a given problem, this should result in 5 or 10 very competitive subsets of neighbourhoods being stored for all problems so that the chances of retrieving a set which will perform successfully on a new problem are very high, provided our similarity measure can successfully identify a matching problem.

Given the results in table 8.2, it is not obvious that running VNS with specially selected neighbourhoods will give noticeably better performance than just using all neighbourhoods with VNS rather than use the CBR system at all. However, with the addition of more and more neighbourhoods and the suggested improvements to the GA, I believe that the consistency of the system can be im-

proved to a level at which VNS with intelligently selected neighbourhoods can consistently outperform VNS with all known neighbourhoods.

The main advantage the CBR system would have over a user just applying the VNS-GA directly to their new problem is that it can save a lot of time. Applying VNS-GA directly to a new problem is likely to give a higher quality result than applying VNS with the neighbourhoods suggested by CBR from some similar problem, but on large problems it can take of the order of weeks to successfully run the GA, especially with high values of v which is essentially a straight time multiplier - increasing v from 1 to 10 will cause the algorithm to take 10 times as long to complete its evolution. This means that for a real life problem where the user may have only two weeks to find a solution, there is not sufficient time to fully utilise the VNS-GA directly on the problem. Retrieving an already evolved good set of neighbourhoods from the case base means that more time can be spent finding a good solution using those neighbourhoods as oppose to evolving them.

Since the exam timetabling problem only needs to be solved two or three times per year at most institutions there is a large amount of time between timetabling sessions during which very little can be done since the problem is not yet fully known. The big advantage of the CBR system is that the VNS-GA algorithm can be working in the background during all this time that the system is not needed, to evolve the best sets of neighbourhoods for the problems in its case base. Therefore, whilst a user may only have two weeks to find a solution to their actual problem, the VNS-GA can spend 20-30 weeks evolving neighbourhoods to problems in the case base. Therefore when the new problem is presented to the system, there has already been a large amount of time spent evolving neighbourhoods for a problem *similar* to the new problem. Clearly the results would potentially not be as good as spending 20 weeks applying VNS-GA to the new problem itself, but since that is totally un-viable, the ability to

still use that 20 weeks in which an exam timetabling system would otherwise be fairly dormant to improve the system's knowledge is a big advantage.

The GA representation proposed here can also be extended to select from the many variants of VNS proposed in section 7.3.2 since many of these can be added or subtracted in a component manner.

8.4.1 More complex timetabling problems

Throughout this thesis I have focused my research on the same set of benchmark problems which have only a single soft constraint - that of spreading clashing exams around the timetable. Of course, real world timetabling problems will have a variety of other constraints, both hard and soft, applied to them and our CBR system will need to be able to handle those if it is to be at its most useful. Some of the most common side constraints applied to real world exam timetabling problems are given in section 1.1.2 and by Carter & Laporte [43]. One of my main motivations in investigating VNS rather than other local search techniques for this project was that its component nature makes it easy to add in new neighbourhoods and many of these can be specifically designed to deal with a variety of side constraints which would be less easy to incorporate into a more standard single neighbourhood trajectory search technique.

Resource constraints

The most important hard constraint after the one considered throughout this paper of conflicting exams not clashing is that there must be sufficient rooms and other resources available for each period for all exams scheduled during that period. A number of algorithms exist for assigning exams to rooms, either as part of an optimisation technique or separate. Other resource constraints can be considered similarly to rooms.

For the work presented in this thesis, I have used a largest degree construction technique to produce initial feasible results for local search techniques which then search only the feasible search space. For an initial solution to be fed to VNS, it is less important that this should be feasible with respect to room capacities since new neighbourhoods can easily be added to VNS specifically to move exams between rooms or to assign or un-assign an exam to a given room. Equally, it could be specified that the initial solution must be feasible with respect to the room constraint also and VNS will never move an exam to a period without sufficient seating capacity. The latter technique may be better for problems which are very highly constrained with respect to room capacities, but the former provides a much more flexible approach and VNS should be capable of finding feasible solutions on less highly constrained problems with respect to capacity. Allowing infeasible solutions with respect to room capacities during the search also means that every Kempe chain move remains feasible, whereas otherwise many of these will become infeasible losing one of the major advantages of the neighbourhood.

Certain problems include constraints that two exams cannot take place in the same room or that an exam should not be split across rooms. Both these constraints can be dealt with in the same way as the standard capacity constraint in terms of whether an exam is allowed to be moved to a given timeslot or not if no room is available. Again, if this constraint causes the problem to be highly constrained with respect to room capacities, it is desirable that the initial solution is feasible with respect to this constraint so that VNS will never enter the infeasible search space. Otherwise, the neighbourhoods should be capable of moving exams around in order to obtain feasible solutions whilst also searching the space of infeasible solutions with respect to this constraint. Also, it is generally undesirable to have exams of different lengths in the same room. Considered as a soft constraint this can be incorporated into the room assignment/swap neigh-

bourhoods.

Time constraints between events

Exams which have precedence constraints or which must be scheduled at the same time as each other or in consecutive time slots can be simply dealt with by enforcing these priorities during initial solution construction after which VNS would regard moves which violate these constraints as infeasible in the same way as the clashing constraint. For a standard local search technique, this reduction in the feasible search space can result in problems with the search space potentially becoming very disconnected, the more infeasible solutions there are, especially since these techniques conduct their search along a trajectory. VNS however is not confined to either a single neighbourhood definition or a trajectory based search therefore provided sufficient neighbourhoods are included, the search should still potentially be able to reach most, if not all areas of the search space. In this case, the problem specific selection of neighbourhoods by the GA can become very important if the user does not wish to include every possible neighbourhood considered.

If these constraints are included as soft constraints they can be incorporated into the objective function meaning that only problems with an equivalent objective function within the case base can be retrieved as a match. The VNS algorithm can then be applied as normal with the addition of specific neighbourhoods which can move a given pair of exams together, for example a pair of Kempe chain moves initiated by the two exams in question so that they are both guaranteed to be scheduled as required with respect to each other.

Time windows and pre-assignments

Many institutions include time window constraints or desire specific exams to be scheduled at a certain time. These constraints can be either hard or soft depend-

ing on the institution and would be dealt with differently depending which is the case. If they are present as hard constraints, pre-assignments can be scheduled before all other exams in the initial solution construction and then forbidden to be moved throughout the rest of the search. Similarly, exams with time window constraints can have these enforced in the initial solution also. As above, this would cause the Kempe chain neighbourhoods to lose their property that all moves within the neighbourhood are feasible.

Alternatively, new neighbourhoods can be introduced which specifically target the exams with time window constraints to move these around the timetable, either allowing temporary infeasibility during the search or in order to repair an infeasible initial solution. If time windows are present as soft constraints, these neighbourhoods would become much more influential allowing the algorithm to focus regularly on those exams involved in time window constraints. Similarly, if pre-assignments are only soft constraints, a special neighbourhood can be included which simply moves the exam in question to its favoured timeslot every time it is called to check if in doing so the objective function improves. Moves which involve taking the exam out of its favoured timeslot could then be considered in the standard way and accepted only if they meet the acceptance criteria - the inclusion of the above mentioned neighbourhood would increase the chances of the exam being moved back to its favoured slot later in the search.

A similar constraint may be placed on large exams, albeit less restricted. It is often desired that large exams are scheduled towards the beginning of the timetable, but with no specified time window as to the latest allowed slot. Again, in this case a number of neighbourhoods can be included which specifically target these large exams, attempting to move them closer to the beginning of the timetable. With a technique such as simulated annealing, this would likely have to be incorporated by specifically targeting the large exams at certain intervals of the search and either just using the single neighbourhood used throughout the

SA algorithm or some other more complicated method whereas VNS can easily include numerous neighbourhoods which target these large exams to move in different ways together with standard neighbourhoods.

Constraints on students

Constraints may exist on the students themselves rather than on the exams. For instance, a variation of the soft constraint considered within this thesis to spread exams around the timetable based on a given set of proximity penalties is that no student should take x exams within y periods. To incorporate this constraint, it is required to keep track of all individual student enrolments rather than just the conflict matrix, but from the point of view of the optimisation technique this constraint is very similar to that of spreading exams if it is included as a soft constraint. If violations of this constraint are recorded, neighbourhoods can be included to target the exams of a specific student. As with the examples of constraint specific neighbourhoods considered above, these would be included together with more standard neighbourhoods which consider the larger picture so that the focus is not entirely on specific students.

If there are part time students taking a particular course, they may be restricted to only being available in evening or weekend periods. This constraint can be considered in an identical way to the constraint on exams not being allowed to be in certain periods. Neighbourhoods targeting these exams would automatically only consider their pre-defined feasible periods.

8.5 Conclusions

In this chapter I have presented a genetic algorithm to intelligently select a subset of neighbourhoods to use within VNS for a given problem. A set of 23 neighbourhoods was considered, but this can be increased by adding many more

diverse neighbourhoods which may be much more suited to certain problems than others. The neighbourhoods so far considered are much more general rather than targeted at specific problem areas. As a result of this and the GA not having been carefully tuned (a deliberate effect), the results of the VNS-GA compared to VNS applied with all 23 neighbourhoods are not as impressive as they perhaps could be, but still yield very high quality results.

Best known solutions compared to those published at the time of writing have been found to four benchmark data sets with solutions equal to the best known found for two more data sets. Results on a further five problems are also highly competitive with state of the art techniques, indicating that the proposed VNS method is capable of producing high quality solutions even better than those presented in Chapter 7 under the right conditions. Those conditions involve the evolution of a set of neighbourhoods which can provide better solutions than simply including all implemented neighbourhoods within VNS.

A number of methods for improving the performance of the GA with particular focus on its consistency have been proposed. Most notable amongst these is to run VNS 10 or more times for each chromosome to take an average result for the fitness function rather than a single run. This should result in a much better evolution of neighbourhoods, but at the cost of significantly increased running time. The proposed conjunction with our CBR system alleviates the problem of running time however by allowing the GA to evolve the best sets of neighbourhoods for problems within the case base whilst the system would otherwise not be being used. This implementation would simplify the case base by including one dominant technique for which neighbourhoods are selected by the matching process rather than incorporating many techniques which would all have to be tuned for individual problems.

Methods for dealing with the additional side constraints which are often included in real world problems were also discussed, showing that the VNS tech-

nique is highly versatile compared to many local search techniques. This is due to its ability to deal with additional side constraints in a largely modular manner by adding new neighbourhoods since the high level VNS method does not need problem specific knowledge itself, only the knowledge it receives from the set of neighbourhoods with respect to their effect on the objective function.

Chapter 9

Conclusions and Future Work

This thesis has investigated some of the key areas relating to the development of a case based reasoning (CBR) meta-heuristic selector for exam timetabling problems. Contributing to a wider project, the work in this thesis concentrates on the low level aspects of a CBR system concerning the domain specific elements. Amongst these, one of the most important aspects is how *similarity* can be measured between two exam timetabling problems in such a way that a meta-heuristic used successfully to solve one will also successfully solve the other.

Involved in this was a study of the structure of a number of benchmark data sets to examine how the definition of certain features must be carefully considered before they are used to measure similarity (Chapter 4) and a study of how these same data sets behaved when optimised, comparing two different objective functions and two different initialisations (Chapter 5). This work led into a more detailed quantitative analysis of many of the features of exam timetabling data sets (Chapter 6) to select which features should be fed into a knowledge discovery process to determine exactly which smaller set of features would be used to measure similarity.

Another important aspect considered in this thesis was the meta-heuristic techniques themselves and the issues which need to be resolved in order for the

CBR theory to hold for the system. A detailed study of current state of the art techniques, both meta-heuristic and other methods was carried out (Chapter 2) to give a good knowledge of which techniques are most successful. It was considered that two major factors had the largest impact on the success of a given method, these being the type of technique itself and the neighbourhood used to perform the local search. With inconclusive results about *why* certain techniques outperformed others on some problems, but not on others, I decided to focus on the neighbourhood rather than the technique as I felt this was the most important factor.

With a potentially large number of different neighbourhoods which could be used within local search for exam timetabling, I opted to implement a variable neighbourhood search (VNS) capable of incorporating all these rather than having to select a single one (Chapter 7). Results from the basic VNS showed that the approach was successful across a range of problems, success which was further increased by the addition of some variations. One of these variations which provided highly successful results and was capable of a high level of generality was the use of a genetic algorithm (GA) to select a subset of neighbourhoods, most suitable to the individual problem, to use within VNS (Chapter 8).

9.1 Measuring Similarity for CBR

Probably the most important part of any CBR system is the measure of similarity between two problems, without which the best case cannot be retrieved for reuse with a new problem. For the project to which this thesis contributes, this similarity measure must match two exam timetabling problems as being *similar* if the same meta-heuristic technique could be applied to both to give high quality results. Ultimately, the set of features to be used within the system would be decided by a knowledge discovery process which takes a set of problem features,

together with ratios between all pairs and aims to discover the feature vector [1] from this set which gives the best performance for the CBR system. Before this could take place however, it was required to study the features of exam timetabling problems more carefully.

The investigation, reported in Chapter 4, of some of the most obvious features which could be used showed that some of these could give a highly misleading measure as to the similarity between two problems. Most obviously, it was found that although the number of students in a problem is often reported as a key feature to show the problem size and potential difficulty, it is a very misleading statistic. Similarly the total number of enrolments was found to be a very bad indicator of problem difficulty. This of course implied that any statistic based on either of these could also not be used to measure similarity unless redundancy in the problem definition could be stripped away. This led to a study regarding which students in the problem definition are actually adding knowledge and which are simply adding 'noise' to the problem definition. A key subset of students, defining the conflict matrix, was identified as being important for problems without capacity constraints, but even this set can be misleading if used in a similarity measure.

In Chapter 5, I examined the effect on the success of a simulated annealing algorithm of changing the objective function slightly and of using different initialisations for the technique. It was found that even with the same side constraints considered, a change in the objective function could lead to very different behaviour of the optimisation algorithm. A data set which is optimised consistently across a number of runs using one objective function was found to give far less consistent results when optimised by SA using a different objective function, showing that whilst SA may be good on this problem with the first objective function, it is less successful when the objectives are changed slightly. Similarly,

[1] set of features

the quality of initial solution was found to have an impact on how successful SA was across the range of problems. Whilst some problems rely quite heavily on a good initialisation, others were less reliant when using the same technique and neighbourhood structure both times.

Due to the fact that knowledge discovery techniques are being used to determine the exact subset of features which will be used to for the similarity measure within our CBR system I have been unable to give concrete conclusions in this thesis as to which features are the most important. From the results presented by Burke et al [37] using slightly different sets of features from those suggested here, it is likely that the actual feature set found by the knowledge discovery process will consist mostly of ratios, between two of the input features, which may not otherwise have been considered as most important. Also, depending on the number of features used, different sets features give best performance for the system.

In Chapter 6, I gave an analysis of many of the features which are considered to be most important in the definition of an exam timetabling problem. Together with some of the features tested in [37], I considered the importance of a number of more complex features such as an analysis of clique structures and the fluidity of the exams within the problem. Clique analysis may be difficult to calculate for a new problem given to the system to be solved, but since it is an important structural aspect I considered it important to include. Ultimately the knowledge discovery process will select from the set of features proposed, and all ratios of their pairs, which give the best performance for the CBR system.

9.2 Developing the meta-heuristic technique(s)

After the definition of a similarity measure, one of the most important parts of a CBR system is the set of cases which form the knowledge of the system. For

this project, each case would consist of a set of feature-value pairs defining the problem, as detailed by the similarity measure together with the meta-heuristic technique(s) which gives best results on the problem and an indication of how successful the technique was. In this thesis I restrict my experiments to a set of benchmark problems to which many techniques have been applied with differing success. These provide enough variation in difficulty, whilst including the key hard constraint of not scheduling clashing exams at the same time, that they can be considered a good starting point to add in further problems.

I have discussed throughout this thesis the importance of two aspects of a local search meta-heuristic, the technique itself and the neighbourhood used within the search. Clearly both aspects are important. However, with a view to measuring similarity between problems in order to select a technique to use, I considered that there was not enough evidence in results presented in the literature to suggest that two different methods (e.g. simulated annealing and great deluge in Burke et al. [12]) using the same neighbourhood structure were sufficiently different across a range of problems.

For the CBR assumption that the same technique will perform equally well on two similar problems, we need to be able to more strongly distinguish between two techniques and know *why* one is successful on certain problems and not on others. Results reported by Thompson & Dowsland [101, 103] indicated that the neighbourhood used within the search could have a large impact on results, albeit that one neighbourhood significantly outperformed another across all problems. For this reason, I chose to focus on developing a variable neighbourhood search (VNS) technique to utilise a potentially large number of neighbourhoods.

In Chapter 7, I introduced the VNS technique together with the initial set of neighbourhoods developed and a description of some of the large number of variations which can be applied. Results presented indicated that the basic VNS method without any variations produced high quality results across the full range

of data sets tested and that adding some variations could improve the results still further to produce one best known result and competitive results on all other problems. This showed clearly that focusing on the neighbourhood structure rather than the technique used with a single neighbourhood can yield a much more general technique.

In order to combine VNS with CBR, I developed a GA to select, from a larger set of neighbourhoods, the best subset to apply within VNS for each problem. The results were reported in Chapter 8 and showed that the proposed method is capable of finding very high quality solutions, with six best known solutions [2] out of 11 benchmark problems and very close results on four others. The VNS-GA technique can be combined with a CBR framework to select a set of neighbourhoods to use with VNS for a new problem now rather than to select a type of meta-heuristic. This has many advantages, meaning that we do not have to test a large number of meta-heuristics with all neighbourhoods and sets of parameters to find the best for each problem in the case base.

I also reported that results when using all 23 neighbourhoods considered were also very competitive so that improvements would need to be made to the GA so that evolution of problem-specific neighbourhoods could more obviously outperform VNS using all neighbourhoods. The proposed VNS-GA does however allow a user to add as many new neighbourhoods as they like, which can be very problem specific to deal with specific constraints, allowing the VNS-GA to determine the best subset(s) of neighbourhoods to store with each problem in the case base. With more problem-specific neighbourhoods, the similarity measure and CBR theory of similar problems being solved using a similar technique is much more solid. A discussion of how new neighbourhoods can be incorporated into VNS to deal with additional side constraints not covered in this paper was also given, showing the versatility of the technique.

[2] Four absolute best known solutions and two equal to the best reported so far

9.3 Future Work

An ambitious project such as this in the area of combinatorial optimisation will always have a number of potential improvements which can be made and some of those are detailed here.

The knowledge discovery process to discover the best set of features from those proposed in Chapter 6 is ongoing and further features may be considered as this progresses. Much of the discussion of cliques in section 6.2.8 centred around features which are difficult to calculate, indeed finding the largest clique of a graph is itself an NP-hard problem. As such, further research needs to be carried out to decide which features of cliques can sensibly be used for a similarity measure and which are more likely to be misleading or add no useful information. Similarly, the issue of side constraints and the objective function discussed in section 6.2.9 can be combined with the suggestions in section 8.4.1 to expand the case base to include a much wider range of side constraints on problems. There are still a relatively limited number of publicly available data sets on exam timetabling, but there are some which include these additional side constraints and more can be developed.

Much of the potential future work resulting from this thesis concerns the further development of the VNS technique both in terms of its performance on benchmark problems and its ability to deal with a much wider range of problems and combine even better with CBR. Basic VNS is a very simple technique, as detailed in figure 7.1 and with a very component-based nature can be easily modified at just about any step of the process. Some of these variations were successfully tested and reported in Chapter 7 whilst others were suggested and either not tested or tested, but without yielding any positive results. Those which were not tested were largely because they would require much larger development and running time than I had available. It is possible that these can improve

the performance of VNS further if studied in detail, for instance the VND variation which uses a variety of neighbourhoods during the local search part of VNS has been found to be very successful in other domains, but would require careful consideration since the current method is very fast and a large increase in running time for the local search would be undesirable since it is called at every iteration of the VNS.

Similarly, I tested the use of both great deluge and simulated annealing in place of the simple steepest descent method for the local search, however both resulted in vastly increased running time with no improvement in solution quality. Further research would need to be conducted to determine if either of these techniques (or others) could be successfully incorporated without causing running times to be excessively long. Also suggested in section 7.3.2 was to investigate a variety of different initialisation strategies. In my work I tested just two, but many other highly successful techniques exist for producing fast initial solutions which could result in further improvement to VNS or a decrease in the running time. The adaptive heuristic ordering technique of Burke & Newall [31] in particular could provide a very good initialisation.

The most obvious further work which can be carried out within the VNS framework is the development of yet more neighbourhoods since these are the most important aspect of the method and its ability to vary the neighbourhood is its biggest strong point. As discussed in section 8.4.1, when dealing with more complex problems with additional side constraints, the VNS method can really come into its own, with very specific neighbourhoods being developed with particular side constraints in mind. Since the standard VNS method used in this thesis selects a move from the neighbourhood at random, it is very simple to implement new neighbourhoods to test without having to consider an exhaustive search of the neighbourhood. The biased neighbourhoods so far implemented were also deliberately chosen to be quick and easy to implement and run by

using a biased selection of exam to move rather than evaluating many moves to select the best.

The addition of many more neighbourhoods should increase the importance of the GA to intelligently select a subset of these for each problem. The proposed GA technique works in the background of the CBR system when it is not in use, to evolve the best subset of neighbourhoods for the problems in the case base. This is an extremely flexible process which can run for as short or long a time as required, with a longer time of course allowing a more thorough search and better evolution of neighbourhoods. Using 10 or more runs of VNS to calculate the fitness of each chromosome from the average result will increase the running time ten-fold for the same population size and as such was infeasible for me to test with any degree of thoroughness, but should significantly improve the ability of the GA to *evolve* the best set of neighbourhoods. Currently the GA searches only slightly more well guided than a random walk would, but increasing the consistency of the fitness function when evaluating a given chromosome will allow it to evolve the set of neighbourhoods which gives consistently good results rather than just a single good result.

Further tuning of the parameters of the proposed VNS-GA could also yield notable improvements in the ability of the GA to find a set of neighbourhoods which, when tested over 100 runs will consistently outperform VNS seeded with all neighbourhoods. This is the most important aspect for the GA to be combined effectively with CBR. For use by itself, it is capable of producing very high quality results given enough time without needing a single set of neighbourhoods to work consistently. However, if the best set(s) of neighbourhoods are to be stored with a case to be retrieved when a similar problem is to be solved, we need to be as sure as we can be that this set of neighbourhoods produces consistently good results. Then, if the similarity measure works successfully, the CBR system will be capable of producing high quality results for a new problem far faster than

the VNS-GA would be able to by itself.

Since only the neighbourhoods and the similarity measure are domain specific in either the VNS algorithm or the CBR system, the technique should easily be able to be used for a wide range of other optimisation problems. This could be done simply by changing the set of neighbourhoods used within VNS to a set suitable for the new domain and conducting an investigation and knowledge discovery processes to develop a suitable similarity measure. This would however require further research to test whether VNS with a good selection of neighbourhoods is capable of producing high quality results in other domains also. My own feeling based on what I have seen from the work reported in this thesis and elsewhere is that the VNS technique should be capable of solving problems in a large number of domains and that it is only the selection of the neighbourhoods which can severely limit its performance. When compared to a technique which uses only a single neighbourhood, it has some significant advantages and is far more flexible.

Bibliography

[1] A. M. Abbas and E. P. K. Tsang. Constraint-based timetabling - A case study. *In: Proceedings of ACS/IEEE International Conference on Computer Systems and Applications*, Beirut, Lebanon, June 26-29, 2001, pp 67-72.

[2] S. Ahmadi, R. Barrone, P. Cheng, P. Cowling and B. McCollum. Perturbation based variable neighbourhood search in heuristic space for examination timetabling problem. *In: Proceedings of MISTA 2003 conference*, Nottingham, August 13-16, 2003, pp 135-171.

[3] D. T. Anh and L. K. Hoa. Combining constraint programming and simulated annealing on university exam timetabling. *In: Proceedings of RIVF 2004 conference*, Hanoi, Vietnam, February 2-5, 2004, pp 205-210.

[4] H. Arntzen and A. Løkketangen. A tabu search heuristic for a university timetabling problem. *In: Proceedings of the fifth meta-heuristics international conference, MIC 2003* (CD-ROM), Kyoto, August 25-28, 2003, MIC03-02.

[5] V. A. Bardadym. Computer-aided school and university timetabling: The new wave. *In*: E. K. Burke and P. Ross (eds). *Practice and theory of automated timetabling: Selected papers from the first international conference.* Volume 1153 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 1996, pp 22-45.

[6] A. Bezirgan. A case-based approach to scheduling constraints. *In*: J. Dorn and K. A. Froeschl (eds). *Scheduling of production processes,* Ellis Horwood Limited, 1993, pp 48-60.

[7] P. Boizumault, Y. Delon and L. Peridy. Constraint logic programming for examination timetabling. *Journal of logic programming*, 1996, 26(2), pp 217-233.

[8] S. C. Brailsford, C. N. Potts and B. M. Smith. Constraint satisfaction problems: Algorithms and applications. *European journal of operational research*, 1999, 119, pp 557-581.

[9] D. Brelaz. New methods to color the vertices of a graph. *Communication of the ACM*, 1979, 22(4), pp 251-256.

[10] B. Bullnheimer. An examination scheduling model to maximize students' study time. *In*: E. K. Burke and M. W. Carter (eds). *Practice and theory of automated timetabling: Selected papers from the second international conference.* Volume 1408 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 1998, pp 78-91.

[11] E.K.Burke, Y. Bykov and S.Petrovic. A multicriteria approach to examination timetabling. *In*: E. K. Burke and W. Erben (eds). *Practice and theory of automated timetabling: Selected papers from the third international conference.* Volume 2079 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 2001, pp 118-131.

[12] E.K. Burke, Y. Bykov, J. P. Newall and S. Petrovic. A Time-predefined local search approach to exam timetabling problems. *IIE transactions on operations engineering*, 2004, 36(6), pp 509-528.

[13] E. K. Burke and M. W. Carter (eds). *Practice and theory of automated timetabling: Selected papers from the second international conference.* Volume 1408 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 1998.

[14] E. K. Burke and P. De Causmaecker (eds). *Practice and theory of automated timetabling: Selected papers from the fourth international conference.* Volume 2740 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 2003.

[15] E. K. Burke, M. Dror, S. Petrovic and R. Qu. Hybrid graph heuristics in hyper-heuristics applied to exam timetabling problems. *University of Nottingham Technical Report* NOTTCS-TR-2004-1, accepted for publication by 9th Informs computing society conference, January 2005.

[16] E. K. Burke, A. J. Eckersley, B. McCollum, S. Petrovic and R. Qu. Similarity measures for exam timetabling problems. *In: Proceedings of MISTA 2003 conference*, Nottingham, August 13-16, 2003, pp 120-136.

[17] E. K. Burke, A. J. Eckersley, B. McCollum, S. Petrovic and R. Qu. Using simulated annealing to study behaviour of various exam timetabling data sets. *In: Proceedings of the fifth meta-heuristics international conference, MIC 2003* (CD-ROM), Kyoto, August 25-28, 2003, MIC03-09.

[18] E. K. Burke, A. J. Eckersley, B. McCollum, S. Petrovic and R. Qu. Analysing similarity in examination timetabling. *To be published in Proceedings of the fifth international conference on the practice and theory of automated timetabling*, Pittsburgh, USA, August 2004.

[19] E. K. Burke, D. G. Elliman, P. H. Ford and R. F. Weare. Examination timetabling in British universities: a survey. *In*: E. K. Burke and P. Ross

(eds). *Practice and theory of automated timetabling: Selected papers from the first international conference.* Volume 1153 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 1996, pp 76-90.

[20] E. K. Burke, D. G. Elliman and R. F. Weare. A university timetabling system based on graph colouring and constraint manipulation. *Journal of research on computing in education*, 1994, 27(1), pp 1-18.

[21] E. K. Burke, D. G. Elliman and R. F. Weare. A genetic algorithm for university timetabling. *In: Proceedings of the AISB Workshop on Evolutionary Computing*, University of Leeds, UK, April 11-13, 1994, Society for the study of artificial intelligence and simulation of behaviour (SSAISB).

[22] E. K. Burke, D. G. Elliman and R. F. Weare. A genetic algorithm based university timetabling system. *In: Proceedings of the 2nd East-West International Conference on Computer Technologies in Education*, Crimea, Ukraine, September 19-23, 1994, vol 1, pp 35-40.

[23] E. K. Burke and W. Erben (eds). *Practice and theory of automated timetabling: Selected papers from the second international conference.* Volume 2079 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 2001.

[24] E. K. Burke, K. Jackson, J. H. Kingston and R. Weare. Automated university timetabling: The state of the art. *The computer journal*, 1997, 40(9), pp 565-571.

[25] E. K. Burke, G. Kendall and E. Soubeiga. A tabu-search hyper-heuristic for timetabling and rostering. *Journal of Heuristics*, 2003, 9, pp 451-470.

[26] E. K. Burke, J. H. Kingston and D. de Werra. Chapter 5.6: Applications to timetabling. *In*: J. Gross and J. Yellen (eds.) *The Handbook of Graph Theory*, Chapman Hall/CRC Press, 2004, pp 445-474.

[27] E. K. Burke, B. MacCarthy, S. Petrovic and R. Qu. Structured cases in CBR - Re-using and adapting cases for timetabling problems. *Knowledge-Based Systems*, 2000, 13(2-3), pp 159-165.

[28] E.K. Burke, B. MacCarthy, S. Petrovic and R. Qu. Case-based reasoning in course timetabling: An attribute graph approach. *In: Proceedings of the 4th International conference on case-based reasoning (ICCBR-2001)*, Vancouver, July 30-August 2, 2001. LNAI 2080. pp 90-104.

[29] E. K. Burke and J. P. Newall. A multi-stage evolutionary algorithm for the timetable problem. *IEEE Transactions on Evolutionary Computation*, 1999, 3(1), pp 63-74.

[30] E. K. Burke and J. P. Newall. Enhancing timetable solutions with local search methods. *In*: E. K. Burke and P. De Causmaecker (eds). *Practice and theory of automated timetabling: Selected papers from the fourth international conference.* Volume 2740 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 2003, pp 195-206.

[31] E. K. Burke and J. P. Newall. Solving examination timetabling problems through adaptation of heuristic orderings. *Annals of operations research*, 2004, 129, pp 107-134.

[32] E. K. Burke, J. P. Newall and R. F. Weare. A memetic algorithm for university exam timetabling. *In*: E. K. Burke and P. Ross (eds). *Practice and theory of automated timetabling: Selected papers from the first international conference.* Volume 1153 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 1996, pp 241-250.

[33] E. K. Burke, J. P. Newall and R. F. Weare. Initialization strategies and diversity in evolutionary timetabling. *Evolutionary computation*, 1998, 6(1), pp 81-103.

[34] E. K. Burke, J. P. Newall and R. F. Weare. A simple heuristically guided search for the timetable problem. *In: Proceedings of the international ICSC symposium on engineering of intelligent systems (EIS'98)*, Rio de Janeiro, August 2-7, 1998, pp 574-579.

[35] E. K. Burke and S. Petrovic. Recent research directions in automated timetabling. *European journal of operational research (EJOR)*, 2002, 140(2), pp 266-280.

[36] E. K. Burke, S. Petrovic and R. Qu. Case-based heuristic selection for examination timetabling. *In: Proceedings of the SEAL'02 conference*, Singapore, November 18-22, 2002, pp 277-281.

[37] E. K. Burke, S. Petrovic and R. Qu. Case-based heuristic selection for timetabling problems. *University of Nottingham Technical Report* NOTTCS-TR-2004-2, accepted for publication in the Journal of Scheduling, 2005.

[38] E. K. Burke & P. Ross (eds): *Practice and theory of automated timetabling: Selected papers from the first international conference.* Volume 1153 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 1996.

[39] Y. Bykov. *The description of the algorithm for international timetabling competition* [online].
Available at http://www.idsia.ch/Files/ttcomp2002/results.htm [20 July 2004]

[40] M. Caramia, P. Dell'Olmo and G. F. Italiano. New algorithms for examination timetabling. *In*: S. Näher and D. Wagner (eds): *Algorithm Engineering 4th International Workshop, Proceedings WAE 2000 (Saarbrücken, Germany)*. Volume 1982 of Lecture notes in computer science, Springer-Verlag, Berlin, Heidelberg, New York, 2001, pp 230-241.

[41] M. W. Carter. A survey of practical applications of examination timetabling algorithms. *Operations research,* 1986, 34, pp 193-202.

[42] M. W. Carter and D. G. Johnson. Extended clique initialisation in examination timetabling. *Journal of the operational research society,* 2001, 52, pp 538-544.

[43] M. W. Carter and G. Laporte. Recent developments in practical examination timetabling. *In*: E. K. Burke and P. Ross (eds). *Practice and theory of automated timetabling: Selected papers from the first international conference.* Volume 1153 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 1996, pp 373-383.

[44] M. W. Carter, G. Laporte and J. W. Chinneck. A general examination scheduling system. *Interfaces*, 1994, 24, pp 109-120.

[45] M. W. Carter, G. Laporte and S. Y. Lee. Examination timetabling: Algorithmic strategies and applications. *Journal of the operational research society,* 1996, 47(3), pp 373-383.

[46] S. Casey and J. Thompson. GRASPing the examination scheduling problem. *In*: E. K. Burke and P. De Causmaecker (eds). *Practice and theory of automated timetabling: Selected papers from the fourth international conference.* Volume 2740 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 2003, pp 232-244.

[47] C. Cheng, L. Kang, N. Leung and G. M. White. Investigations of a constraint logic programming approach to university timetabling. *In*: E. K. Burke and P. Ross (eds). *Practice and theory of automated timetabling: Selected papers from the first international conference.* Volume 1153 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 1996, pp 112-129.

[48] T. B. Cooper and J. H. Kingston. The complexity of timetable construction problems. *In*: E. K. Burke and P. Ross (eds). *Practice and theory of automated timetabling: Selected papers from the first international conference.* Volume 1153 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 1996, pp 183-295.

[49] J-F. Cordeau, B. Jaumard and R. Morales. *Efficient timetabling solution with tabu search* [online].
Available at http://www.idsia.ch/Files/ttcomp2002/results.htm [20 July 2004]

[50] D. Corne, P. Ross and H-L Fang. Fast practical evolutionary timetabling. *In: Proceedings of Evolutionary Computing AISB Workshop*, Leeds, 1994, Volume 865 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 1994, pp 251-263.

[51] D. Corne, P. Ross and H-L Fang. Evolutionary timetabling: Practice, prospects and work in progress. *In*: P. Prosser (ed). *Proceedings of UK planning and scheduling SIG workshop*, Strathclyde, September 1994.

[52] P. I. Cowling, G. Kendall and L. Han. An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. *In: Proceedings of congress on evolutionary computation (CEC2002)*, 2002, pp 1185-1190.

[53] P. Cunningham and B. Smyth. Case-based reasoning in scheduling: Reusing solution components. *International journal for production research*, 1996, 35(11), pp 2947-2960.

[54] P. David. A Constraint-based approach for examination timetabling using local repair techniques. *In*: E. K. Burke and M. W. Carter (eds). *Practice and theory of automated timetabling: Selected papers from the second international conference.* Volume 1408 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 1998, pp 169-186.

[55] D. de Werra. An introduction to timetabling. *European journal of operational research*, 1985, 19, pp 151-162.

[56] M. Dimopoulou and P. Miliotis. Implementation of a university course and examination timetabling system. *European journal of operational research*, 2001, 130, pp 202-213.

[57] L. Di Gaspero. Recolour, shake and kick: A recipe for the examination timetabling problem. *In*: E. K. Burke and P. De Causmaecker (eds): *Proceedings of the fourth international conference on the practice and theory of automated timetabling*, Gent, Belgium, August 2002, pp 404-407.

[58] L. Di Gaspero and A. Schaerf. Tabu search techniques for examination timetabling. *In*: E. K. Burke and W. Erben (eds). *Practice and theory of automated timetabling: Selected papers from the third international conference.* Volume 2079 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 2001, pp 104-117.

[59] L. Di Gaspero and A. Schaerf. *Timetabling competition TTComp 2002: Solver description*. [online]
Available at http://www.idsia.ch/Files/ttcomp2002/results.htm [20 July 2004]

[60] K. A. Dowsland, N. Pugh and J. Thompson. Examination timetabling with ants. *In*: E. K. Burke and P. De Causmaecker (eds): *Proceedings of the fourth international conference on the practice and theory of automated timetabling*, Gent, Belgium, August 2002, pp 397-399.

[61] W. Erben. A grouping genetic algorithm for graph colouring and exam timetabling. *In*: E. K. Burke and W. Erben (eds). *Practice and theory of automated timetabling: Selected papers from the third international conference.* Volume 2079 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 2001, pp 132-156.

[62] C. G. Guéret, N. Jussien, P. Boizumault and C. Prins. Building university timetables using constraint logic programming. *In*: E. K. Burke and P. Ross (eds). *Practice and theory of automated timetabling: Selected papers from the first international conference.* Volume 1153 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 1996, pp 130-145.

[63] L. Han, G. Kendall and P. Cowling. An adaptive length chromosome hyper-heuristic genetic algorithm for a trainer scheduling problem. *In: Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning (SEAL'02)*, Singapore, November 18-22, 2002, pp 267-271.

[64] L. Han and G. Kendall. Guided operators for a hyper-heuristic genetic algorithm. *In: Proceedings of the 16th Australian conference on artificial intelligence*, Perth, Australia, 2003, pp 807-820.

[65] L. Han and G. Kendall. Investigation of a tabu assisted hyper-heuristic genetic algorithm. *In: Proceedings of congress on evolutionary computation*, 2003, vol 3, pp 2230-2237.

[66] P. Hansen and N. Mladenović. Variable neighbourhood search: Principles and applications. *European journal of operational research*, 2001, 130, pp 449-467.

[67] D. Henderson, S. H. Jacobson and A. W. Johnson. The theory and practice of simulated annealing. *In*: F. Glover and G. A. Kochenberger (eds): *Handbook of Metaheuristics*, Kluwer Academic Publishers, 2003, pp. 287-319.

[68] W. K. Ho, A. Lim and W. C. Oon. Maximizing paper spread in examination timetabling using a vehicle routing method. *In: Proceedings of 13th IEEE international conference on tools with artificial intelligence (ICTAI'01),* 2001, pp 359-366.

[69] G. Kendall and N. Mohd Hussin. An investigation of a tabu search based hyper-heuristic for examination timetabling. *In: Proceedings of MISTA 2003 conference*, Nottingham, August 13-16, 2003, pp 226-233.

[70] J. H. Kingston. Modelling timetabling problems with STTL. *In*: E. K. Burke and W. Erben (eds). *Practice and theory of automated timetabling: Selected papers from the third international conference.* Volume 2079 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 2001, pp 309-321.

[71] S. Kirkpatrick, C. Gelatt and P. Vecchi. Optimization by simulated annealing. *Science*, 1983, 220, pp 671-679.

[72] J. Kolodner. *Case-based reasoning*. Morgan Kaufmann Publishers, Inc. San Mateo, 1993.

[73] P. A. Kostuch. *Timetabling competition - SA-based heuristic* [online]. Available at http://www.stats.ox.ac.uk/ kostuch/TTcomp.pdf

[74] P. Koton. SMARTplan: A case-based reasoning allocation and scheduling system. *In: Proceedings of workshop on case-based reasoning (DARPA)*, 1989, pp 285-289.

[75] J. D. Landa Silva, E. K. Burke and S. Petrovic. An introduction to multiobjective metaheuristics for scheduling and timetabling. *In*: X. Gandibleux, M. Sevaux, K. Sorensen and V. T'kindt (eds): *Multiple objective metaheuristics*. Volume 535 of Lecture notes in economics and mathematical systems. Springer-Verlag, Berlin, Heidelberg, 2004, pp 91-129.

[76] A. Lim, J. C. Ang, W. K. Ho and W. C. Oon. A campus-wide university examination timetabling application. *In: Proceedings of the seventeenth national conference on artificial intelligence and twelfth conference on innovative applications of artificial intelligence*, 2000, pp 1020-1025.

[77] A. Lim, J. C. Ang, W. K. Ho and W. C. Oon. UTTSExam: A campus-wide university exam-timetabling system. *In: Proceedings of the eighteenth national conference on artificial intelligence and fourteenth conference on innovative applications of artificial intelligence*, 2002, pp 838-844.

[78] S. L. M. Lin. A broker algorithm for timetabling problem. *In*: E. K. Burke and P. De Causmaecker (eds): *Proceedings of the fourth international conference on the practice and theory of automated timetabling*, Gent, Belgium, August 2002, pp pp 372-386.

[79] B. MacCarthy and P.Jou. Case-based reasoning in scheduling. *In*: M. K. Khan and C. S. Wright (Eds): *Proceedings of the symposium on advanced manufacturing processes, systems and techniques (AMPST96)*, MEP Publications, 1996, pp 211-218.

[80] N. Mladenović and P. Hansen. Variable neighbourhood search. *Computers Ops. Res.*, 1997, 24(11), pp 1097-1100.

[81] A. Meisels, J. Ell-Sana and E. Gudes. Decomposing and solving timetabling constraint networks. *Computational intelligence*, 1997, 13(4), pp 486-505.

[82] L. T. G. Merlot, N. Boland, B. D. Hughes and P. J. Stuckey. A hybrid algorithm for the examination timetabling problem. *In*: E. K. Burke and P. De Causmaecker (eds): *Proceedings of the fourth international conference on the practice and theory of automated timetabling*, Gent, Belgium, August 2002, pp 207-231.

[83] K. Miyashita. Case-based knowledge acquisition for schedule optimization. *Artificial intelligence in engineering*, 1995, 9, Elsevier Science Ltd. pp 277-287.

[84] K. Miyashita and K. Sycara. Adaptive case-based control of scheduling revision. *In*: M. Zweben and M. S. Fox (eds): *Intelligent scheduling*, Morgan Kaufmann, 1994, pp 291-308.

[85] K. Miyashita amd K. Sycara. CABINS: a framework of knowledge acquisition and iterative revision for schedule improvement and reactive repair. *Artificial intelligence*, 1995, 76, pp 377-426.

[86] J. P. Newell. *Hybrid methods for automated timetabling*. University of Nottingham Ph.D. thesis, 1999.

[87] W. P. M. Nuijten, G. M. Gunnen, E. H. L. Aarts and F. P. M. Dignum. Examination time tabling: A case study for constraint satisfaction. *In: Proceedings of the ECAI '94 workshop on constraint satisfaction issues raised by practical applications*, 1994, pp 11-19.

[88] L. Pacquete and C. M. Fonseca. A study of examination timetabling with multiobjective evolutionary algorithm. *In: Proceedings of the 4th Metaheuristics International Conference (MIC 2001)*, 2001, pp 149-154.

[89] L. Pacquete and T. Stützle. Empirical analysis of tabu search for the lexicographic optimization of the examination timetabling problem. *In*: E. K. Burke and P. De Causmaecker (eds): *Proceedings of the fourth international conference on the practice and theory of automated timetabling*, Gent, Belgium, August 2002, pp 413-420.

[90] L. Pacquete and T. Stützle. An experimental investigation of iterated local search for coloring graphs. *In*: S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, and G. Raidl (eds): *Applications of evolutionary computing - EvoWorkshops 2002*. Volume 2279 of Lecture notes in computer science, Springer-Verlag, Berlin, Heidelberg, 2002, pp 122-131.

[91] S. Petrovic and E. K. Burke. Chapter 45: University timetabling. *In*: J. Leung (ed): *Handbook of scheduling: Algorithms, models, and performance analysis*, CRC Press, April 2004.

[92] S. Petrovic and Y. Bykov. A multiobjective optimisation technique for exam timetabling based on trajectories. *In*: E. K. Burke and P. De Causmaecker (eds). *Practice and theory of automated timetabling: Selected papers from the fourth international conference.* Volume 2740 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 2003, pp 181-194.

[93] S. Petrovic, G. Kendall and Y. Yang. A tabu search approach for graph-structured case retrieval. *In: Proceedings of the Starting artificial intelligence researchers symposium*, IOS Press, 2002, pp 55-64.

[94] S. Petrovic and R. Petrovic. Eco-Ecodispatch: DSS for multi-criteria loading of thermal power generators. *Journal of decision systems*, 1995, 4(4), pp 279-295.

[95] S. Petrovic, Y. Yang and M. Dror. Case-based initialisation of metaheuristics for examination timetabling. *In: Proceedings of MISTA 2003 conference*, Nottingham, August 13-16, 2003, pp 137-154.

[96] I. Phillips: *The Department of Computing Examination Timetabling Problem*. [online]
Available at http://theory.doc.ic.ac.uk/ iccp/papers/ [20 July 2004]

[97] R. Qu. Case based reasoning for course timetabling problems. University of Nottingham Ph.D. thesis, 2002.

[98] P. Ross, E. Hart and D. Corne. Some observations about GA-based exam timetabling. *In*: E. K. Burke and M. W. Carter (eds). *Practice and theory of automated timetabling: Selected papers from the second international conference.* Volume 1408 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 1998, pp 115-129.

[99] A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 1999, 13(2), pp 87-127.

[100] H. Terashima-Marín, P. Ross and M. Valenzuela-Rendón. Evolution of constraint satisfaction strategies in examination timetabling. *In: Proceedings of the genetic and evolutionary conference*, Orlando, Florida, July 13-17 1999, pp 635-642.

[101] J. Thompson and K. Dowsland. Variants of simulated annealing for the examination timetabling problem. *Annals of Operational Research*, 1996, 63, pp 105-128.

[102] J. Thompson and K. Dowsland. General cooling schedules for a simulated annealing based timetabling system. *In*: E. K. Burke and P. Ross (eds). *Practice and theory of automated timetabling: Selected papers from the first international conference.* Volume 1153 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 1996, pp 345-363.

[103] J. Thompson and K. Dowsland. A robust simulated annealing based examination timetabling system. *Comput. Oper. Res.*, 1998, 25, pp 637-648.

[104] G. M. White. Constrained satisfaction, not so constrained satisfaction and the timetabling problem. *In*: E. K. Burke and W. Erben (eds): *Proceedings of the third international conference on the practice and theory of automated timetabling*, Konstanz, Germany, August 2000, pp 32-47.

[105] G. M. White and B. S. Xie. Examination timetables and tabu search with longer-term memory. *In*: E. K. Burke and W. Erben (eds). *Practice and theory of automated timetabling: Selected papers from the third international conference.* Volume 2079 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 2001, pp 85-103.

[106] A. Wren. Scheduling, timetabling and rostering - A special relationship? *In*: E. K. Burke and P. Ross (eds). *Practice and theory of automated timetabling: Selected papers from the first international conference.* Volume 1153 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 1996, pp 46-75.