

HEURISTIC APPROACHES FOR UNIVERSITY TIMETABLING PROBLEMS

by

Salwani Abdullah

BSc (UTM), MSc (UKM) Malaysia

Thesis submitted to The University of Nottingham
for the degree of Doctor of Philosophy

The School of Computer Science and Information Technology

June 2006

*Dedicated to,
my lovely husband Wan Abd Malik
my sons Faris, Shahmi, Adib, Amir and Naqib
my parents Abdullah and Zaharah
my brother and sisters,
I love you all.*

TABLE OF CONTENTS

List of Figures.....	vi
List of Tables	x
Abstract.....	xii
Acknowledgements.....	xiii
Declaration.....	xiv
PART I. The University Timetabling Problem.....	1
1. Introduction.....	2
1.1 Background and Motivation	2
1.2 Aims of the Research.....	3
1.3 Overview of the Thesis.....	5
2. A Review of University Timetabling Problems and Approaches	8
2.1 Introduction	8
2.2 What is Timetabling?	9
2.3 Classification of Educational Timetabling Problems	10
2.3.1 School Timetabling.....	10
2.3.2 University Timetabling.....	11
2.4 The Examination Timetabling Problem	11
2.5 The Course Timetabling Problem	14
2.6 A Graph Colouring Model for the University Timetabling Problem.....	16
2.7 Techniques Applied to the University Timetabling Problem.....	18
2.7.1 Examination Timetabling	19
2.7.1.1 Constraint-based Methods	19
2.7.1.2 Graph-based Approaches.....	20
2.7.1.3 Cluster-based Methods	22
2.7.1.4 Population-based Approaches	23
2.7.1.5 Hill Climbing.....	27
2.7.1.6 Meta-heuristic Methods.....	27

2.7.1.7	Multi-criteria Approaches.....	37
2.7.1.8	Case-based Reasoning (CBR) and Fuzzy-based Approaches	38
2.7.1.9	Hyper-heuristic and Self Adaptive Approaches	40
2.7.2	Course Timetabling	41
2.7.2.1	Constraint-based Methods	41
2.7.2.2	Graph-based Approaches.....	42
2.7.2.3	Population-based Approaches	42
2.7.2.4	Meta-heuristic Methods.....	44
2.7.2.5	Multi-criteria Approaches.....	47
2.7.2.6	Case-based Reasoning (CBR), Knowledge-based and Fuzzy-based Approaches.....	48
2.7.2.7	Hyper-heuristic Approaches.....	49
2.8	Brief Summary	49
3.	Specification and Datasets for University Timetabling Problems.....	50
3.1	Introduction	50
3.2	Specification of the Examination Timetabling Problem	50
3.2.1	The Uncapacitated Examination Timetabling Problem	51
3.2.2	The Capacitated Examination Timetabling Problem.....	56
3.3	Specification of the Course Timetabling Problem	59
3.3.1	Problem Definition	60
3.3.2	An Evaluation Example of Soft Constraint Violation	60
3.4	Benchmark Datasets	69
3.4.1	The Examination Timetabling Datasets.....	69
3.4.2	The Course Timetabling Datasets.....	71
3.5	Summary.....	71
PART II.	New Examination Timetabling Approaches.....	73
4.	Investigating Ahuja-Orlin's Large Neighbourhood Approach for Examination Timetabling.....	74
4.1	Introduction	74
4.2	A Very Large Scale Neighbourhood Search: A Literature Overview.....	76

4.3	Modelling the Examination Timetabling Problem	79
4.3.1	Partitioning the Problem	79
4.3.2	Cyclic Exchange Neighbourhood Structure	80
4.3.3	Improvement Graph	81
4.4	Search Algorithm	84
4.5	Experiments and Results	89
4.5.1	The Uncapacitated Problem	89
4.5.2	The Capacitated Problem	92
4.6	Summary	95
5.	A Tabu-based Large Neighbourhood Search for the Capacitated Examination Timetabling Problem	97
5.1	Introduction	97
5.2	A Tabu-based Large Neighbourhood Search	98
5.3	Experiments and Results	101
5.4	Summary	104
6.	A Multi-start Large Neighbourhood Search Approach with Local Search Methods for the Examination Timetabling Problem	106
6.1	Introduction	106
6.2	Intensification and Diversification Strategies	107
6.3	A "Multi-start Two-phase" Approach	108
6.4	Experiments and Results	113
6.5	Summary	120
	PART III. New Course Timetabling Approaches	121
7.	An Investigation of the Variable Neighbourhood Search Approach for University Course Timetabling	122
7.1	Introduction	122
7.2	Variable Neighbourhood Search (VNS) for the Course Timetabling Problem	123
7.2.1	Initial Solution: Constructive Heuristic	124
7.2.2	Neighbourhood Structure within VNS	124

7.2.3	Acceptance Criteria	126
7.2.4	Tabu List.....	126
7.2.5	Stopping Criterion	127
7.2.6	Local Search	127
7.3	Experiments and Results	129
7.4	Summary.....	147
8.	Using a Randomised Iterative Improvement Algorithm with Composite Neighbourhood Structures for University Course Timetabling.....	148
8.1	Introduction	148
8.2	Composite Neighbourhood Structures: A Literature Overview.....	149
8.3	The Randomised Iterative Improvement Algorithm	150
8.3.1	The Neighbourhood Structures.....	150
8.3.2	The Algorithm	152
8.4	Experiments and Results	154
8.5	Summary.....	166
9.	A Memetic Approach for University Course Timetabling.....	167
9.1	Introduction	167
9.2	Memetic Algorithm for the Course Timetabling Problem	168
9.2.1	Solution Representation.....	168
9.2.2	Initial Population Generation.....	168
9.2.3	The Evolutionary Operator: Mutation	169
9.2.4	Selection	169
9.2.5	The Local Search: Randomised Iterative Improvement Algorithm.....	170
9.2.6	The Algorithm	170
9.3	Experiments and Results	171
9.4	Summary.....	172

10. Conclusions and Future Work.....	174
10.1 Research Work Summary	174
10.2 Contributions	175
10.3 Future Work.....	177
10.3.1 Improving the Presented Approaches.....	177
10.3.2 Hybridisation or Multi-start Approaches.....	178
10.3.3 Applications of the Presented Approaches to Real-life Problems	178
10.4 Dissemination	179
References	181

LIST OF FIGURES

2.1	An undirected graph $G = (V, E)$	17
2.2	A graph model for a simple course timetabling problem.....	18
2.3	The basic VNS algorithm (adapted from Hansen and Mladenović 2001)...	35
3.1	The pseudo-code to generate a <i>conflict matrix</i> , C	53
3.2	Proximity cost for examination e_3	55
3.3	The total proximity cost for the solution in Table 3.2.....	55
3.4	Vector of timeslots in days.....	56
3.5	Vector of timeslots	56
3.6	The adjacent period cost for examination e_3	58
3.7	The adjacent period cost for the solution in Table 3.4.....	59
3.8	The pseudo-code to generate a <i>student-event matrix</i>	61
3.9	The pseudo-code to generate a <i>suitable-room matrix</i>	64
3.10	An example of a vector of timeslots	65
3.11	An example of a vector of rooms	65
3.12	The pseudo-code to generate a <i>student-availability matrix</i>	66
3.13	The pseudo-code for the first soft constraint violation	68
3.14	The pseudo-code for the second soft constraint violation.....	68
3.15	The pseudo-code for the third soft constraint violation	68
3.16	Formula to calculate the total penalty for the course timetabling problem	68
4.1	Cells before a cyclic exchange operation.....	80
4.2	A cyclic exchange operation takes place	81
4.3	Cells after a cyclic exchange operation.....	81
4.4	The directed arc (7,1) of inserting examination e_7 and ejecting examination e_1	82
4.5	The improvement graph of the example in Section 4.3.2	83
4.6	The pseudo-code for the large neighbourhood search algorithm applied to the examination timetabling problem	84
4.7	The pseudo-code for the modified shortest label-correcting algorithm.....	87
4.8	The pseudo-code for updating the improvement graph	88
4.9	The behaviour of the large neighbourhood search algorithm on <i>ute-s-92</i> ...	90
4.10	The behaviour of the large neighbourhood search algorithm <i>yor-f-83</i>	90

4.11	The behaviour of the large neighbourhood search algorithm on <i>car-f-92</i> ...	94
4.12	The behaviour of the large neighbourhood search algorithm on <i>car-s-91</i> ...	94
4.13	The behaviour of the large neighbourhood search algorithm on <i>kfu-s-93</i> ...	94
4.14	The behaviour of the large neighbourhood search algorithm on <i>tre-s-92</i>	95
4.15	The behaviour of the large neighbourhood search algorithm on <i>uta-s-92</i> ...	95
5.1	The pseudo-code for the tabu-based large neighbourhood search applied to the examination timetabling problem	99
5.2	The pseudo-code for updating the improvement graph in the tabu-based large neighbourhood search approach.....	101
5.3	The performance of the tabu-based large neighbourhood search algorithm with different TT on the <i>tre-s-92</i> dataset	103
5.4	The performance of the tabu-based large neighbourhood search algorithm with different TT on the <i>kfu-s-93</i> dataset.....	103
5.5	The behaviour of <i>car-f-92</i> dataset using tabu-based large neighbourhood search algorithm.....	104
6.1	The pseudo-code for the “multi-start” approach.....	109
6.2	The pseudo-code for the great deluge algorithm applied to the examination timetabling problem	111
6.3	The pseudo-code for the simulated annealing algorithm applied to the examination timetabling problem	112
6.4	The behaviour of the “multi-start” large neighbourhood search algorithm on the <i>hec-s-92</i> dataset.....	116
6.5	The behaviour of the “multi-start” large neighbourhood search algorithm on the <i>sta-f-83</i> dataset	116
6.6	The behaviour of the “multi-start” large neighbourhood search algorithm on the <i>yor-f-83</i> dataset.....	116
6.7	The behaviour of the “multi-start” large neighbourhood search algorithm on the <i>ute-s-92</i> dataset.....	117
6.8	The behaviour of the “multi-start” large neighbourhood search algorithm on the <i>ear-f-83</i> dataset.....	117
6.9	The behaviour of the “multi-start” large neighbourhood search algorithm on the <i>tre-s-92</i> dataset	117

6.10	The behaviour of the “multi-start” large neighbourhood search algorithm on the <i>lse-f-91</i> dataset.....	118
6.11	The behaviour of the “multi-start” large neighbourhood search algorithm on the <i>kfu-s-93</i> dataset.....	118
6.12	The behaviour of the “multi-start” large neighbourhood search algorithm on the <i>car-f-92</i> dataset.....	118
6.13	The behaviour of the “multi-start” large neighbourhood search algorithm on the <i>uta-s-92</i> dataset	119
6.14	The behaviour of the “multi-start” large neighbourhood search algorithm on the <i>car-s-91</i> dataset	119
7.1	Example 1 of “move a whole timeslots” neighbourhood structure.....	124
7.2	Example 2 of “move a whole timeslots” neighbourhood structure.....	125
7.3	The pseudo-code of a modified VNS for the course timetabling problem	128
7.4	The pseudo-code for VNS-Tabu	129
7.5a-e	The behaviour of the algorithm on the <i>small</i> datasets (VNS-EMC).....	132
7.6a-e	The behaviour of the algorithm on the <i>medium</i> datasets (VNS-EMC).....	134
7.7	The behaviour of the algorithm on the <i>large</i> dataset (VNS-EMC).....	134
7.8a-e	The behaviour of the algorithm on the <i>small</i> datasets (with-ordering).....	137
7.9a-e	The behaviour of the algorithm on the <i>medium</i> datasets (with-ordering)..	138
7.10	The behaviour of the algorithm on the <i>large</i> dataset (with-ordering).....	139
7.11a-b	The behaviour of VNS-Tabu and VNS-EMC algorithms applied to the <i>medium3</i> and <i>medium5</i> datasets, respectively.....	142
7.12	The behaviour of VNS-Tabu and VNS-EMC algorithms applied to the <i>large</i> dataset	142
7.13a-e	The behaviour of the VNS-Tabu algorithm on the <i>small</i> datasets.....	144
7.14a-e	The behaviour of the VNS-Tabu algorithm on the <i>medium</i> datasets.....	146
7.15	The behaviour of the VNS-Tabu algorithm on the <i>large</i> dataset.....	147
8.1	The earlier neighbourhood structure before the kempe chain move execution	151
8.2	The result of a kempe chain move	152
8.3	Schematic overview of the randomised iterative improvement algorithm	152

8.4	The pseudo-code for the randomised iterative improvement algorithm applied to the course timetabling problem	153
8.5a-e	The behaviour of the randomised iterative improvement algorithm on the <i>small</i> datasets	157
8.6a-e	The behaviour of the randomised iterative improvement algorithm on the <i>medium</i> datasets	158
8.7	The behaviour of the randomised iterative improvement algorithm on the <i>large</i> dataset	159
8.8	The neighbourhood structures used for the <i>small</i> datasets.....	160
8.9	The neighbourhood structures used for the <i>medium</i> datasets.....	160
8.10	The neighbourhood structures used for the <i>large</i> dataset	160
8.11a-e	The behaviour of the randomised iterative improvement algorithm using single and composite neighbourhood structures applied to the <i>small</i> datasets	164
8.12a-e	The behaviour of the randomised iterative improvement algorithm using single and composite neighbourhood structures applied to the <i>medium</i> datasets	165
8.13	The behaviour of the randomised iterative improvement algorithm using single and composite neighbourhood structures applied to the <i>large</i> dataset.....	166
9.1	Solution representation.....	168
9.2	The pseudo-code to generate populations	169
9.3	The pseudo-code for the mutation operation	169
9.4	A schematic overview of the memetic algorithm applied to the course timetabling problem	170
9.5	The pseudo-code of the memetic algorithm applied to the course timetabling problem	171

LIST OF TABLES

2.1	Basic terminology used in timetabling.....	9
3.1	An example of a <i>conflict matrix</i> , C	53
3.2	Examinations-timeslots assignment	54
3.3	Example of the proximity coefficient matrix	54
3.4	Examinations-timeslots-days assignment	57
3.5	The example of the adjacent period coefficient	58
3.6a	The <i>student-event matrix</i> for course e_i , $i \in \{1, \dots, 15\}$	61
3.6b	The <i>student-event matrix</i> for course e_i , $i \in \{16, \dots, 30\}$	61
3.6c	The <i>student-event matrix</i> for course e_i , $i \in \{31, \dots, 45\}$	62
3.6d	The <i>student-event matrix</i> for course e_i , $i \in \{46, \dots, 60\}$	62
3.6e	The <i>student-event matrix</i> for course e_i , $i \in \{61, \dots, 75\}$	62
3.6f	The <i>student-event matrix</i> for course e_i , $i \in \{76, \dots, 90\}$	62
3.6g	The <i>student-event matrix</i> for course e_i , $i \in \{91, \dots, 100\}$	63
3.7	A <i>suitable-room matrix</i>	65
3.8a	The <i>student-availability matrix</i> for timeslot t_i , $i \in \{0, \dots, 16\}$	66
3.8b	The <i>student-availability matrix</i> for timeslot t_i , $i \in \{17, \dots, 33\}$	67
3.8c	The <i>student-availability matrix</i> for timeslot t_i , $i \in \{34, \dots, 44\}$	67
3.9	The uncapacitated benchmark datasets	70
3.10	The capacitated benchmark datasets	71
3.11	The parameter values for the course timetabling problem categories.....	71
4.1	<i>Conflict matrix</i> of the example in Section 4.3.2.....	82
4.2	Example of the feasible initial solution.....	83
4.3	Results on the uncapacitated problem using proximity cost	91
4.4	Result on the uncapacitated problem	92
5.1	Result on the capacitated problem using tabu-based large neighbourhood search algorithm	102
6.1	Improvement results at phase 1 and 2	114
6.2	Comparison of results using the “multi-start two phase” approach applied to the uncapacitated examination timetabling problem using proximity cost...	115
7.1	Results on the course timetabling problem	130
7.2	Comparison results on the course timetabling problem using VNS	140

7.3	The results on VNS-Tabu	141
8.1	Comparison results on the course timetabling problem using the randomised iterative improvement algorithm.....	156
8.2	Comparison of the performance of the randomised iterative improvement algorithm on single and composite neighbourhood structures.....	162
9.1	Comparison results on the course timetabling problem using a memetic algorithm	173

ABSTRACT

University timetabling represents a difficult optimisation problem and finding a high quality timetable is a challenging task. With a large number of events involved and various hard constraints to be fulfilled, finding an optimal timetable is complicated and time consuming. Many approaches in the literature have addressed this problem. The research work presented in this thesis aims to build upon the state of the art in search methodologies for university timetabling. The research focuses on both examination and course timetabling problems. Towards this goal, several ideas are introduced to increase the overall performance of timetabling algorithms. The research first highlights an initial investigation into a very large scale neighbourhood search for examination timetabling problem. This differs from much of the literature because most of the search mechanisms applied to this problem use small neighbourhood structures. This research is based on graph theoretical algorithms implemented on a so called improvement graph. It identifies improvement moves by solving negative cost partition-disjoint graph cycles using a modified shortest path label-correcting algorithm. Then, the research programme explores a hybridisation method that incorporates the large scale neighbourhood approach with tabu search. It is applied to the capacitated examination timetabling problem with the assumption that the strength of different approaches can combine to produce a more powerful new approach. Next, the research proposes a “multi-start two phase” approach that involved the large scale neighbourhood approach at the first phase and local searches at the second phase. The aim is to generate solutions that work well across all examination test instances in which a single technique may work well on certain instances and poorly on others. The next three research chapters concentrate on the course timetabling problem where the first technique to be investigated is variable neighbourhood search followed by a randomised iterative improvement algorithm using composite neighbourhood structures. This is then incorporated with a population based technique (referred to as a memetic algorithm). Computational results based on standard university benchmark instances are reported to demonstrate the effectiveness of the approaches studied here.

ACKNOWLEDGEMENTS

In the name of God, Most Gracious, Most Merciful.

I would like to express my gratitude to my exceptional supervisor Prof. Edmund K. Burke for his guidance, support and his continuous enthusiasm and interest during my study. I am also very grateful and extend my sincere thanks to Prof. Moshe Dror, Dr. Barry McCollum, Dr. Samad Ahmadi, Dr. Djamila Ouelhadj and Dr. Yuri Bykov for their rewarding discussion and unselfish support. Many thanks to the rest of the ASAP research group for their friendship. It is my pleasure to acknowledge the Public Services Department of Malaysia and University Kebangsaan Malaysia for their sponsorship.

To my lovely husband Wan Abd Malik Wan Mohamed, my sons Faris, Shahmi, Adib, Amir, Naqib, my parents, my sisters and brother for their unconditional understanding, encouragement and constant support throughout my study. I love you all.

May God bless us.

DECLARATION

I hereby declare that this thesis has not been submitted, either in the same or different form, to this or any other university for a degree.

Signature:

PART I

The University Timetabling Problem

Chapter 1

Introduction

1.1 Background and Motivation

University timetabling problems, particularly examination and course timetabling, are difficult tasks faced by educational institutions. Solving a real world university timetabling problem manually often requires a large amount of time and expensive resources. In order to handle the complexity of the problems and to provide automated support for human timetables, much research in this area has been invested over the years. A wide variety of papers, from the fields of operational research and artificial intelligence, have addressed the broad spectrum of university timetabling problems. Early timetabling research focused on sequential heuristics which represented a simpler and easier method for solving graph colouring problems, the principle idea being to schedule events one by one starting with the most difficult first (Carter and Laporte 1996).

In recent years, interest in meta-heuristic approaches such as simulated annealing, tabu search and genetic algorithms (for university timetabling) has increased due to the ability of these approaches to generate solutions which are better than those generated from sequential heuristics alone (Schaerf 1999a, Burke and Petrovic 2002). Normally in university timetabling, an initial solution is constructed using an appropriate heuristic, then the improvement is carried out using these meta-heuristics. However, some meta-heuristics are dependent on certain parameters. For example, simulated annealing depends on a cooling schedule; tabu search requires (among other parameters) an appropriate length of tabu list. Genetic algorithms, for example, might need to tune the length of a chromosome. The performance of

approaches may vary from one instance to another which might depend on the setup of these parameters, the neighbourhood structure and the search algorithm itself. Thus, meta-heuristic implementations often have a *tailor-made* aspect to their nature and this is, indeed, the nature of the work presented in this thesis. However, many meta-heuristic methods work well on certain problem instances but often are not readily applicable and are expensive to adapt to new problems. Hence, the development of a more general framework that can work effectively across different problems has been studied in recent years. To this end, the area of hyper-heuristics (i.e. heuristics to choose heuristics) is currently being investigated (Burke et al. 2003c) in contrast to some meta-heuristic research which focuses upon comprehensive problem-specific knowledge to arrive at good solutions for specific problems. In the last few years, researchers have also attempted to investigate knowledge-based such as case-based reasoning (Burke et al. 2006b), fuzzy approaches (Asmuni et al. 2004, 2005a, 2005b) and constraint-based reasoning (Boizumault et al. 1996, Cheng et al. 1996, Henz and Würtz 1996) for solving timetabling problems.

A brief observation of the recent timetabling literature (see Chapter 2) shows that most of the meta-heuristic methods use small neighbourhood structures in their search algorithms. This motivated the investigation of a large rather than small neighbourhood structure to solve the problem. In general, this thesis describes an investigation into the development of a large neighbourhood search approach that is based on graph algorithms, meta-heuristic approaches and hybridisations to automate the university timetabling problem. This thesis is derived from an interest in developing automated algorithms to tackle this important class of problems in a more effective way than currently exists.

1.2 Aims of the Research

This thesis investigates a large neighbourhood search approach and meta-heuristic approaches to university timetabling problems. The overall aim of the work is to investigate how these approaches can improve the quality of the timetable. We consider both examination and course timetabling problems. In order to accomplish this primary aim, several objectives are outlined:

1. An investigation of the use of a large neighbourhood search approach to solve the problem: The approach is based on a theoretical graph algorithm that integrates the following key points:
 - The definition of examination timetabling as a partitioning problem.
 - The definition of a large neighbourhood structure through a cyclic exchange operation.
 - The establishment of an improvement graph (the details of the improvement graph are discussed in Chapter 4).
 - The identification of a profitable exchange using a shortest path label-correcting algorithm (the details of the shortest path label-correcting algorithm are also discussed in Chapter 4).
2. A hybridisation with tabu search: This method enables a tabu-based approach to be integrated with a large neighbourhood search approach. This research investigates whether the integration of meta-heuristic methods and algorithms based on graph theory is able to generate good quality solutions.
3. The development of a two phase approach that tries to improve the solution obtained from the first phase by employing local search methods: The research issue is to see whether the approach can generate solutions that are good across all standard benchmarks on the uncapacitated examination timetabling problem. The uncapacitated problem is a particular instance of the examination timetabling problem and is discussed in Chapters 3, 4 and 6.
4. An empirical investigation into variable neighbourhood search: Here the issue is whether to sequence or not the neighbourhood structures and whether to accept or not a worse solution with a certain probability in a variable neighbourhood search approach that can contribute significantly to the solution quality.
5. The development of a powerful new local search method based heuristic for the course timetabling problem: This involves an empirical investigation about whether a single or a composite neighbourhood structure is most appropriate when employed within a local search method. This is a question of great practical importance when dealing with more complex problems.
6. The development of a population-based approach that tries to further enhance the quality of the solution obtained from local search methods: A memetic

algorithm is developed and standard benchmark datasets are used to provide comparisons against other approaches in the literature.

1.3 Overview of the Thesis

This thesis consists of ten chapters. This chapter presents the background motivation and the aims of the research. The remainder of this thesis is organised in the following way:

Chapter 2 introduces various timetabling problems and concentrates upon particular research issues concerned with university timetabling problems. It reviews and analyses the current published research on the subject of university timetabling.

Chapter 3 explores the specification of university timetabling problems. It includes formal mathematical statements and the standard benchmark instances (i.e. capacitated and uncapacitated datasets for examination timetabling and course timetabling datasets) that are used in the experiments carried out in the subsequent chapters.

An investigation into the application of a large neighbourhood search based on Ahuja-Orlin's methodology (Ahuja et al. 2001a) to the examination timetabling problem is presented in Chapter 4. This initial study aims to investigate the strength of a large neighbourhood technique when compared to other approaches which usually used small neighbourhoods to solve this problem. The approach is tested on standard uncapacitated examination benchmark datasets.

In Chapter 5, a hybridisation of the large neighbourhood method with tabu search is applied to the capacitated examination timetabling problem. A number of experiments involving different values of the tabu length are performed and discussed in this chapter.

In Chapter 6, a “multi-start two phase” approach applied to the uncapacitated examination timetabling problem is carried out. The first phase employs a large neighbourhood search approach and the second phase employs a local search (i.e. the

great deluge and simulated annealing algorithms). In the first phase, a large neighbourhood search approach is restarted several times if there is no improvement in the solution quality after a certain number of iterations. Then, the second phase is applied using great deluge and simulated annealing separately.

Moving onto course timetabling, Chapter 7 deals with variable neighbourhood search (Mladenović and Hansen 1997). This method requires several neighbourhood structures of a different nature. These neighbourhood structures are investigated in a particular manner. If one fails to improve the solution, the other one may still have a chance. In this approach, the neighbourhood structures are ordered based on a pre-defined sequence. Also discussed in this chapter is a hybridisation between variable neighbourhood search and tabu search. The experimental results on the course timetabling problem show that the hybrid method outperforms a variable neighbourhood search method alone.

Chapter 8 presents a composite neighbourhood structures (with randomised iterative improvement) algorithm. The difference between this algorithm and the algorithm discussed in Chapter 7 is that this algorithm does not depend upon the sequence of the neighbourhood structures. Given a solution, the search algorithm will explore the solution space using all the neighbourhood structures. This means that each neighbourhood structure has a chance to be employed on the same solution. The best return solution from each neighbourhood structure will be accepted and the process will be repeated. This differs from the approach taken in Chapter 7 because the next neighbourhood structure in the sequence will only be applied if there is no more improvement in the solution quality. Experimental results show that a composite neighbourhood structures with randomised iterative improvement algorithm produced better results compared to the variable neighbourhood search approach (as in Chapter 7) when applied to the course timetabling problem.

The strength of a population-based approach for the examination timetabling problem is discussed in the literature. Chapter 9 focuses on a population-based approach with local search which is sometimes called a memetic algorithm. This algorithm is tested on the course timetabling problem.

Finally, the overall conclusions of the work presented in this thesis and research directions for future work in this area are presented in Chapter 10.

Chapter 2

A Review of University Timetabling Problems and Approaches

2.1 Introduction

This literature review chapter attempts to place an emphasis on the fundamental aspects of the research area. It introduces the definition of the general timetabling problem, specifically the university timetabling problem, the constraints that such a problem must consider and the key approaches related to the university timetabling problem that have been carried out to date.

This chapter comprises seven sections. Section 2.1 describes the definition of timetabling. Section 2.2 briefly explains the general timetabling problem followed by the classification of educational timetabling problems. Since this thesis focuses on the university timetabling problem, Sections 2.4 and 2.5 present more details on the examination and course timetabling problems which include related soft and hard constraints. Section 2.6 discusses the university timetabling problem model using graph colouring. The summary of the papers which have been published and the techniques applied to university timetabling problems are presented in Section 2.7. Section 2.8 provides a summary of this chapter.

2.2 What is Timetabling?

Timetabling problems are a specific type of scheduling problem and are mainly concerned with the assignment of events to timeslots subject to constraints with the resultant solution constituting a timetable. Wren (1996) defined timetabling in the following way:

“Timetabling is the allocation, subject to constraints, of given resources to objects being placed in space time, in such a way as to satisfy as nearly as possible a set of desirable objectives.”

Based on the definition given by Wren (1996), we need to know whether there are sufficient resources available for the given event to take place at its specified time as well as which resources are allocated. The goal is to optimise some objective function depending on the application domain at hand. For example, in examination timetabling environments, the function to optimise is usually the gap between two examinations that a student has to sit in i.e. try to spread the examinations throughout the examinations periods of time. The basic terminology used in timetabling problems is summarised in Table 2.1.

Table 2.1. Basic terminology used in timetabling

Terminology	Definition
Event	An activity to be scheduled. Examples include examinations and courses.
Timeslot (period)	An interval of time in which events can be scheduled.
Resource	Resources required by events. Examples include rooms and equipment (i.e. projectors).
Constraint	A restriction to schedule the events. Examples include <i>room capacity</i> and <i>specific timeslot</i> .
Individual	A person who has to attend the events.
Conflict	Two events are clashing with each other if they have at least a common individual and are scheduled in the same timeslot.

The constraints in timetabling can be divided into two categories: hard and soft. Hard constraints cannot be violated. Soft constraints are not essential but their satisfaction is highly desirable in order to produce a good quality timetable.

A general timetabling problem consists of assigning a number of events like courses, examinations, lectures, lab sessions etc. into a limited number of timeslots and rooms while minimising the violations of a set of constraints. In general, the timetable problem has a set of events, E , a set of timeslots, T , and a set of hard constraints, C . The intention is to arrange all of the events, E , into the timeslots, T , in such a way that no hard constraint, C , is violated in order to produce a feasible timetable. Two common hard constraints in timetabling are: (i) no two events can be scheduled at the same time and place and (ii) there must be sufficient resources available for all the events scheduled for each timeslot.

2.3 Classification of Educational Timetabling Problems

Schaerf (1999a) classified educational timetabling into three main classes i.e. school timetabling, course timetabling and examination timetabling. They share the same basic characteristics of the general timetabling problem but can still have significant differences between them. Each one of them has its own constraints, requirements and rules. More details on educational timetabling can be found in Burke et al. (2004e). In this section, a classification of educational timetabling and its properties are discussed. We divided educational timetabling into two categories i.e. school timetabling and university timetabling (which consists of examination timetabling and course timetabling).

2.3.1 School Timetabling

The school timetabling problem is concerned with the weekly scheduling for all the lessons of a school. The problem consists of a set of teachers, classes, subject/lessons and weekly periods. These weekly periods are predefined. This problem tries to assign lessons to periods and, a teacher to a particular class at a given time while satisfying a set of constraints in order to produce a feasible timetable. Some examples of constraints in the school timetabling problem are capacities, locations,

teacher loads, rest time between two lessons and other personal preferences. Examples of research on school timetabling can be found in Abramson (1991) who employed simulated annealing, Carrasco and Pato (2001) who employed a multi-objective genetic algorithm and Legierski (2003) who applied a constraint-based approach.

2.3.2 University Timetabling

The university timetabling problem can be grouped into two categories: (i) course (or lecture) timetabling and (ii) examination timetabling. The course timetabling problem is the process of assigning timeslots and rooms so that meetings between lecturers and students can take place. The examination timetabling problem refers to the assignment of timeslots and rooms so that students can take examinations. These two (examination and course) timetabling problems are fairly similar in some superficial ways, but there are some distinct underlying differences between them. In examination timetabling, several examinations can be assigned to one (large) room at the same time. However, this is not possible for course timetabling where only one course can be assigned to one room.

This thesis focuses on the university (examination and course) timetabling problems. The details of both problems are discussed in Sections 2.4 and 2.5, respectively.

2.4 The Examination Timetabling Problem

The examination timetabling problem represents a major administrative activity for academic institutions. It is often a difficult and demanding process and it affects a significant number of people. Romero (1982) reports that there are three broad categories of people that are affected by its outcome: administrators, academic staff and students. Many universities are seeing an increasing number of student enrolments into a wider variety of courses and an increasing number of combined degree courses. This is contributing to the growing challenge of developing examination timetabling software to cater for the broad spectrum of constraints and demands that are required by educational institutions across the world and therefore the quality of a timetable should be evaluated from several points of view.

Carter and Laporte (1996) defined the examination timetabling problem as:

“The assigning of examinations to a limited number of available time periods in such a way that there are no conflicts or clashes”

The examination timetabling problem is very common in both schools and universities. It is concerned with allocating a set of examinations, into a limited number of timeslots (periods), subject to a set of constraints. Carter et al. (1994) quoted that the basic challenge of examination timetabling is to schedule examinations over a limited number of timeslots so as to avoid conflicts and to satisfy a number of side constraints. In this case, the conflict is referred to as a hard constraint and side constraints are referred to as soft constraints.

The responses from a survey on examination timetabling problem among 56 universities conducted by Burke et al. (1996a) found that the constraints vary widely from institution to institution. Some examples of hard constraints for examination timetabling are:

- Certain examinations must be consecutive or must take place in a specific order (before/after each other).
- Examinations with the largest number of students should be scheduled earlier in the timetable to allow more time for marking.
- Examinations given by the same instructor, if scheduled in the same timeslots, have to be assigned to nearby classrooms.
- There should be no more than x conflicting examinations in y ($>x$) consecutive timeslots.
- Certain examinations must take place in a specific room.
- There must be enough seating capacity in the room for the number of students scheduled in it.
- No student should be required to sit two examinations simultaneously.

The generally accepted hard constraints for the examination timetabling problem are (i) there must be enough seating capacity and (ii) no student should be required to sit two examinations at the same time. Solutions that satisfy all the hard constraints are

called *feasible*. On the other hand, there might be some requirements that are not essential. These are referred to as soft constraints. Common constraints, as reported in Burke et al. (1996a), are:

- Students should not be scheduled to sit more than one examination in a day.
- Students should not be scheduled to sit examinations in two consecutive timeslots.
- Each student's examinations should be spread as evenly as possible over the schedule.
- Some examinations may only be assigned within a particular set of timeslots.
- Examinations of the same length may be scheduled in the same room.
- Examinations must be scheduled to the rooms which are near to the relevant department.
- Examinations with questions in common must be scheduled in the same timeslot.

In a real world situation, it is, of course, usually impossible to satisfy all the soft constraints, but minimising these violations will increase the quality of the solution by calculating the penalty function to the extent to which a timetable has violated its soft constraints.

de Werra (1985) presented a formal approach for the examination timetabling problem based on a mathematical programming model. Let us consider the following notation:

- E_i is a collection of n examinations ($i = 1, \dots, N$)
- T is the number of timeslots.
- C_{it} is the cost of scheduling examination i in timeslot T .
- $Y_{it} = 1$ if examination i is scheduled in timeslot T and 0 otherwise.
- $X_{ij} = 1$ if examination i clashes with examination j and 0 otherwise

A timetable is considered feasible if the following hard constraints are satisfied: (i) every examination must be scheduled once and (ii) no conflicting examinations should be scheduled in the same timeslot. The problem is required to generate a timetable using T timeslots without violating the hard constraints. The objective is to

minimise the cost of scheduling examination i in period T . This examination timetabling problem can be represented as (adopted from Terashima-Marín, 1998):

$$\min \sum_{n=1}^N \sum_{t=1}^T C_{nt} Y_{nt} \quad (2.1)$$

subject to

$$\sum_{t=1}^T Y_{nt} = 1 \quad n = 1, \dots, N \quad (2.2)$$

$$\sum_{n=1}^N \sum_{m=1}^N \sum_{t=1}^T Y_{nt} Y_{mt} X_{nm} = 0 \quad (2.3)$$

The other hard constraint that can also be considered is concerned with room capacity. Let X_t be the maximum number (capacity) of examinations that can be scheduled in period t , so the hard constraints can be represented as:

$$\sum_{n=1}^N Y_{nt} \leq X_t \quad t = 1, \dots, T \quad (2.4)$$

It can be said that the final purpose of the examination timetabling problem is to guarantee that all students can take any examination that they are required to. However, we also need to maintain a reasonable use of resources (rooms, timeslots, etc). In order to accomplish this, certain types of constraint should be satisfied and this degree of satisfaction establishes a quality measure for the timetable. The details of the techniques applied to solve university timetabling problems will be discussed in Section 2.7.

2.5 The Course Timetabling Problem

Carter and Laporte (1998) defined course timetabling as:

“a multi-dimensional assignment problem in which students, teachers (or faculty members) are assigned to courses, course sections or classes; events (individual meetings between students and teachers) are assigned to classrooms and times”

In course timetabling (which is also sometimes known as class/teacher timetabling), a set of courses is scheduled into a given number of rooms and timeslots within a week and, at the same time, students and teachers are assigned to courses so that the meetings can take place. Some combinatorial models which draw upon graph colouring for simple class-teacher timetabling problems can be found in de Werra (1996b, 1997b). As in examination timetabling, course timetabling also involves hard and soft constraints. Examples of hard constraints for the course timetabling problem are:

- A student and a teacher cannot be in two places at the same time.
- Only one course is allowed to be assigned to a timeslot in each classroom.
- The classroom capacity should be equal to or greater than the number of students attending the course at a particular timeslot.
- The classroom assigned to the course should satisfy the features required by the course.

Some related soft constraints for course timetabling reported by Socha et al. (2002) are:

- Students should not have a single course on a day.
- Students should not have to attend more than two consecutive courses on a day.
- Students should not be scheduled to attend a course that is assigned to the last timeslot of the day.

Carter and Laporte (1998) decomposed the course scheduling problem into five sub problems i.e. course timetabling, class-teacher timetabling, student scheduling, teacher assignment and classroom assignment. Laporte and Desroches (1986) presented a problem of assigning students to course sections in a large engineering school.

Rudová and Murray (2003) presented a university course timetabling problem with soft constraints for Purdue University. An example of comprehensive course timetabling and models for some special types of class-teacher timetabling problems can be found in Carter (2001) and de Werra et al. (2002). This thesis deals mainly with the course timetabling problem that schedules courses to timeslots and assigns

them to rooms subject to several constraints. The details of the specification of the version of the course timetabling problem that we tackle will be discussed in Chapter 3.

2.6 A Graph Colouring Model for the University Timetabling Problem

de Werra (1985) illustrates how a timetabling problem can be modelled using a graph. This section presents a graph colouring model that can be used in modelling the university timetabling problem.

Graph colouring is concerned with colouring the vertices of a given graph using a given number of colours. Let us consider the examination timetabling problem. We need to schedule all the examinations within a limited number of timeslots in such a way that any clashing examinations (i.e. examinations that have at least a common student) are scheduled in different timeslots, so this problem can be viewed as a graph colouring model where the vertices represent the examinations, the colours represent the slots and the edges represent the conflicts between examinations. Each vertex of a graph should be coloured using p colours so that no two vertices connected by an edge are both assigned the same colour and normally there are a limited number of colours available. The graph colouring problem and its relationship to timetabling is widely discussed in the literature (see examples in de Werra 1996a, de Werra 1997a, Burke and Ross 1996 and Burke et al. 2004a).

A definition of the concepts and terms that relate to a graph is given before progressing with the explanation of this model. An undirected graph $G = (V, E)$ is a representation that consists of a set of vertices, $V = \{v_1, \dots, v_n\}$, and a set of edges, E . If (v_i, v_j) is an edge in a graph $G = (V, E)$, then vertex v_i is adjacent to vertex v_j (Burke et al., 2004a). Figure 2.1 shows the representation of an undirected graph on the vertex set $\{v_1, v_2, v_3, v_4, v_5\}$.

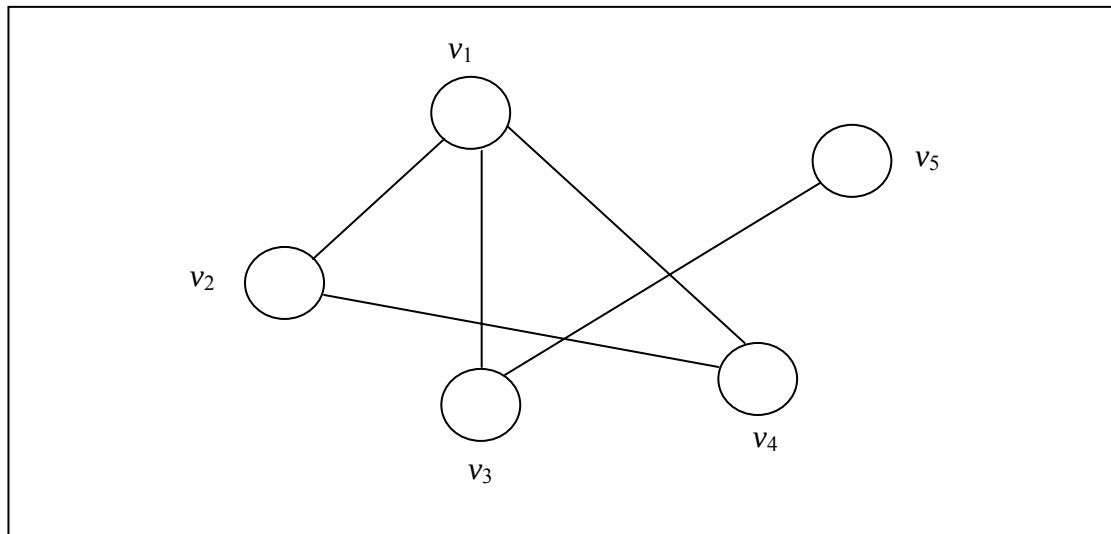


Figure 2.1. An undirected graph $G = (V, E)$

Other related definitions are:

- The degree of a vertex is the number of edges connected to it. For example, from Figure 2.1, vertex v_1 has a degree of 3.
- The chromatic number of a graph is the minimum number of colours necessary to colour the vertices, so that no two vertices connected by an edge are both assigned the same colour.

For a better understanding about the relationship between the graph colouring problem and the timetabling problem, an example of course timetabling is presented in Figure 2.2.

From Figure 2.2, we can see that there are five different courses coded as A, B, C, D and E . One possible goal is to find the minimum number of timeslots that are needed to schedule the five courses. A set of edges represents clashes between courses. If there is an edge between vertices, it means that these courses cannot be scheduled in the same timeslot. In our example, course A cannot be scheduled at the same time as course B and C . Course B cannot be scheduled at the same time as course A and D and so on. Clearly 3 colours (timeslots) are needed to schedule this problem. Course A and E could be coloured red, course D could be coloured yellow, course B could be coloured blue and course C could be coloured yellow or blue. The colours correspond to timeslots. The graph colouring problem is concerned with finding the

chromatic number of a graph (which is the minimum number of colours required to colour the graph). From the graph in Figure 2.2, it is easy to see that the chromatic number is 3.

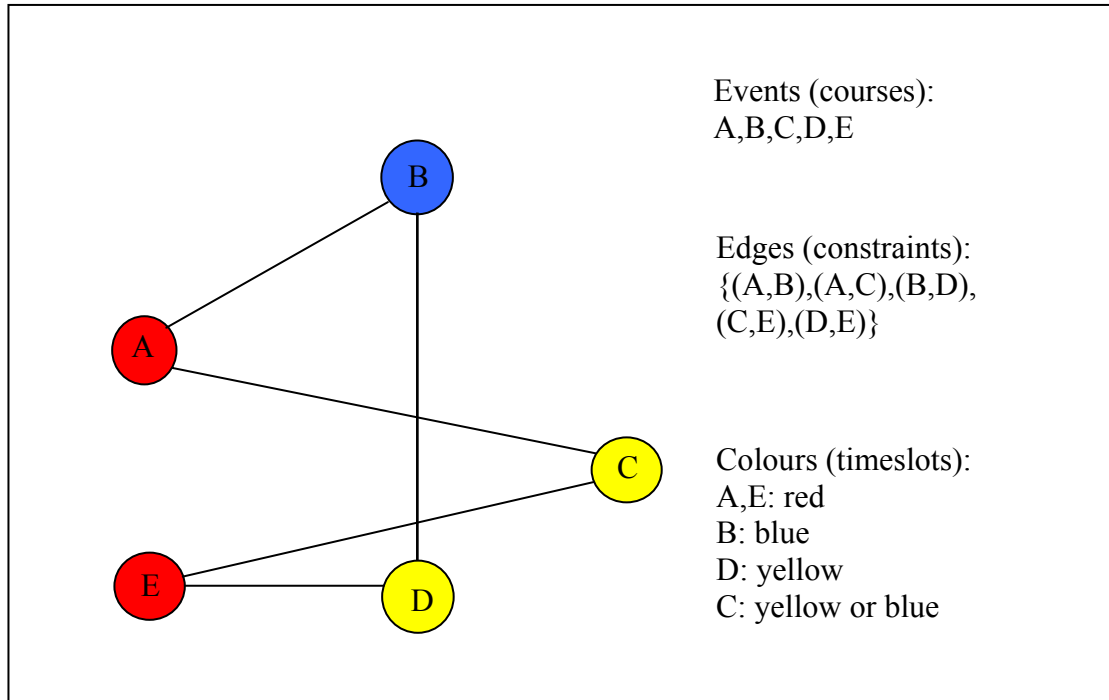


Figure 2.2. A graph model for a simple course timetabling problem

A variety of graph colouring based heuristics for constructing a clash-free timetable is available in the literature. For example see Brelaz (1979), Burke et al. (1994a) and Carter and Laporte (1996).

2.7 Techniques Applied to the University Timetabling Problem

Several review papers discuss the major approaches to timetabling (see de Werra 1985, Carter 1986, Balakrishnan et al. 1992, Carter and Laporte 1996, Bardadym 1996, Burke et al. 1997, Schaerf 1999a, Burke and Petrovic 2002 and Petrovic and Burke 2004). Carter and Laporte (1996) roughly divided these approaches into four categories i.e. cluster, sequential, generalised search (meta-heuristics) and constraint-based methods. Petrovic and Burke (2004) added the following categories: multi-criteria approaches, case-based reasoning techniques and hyper-heuristic/self adaptive methods.

Carter and Laporte (1998) present some developments in practical course timetabling. They classified the algorithms used to solve various components of the course timetabling problem into global algorithms, constructive heuristics, improvement heuristics and several methods that include user interaction with a computer system.

This section divides the related techniques applied to university timetabling problems into eight categories i.e. constraint-based methods, graph-based approaches, cluster-based methods, population-based approaches, meta-heuristic methods, multi-criteria approaches, hyper-heuristic/self adaptive approaches, case-based reasoning, knowledge-based and fuzzy-based approaches. The details of these categories are discussed in the following sub-sections.

2.7.1 Examination Timetabling

2.7.1.1 Constraint-based Methods

Constraint-based methods have been used to a significant extent for solving examination timetabling problems. Generally, in a constraint-based approach, a problem is modelled as a set of variables with a finite domain. The method assigns values to variables that fulfil a number of constraints. Nuijten et al. (1994) applied a general constraint satisfaction technique to the examination timetabling problem at the Eindhoven University of Technology, and tested it on a relatively small real dataset (275 examinations, 7000 students, 33 examination timeslots over three weeks and about 3000 constraints). Successful results were obtained in-between two and twelve minutes of CPU time on a SUN workstation. Different variations of the logic programming language have been employed in the wide variety of constraint-based methods that have appeared in the literature. Kang and White (1992) employed a constraint logic programming approach using Prolog to generate a feasible examination timetable. Lajos (1996) also used Prolog to create a feasible modular timetable for the University of Leeds.

Henz and Würtz (1996) presented a constraint logic programming using Oz to construct a timetable at a German college. Boizumault et al. (1996) used constraint logic programming for the examination timetabling problem by including room

assignment rules in addition to common rules such as conflicting and consecutive examinations, precedence constraints and preassignment. This work was tested on data from l'Université Catholique de l'Ouest in Angers. The approach used the *best fit* strategy. It was developed using CHIP (Constraint Handling in Prolog) and tested on 308 examinations and 2600 constraints. The solution is obtained in less than one minute of CPU time. Cheng et al. (1996) investigated a constraint logic programming approach for university timetabling which is based on a Prolog (it is written using WPROLOG) description of the constraints and goals. David (1998) presented the problem as a constraint satisfaction problem to generate an examination timetable where an incomplete assignment algorithm based upon a local repair technique was implemented. If no solution was found, the program was rerun by relaxing some of the constraints (i.e. allowing the violation of some constraints).

In general, constraint-based approaches alone can generate feasible solutions efficiently. However, most search approaches lack the ability to further enhance the quality of the generated solution. Brailsford et al. (1999) said that pure constraint-based approaches cannot compete with the state of the art local search methods. Therefore, they are widely applied as a hybridisation approach (with other local search methods). For example, White and Zhang (1998) presented a combination between constraint logic programming and tabu search. Constraint logic programming is used to provide a starting point for the tabu search method since it is able to quickly generate a solution. Merlot et al. (2003) implemented a hybridisation between constraint programming and local search. A constraint programming method is used to obtain a feasible initial timetable. A detailed discussion about combining local search with constraint programming can also be found in Focacci et al. (2003).

2.7.1.2 Graph-based Approaches

Early approaches to examination timetabling incorporated the concept of heuristic ordering (e.g. Foxley and Lockyer 1968). These approaches are based on graph-colouring heuristics (de Werra 1997a, Burke et al. 1994a and Welsh and Powell 1967). Graph-colouring heuristics are often called sequential heuristics. The main idea is to assign examinations to a timeslot, one by one, based on a *sequencing*

strategy (Carter and Laporte, 1996). Carter et al. (1996) list several sequencing strategies, which can be described as follows:

- Largest degree: examinations with the largest number of conflicts are scheduled first.
- Largest enrolment: this is a modification of largest degree where the examination with the largest student enrolment is scheduled first.
- Saturation degree: this is a dynamic sequencing strategy where the next selected examination to be scheduled is based on the number of available periods. The examination with the least number of available periods will be scheduled next.
- Largest weighted degree: this is a modification of largest degree where priority is given to the examination that has the largest weighted conflict. Each conflict is weighted based on the number of students enrolled in two conflicting examinations.
- Random ordering: the examinations to be scheduled are selected at random.

Brelaz (1979) presented new heuristic methods to colour the vertices of a graph by defining a *saturation degree* of a vertex as the number of different colours of the vertex's adjacent vertices that have already been coloured. In Brelaz's algorithm, the vertex with the largest saturation degree is the most difficult to be coloured since it has less colours to choose from.

Burke et al. (1994a) presented graph colouring and room allocation algorithms for the university timetabling problem. A graph colouring algorithm is used to split the examinations into non-conflicting clusters and the room allocation algorithm is used to place the examinations into rooms. This work was further refined by Burke et al. (1993) by adding more constraints to the problem.

Carter et al. (1994) reported that sequential heuristics had proved to be very efficient when incorporating a backtracking procedure. The backtracking procedure is employed when it is not possible to schedule an examination into the timeslots because of earlier assignments. It functions by un-scheduling certain examinations that have conflicts with the current examination and then allowing the current examination to be scheduled. Carter et al. (1996) implemented a backtracking

procedure. They found that the use of this procedure can reduce the number of timeslots needed for the timetable compared to sequential heuristics without backtracking. Burke et al. (1998a) employed a heuristic procedure without backtracking but incorporated a random element. The results obtained show that the heuristic approaches can be improved with the addition of a random element. Asmuni et al. (2005a) proposed a fuzzy heuristic ordering. Instead of using one heuristic, the authors considered the combination of two ordering heuristics out of three heuristics (i.e. largest degree, largest enrolment and saturation degree). These multiple orderings are used simultaneously. The orderings are determined by a fuzzy model. The approach has been tested on standard benchmark examination timetabling datasets and it was seen that better solutions could be obtained by combining more heuristics. Burke and Newall (2004) considered heuristic orderings and introduce a *heuristic modifier* to determine the difficulty of examinations (at each iteration) in order to adapt the ordering over time. Experimental results showed that the adaptive method has a significant amount of independence over the initial choice of heuristic. It even turned deliberately bad heuristics into good ones.

In general, sequential heuristics are found to be effective and yet simple approaches for finding a feasible timetabling solution. However, they might not be able to produce a high quality solution with respect to the satisfaction of the soft constraints. To address this situation, hybrid approaches have been studied which incorporate sequential heuristics with other techniques. For example, Burke et al. (1998b) seeded an evolutionary algorithm with heuristic ordering strategies to produce an initial population by adding a random element to the heuristic ordering strategies. Experimental results showed that the solution obtained from such initialisation of a population is superior to random initialisation in terms of time and quality. Carter and Johnson (2001) incorporated the clique initialisation technique (to identify the maximum clique of the problem and assign the clashing examinations to the timetable) with a sequential heuristic that schedules the remaining examinations.

2.7.1.3 Cluster-based Methods

Cluster methods were classified as one of the four major approaches by Carter and Laporte (1996). The idea of the cluster method was first coined by Desroches et al.

(1978). White and Chan (1979) and White and Haddad (1983) describe cluster methods as which can be thought of as representing a three phase approach. In the first phase, the examinations are grouped into timeslots to construct a feasible timetable. The second phase attempts to reduce second order conflicts by considering permutations of timeslots. Then the third stage is employed with the aim of improving the solution quality further. This is done by moving a particular examination between timeslots such as by employing a hill climbing local search.

2.7.1.4 Population-based Approaches

Genetic Algorithms

Genetic algorithms were popularised by Holland (Holland 1975). This methodology employs operators known as genetic operators (such as *selection*, *crossover* and *mutation*) that manipulate individual solutions (referred to as *chromosomes*) in a population for a number of generations in order to improve the objective function (often called the fitness). The chromosome is represented as a fixed length string where each position is called a *gene* and contains a unit of solution information. Using selection operators (such as *roulette wheel*), the best solutions are selected to become *parents*. The crossover operator is used to create one or more *offspring* from two existing parents. Some common crossover operations are one-point crossover, two-point crossover, cycle crossover and uniform crossover. Several parameters need to be considered when applying a genetic algorithm to a given problem such as population size, crossover rate, mutation rate and the number of generations (Goldberg 1989, Pham and Karaboga 2000, Burke and Kendall 2005).

Corne et al. (1993) employed a genetic algorithm for solving the examination timetabling problem. The value of the required parameters was set empirically. For example, the length of the chromosome was set as the number of examinations. Each gene represents a timeslot where the examinations are assigned. They applied a repair mechanism to overcome the infeasibilities due to the direct chromosome representation that generate infeasible offspring solutions. The solution obtained from this algorithm was found to be better than a manual solution. In order to avoid infeasible solutions, Ross et al. (1994) proposed using only the mutation operator to

generate offspring solutions. Experimental results showed that their approach outperformed the genetic algorithm that used a uniform crossover operator.

Burke et al. (1994b) describe the use of a genetic algorithm to solve timetabling problems. The problem is addressed using a direct representation scheme which considers when and where an examination will be taken. Domain specific knowledge is used in the form of heuristic assignment algorithms that deal with timeslots and rooms and the search algorithm only needs to cover feasible solutions. A comparison between a genetic algorithm, simulated annealing and stochastic hill climbing can be found in Ross and Corne (1995). Ergül (1996) implemented a university examination timetabling method based on a genetic algorithm for the Middle East Technical University. It addressed two cases involving 682 and 1449 examinations. Two adaptive mutation operators which are called *linear mutation* and *quadratic mutation* are used. The fitness value is employed to determine the mutation rate. Experimental results show that the adaptive mutation operators accept solutions without altering the genetic algorithm parameters

Ross et al. (1998) made some observations about genetic algorithms that use direct encoding for examination timetabling and found that the algorithms have a tendency to fail in solving different parts of problems (i.e. what they call a moderate-constrained problem in their experiment). They suggest that rather than employing genetic algorithms to search for a solution for a specific problem, it might be better to use genetic algorithms to search for a good algorithm. Chu and Fang (1999) compared genetic algorithm and tabu search approaches for examination timetabling. Experimental results show that tabu search (see Glover and Laguna 1997, Burke and Kendall 2005) is able to produce better results than genetic algorithms with less computing time.

A grouping genetic algorithm has been applied by Erben (2001) for graph colouring and examination timetabling problems. In a grouping genetic algorithm, a chromosome is made up from a group (of *genes*) which is different from a straightforward direct representation in a standard genetic algorithm (where the solution is represented as a chromosome that consists of N genes if it has N entities).

In the application of examination timetabling, the mutation operators used in this algorithm will choose two groups in the chromosome at random and do the swapping, and will choose more than two groups, exchange them and then reverse the order of the scheduled groups. The algorithm is tested on standard benchmark examination timetabling problems and obtains quite promising results. Other examples of applying genetic algorithms for examination timetabling can be found in Burke et al. (1994c, 1995a, 1995b), Terashima-Marín et al. (1999a, 1999b) and Corne et al. (1994).

Ant Colony Optimisation

Ant colony optimisation is another population based meta-heuristic proposed by Dorigo et al. (1996). The basic idea behind this optimisation technique is based on the observation that ants find their way to a food source and back to their nest. As they move, a trail of chemical substance (called *pheromone*) is deposited by them that will help other ants to find the same food source.

Costa and Hertz (1997) developed a method called ANTCOL for addressing graph colouring problems using ant colony optimisation and a sequential heuristic. In successive generations, each ant colours the vertices using static (i.e. *random*, *largest first*, *smallest last*) or dynamic (i.e. *saturation degree*, *recursive largest first*) constructive methods. The probability value of the pheromone is used to select the colour for each vertex. Experimental results show that the dynamic methods perform significantly better than static methods. This research highlights the promise of using ant colony optimisation in successfully solving examination timetabling problems.

Dowsland and Thompson (2005) investigated the application of ant colony optimisation to the examination timetabling problem. The objectives of this research were: firstly to compare the performance of ANTCOL on typical timetabling graphs with a set of random graphs observed by Costa and Hertz (1997); secondly, the authors wished to identify promising constructive heuristic combinations, trail calculations and ANTCOL parameter values. Experimental results show that the modification of ANTCOL applied to the examination timetabling problem is

competitive with the best published approaches in the literature in minimising the number of timeslots required for a feasible timetable.

Memetic Algorithms

Memetic algorithms represent evolutionary based approaches combined with local search techniques. This hybridisation method (evolutionary algorithms together with local search) has been given various names in the literature such as hybrid genetic algorithms and genetic local search algorithms (Hart et al. 2004). The majority of memetic algorithms discussed in the literature are an outcome of incorporating local search techniques with a genetic algorithm.

Burke et al. (1996b) employed a memetic algorithm for university examination timetabling where two evolutionary operators are used (light and heavy mutation) in the initial phase followed by a hill climbing operator. Burke and Newall (1999) presented a multi-stage memetic algorithm for the capacitated examination timetabling problem. The algorithm is applied to a subset of examinations at a time which is subsequently fixed in the timetable before moving to the next subsets (which represents a decomposition process). In order to evaluate the effectiveness of this approach, real datasets were used. Experimental results show that the solution quality was better than when employing the memetic approach alone.

Burke and Landa Silva (2004) identify and discuss the following effective strategies when designing memetic algorithms for scheduling and timetabling problems:

- (1) dealing with infeasibility because it is difficult to keep the solutions feasible during the search. One way to deal with infeasibility is simply to prevent the occurrence of infeasible solutions or by applying a repair mechanism to repair an infeasible solution (see Colorni et al. 1998).
- (2) approximate fitness evaluation in order to reduce the computation time taken by the memetic algorithm.
- (3) the design of data structures using linked lists to allow efficient implementation of approximate fitness evaluation.
- (4) the fitness landscape differing from one problem to another.

- (5) establishing the right balance between genetic and local search methods (see also Inshibuchi et al. 2003).

More details about memetic algorithms can be found in Moscato (1999, 2002) and Krasnogor and Smith (2005).

2.7.1.5 Hill Climbing

Hill climbing is the simplest local search method which iteratively evaluates all neighbouring solutions and replaces the current solution with a candidate solution if there is an improvement in the cost for the problem in hand. In general, the performance of hill climbing alone is relatively poor since it is easily trapped in local optima. However, hybridised methods which incorporate hill climbing with another method can be effective. For example, Merlot et al. (2003) employed a hybridisation between constraint programming, simulated annealing and hill climbing methods. Also Kendall and Hussin (2005b) successfully applied hyper-heuristic and hill climbing methods for the examination timetabling problem. Moreover, hill climbing forms the basis of many meta-heuristic methods such as tabu search, simulated annealing (see below) and memetic algorithms (see above).

2.7.1.6 Meta-heuristic Methods

The term meta-heuristic was first introduced in Glover (1986). Some definitions of meta-heuristics are:

“A meta-heuristic refers to a master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality. The heuristics guided by such a meta-strategy may be high level procedures or may embody nothing more than a description of available moves for transforming one solution into another, together with an associated evaluation rule.”

(Glover and Laguna 1997)

“A meta-heuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and

exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions.”

(Osman and Laporte 1996)

“A meta-heuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method.”

(Voß et al. 1999)

“A meta-heuristic is a set of concepts that can be used to define heuristic methods that can be applied to a wide set of different problems. In other words, a meta-heuristic can be seen as a general algorithmic framework which can be applied to different optimisation problems with relatively few modifications to make them adapted to a specific problem.”

(Meta-heuristic Network Website 2005)

Note that we have already discussed genetic algorithms, memetic algorithms and ant colony optimisation in detail. Other meta-heuristics will be discussed below.

Tabu Search

The idea of tabu search was proposed by Fred Glover (1986). Glover and Laguna (1997) define tabu search as:

“A meta-heuristic that guides a local heuristic search procedure to explore the solution space beyond local optimality”

The basic mechanism of tabu search is a hill climbing algorithm. However, to prevent the search from getting stuck in local optima, a so-called tabu list is maintained i.e. a list that contains moves that satisfy some tabu restriction criteria. These previously accepted moves are forbidden to be performed for a certain number of iterations (called the tabu tenure). The tabu tenure determines how long a move remains tabu. However, a mechanism called the aspiration criterion is sometimes used to override the tabu status of a move. A common aspiration criterion is better fitness (i.e. an improvement of the cost function) where a tabu move is changed to a non-tabu move if it produces a better solution. A basic introduction to tabu search can be found in Gendreau and Potvin (2005) and a comprehensive treatment can be found in Glover and Laguna (1997).

White and Xie (2001) implemented a tabu search algorithm which is called OTTABU. It used both recency-based short-term memory and frequency-based longer-term memory to prevent cycling and to effectively diversify the search space in getting a better solution. A node which represents an examination is made tabu whenever it is moved from one timeslot to another. Another strategy used in this implementation is to relax the tabu lists by emptying all entries in the tabu lists when a given number of iterations have passed and the tabu list is full or when the current solution is much worse than the last best solution found. A mechanism to balance the diversification and intensification has also been introduced in this work. The complete system is constructed in four phases where the output of one phase is treated as an input of the next. Experimental results on two datasets (from Carter et al. 1996) show that the OTTABU with longer-term tabu lists compares favourably with other algorithms used for the same problem. White et al. (2004) continue the work done in 2001 by applying longer-term tabu lists and relaxation. Experimental results on twelve datasets (from Carter et al. 1996) show that the use of a longer-term tabu list in addition to a short-term tabu list has enhanced the quality of the final solution. Also, the tabu relaxation is a good optimisation strategy because it helps to drive the solution into new areas of the search space.

Di Gaspero and Schaerf (2001) presented an examination timetabling algorithm that is based on tabu search and graph colouring heuristics. In order to guide the search to

explore different areas of the solution space, they modified the objective function by changing the weights. A dynamic size of the tabu list in the interval (k_{min} and k_{max}) is used to store the most recently accepted moves. The same aspiration criteria as in Schaerf (1999b) is employed and the algorithm is tested on standard benchmark and random instances of examination timetabling. Di Gaspero (2002) continues the research from Di Gaspero and Schaerf (2001) by applying a novel multi-neighbourhood local search algorithm based on a combination of two kinds of moves rather than simple moves as in Di Gaspero and Schaerf (2001) with the objective of diversifying the search away from local optima. Two types of neighbourhoods are defined called *recolour* (i.e. an examination that involved at least one soft or hard constraint violation is considered to be moved which aims to focus on examinations that can contribute to the cost) and *shake* (that exchanges the timeslots of two groups of examinations). To improve further the solution, a form of perturbation called *kick* is performed. The author called this solver *Recolour, Shake and Kick*. Preliminary results show that these tabu search algorithms outperform the plain tabu search by Di Gaspero and Schaerf (2001).

Kendall and Hussin (2005a, 2005b) applied a tabu-based hyper-heuristic for examination timetabling. The hyper-heuristic represents a sort of reinforcement learning that helps in making an intelligent decision. It monitors the performance of each low level heuristic. A tabu list is used to store information about heuristics. The used heuristic is made tabu for a certain number of times which is equal to a fixed length of the tabu tenure in order to allow other heuristics to be applied. The approach is tested on standard benchmark instances and shows that it is able to generate good quality solutions. In Kendall and Hussin (2005b), the authors solved a new examination timetabling dataset from the MARA University of Technology that consists of 2,159 examinations and 40 timeslots. The objective is to minimise the proximity cost (Carter et al. 1996) and weekend cost (cost for examinations that are scheduled during the weekends). They employed three different hyper-heuristics i.e. a simple tabu search hyper-heuristic where only the heuristic that gives the best improvement will be applied, a tabu search hyper-heuristic with hill climbing and a tabu search hyper-heuristic with great deluge (Dueck 1993). The approach employed deterministic and random dynamic tabu tenures (where a tabu tenure value is

selected at random from a given range between t_{min} and t_{max} . The method is able to produce feasible and good quality solutions which were at least 80% better than the manual solutions with respect to the proximity cost. Other examples which discuss tabu search with respect to examination timetabling problems can be found in Dowsland (1998) and Pacquete and Stützle (2002).

Simulated Annealing

Another hill climbing based method that has a mechanism to escape from local optima is called simulated annealing. Annealing is the physical process of heating up a solid at high temperature and slowly cooling it down until it crystallises and no further changes occur and the system reaches a steady state. This is called thermalisation. A sequence of temperatures that has been applied to thermalise the system comprises an annealing schedule or cooling schedule. Thus, the simulated annealing algorithm was derived from the annealing process and used to address combinatorial optimisation problems. It was proposed by Kirkpatrick et al. (1983). Simulated annealing starts from a randomly generated initial solution. The process operates by evaluating the randomly selected neighbour (move) of the current solution and improving moves with respect to the objective function are always accepted while worse candidate moves are accepted with a certain probability determined by the Boltzmann probability, P , calculated by formula $P = e^{-\alpha/t}$ where α is the difference of the objective function evaluation between the current and the candidate solutions and t is a parameter (called the temperature) which periodically decreases during the search process according to some cooling schedule. The choice of cooling schedule fundamentally impacts upon the quality of the final solution. The faster the cooling schedule, the faster the local optima can be obtained. On the other hand, the slower the cooling schedule, the more comprehensive the search will be and this generally results in a higher quality solution, but it takes considerably more time to do so. This choice of reducing the temperature during the search process can be done in several ways such as using a progress formula (geometric cooling schedule) or the temperature can be reduced based on a certain number of moves or successful moves only. A tutorial can be found in Burke and Kendall (2005).

Simulated annealing has also been extensively studied in the area of examination timetabling with some success. Johnson (1990) employed simulated annealing in real world examination timetabling and obtained a good solution compared to manual approaches. Thompson and Dowsland (1996a, 1996b) also investigated simulated annealing techniques for examination timetabling. The problem is solved in a 2 phase approach. The first phase is concerned with finding a feasible solution and the second phase attempts to optimise the soft constraints to produce better quality timetables. They continued the work by implementing an adaptive cooling schedule rather than a geometric cooling schedule. The computational results show that an adaptive cooling schedule outperformed a simple geometric cooling approach.

The choice of solution space and neighbourhood as well as the cooling schedule can significantly affect the quality of the solution produced by any simulated annealing implementation. Due to this situation, Thompson and Dowsland (1998) again carried out a robust simulated annealing approach in which they test different neighbourhoods and cooling schedules over a variety of examination timetabling problems. They also examine whether the sampling of the neighbourhood has a significant effect on the solution quality. The investigation conducted on real examination instances from different universities shows the superiority of the kempe chain neighbourhood (see Figure 8.2 in Chapter 8). This is due to its ability to allow a large number of examinations to move and makes a significant improvement to the quality of the solution.

Bullnheimer (1998) also investigated simulated annealing for examination timetabling. He focuses on small scale problems and one real world problem in particular, in which he breaks the problem into several sub-problems according to different types of constraints. This approach was able to give a good solution. Wright (2001) presents sub-cost guided search incorporated with simulated annealing applied to school timetabling problems. The sub-costs were incorporated into simulated annealing where they modify the standard probability function of accepting worse solutions by using an “adjusted cost increase” in the probability formula. Experimental results show that the incorporation of sub-cost information significantly improves the results of the simulated annealing method. Other examples

of the application of simulated annealing in examination timetabling can be found in Burke et al. (2003b, 2004b, 2004c).

A threshold acceptance algorithm was proposed by Dueck and Scheuer (1990), and is quite similar to simulated annealing. An improving solution is always accepted and a non-improving solution is accepted if the difference between the current and candidate solutions is less than a given threshold. The threshold can be fixed or changed during the search.

Great Deluge

The great deluge algorithm was introduced by Dueck (1993). It is a local search procedure which has certain similarities with simulated annealing but has been introduced as an alternative. This approach is far less dependent upon parameters than simulated annealing. It needs just two parameters: the amount of computational time that the user wishes to “spend” and an estimate of the quality of solution that a user requires. Apart from accepting a move that improves the solution quality, the great deluge algorithm also accepts a worse solution if the quality of the solution is less than (for the case of minimisation) or equal to some given upper boundary value B (in the paper by Dueck it was called a “*level*”). In this work, the “*level*” is initially set to be the objective function value of the initial solution. During its run, the “*level*” is iteratively lowered by a constant β where β is a decreasing rate.

The great deluge algorithm was applied to examination timetabling in Burke et al. (2004b). Burke and Newall (2003) employed a combination of adaptive initialisation strategies to seed this great deluge algorithm for examination timetabling. Experimental results show that the great deluge algorithm can improve the quality of solutions. Kendall and Hussin (2005b) hybridise a great deluge algorithm with a tabu search hyper-heuristic approach in examination timetabling. The experiments carried out show that this hybridisation approach was able to produce better results when compared to a single tabu search hyper-heuristic. Petrovic et al. (2005a) incorporate the great deluge algorithm with a case-based reasoning technique for examination timetabling. Computational experiments demonstrate that this solution approach gives comparable or better results than solutions generated by the single great deluge

algorithm in the literature. Other research on the great deluge algorithm for examination timetabling can be found in Burke et al. (2001a) and Yang and Petrovic (2005). More details about great deluge algorithm are discussed in Dueck (1993).

Variable Neighbourhood Search (VNS)

The success of finding good solutions for optimisation problems by employing meta-heuristic techniques is determined by the technique itself and the neighbourhood structure used during the search. Ahuja et al. (2000) highlighted the importance of the neighbourhood structure in the local or neighbourhood search.

“A critical issue in the design of a neighbourhood search approach is the choice of the neighbourhood structure that is the manner in which the neighbourhood is defined.”

The above statement shows the importance of the neighbourhood structure. This provides us with the motivation to explore the variable neighbourhood search approach. Other techniques in the literature like simulated annealing and tabu search generally use a single neighbourhood structure throughout the search and usually focus more on the parameters that affect the acceptance of the moves rather than on the neighbourhood structure. Thompson and Dowsland (1996b, 1998) discussed how the choice of the neighbourhood structure affects the quality of solutions obtained.

VNS was introduced by Mladenović and Hansen (1997). It is based on the strategy of using more than one neighbourhood structure and changing them systematically during the local search. This helps the VNS to explore a variety of possibilities and jump to a new solution if required.

Let us denote a set of predefined neighbourhood structures by $n_k(s)$ i.e. this is the set of solutions in the k^{th} neighbourhood of s . The steps of the basic variable neighbourhood search are presented in Figure 2.3, which is made up of three stages i.e. *shaking*, *local search* and *move*. More details can be found in Hansen and Mladenović (2001) and Burke and Kendall (2005). A local search technique is applied repeatedly to obtain the local optimum (Mladenović and Hansen 1997). Originally, the basic VNS approach was a descent method. It does not accept a

worsening solution to get out of local optima since the neighbourhood structures are varied regularly. Since a local optimum in one neighbourhood structure is not necessarily a local optimum in another neighbourhood structure, the change of the neighbourhood structures can be undertaken during the search.

```
Initialisation: Select the set of neighbourhood structures  $n_k$ ,  
                 $k=1,\dots,K$  that will be used in the search; find the initial  
                solution  $s$ ; choose a termination criterion;  
Repeat until the termination criterion is met:  
  (1) Set  $k \leftarrow 1$ ;  
  (2) Until  $k = K$ , repeat:  
    (a) Shaking: Generate a point  $s'$  at random from the  $n_k$   
        neighbourhood of  $s$  ( $s' \in n_k(s)$ );  
    (b) Local Search: Apply a local search method with  $s'$  as  
        initial solution until local optimum  $s''$  is obtained.  
    (c) Move or not: Accept  $s''$  ( $s \leftarrow s''$ ) if it is better than  
        incumbent solution and continue the search with  $n_k$  ( $k \leftarrow k+1$ );  
        otherwise  
         $k \leftarrow k+1$ ;
```

Figure 2.3. The basic VNS algorithm (adapted from Hansen and Mladenović 2001)

The termination criterion may be selected as the maximum number of iterations, the CPU time or the number of iterations without improvement. The point s' in step 2(a) in Figure 2.3 is randomly generated (rather than deterministically) in order to avoid cycling.

Avanthay et al. (2003) implemented an adaptation of VNS for the graph colouring problem using the Tabucol algorithm (Hertz and de Werra 1990) as a local search method. The experiments show that their VNS is not superior to the hybrid algorithm (tabu search and a genetic algorithm) proposed by Galinier and Hao (1999).

Current work within the Automated Scheduling Optimisation and Planning (ASAP) group at the University of Nottingham is exploring variable neighbourhood search for examination timetabling (see Burke et al. 2006c).

Greedy Randomised Adaptive Search Procedure (GRASP)

GRASP is a simple meta-heuristic that is composed of two phases i.e. solution construction and local search (Blum and Roli 2003). The first phase constructs a solution based on a probabilistic greedy algorithm where elements are normally ordered based on some criteria and the method probabilistically selects one element from the top n candidate list to be scheduled. The second phase will optimise a solution using a local descent formation. Casey and Thompson (2003) present a GRASP technique for the examination timetabling problem to minimise the proximity cost between examinations (see Chapter 3 for an explanation of the proximity cost) with respect to a feasible timetable. In the construction stage, examinations are ordered according to one of the criteria in Carter et al. (1996). The next examination to be scheduled is chosen using roulette wheel selection from the top n in the list. A backtracking technique with tabu list is employed if there is no feasible timeslot available. The use of a tabu list is to forbid cycling. In the second stage, examinations are considered based on their contribution to the quality of the solution. The best feasible move (that can reduce the cost of the objective function) is selected. The process continues until no further improvement is found. Then the algorithm will start again at the construction stage with an empty timetable. Experimental results show that the saturation degree heuristic strategy employed in the construction stage is able to produce the best result compared to other heuristic strategies.

Hybrid Meta-heuristics

The aim of the hybrid approach is to take the best idea from one approach and to incorporate it with another good (or better) idea from another (or more) approach (es). Note that many of the methods already described represent hybrid methods (to a greater or lesser extent). Hybridisation has proven to be very effective in the examination timetabling literature; for example, Caramia et al. (2001) obtained the best known results on several of the benchmark instances. The authors used improvement steps after employing a greedy scheduler that assigns examinations (which are ordered based on the degree of conflict) in turn to the lowest available timeslot with respect to the conflict free requirement.

Merlot et al. (2003) applied a hybrid method that consists of constraint programming, simulated annealing and hill climbing for uncapacitated and capacitated examination timetabling and obtained the best known results on some benchmarks. Constraint programming is used to generate feasible initial solutions (as presented in the Constraint-based Methods section) and the quality of the timetable is improved using simulated annealing where a kempe chain neighbourhood (Thompson and Dowsland 1996a, 1998) is employed. Then the hill climbing technique is utilised to further refine the timetable. Experimental results show that this method is superior to the method currently used by the University of Melbourne and performs well in comparison with other standard benchmark instances. Burke and Newall (2003) hybridised the approach in Burke and Newall (2004) with the great deluge method by Burke et al. (2004b) to build a method which has the best known results on certain examination benchmark problems. Another example of a hybrid approach applied to the examination timetabling problem can be found in Côté et al. (2005).

2.7.1.7 Multi-criteria Approaches

In the majority of the timetabling algorithms in the literature, a single cost function is used to evaluate the solution quality. However, multi-criteria approaches to timetabling offer a more flexible way of handling different types of constraints simultaneously (see Petrovic and Burke 2004). Burke et al. (2001a) discussed a multi-criteria approach where each criterion measures the number of violations of the corresponding constraints. There are nine criteria which are divided into three groups (room capacities, proximity of examinations, time and order of examinations). The overall aim of this approach is to minimise each of the nine criteria based on compromise programming (Petrovic and Bykov 2003) because some of these criteria are conflicting with each other.

Pacquete and Fonseca (2001) studied a multi-objective evolutionary algorithm applied to examination timetabling which considered the assignment of 249 examinations to 30 timeslots. In their study, each objective is evaluated separately and the quality of the final solution is based on pareto-ranking and the sum of the objectives. Experiments were carried out to test various objectives regarding the comparison between the pareto-ranking and linear-ranking. Experimental results

show that the solution quality obtained from the pareto-ranking approach gives a better performance than the linear sum of objectives. Carrasco and Pato (2001) presented an automated timetabling methodology using a multi-objective genetic algorithm applied to real instances taken from a university in Portugal. Two distinct objectives are designed to minimise the constraint violations for teachers and classes.

Petrovic and Bykov (2003) applied a multi-objective optimisation technique for examination timetabling based on trajectories. In this method, a user needs to specify a reference solution and the algorithm (which is a modified great deluge algorithm from Burke et al. 2004b), will change the user preferences into dynamic weights (during the search) which direct the search of the solution space along a trajectory by changing the acceptance level of the cost function values. Experimental results show that the algorithm can produce a final solution that satisfies the user's preferences. More details on the multi-criteria approach can be found in Burke and Petrovic (2002), Petrovic and Burke (2004) and Landa Silva et al. (2004) which presents other applications of multi-objective meta-heuristics in scheduling and timetabling.

2.7.1.8 Case-based Reasoning (CBR) and Fuzzy-based Approaches

A discussion of the applicability of Artificial Intelligence (AI) scheduling techniques in practice can be found in Wiers (1997). CBR is an AI technique that is supported by the study of cognitive science. It is motivated by the observation that humans use past experience to solve similar problems and reuse that experience with some modification to suit different requirements (Kolodner 1993). In CBR, problems are represented as cases that are defined by Kolodner and Leake (1996) as:

“A case is a contextualised piece of knowledge representing an experience that teaches a lesson fundamental to achieving the goals of the reasoner.”

A case usually has two parts: the description and the solution of the problem. These cases are stored in a knowledge base called a case base. There are four cyclic processes in CBR as noted by Aamodt and Plaza (1994):

- Retrieve – the most similar cases need to be retrieved from the case base.
- Reuse – the retrieved solution is used to solve the new case (problem).

- Revise – the retrieved solution is revised regarding new requirements.
- Retain – the solution of the new problem is retained for the case base.

The methodology involved in the development of a CBR system is: (i) case representation where the cases need to be represented in a way that can describe the features of the problem (ii) an indexing process that can carry out the case matching process efficiently (iii) case base maintenance and management that can maintain, manage and retain the solved cases automatically (iv) adaptation which alters an old solution to fit a new solution and (v) similarity measures which consider the proper features for carrying out the comparison between cases. More details can be found in Kolodner (1993).

Petrovic and Burke (2004) note that CBR can play two roles in solving timetabling either as a *solution reuse* or as a *methodology reuse* technique. Burke et al. (2003b) presents some work that treats CBR as a methodology reuse technique where they conduct an initial investigation of some of the main features of standard benchmark instances (in examination timetabling) to measure the similarities between problems which represent an important element for a case-based reasoning system. They studied the impact of the objective function on the behaviour of the standard examination benchmark instances using simulated annealing. The findings from the experiments show that the objective function plays a major role in identifying whether two problems are believed to be similar or not. Burke et al. (2004c) continue to investigate the major features of examination timetabling in order to develop a similarity measure. The aim of this work is to create a similarity measure that can intelligently retrieve heuristic or meta-heuristic(s) techniques for a particular problem from the case base. A case-based heuristic selection prototype for the examination timetabling problem can be found in Burke et al. (2004d). CBR for examination timetabling has also been investigated in Petrovic et al. (2004, 2005a).

Fuzzy reasoning has recently been investigated with some success for examination timetabling by Asmuni et al. (2004, 2005a, 2005b). A comparison between fuzzy ordering heuristics and standard heuristics applied to examination timetabling is discussed in these papers in which three ordering heuristics based on graph colouring

strategies are considered (i.e. largest degree, saturation degree and largest enrolment). The methods were tested on standard benchmark instances. Experimental results show that in most of the cases, the fuzzy-based ordering outperforms the linear combination ordering. Experimental results also show that using three ordering heuristics always performs better than when using two ordering heuristics. Petrovic et al. (2005b) considered fuzzy constraint satisfaction where two constraints are taken into consideration i.e. a larger examination should be scheduled earlier and students should have a sufficient break between two consecutive examinations. Fuzzy sets are used to describe the degree of constraint satisfaction and fuzzy rules are used in order to derive the satisfaction degree for each constraint. This approach was tested on standard benchmark instances.

2.7.1.9 Hyper-heuristic and Self Adaptive Approaches

Burke et al. (2003c) define a hyper-heuristic as:

*“The process of using meta-heuristics to choose (meta) heuristics
to solve the problem in hand”*

Hyper-heuristics are emerging as powerful approaches which are raising the level of generality of timetabling and other problem solving systems. A hyper-heuristic can be thought of as a high-level heuristic that modifies the solution indirectly by employing a set of low level heuristics based on learning mechanisms (compared to other techniques that are applied directly to the problem). So hyper-heuristics can potentially act as more general purpose methods that can generate good quality solutions for various problems by implementing the same method with very limited problem-specific knowledge. Hyper-heuristics are now widely explored with respect to timetabling and scheduling problems. More details can be seen in Burke et al. (2003c) and Burke and Kendall (2005).

Burke et al. (2007) investigated a generic hyper-heuristic approach upon a set of graph colouring heuristics for examination timetabling. A tabu search approach is employed within the hyper-heuristic framework where it employs permutations of graph based heuristics. Its objective is to find the heuristic list that can generate the best quality solution. Experimental results show that this approach is able to find

good solutions while maintaining the generality of the hyper-heuristic framework. A hybrid method which combines variable neighbourhood search and a hyper-heuristic is presented in Qu and Burke (2005). Another example that employed a hyper-heuristic with tabu-search is presented in Kendall and Hussin (2005b).

Burke and Newall (2004) have presented an adaptive heuristic approach which drew upon the squeaky wheel optimisation methodology developed by Joslin and Clements (1999). This method reduces the dependency on the choice of heuristics. Indeed, in Burke and Newall (2004) it is shown that poor examination timetabling heuristics can be automatically “turned into” good ones.

2.7.2 Course Timetabling

2.7.2.1 Constraint-based Methods

Zervoudakis and Stamatopoulus (2001) applied a constraint programming generic object-oriented model using ILOG SOLVER C++ library for the university course timetabling faced by the Department of Informatics and Telecommunications at the University of Athens. The problem involved 68 lectures that need to be scheduled in five days of nine teaching timeslots. A variety of search methods (for example depth first search) and variable ordering heuristics were used for the search for near optimal solutions.

Deris et al. (2000) formulated a timetable planning problem as a constraint-based reasoning technique implemented in an object oriented approach for colleges that involved 378 timeslots, 1673 subject sections and 10 rooms. Certain constraint categories are introduced (for example time, space and dispersion constraints). To facilitate the search for a solution, the timetabling problem is represented as a graph tree. In order to find the solution faster (by reducing the search space explored by the backtracking search), variable orderings are introduced based on size (for example the size of the domain and the number of constraints of the variables). Experimental results show that the feasible and best solutions can be found in a reasonable time.

Deris et al. (1999) incorporated constraint-based reasoning within a genetic algorithm to find a feasible and near optimal solution for course timetabling where a

constraint-based reasoning method is used to validate individuals generated by the genetic algorithm operators. This study shows that the approach is able to produce faster convergence as the constraint-based reasoning significantly reduces the search space by only considering feasible solution spaces. Experimental results show that the approach is capable of finding a near optimal solution and is able to avoid local optima (if fed with a correct representation of the chromosome, operators and fitness function).

2.7.2.2 Graph-based Approaches

Selim (1988) employed a graph colouring methodology for the faculty timetable problem where the vertices were split in order to reduce the chromatic number. The approach was tested on real data from the Faculty of Science of the American University in Cairo. de Werra (1996a) applied colouring models for course timetabling problems which are repeated week after week. Neufeld and Tartar (1974) implemented a graph colouring method for a class-teacher timetabling problem. This method has also been used by Asratian and de Werra (2002) to solve a class-teacher problem where it is able to handle several disjoint groups of lectures.

2.7.2.3 Population-based Approaches

Genetic Algorithms

Erben and Keppler (1996) dealt with a weekly-course timetabling problem to schedule classes, teachers, course modules and rooms to a number of timeslots in a week. An approach based on a genetic algorithm was applied to solve this problem. Random populations of feasible solutions were created during the initialisation step. The mutation was carried out by assigning new timeslots and rooms at random and a so-called *cycle* crossover operator was employed to generate feasible offspring. A large data sample was used to test the algorithm. The experiments carried out show that the algorithm was able to obtain promising results.

Ueda et al. (2001) presented a two-phase genetic algorithm for the course timetabling problem. The problem was derived from the curriculum of the Faculty of Information Sciences at Hiroshima City University in 1997. Two kinds of population are generated i.e. a class scheduling population (used in the first phase) and a room

allocation population (used in the second phase). These two populations were evolved separately. The algorithm was able to find a feasible solution.

Blum et al. (2002) presented a genetic algorithm to assign courses to timeslots and rooms that can also deal with infeasible solutions. They applied one-point or uniform crossover and a mutation operator to a population of individual solutions. A number of heuristic rules were formulated to construct good timetables such as *smallest possible room* and *the course that has the highest number of students*. Konstantinow and Coakley (2004) investigated the use of genetic algorithms to repair the schedule (once initial schedules have been adopted) in reactive scheduling. Two schedule perturbations were employed i.e. surges (for example due to the changes in student load that can affect the number of instructors that need to be assigned) and encroachments (which represent carry forward situation where students must repeat certain classes on the previous year's performance). Experimental results show that genetic algorithms are appropriate methodologies for coping with a variety of changes in student load.

Lewis and Paechter (2004) proposed a number of different crossover operators (called, for example, *sector-based*, *day-based*, *student-based* and *conflict-based* crossover). They applied a genetic repair function to preserve feasibility during crossover and mutation. The algorithm was tested on twenty problem instances for an international timetabling competition (which can be freely downloaded at <http://www.idsia.ch/Files/ttcomp2002/>) and the results show that conflict based crossover seems to be the more effective crossover method and that the algorithm is also able to produce a large number of diverse and feasible timetables in a reasonable amount of time. More research on genetic algorithms applied to course timetabling can be found in Rich (1996) and Lewis and Paechter (2005).

Ant Colony Optimisation

Socha et al. (2002) applied a MIN-MAX ant system for university course timetabling. The course timetabling problem is transformed into an optimal path problem which can be tackled by generating a *construction graph*. The assignment of courses to the timeslots is dependent on the pheromone value within the bounds.

They compare their algorithm with a random restart local search (this algorithm iterates the same local search as used by the MIN-MAX ant system where it starts from a random solution and keeps the best solution found). The results show that the MIN-MAX ant system performs better than the random restart local search method. More details on the comparison between the ant algorithm and other meta-heuristics applied to university course timetabling can be found in Socha et al. (2003).

Rossi-Doria et al. (2003) presented a comparison of five different meta-heuristics applied to a university course timetabling problem. One of the meta-heuristics used is ant colony optimisation. In this approach, an ant constructs a timetable using a sequential strategy where it chooses a course from a pre-defined list and assigns it to a timeslot in a probabilistic manner. Experimental results show that the performance of ant colony optimisation is slightly worse than simulated annealing and tabu search. However, it is better than a genetic algorithm. More details about ant colony optimisation can be found in Dorigo and Di Caro (1999) and Dorigo and Stützle (2003, 2004).

Memetic Algorithms

In educational timetabling, Paechter et al. (1996) implemented a memetic algorithm for the lecture timetabling problem which employed several types of mutation strategies. The system was tested with a standard dataset from the Napier University to see the effect of different types of mutation. Paechter et al. (1998) extended their work by enhancing the user interface and timetabling engine and applied it to the whole institution. The authors define two concepts i.e. features (properties satisfied by some resources) and containers (resources that can hold other similar resources) to help define constraints in timetabling problems. Experimental results show that a memetic (local search) algorithm with lamarckian evolution works well for this problem.

2.7.2.4 Meta-heuristic Methods

Tabu Search

Costa (1994) used tabu search for constructing different real course schedules. Two tabu lists were introduced. The first is a list of lectures which are moved from one

timeslot to another. Each lecture is not allowed to be moved while it is in the list. The second tabu list is made up of pairs consisting of information about a lecture and the previous timeslot, (l, t) with the aim that lecture l cannot be replaced at timeslot t while the pair remains on the list. In order not to focus only on regions that contain good solutions, the author introduced a diversification strategy that drives the search to other areas that are not examined so far by drastically reducing the weights (where these weights force the search process to visit solutions which are relatively good) and concentrating on less important relaxed constraints. Experimental results show that the algorithm is able to obtain satisfactory solutions to the timetabling problem. However, several parameters need to be tuned in the algorithm such as the weights and the lengths of the tabu lists.

Nonobe and Ibaraki (1998) proposed a tabu search method for a weighted constraint satisfaction problem with the objective of minimising the total weights of the unsatisfied constraints. The authors developed a general problem solver for optimisation problems that incorporates a control mechanism of weights. They tested their algorithm on university timetables using fixed and controlled weights. Colormi et al. (1998) presented an investigation of three different meta-heuristics (i.e. simulated annealing, tabu search and a genetic algorithm) on the high school timetabling problem instance. A variable size of tabu list where the length of the list is changed after a constant number of iterations is used. A relaxation procedure is also incorporated into the tabu search algorithm. Experimental results show that tabu search was the consistently best performing algorithm when compared against simulated annealing and the genetic algorithm.

Schaerf (1999b) applied tabu search techniques to scheduling lectures to timeslots for a large high school. A variable tabu list size is used. Each move is added into the tabu list where the size of the tabu list is randomly selected from a pre-determined range. Therefore the tabu tenure varies for each move in the tabu list. A common aspiration criterion is employed. Experimental results show that the algorithm is able to schedule 90-95% of the lectures (and is better than the manual timetable).

Alvarez-Valdes et al. (2000) used tabu search for assigning students to course sections in the Faculty of Mathematics at the University of Valencia. The approach was carried out in two phases. The first phase generated a set of best solutions for each student. These solutions were combined and a tabu search algorithm with strategic oscillation and a fixed tabu list was employed to enhance the quality of the solution for each student. Alvarez-Valdes et al. (2002) developed a course timetable using tabu search. They constructed a three phase algorithm in which the initial timetable is obtained in the first phase. The improving procedure to enhance the quality of the timetable is carried out in the second phase in which the tabu search algorithm plays its role (which is considered to be the most important part of this algorithm). The final phase concentrates upon improving the room assignment. Experiments were carried out on a different selection of moves (i.e. simple move, swap and multi-swap) using a static and dynamic tabu list with and without the candidate list.

Burke et al. (2003a) applied a tabu search hyper-heuristic technique to both course timetabling problems and nurse rostering. In this algorithm, a set of low level heuristics are competing with each other. When a heuristic has been applied, the change in the cost function value from the previous to a new solution is noted. A variable length dynamic tabu list of low level heuristics is maintained which stops certain heuristics from being employed for a certain duration. The status of the heuristics in the tabu list will be changed from tabu active to non-tabu active if there is an improvement in the cost function. The authors believe that there is no point in keeping a heuristic tabu once the current solution has been updated (i.e. there is an improvement in the current solution). Experimental results show that this technique is capable of producing acceptable solution qualities for course timetabling problems but the main focus of the paper was on robustness across different problem types. Other examples which discuss tabu search with respect to course timetabling problems can be found in Hertz and de Werra (1990) and Hertz (1991, 1992).

Simulated Annealing

Elmohamed et al. (1998) applied a simulated annealing algorithm with different cooling schedules (geometric, adaptive and adaptive with reheating) to a course

timetabling problem. The algorithms were tested on real data at Syracuse University. Experimental results show that the simulated annealing with adaptive cooling and reheating algorithm outperformed other methods.

Great Deluge

Burke et al. (2003d) presented a course timetabling version of the great deluge algorithm which involves two parameters i.e. estimated search time and level of solution quality. The method was tested on the course timetabling problems from an international timetabling competition. It was confirmed as an effective method where 7 out of 20 best known results were obtained among 21 compared algorithms.

Variable Neighbourhood Search (VNS)

Abdullah et al. (2005a) employed a variable neighbourhood search method for course timetabling and tested it on standard benchmark problems. The details of this approach are presented in Chapter 7.

Hybrid Meta-heuristics

Kostuch (2005) employed a three-phase approach for the course timetabling problem which combined graph colouring and simulated annealing. In the first phase, an initial feasible timetable is generated using graph colouring heuristics. Improvement is made in the second phase using simulated annealing. The final phase is applied to make further improvement using local search guided by simulated annealing. The basic algorithm was entered for the timetabling competition mentioned above and was able to achieve best results on 13 out of the 20 instances when compared to the other approaches in the competition. This method was the overall competition winner. The final method described in Kostuch (2005) significantly improves upon Kostuch's own results in the international timetabling competition.

2.7.2.5 Multi-criteria Approaches

Badri (1996) formulates a multi-objective method for course scheduling for the United Arab Emirates University. This is a two-stage optimisation problem. Firstly, the model tries to maximise faculty course preferences when assigning faculty members to courses. Secondly, when allocating courses to timeslots, the approach

tries to maximise faculty time preferences. Experimental results show that the model was able to offer an assignment that fulfils departmental policies, course offerings and personnel preferences.

2.7.2.6 Case-based Reasoning (CBR), Knowledge-based and Fuzzy-based Approaches

There are some papers that discuss the use of CBR in university timetabling such as Burke et al. (2000) who applied CBR to course timetabling. The course timetabling problem is modelled as attribute graphs where nodes represent the courses, edges represent conflicts and attributes on both nodes and edges represent information about the problem structure. The similarity between the target problem and the candidate cases is calculated and the most similar case(s) are selected for adaptation. The adaptation process is carried out by using a graph heuristic method which attempts to minimise constraint violations. Burke et al. (2001b) continue with this line of research and discuss this approach for solution reuse. An overview of case-based reasoning for university timetabling can be seen in Burke and Petrovic (2002), Petrovic and Burke (2004) and Burke et al. (2006a, 2006b). More details on the use of CBR applied to course timetabling can be found in Qu (2002).

Knowledge-based approaches using a rule-based system to solve problems by simulating knowledge from an experienced human expert can be modelled as a set of rules. Rules are stored in a knowledge base and are inferred by the inference engine. Partovi and Arinze (1995) applied such an approach to the faculty-course assignment problem at Drexel University. The system was developed in Prolog. The knowledge is represented in predicate logic form and there are three inferencing strategies (i.e. backward chaining, depth-first search and monotonic reasoning). The design incorporates heuristics for the purpose of efficient search of the solution space. Kong and Kwok (1999) implemented a conceptual model of a knowledge-based timetabling system for high school timetabling. There are a large number of rules to represent various types of constraints (for example, constraints for insufficient rooms, splitting and combining classes). Different heuristics were developed to infer rules in order to produce a feasible solution. The challenging work in this research was to conduct the knowledge elicitation process.

Asmuni et al. (2005b) applied fuzzy multiple heuristic ordering for the course timetabling problem which draws upon their previous work in Asmuni et al. (2005a). A fuzzy c-means clustering has been employed by Amintoosi and Haddadnia (2005) to allocate students from a course into smaller sections while satisfying student course selection, section enrolment, section capacities and student schedules in each section.

2.7.2.7 Hyper-heuristic Approaches

Petrovic and Qu (2002) presented a hyper-heuristic method that used CBR to solve the course timetabling problem. In this approach, CBR is used to select the best heuristic for the problem in hand. Details about CBR are discussed in Sections 2.7.1.8 and 2.7.2.6. Burke et al. (2003a) investigated a hyper-heuristic approach for university course timetabling and nurse rostering problems which has been discussed in the tabu search section. Burke et al. (2007) also investigated a graph-based hyper-heuristic for the course timetabling problem. Experimental results show that the approach works more efficiently when a larger number of low level heuristics are used.

2.8 Brief Summary

This chapter has introduced various types of timetabling problems. The university timetabling problem which is the applicative domain for this thesis has also been discussed together with a range of hard and soft constraints. A number of algorithmic approaches applied to university timetabling have also been highlighted. In the sixties and seventies, graph colouring heuristics were adopted to solve this problem. Then in the eighties and nineties, more advanced techniques, such as meta-heuristics (e.g. simulated annealing, tabu search and constraint-based reasoning) became more popular. New ideas have been developed in recent years to improve the quality of the solution by introducing the hybridisation of different approaches. Furthermore, recent approaches applied to university timetabling have been highlighted such as case-based reasoning, hyper-heuristics and fuzzy methods.

Chapter 3

Specification and Datasets for University Timetabling Problems

3.1 Introduction

This chapter provides a general problem specification and a presentation of the constraints of the real-world examination and course timetabling datasets that are used as benchmark problems. Formal mathematical statements are also included.

The chapter is organised as follows: Section 3.2 gives a specification of the examination timetabling problem with the formulation and examples for the uncapacitated and capacitated examination timetabling problems. Section 3.3 describes the specification for the course timetabling problem which consists of the problem definition and the example of the penalty cost calculation from the solution obtained. Section 3.4 depicts the benchmark datasets for examination and course timetabling problems, followed by some brief concluding comments in Section 3.5.

3.2 Specification of the Examination Timetabling Problem

The problem description employed in this work is adopted from Burke et al. (2004b). The input for the examination timetabling problem can be stated as follows:

- N is the number of examinations
- D is the number of days
- T is the given number of available timeslots
- M is the number of students

- $C = (c_{ij})_{N(N+2)}$ is the *conflict matrix* where each element (after the second column, see Table 3.1 for an example) denoted by c_{ij} , $i, j \in \{1, \dots, N\}$ is the number of students taking examinations i and j .
- t_k ($1 \leq t_k \leq T$) specifies the assigned timeslot for examination k ($k \in \{1, \dots, N\}$)

The examination timetabling problem in this work is divided into 2 categories: (i) the uncapacitated problem and (ii) the capacitated problem.

3.2.1 The Uncapacitated Examination Timetabling Problem

In this problem, an objective function is used to space out students' examinations throughout the examination period. The uncapacitated examination timetabling problem can be formulated as the minimisation of:

$$\frac{\sum_{i=1}^N F_1(i)}{2M} \quad (3.1)$$

where

$$F_1(i) = \sum_{j=i}^N c_{ij} * proximity(t_i, t_j) \quad (3.2)$$

is a cost for examination i (since $c_{ij} = 0$ when $i = j$) and

$$proximity(t_i, t_j) = \begin{cases} 32 / 2^{|t_i - t_j|} & \text{if } 1 \leq |t_i - t_j| \leq 5 \\ 0 & \text{otherwise} \end{cases} \quad (3.3)$$

subject to:

$$\sum_{i=1}^{N-1} \sum_{j=i+1}^N c_{ij} * \lambda(t_i, t_j) = 0 \text{ where } \lambda(t_i, t_j) = \begin{cases} 1 & \text{if } t_i = t_j \\ 0 & \text{otherwise} \end{cases} \quad (3.4)$$

Equation (3.2) presents a cost for an examination i that is given by the proximity value multiplied by the number of students in conflict. Equation (3.3) represents a proximity value between two examinations (as introduced by Carter et al. 1996). For example, if a student has two consecutive examinations then a proximity value of 16 is assigned. If a student has two examinations with a free timeslot in between then a value of 8 is assigned. The value will be 4 if there are 2 timeslots in between and so on. These values are summed up and divided by $2M$, to give an average penalty per

student (Equation (3.1)) since all clashes would otherwise be counted twice (e.g. c_{23} and c_{32}). Equation (3.4) represents a clash-free requirement so that no student is asked to sit two examinations at the same time. The clash-free requirement is considered as a hard constraint. This problem (without room capacity requirement) is tackled in Chapters 4 and 6.

A small illustrative example of the proximity cost examination timetabling problem is given below. To show how the proposed formulation works, assume that there are 15 examinations ($N = 15$) which are labelled as $e_1, \dots, e_{15}$, 6 timeslots ($T = 6$) which are labelled as $t_1, \dots, t_6$, 100 students ($M = 100$) and the *conflict matrix* C as shown in Table 3.1. The pseudo-code to generate a *conflict matrix* C is given in Figure 3.1.

Table 3.1 shows an example of a *conflict matrix* C . The specification of Table 3.1 can be outlined as follows:

- First column: represents examination index i .
- Second column: represents the number of students taking examination i .
- Third column: represents the number of examinations in conflict with examination i .
- Next N columns: represents the number of students in conflict.

Table 3.1 shows that the examinations are sorted in decreasing order based on the number of examinations in conflict. The examination with the highest number of examinations in conflict will be in the first row in the matrix and the examination with the lowest number of examinations in conflict will be in the last row in the matrix. Taking, for example, the first examination (e_3) we can see that it has 10 examinations in conflict, examination e_5 has 8 examinations in conflict and the last examination (i.e. examination e_{15}) has 3 examinations in conflict. The entries (for examination e_3) in the *conflict matrix* C can be read as examination e_3 conflicts with examination e_1 with 4 common students. Examination e_3 also conflicts with examination e_2 with 6 common students and so on. Finally examination e_3 conflicts with examination e_{15} with 2 common students.

Table 3.1. An example of a *conflict matrix, C*

			e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}	e_{13}	e_{14}	e_{15}
e_3	20	10	4	6	0	0	2	0	2	1	7	0	1	3	5	0	2
e_5	12	8	11	0	2	10	0	0	2	0	1	0	3	0	4	0	5
e_1	14	7	0	0	4	0	11	1	0	5	7	0	0	3	0	4	0
e_9	14	7	7	0	7	0	1	1	5	4	0	0	2	0	0	0	0
e_{11}	10	6	0	3	1	0	3	0	8	2	1	0	0	0	0	0	0
e_2	14	5	0	0	6	0	0	0	0	0	0	3	0	4	0	4	7
e_4	20	5	0	0	0	0	10	0	0	0	0	3	0	4	0	4	7
e_7	13	5	0	0	2	0	2	0	0	0	5	3	8	0	0	0	0
e_8	18	5	5	0	1	0	0	12	0	0	4	0	0	0	1	0	0
e_{10}	10	5	0	2	0	3	0	0	3	0	0	0	1	0	0	8	0
e_{12}	10	5	3	0	3	4	0	6	0	0	0	0	0	0	4	0	0
e_{13}	13	5	0	5	5	0	4	0	0	1	0	0	0	4	0	0	0
e_6	15	4	1	0	0	0	0	0	0	12	1	0	0	6	0	0	0
e_{14}	12	4	4	2	0	4	0	0	0	0	0	8	0	0	0	0	0
e_{15}	10	3	0	0	2	7	5	0	0	0	0	0	0	0	0	0	0

```

for every student:
    read the events taken by the student;
end for

for every event:
    calculate the number of students taking each event;
end for

for every event i:
    for every event j:
        calculate the number of students in common for every
        pair of events (examination or course) i and j where
         $i, j \in \{1, \dots, N\}$ ;
        calculate the number of events in conflict;
        note the conflicting events;
    end for
end for

for every event:
    output the event index;
    output the number of students taking each event;
    output the number of events in conflict;
    for every event:
        output the conflicting events;
        output the number of events in conflict;
    end for
end for

```

Figure 3.1. The pseudo-code to generate a *conflict matrix, C*

Assume that a feasible solution of assigning examinations to timeslots for this example problem is given as in Table 3.2.

Table 3.2. Examinations-timeslots assignment

Examination	e_3	e_5	e_1	e_9	e_{11}	e_2	e_4	e_7	e_8	e_{10}	e_{12}	e_{13}	e_6	e_{14}	e_{15}
Timeslot	t_1	t_2	t_3	t_4	t_3	t_2	t_1	t_5	t_2	t_4	t_5	t_4	t_6	t_5	t_6

Table 3.2 shows that examination e_3 is assigned to timeslot t_1 , examination e_5 is assigned to timeslot t_2 , and so on. Finally examination e_{15} is assigned to timeslot t_6 .

The total proximity coefficient for the given solution is calculated using Equation (3.3) as shown in Table 3.3. The entries in Table 3.3 can be read as “the proximity coefficient between examination e_3 and examination e_1 is 8. The proximity coefficient is 16 between examination e_3 and examination e_2 ”.

Table 3.3. Example of the proximity coefficient matrix

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}	e_{13}	e_{14}	e_{15}
e_3	8	16	0	0	16	2	16	4	8	0	0	0	2	4	1
e_5	16	0	16	16	0	0	4	0	8	0	16	0	8	0	2
e_1	0	0	8	0	16	4	0	16	16	0	0	8	0	8	0
e_9	4	0	4	0	8	8	16	8	0	0	16	0	0	0	0
e_{11}	0	16	8	0	16	0	8	16	16	0	0	0	0	0	0
e_2	0	0	16	0	0	0	0	0	0	8	16	0	8	4	0
e_4	0	0	0	0	16	0	0	0	0	8	16	0	8	4	0
e_7	0	0	2	0	4	0	0	0	16	16	8	0	0	0	0
e_8	16	0	16	0	0	2	0	0	8	0	0	0	8	0	0
e_{10}	0	8	0	4	0	0	16	0	0	0	16	0	0	16	0
e_{12}	8	0	2	2	0	16	0	0	0	0	0	0	16	0	0
e_{13}	0	8	4	0	8	0	0	8	0	0	0	16	0	0	0
e_6	4	0	0	0	0	0	0	2	8	0	0	16	0	0	0
e_{14}	8	2	0	2	0	0	0	0	0	16	0	0	0	0	0
e_{15}	0	0	1	1	2	0	0	0	0	0	0	0	0	0	0

Based on the solution in Table 3.2, the proximity cost ($F_1(i)$) for one examination, i , (i.e. examination e_3) is calculated using Equation (3.2) as shown in Figure 3.2.

$$\begin{aligned}
&= (4*8) + (6*16) + (2*16) + (2*2) + (1*16) + (7*4) + (1*8) + (3*2) + (5*4) + \\
&\quad (2*1) \\
&= 244
\end{aligned}$$

Figure 3.2. Proximity cost for examination e_3

So, the proximity cost for examination e_3 is equal to 244. Note that since, in this representation, there is a duplication in the calculation of the proximity cost for the examinations (for instance the proximity cost between examinations e_3 and e_2 is calculated and the proximity cost between examinations e_2 and e_3 is calculated once again), thus the total proximity cost for the solution in Table 3.2 is a summation of the proximity costs of each examination divided by 2 i.e. $3045/2$ (see Figure 3.3). Hence, the overall penalty is obtained by dividing the total proximity cost of the solution ($3045/2$) by the total number of students (100 in this example) giving 15.23. This value represents the average penalty cost per student.

$$\begin{aligned}
&= \{(4*8) + (6*16) + (2*16) + (2*2) + (1*16) + (7*4) + (1*8) + (3*2) + (5*4) + \\
&\quad (2*1)\} + \\
&\quad \{(11*16) + (2*16) + (10*16) + (2*4) + (1*8) + (3*16) + (4*8) + (5*2)\} + \\
&\quad \{(4*8) + (11*16) + (1*4) + (5*16) + (7*16) + (3*8) + (4*8)\} + \\
&\quad \{(7*4) + (7*4) + (1*8) + (1*8) + (5*16) + (4*8) + (2*16)\} + \\
&\quad \{(3*16) + (1*8) + (3*16) + (8*8) + (2*16) + (1*16)\} + \\
&\quad \{(6*16) + (2*8) + (3*16) + (5*8) + (2*4)\} + \\
&\quad \{(10*16) + (3*4) + (4*2) + (4*2) + (7*1)\} + \\
&\quad \{(2*2) + (2*4) + (5*16) + (3*16) + (8*8)\} + \\
&\quad \{(5*16) + (1*16) + (12*2) + (4*8) + (1*8)\} + \\
&\quad \{(2*8) + (3*4) + (3*16) + (1*16) + (8*16)\} + \\
&\quad \{(3*8) + (3*2) + (4*2) + (6*16) + (4*16)\} + \\
&\quad \{(5*8) + (5*4) + (4*8) + (1*8) + (4*16)\} + \\
&\quad \{(1*4) + (12*2) + (1*8) + (6*16)\} + \\
&\quad \{(4*8) + (2*4) + (4*2) + (8*16)\} + \\
&\quad \{(2*1) + (7*1) + (5*2)\} \\
&= 3045 / 2
\end{aligned}$$

Figure 3.3. The total proximity cost for the solution in Table 3.2

3.2.2 The Capacitated Examination Timetabling Problem

This problem considers the room capacity constraint in addition to the clash-free requirement (Equation (3.4)). In this problem, an objective function that minimises the number of students having two examinations in a row on the same day (adapted from Burke et al. 1996b) is used. Assume that examinations start on Monday. Each week day has 3 timeslots, Saturday has 1 timeslot and Sunday has none. This can be represented as a vector. Indeed, Burke et al. (2004b) present the following vector (Figure 3.4) which clearly demonstrates the idea:

(1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5, 6, 8, 8, 8, 9, 9, 9, 10, 10, 10, 11, 11, 11, 12, 12, 12, 13, 15, 15, 15, 16, 16, 16, 17, 17, 17...).

Figure 3.4. Vector of timeslots in days

It can be seen that there is just one “6” entry (one timeslot on the first Saturday – 6th day) and one “13” entry (one timeslot on the second Saturday – 13th day). Sundays (days 7 and 14) are missing because there are no timeslots on Sundays. The corresponding timeslot vector is represented in Figure 3.5.

(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,...,T).

Figure 3.5. Vector of timeslots

The day for a particular timeslot t (where $t \in \{1, \dots, T\}$) is represented by d_t . For example, d_4 would be day 2, d_7 would be day 3 and d_{17} would be day 8. The days should not be confused with the timeslots.

The problem can be represented as follows where the objective is to minimise the value of the sum (Equation (3.5)).

$$\frac{\sum_{i=1}^N F_2(i)}{2} \quad (3.5)$$

where

$$F_2(i) = \sum_{j=1}^N c_{ij} * Adj(t_i, t_j) \quad (3.6)$$

and

$$Adj(t_i, t_j) = \begin{cases} 1 & \text{if } (|t_i - t_j| = 1) \& \& (d_{t_i} == d_{t_j}) \\ 0 & \text{otherwise} \end{cases} \quad (3.7)$$

and d_{t_k} ($1 \leq d_{t_k} \leq D$) specifies the assigned day for examination k ($k \in \{1, \dots, N\}$) that is scheduled in timeslot t ($t \in \{1, \dots, T\}$), subject to Equation (3.4) and

$$Student_t \leq Seats \text{ for } t = 1, \dots, T \quad (3.8)$$

where $Student_t$ is the number of students taking examinations in timeslot t and $Seats$ is the number of seats available in each timeslot. Equations (3.6) and (3.7) represent the number of students taking examination i who are forced to take other examinations in adjacent timeslots on the same day. Inequality (3.8) represents the restriction on the room capacity. The experiments for the capacitated examination timetabling problem are addressed in Chapters 4 and 5.

Using the same example as in Section 3.2.1, the days are assigned to the examinations as shown in Table 3.4.

Table 3.4. Examinations-timeslots-days assignment

Examination	e_3	e_5	e_1	e_9	e_{11}	e_2	e_4	e_7	e_8	e_{10}	e_{12}	e_{13}	e_6	e_{14}	e_{15}
Timeslot	t_1	t_2	t_3	t_4	t_3	t_2	t_1	t_5	t_2	t_4	t_5	t_4	t_6	t_5	t_6
Day	d_1	d_1	d_1	d_2	d_1	d_1	d_1	d_2	d_1	d_2	d_2	d_2	d_2	d_2	d_2

Table 3.4 shows that examination e_3 is scheduled in timeslot t_1 on day d_1 , examination e_5 is scheduled in timeslot t_2 on day d_1 and finally examination e_{15} is scheduled in timeslot t_6 on day d_2 . From the input in Table 3.4, the adjacent period coefficient is calculated using Equation (3.7). For instance, the adjacent period coefficient for examinations e_3 and e_2 is 1 (i.e. $t_2 - t_1 = 2-1$) because the difference between the timeslots assigned to examinations e_3 and e_2 is 1 and both of them are scheduled on the same day (i.e. d_1). The coefficient value 0 means that either two conflicting examinations are not scheduled in the adjacent period on the same day or they are not conflicting (refer to the *conflict matrix* in Table 3.1 to see the conflicting

examinations). The adjacent period coefficients for all the examinations in this example are shown in Table 3.5.

Table 3.5. The example of the adjacent period coefficient

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}	e_{13}	e_{14}	e_{15}
e_3	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0
e_5	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0
e_1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0
e_9	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
e_{11}	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
e_2	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
e_4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
e_7	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
e_8	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
e_{10}	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
e_{12}	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0
e_{13}	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
e_6	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
e_{14}	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
e_{15}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

By assuming that the Formula (3.8) is always satisfied (i.e. the room capacity requirement), the calculation for the cost of the adjacent period for one examination (taking examination e_3 as an example) is given in Figure 3.6.

$$= (4*0) + (6*1) + (2*1) + (2*0) + (1*1) + (7*0) + (1*0) + (3*0) + (5*0) + (2*0) \\ = 9$$

Figure 3.6. The adjacent period cost for examination e_3

The total cost of the adjacent period for the given solution in Table 3.4 can be calculated as shown in Figure 3.7.

$$\begin{aligned}
&= \{(4*0) + (6*1) + (2*1) + (2*0) + (1*1) + (7*0) + (1*0) + (3*0) + (5*0) + (2*0)\} \\
&+ \\
&\{(11*1) + (2*1) + (10*1) + (2*0) + (1*0) + (3*1) + (4*0) + (5*0)\} + \\
&\{(4*0) + (11*1) + (1*0) + (5*1) + (7*0) + (3*0) + (4*0)\} + \\
&\{(7*0) + (7*0) + (1*0) + (1*0) + (5*1) + (4*0) + (2*0)\} + \\
&\{(3*1) + (1*0) + (3*1) + (8*0) + (2*0) + (1*0)\} + \\
&\{(6*1) + (2*0) + (3*1) + (5*0) + (2*0)\} + \\
&\{(10*1) + (3*0) + (4*0) + (4*0) + (7*0)\} + \\
&\{(2*0) + (2*0) + (5*1) + (3*1) + (8*0)\} + \\
&\{(5*1) + (1*1) + (12*0) + (4*0) + (1*0)\} + \\
&\{(2*0) + (3*0) + (3*1) + (1*0) + (8*0)\} + \\
&\{(3*0) + (3*0) + (4*0) + (6*1) + (4*1)\} + \\
&\{(5*0) + (5*0) + (4*0) + (1*0) + (4*1)\} + \\
&\{(1*0) + (12*0) + (1*0) + (6*1)\} + \\
&\{(4*0) + (2*0) + (4*0) + (8*1)\} + \\
&\{(2*0) + (7*0) + (5*0)\} \\
&= 134/2
\end{aligned}$$

Figure 3.7. The adjacent period cost for the solution in Table 3.4

Note that since the same representation as in Section 3.2.1 is being used. For instance, the cost of the adjacent period between examinations e_3 and e_2 is calculated twice (i.e. another calculation is between examinations e_2 and e_3 which carries the same adjacent period cost), so that the total cost of the adjacent period is given by $134/2$ which is equal to 67.

3.3 Specification of the Course Timetabling Problem

In this work, we used the hard and soft constraints as presented by Socha et al. (2002). The hard constraints are:

- No student can be assigned to more than one course at the same time.
- A room should satisfy the features required by the course assigned to it.
- The number of students attending the course should be less than or equal to the capacity of the room.
- No more than one course is allowed at a timeslot in each room.

The soft constraints that are equally penalised are:

1. A student has a course scheduled in the last timeslot of the day.
2. A student has more than two consecutive courses.
3. A student has a single course in a day

3.3.1 Problem Definition

In this work, the same course timetabling problem as described in Rossi-Doria et al. (2002, 2003) and Socha et al. (2002) is considered. The problem consists of:

- N is the number of courses labelled as $\{e_1, e_2, \dots, e_N\}$
- T is the number of timeslots which is equal to 45 (5 days with 9 timeslots each day)
- R is the number of rooms
- A set of F room features
- M is the number of students.

The objective of this problem is to minimise the number of soft constraint violations in a feasible solution. A solution is represented as an ordered list of length N where the position corresponds to the courses i.e. position i corresponds to course e_i for $i = 1, \dots, N$. Each position has two values which are numbered between 0 to $T-1$ and 0 to $R-1$ that correspond to the timeslot index and room index, respectively. For example, given a timeslot vector of $(0, 17, 30, \dots, 10)$ and a room vector as $(4, 3, 0, \dots, 3)$. We can see that course e_1 is scheduled in timeslot 0 at room 4, course e_2 is scheduled in timeslot 17 at room 3 and, finally, course e_N is scheduled in timeslot 10 at room 3.

3.3.2 An Evaluation Example of Soft Constraint Violation

This section uses standard benchmark problems from *small* instances proposed by the Meta-heuristics Network¹. In the *small* instances, there are 100 courses ($N = 100$), 5 rooms ($R = 5$), 5 room features ($F = 5$) and 80 students ($M = 80$). The details of the benchmark problems for course timetabling are discussed in Section 3.4. Using the inputs of the course timetabling problem (as in Section 3.3.1), a *student-event matrix*, a *suitable-room matrix* and a *conflict matrix* are generated. A *student-event matrix* is an MN matrix where each element in the matrix is represented by “1”

¹ <http://www.metaheuristics.net>

or “0”. The value “1” shows the courses attended by the student i where $i \in \{1, \dots, M\}$. Otherwise the element in the *student-event matrix* is represented by “0”. The pseudo-code to generate a *student-event matrix* is given in Figure 3.8.

```

for i = 0 to i < number of students
  for j = 0 to j < number of courses
    if student[i] attends event[j] then
      student-event[i][j] = 1;
    else
      student-event[i][j] = 0;
    end for
  end for
end for

```

Figure 3.8. The pseudo-code to generate a *student-event matrix*

Tables 3.6a-g show examples of the *student-event matrix* for one of the *small* instances. The specification of the *student-event matrix* is as follows:

- First column: represents student m where $m \in \{1, \dots, M\}$
- Remaining columns: represent courses taken by student m

Table 3.6a. The *student-event matrix* for courses $e_i, i \in \{1, \dots, 15\}$

$e_i, i \in \{1, \dots, 15\}$ / Student	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
80	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0

Table 3.6b. The *student-event matrix* for courses $e_i, i \in \{16, \dots, 30\}$

$e_i, i \in \{16, \dots, 30\}$ / Student	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
80	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0

Table 3.6c. The student-event matrix for courses $e_i, i \in \{31, \dots, 45\}$

$e_i, i \in \{31, \dots, 45\}$ / Student	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
2	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.6d. The student-event matrix for courses $e_i, i \in \{46, \dots, 60\}$

$e_i, i \in \{46, \dots, 60\}$ / Student	46	47	48	49	50	51	52	52	54	55	56	57	58	59	60
1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
80	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0

Table 3.6e. The student-event matrix for courses $e_i, i \in \{61, \dots, 75\}$

$e_i, i \in \{61, \dots, 75\}$ / Student	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
80	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.6f. The student-event matrix for courses $e_i, i \in \{76, \dots, 90\}$

$e_i, i \in \{76, \dots, 90\}$ / Student	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90
1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
80	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0

Table 3.6g. The *student-event matrix* for courses $e_i, i \in \{91, \dots, 100\}$

$e_i, i \in \{91, \dots, 100\}$ / Student	91	92	93	94	95	96	97	98	99	100
1	0	0	0	0	0	0	0	1	0	0
2	0	0	0	0	0	0	0	1	0	0
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
80	0	0	0	0	0	0	0	1	0	0

Note that only parts of the elements of the *student-event matrix* are represented in Tables 3.6a-g. The entries in Tables 3.6a-g can be read as “student 1 taking courses $e_{11}, e_{41}, e_{55}, e_{56}, e_{63}, e_{77}, e_{88}$ and e_{98} ” (refer to the first row).

A *suitable-room matrix* is generated using the input of courses that require a specific room and the input of a set of room features (F). The elements in the *NR suitable-room matrix* are also represented by “1” and “0” indicating, respectively, that the room can or cannot be used for course e_i where $i \in \{1, \dots, N\}$. The pseudo-code to generate a *suitable-room matrix* is given in Figure 3.9.

Table 3.7 shows an example of the *suitable-room matrix* with 100 courses and 5 rooms. The specification of the *suitable-room matrix* is given as:

- First column: represents course e_i where $i \in \{1, \dots, N\}$
- Next R columns: represent the room suitability for room indexed as $[0, R-1]$

```

for i = 0 to i < number of courses
  for j = 0 to j < number of features
    Assign features needed by each courses,
    event-features[i][j] = 0 or 1;
  end for
end for

for i = 0 to i < number of rooms
  for j = 0 to j < number of features
    Assign the features for each room,
    room-features[i][j] = 0 or 1;
  end for
end for

for i = 0 to i < number of rooms
  Assign capacity for each room, room-size[i];
end for

for i = 0 to i < number of courses
  for j = 0 to j < number of students
    Calculate the number of students for each course as
    studentNumber[i] = studentNumber[i] + student-
    event[j][i];
  end for
end for

for i = 0 to i < number of courses
  for j = 0 to j < number of rooms
    if (room-size[j] >= studentNumber[i])
      for k = 0 to k < number of features
        if (event-features[i][k] == 1 &&
            room-features[j][k] == 0)
          suitable-room[i][j] = 0;
        else
          if (event-features[i][k] == 1 &&
              room-features[j][k] == 1)
            suitable-room[i][j] = 1;
          end if
        end if
      end for
    end if
  end for
end for

```

Figure 3.9. The pseudo-code to generate a *suitable-room matrix*

The entries in Table 3.7 show that rooms 0 and 4 are suitable for course e_1 , room index 0 is suitable for course e_2 and rooms 0 and 1 are suitable for course e_{100} . This means that course e_1 can only be scheduled in room 0 or room 4. Course e_2 can only be scheduled in room 0 and course e_{100} can only be assigned to room 0 or 1.

Table 3.7. A suitable-room matrix

	0	1	2	3	4
e_1	1	0	0	0	1
e_2	1	0	0	0	0
e_3	1	1	0	0	0
e_4	0	0	0	1	1
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
e_{100}	1	1	0	0	0

A *conflict matrix* is a $N(N+2)$ matrix in which the elements are similar to the *conflict matrix* of the examination timetabling problem as discussed in Section 3.2.1. The pseudo-code to generate a *conflict matrix* for course timetabling is similar to examination timetabling, as shown in Figure 3.1. Assume that a feasible solution of assigning courses to timeslots and rooms for this example problem is given as a vector of timeslots and rooms as shown in Figures 3.10 and 3.11, respectively.

(21, 15, 19, 18, 16, 21, 9, 20, 9, 14, 8, 15, 15, 4, 19, 14, 18, 17, 16, 7, 13, 12, 20, 15, 7, 17, 2, 5, 11, 14, 10, 8, 14, 4, 8, 13, 19, 2, 13, 12, 6, 12, 11, 11, 9, 10, 10, 9, 8, 9, 8, 7, 21, 6, 7, 1, 22, 6, 5, 13, 3, 13, 5, 14, 2, 5, 4, 10, 12, 1, 20, 3, 4, 22, 6, 17, 2, 3, 7, 16, 5, 4, 2, 1, 0, 0, 12, 0, 3, 6, 16, 1, 1, 0, 11, 0, 10, 34, 19, 3)

Figure 3.10. An example of a vector of timeslots

(0, 0, 1, 3, 1, 4, 3, 4, 2, 1, 3, 3, 2, 3, 4, 0, 4, 4, 4, 3, 0, 0, 3, 4, 2, 3, 2, 3, 0, 2, 0, 0, 4, 2, 2, 4, 0, 3, 1, 4, 2, 1, 1, 4, 0, 1, 4, 4, 4, 1, 1, 0, 3, 4, 4, 3, 3, 0, 2, 2, 3, 3, 4, 3, 1, 0, 4, 3, 3, 2, 0, 4, 0, 0, 1, 0, 4, 0, 1, 3, 1, 1, 0, 1, 2, 1, 2, 4, 2, 3, 0, 0, 4, 0, 3, 3, 2, 0, 3, 1)

Figure 3.11. An example of a vector of rooms

From Figures 3.10 and 3.11, the solution can be read as “course e_1 is scheduled in timeslot 21 (refer to the first element in Figure 3.10) in room 0 (refer to the first element in Figure 3.11). Course e_2 is scheduled in timeslot 15 in room 0. Finally, course e_{100} is scheduled in timeslot 3 in room 1. Using the input from the *student-*

event matrix and the vector of timeslots as in Tables 3.6a-g and 3.10, a *student-availability matrix* is generated. The *student-availability matrix* shows the courses taken by the student and the timeslots assigned to them. It is an *MT* matrix where each element in the matrix is represented by 0 or 1 indicating respectively that the student m , where $m \in \{1, \dots, M\}$, attends or does not attend the course that is assigned to timeslot k where $k \in \{0, \dots, T-1\}$. The pseudo-code to generate the *student-availability matrix* is given in Figure 3.12.

```

for i = 0 to i < number of students
  for j = 0 to j < number of courses
    for k = 0 to k < number of timeslots
      if student-event[i][j]==1 then
        Get the timeslot k;
        Assign the student-availability
        matrix[i][k] = 1;
      else
        Assign the student-availability
        matrix[i][k] = 0;
      end for
    end for
  end for
end for

```

Figure 3.12. The pseudo-code to generate a *student-availability matrix*

Tables 3.8a-c show the representation of the *student-availability matrix*. Note that only some of the elements of the *student-availability matrix* are represented. The entries in Tables 3.8a-c (refer to the first row) can be read as “student 1 takes courses that are scheduled in timeslots 0, 1, 2, 5, 6, 7, 8 and 34, student 2 takes courses that are assigned to timeslots 0, 1, 2, 5, 9, 10, 12, 15, 19, 20 and 34. Finally, student 80 takes courses that are scheduled in timeslots 0, 2, 5, 7, 10, 17 and 34.

Table 3.8a. The *student-availability matrix* for timeslot t_i , $i \in \{0, \dots, 16\}$

$t_i /$ Student	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0
2	1	1	1	0	0	1	0	0	0	1	1	0	1	0	0	1	0
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
80	1	0	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0

Table 3.8b. The student-availability matrix for timeslot t_i , $i \in \{17, \dots, 33\}$

$t_i /$ Student	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
80	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.8c. The student-availability matrix for timeslot t_i , $i \in \{34, \dots, 44\}$

$t_i /$ Student	34	35	36	37	38	39	40	41	42	43	44
1	1	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.
80	1	0	0	0	0	0	0	0	0	0	0

The quality of the solution is calculated based on the violation of the soft constraints. Please note that the soft constraints considered in course timetabling can be seen in Section 3.3. The pseudo-code for the calculation of each soft constraint violation is given as follows:

- (i) First soft constraint: A student has more than two courses in a row (see Figure 3.13).
- (ii) Second soft constraint: A student has one course on a day (see Figure 3.14).
- (iii) Third soft constraint: A student has a class in the last timeslot of the day (see Figure 3.15).

```

set first soft constraint penalty, P1 = 0;
for each student
  for each day
    set the number of courses in a row, NumOfCourses = 0;
    for each timeslots, t for each day
      if student-availability matrix at timeslot t == 1
        NumOfCourses++;
      else
        NumOfCourses = 0;
        if (NumOfCourses >= 3)
          P1++;
        end for
      end for
    end for
  end for
end for

```

Figure 3.13. The pseudo-code for the first soft constraint violation

```

set second soft constraint penalty, P2 = 0;
for each student
  for each day
    set the number of courses in a row, NumOfCourses = 0;
    for each timeslots, t for each day
      if student-availability matrix at timeslot t == 1
        NumOfCourses++;
        if (NumOfCourses == 1)
          P2++;
        end for
      end for
    end for
  end for
end for

```

Figure 3.14. The pseudo-code for the second soft constraint violation

```

set third soft constraint penalty, P3 = 0;
for each student
  if student-availability matrix at timeslots 8 or 17 or
  26 or 35 or 44 ==1
    P3++;
  end for
end for

```

Figure 3.15. The pseudo-code for the third soft constraint violation

The total penalty can be calculated as in Figure 3.16.

$$\text{Total penalty} = P1 + P2 + P3$$

Figure 3.16. Formula to calculate the total penalty for the course timetabling problem

From the *student-availability matrix* (Tables 3.8a-c), it can be seen that for the case of:

- student 1: first soft constraint penalty = 3 (at timeslots 2, 7 and 8)
second soft constraint penalty = 1 (at timeslot 34)
third soft constraint penalty = 1 (at timeslot 8)
- student 2: first soft constraint penalty = 1 (at timeslot 2)
second soft constraint penalty = 1 (at timeslot 34)
third soft constraint penalty = 0
- student 80: first soft constraint penalty = 0
second soft constraint penalty = 1 (at timeslot 34)
third soft constraint penalty = 1 (at timeslot 17).

Note that the first, second and third soft constraint penalties need to be checked for all the students. The summation of these soft constraint penalties will result in the total penalty of the solution using the formula in Figure 3.16.

3.4 Benchmark Datasets

This section gives standard benchmark problems for both examination and course timetabling.

3.4.1 The Examination Timetabling Datasets

All experiments for the examination timetabling problem discussed in this thesis were carried out from the following open sources:

- Michael Carter's collection of examination timetabling data, which can be freely downloaded from <ftp://ftp.mie.utoronto.ca/pub/carter/testprob/> comprising 13 datasets which were collected from different universities.
- A dataset from The University of Nottingham in 1994 that includes 800 examinations, 7896 students and 33997 enrolments that can be downloaded from <ftp://ftp.cs.nott.ac.uk/ttp/Data/>.

The experiments for the uncapacitated problems considered the datasets from Michael Carter's collection. The numbers of timeslots are obtained from the results reported by Carter et al. (1996). For the capacitated examination timetabling

problem, certain benchmarks from Michael Carter’s collection and the University of Nottingham dataset are taken into account. The specifications of the uncapacitated and capacitated examination timetabling problems are shown in Tables 3.9 and 3.10, respectively.

The number of other examinations that each examination conflicts with is summed up and divided by the total number of examinations, to give an average *conflict matrix* density for each dataset in Carter et al. (1996). The same formula as above is used to calculate the *conflict matrix* density for the *nott-94* dataset as shown in Table 4.10 (since it is not available in Burke et al. 1996b).

Table 3.9. The uncapacitated benchmark datasets

Datasets	Institution	Number of timeslots	Number of examinations	Number of students	Conflict matrix density
<i>car-f-92</i>	Carleton University, Ottawa	32	543	18419	0.14
<i>car-s-91</i>	Carleton University, Ottawa	35	682	16925	0.13
<i>ear-f-83</i>	Earl Haig Collegiate Institute, Toronto	24	190	1125	0.29
<i>hec-s-92</i>	Ecole des Hautes Etudes Commerciales, Montreal	18	81	2823	0.42
<i>kfu-s-93</i>	King Fahd University, Dharan	20	461	5349	0.06
<i>lse-f-91</i>	London School of Economics	18	381	2726	0.06
<i>rye-s-93</i>	Ryerson University, Toronto	23	486	11483	0.07
<i>sta-f-83</i>	St. Andrew’s Junior High School, Toronto	13	139	611	0.14
<i>tre-s-92</i>	Trent University, Peterborough, Ontario	23	261	4360	0.18
<i>uta-s-92</i>	Faculty of Arts and Sciences, University of Toronto	35	622	21267	0.13
<i>ute-s-92</i>	Faculty of Engineering, University of Toronto	10	184	2750	0.08
<i>yor-f-83</i>	York Mills Collegiate Institute, Toronto	21	181	941	0.27

Table 3.10. The capacitated benchmark datasets

Datasets	Number of examinations	Number of timeslots	Room capacity	Conflict matrix density
<i>car-f-92</i>	543	31	2000	0.14
<i>car-s-91</i>	682	51	1550	0.13
<i>kfu-s-93</i>	461	20	1955	0.06
<i>tre-s-92</i>	261	35	655	0.18
<i>uta-s-92</i>	622	38	2800	0.13
<i>nott-94</i>	800	26	1550	0.03

3.4.2 The Course Timetabling Datasets

The experiments for the course timetabling problem discussed in this thesis were tested on benchmark course timetabling problems proposed by the Meta-heuristics Network. They need to schedule 100-400 courses into a timetable with 45 timeslots corresponding to 5 days of 9 hours each, while satisfying room features and room capacity constraints. They are divided into three categories: *small*, *medium* and *large* problems where 11 instances are generated i.e. 5 *small*, 5 *medium* and 1 *large*. The parameter values defining the categories are given in Table 3.11.

Table 3.11. The parameter values for the course timetabling problem categories

Category	<i>small</i>	<i>medium</i>	<i>large</i>
Number of courses	100	400	400
Number of rooms	5	10	10
Number of features	5	5	10
Number of students	80	200	400
Maximum courses per student	20	20	20
Maximum student per courses	20	50	100
Approximation features per room	3	3	5
Percent feature use	70	80	90

3.5 Summary

This chapter has described the specification and the formulation of the basic variant of examination timetabling which tries to spread examinations as evenly as possible throughout the schedule. This specification is categorised as the uncapacitated problem which takes into consideration only one hard constraint i.e. no student can

take two or more examinations simultaneously. Apart from this, there is another specification for the examination timetabling problem that is known as the capacitated problem. In this problem, the room capacities constraint is an additional constraint beside a clash-free requirement (no student can take two or more examinations at the same time). The objective function applied to the capacitated problem evaluates the closeness of the examinations on the same day.

The specification of the course timetabling problem is also discussed in this chapter. The goal is to minimise the number of students who have a course in the last timeslot of the day, more than two courses in a row and only one course in a day while satisfying the hard constraints. The standard benchmark problems for both examination and course timetabling problems are also presented in this chapter. The next six chapters (i.e. Chapters 4 to 9) used the datasets presented in this chapter to test the efficiency of each developed approach.

PART II

New Examination Timetabling Approaches

Chapter 4

Investigating Ahuja-Orlin's Large Neighbourhood Approach for Examination Timetabling

4.1 Introduction

This chapter presents an initial investigation into a very large scale neighbourhood search for the examination timetabling problem as described in Chapter 3. This solution search methodology, originally developed by Ahuja et al. (2001a), has been applied successfully in the past to a number of difficult combinatorial optimisation problems. The main idea is to build upon a sequential solution improvement technique which searches efficiently over a very large set of “adjacent” (neighbourhood) solutions. It is based on searching a very large neighbourhood of solutions using graph theory based algorithms implemented on a so called improvement graph. It identifies improvement moves by solving negative cost partition-disjoint graph cycles using a modified shortest path label-correcting algorithm adjusted from Ahuja et al. (1993). As can be seen in Chapter 2, most of the search methodologies for this problem in the literature use small neighbourhood structures. At the time of writing this thesis, no similar research has been carried out on applying large neighbourhood search to examination timetabling. Thus, the aim of this research is to explore a new search methodology in this important area.

The overall contributions that are presented in this chapter can be outlined as follows:

- The examination timetabling problem is treated as a variant of the problem in which examinations are partitioned into cells. Each such cell is assigned a timeslot.
- A very large scale neighbourhood structure is created by employing a cyclic-exchange neighbourhood that is substantially larger than a two-exchange neighbourhood structure, which is perhaps exponential in terms of the input size (by moving examinations from one cell to another and so on).
- Large neighbourhood search techniques are developed which are based upon an improvement graph to identify an improved neighbour implicitly without evaluating all the neighbours in the neighbourhood.
- The evaluation of the algorithm is undertaken by presenting a series of computational results on benchmark instances for examination timetabling (see Chapter 3, Tables 3.9 and 3.10).

The work presented is accepted for publication in OR Spectrum (Abdullah et al. 2006a). An earlier version of this work appeared as an abstract in the proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT V) (Abdullah et al. 2004).

This chapter is organised as follows: Section 4.2 discusses literature reviews arising from the investigation of a very large scale neighbourhood search. Section 4.3 translates the problem into the model and discusses the neighbourhood structure which includes the representation of the problem as a partitioning problem, a cyclic-exchange operation to create a neighbourhood and, the major part of this algorithm, an improvement graph. Section 4.4 presents the pseudo-code used in this work. The experiments and results are discussed in Section 4.5 and finally, Section 4.6 presents a summary of the chapter.

4.2 A Very Large Scale Neighbourhood Search: A Literature Overview

A neighbourhood search algorithm is used to find a better solution by searching the neighbours of the current solution (Ahuja et al. 2002a). This local improvement algorithm works as follows:

- Start with a feasible solution x .
- Define a neighbourhood of x , say $N(x)$
- Replace x by an improved neighbour, $x' \in N(x)$ with respect to the fitness function and repeat the process until the stopping condition is met.

Most of the neighbourhood search algorithms in the literature use small neighbourhoods (see Chapter 2). In the very large scale neighbourhood search algorithms, this is not the case, thus it is difficult to explicitly carry out the searching process to find an improving neighbour. Implicit enumeration methods are used instead. This searching strategy has been successfully implemented in the past and applied to a number of difficult combinatorial optimisation problems. Some parts of this thesis (i.e. Chapters 4, 5 and 6) have adapted the search idea from Ahuja et al. (2001a) for the examination timetabling problem for which the size of the neighbourhood considered is very large. The detail of this very large neighbourhood search is discussed in Section 4.4. To date there has been significant research activities devoted to a very large scale neighbourhood structures in various domains such as transportation, medical, telecommunication network design and in defence-related applications. This section gives a brief description of the work in these areas.

Ahuja et al. (2000) proposed two very large scale neighbourhood search algorithms for the capacitated minimum spanning tree problem to determine a minimum cost spanning tree in a telecommunication network where nodes have specified demands. These two very large scale neighbourhood searches are called node-based and tree-based. The neighbourhood structure is obtained by performing multi-exchanges between several trees where each tree contributes to a single node and a sub-tree, respectively. The experiments for both very large scale neighbourhood search algorithms give the best performance in two different cases where node-based

performed well in homogenous demands and tree-based performed well in heterogeneous demands. The extension of these two very large scale neighbourhood searches in which the neighbourhood is created through the unification of node-based and tree-based exchanges has been carried out by the same authors (Ahuja et al. 2003a). It is called a composite very large scale neighbourhood structure and has also been applied to the capacitated minimum spanning tree problem. This composite neighbourhood structure outperforms both the previous neighbourhood search algorithms applied to the same capacitated minimum spanning tree problem.

Ahuja et al. (2002a) continue the research on a very large scale neighbourhood by solving the quadratic assignment problem. This problem consists of assigning m facilities to m locations. The authors define a multi-exchange neighbourhood structure that corresponds to a cyclic sequence of the facilities in which this multi-exchange is referred to as k -exchange i.e. we have k different cyclic sequences. For instance, a cyclic sequence $2 \rightarrow 1 \rightarrow 5 \rightarrow 2$ indicates a 3-exchange, thus the larger the cycle length, the greater the neighbourhoods. It means that this exchange operation produces a large neighbourhood structure. From the experiments carried out, the authors show that the use of the concept of an improvement graph that is generated from multi-exchange neighbourhoods gives an added value to the commonly used 2-exchange neighbourhoods.

Ahuja et al. (2004a) carried out research on a fleet assignment model, in which the size of the neighbourhood is very large, by performing "A-B swaps" where A and B represent fleet types. "A-B swap" operations change fleet types of some legs from A to B and vice versa. The neighbourhood search algorithm using "A-B swaps" falls within the category of a very large scale neighbourhood search algorithm (Thompson and Orlin, 1989). The improved neighbour in the A-B improvement graph is identified heuristically. The implemented experiments show encouraging results on this problem domain.

In the medical domain, Ahuja and Hamacher (2005) introduced a very large scale neighbourhood structure within radiation therapy treatment planning for cancer patients where beams of radiation are sent from different angles to the cancerous and

healthy tissues, so that both healthy and cancerous tissues will receive the right dose. The neighbourhood is created by changing the cross-section of the beam.

The capacitated facility location problem deals with the allocation of facilities to serve a set of customers at minimum cost. Ahuja et al. (2002b) proposed a very large scale neighbourhood technique for a facility location problem. In their research, the neighbourhood structure is created by exchanging either a single customer or partitions of customers among facilities in a cyclic mode called the single customer or multi-customer neighbourhood, respectively. The computational results show the effectiveness of the multi-exchange neighbourhood in solving this problem. In a defence application, Ahuja et al. (2003b) conducted research into a weapon target assignment problem that assigns n weapons to m targets optimally with the minimum survival value of the targets. This problem is formed as a partition problem where the set of weapons is partitioned into n partitions of targets. The neighbourhood structure is produced by performing a multi-exchange (cyclic multi-exchange and path multi-exchange) that is defined as a sequence of weapons which yields a large number of neighbours. This search algorithm gives either optimal or almost optimal solutions for all instances used in the experiments. Ahuja et al. (2004b) implemented a new technique which is classified as a very large scale optimisation technique to handle a United States railroad shipments problem. The authors developed a very large scale neighbourhood search technique and, based on the experiments carried out, they presented very encouraging results which reduced the cost of the railroad shipment.

Agarwal et al. (2003) applied a very large scale neighbourhood search algorithm to the vehicle routing problem by defining the concept of a composite improvement graph. The experiment implemented shows that a very large scale neighbourhood search algorithm provides a very high-quality local optimum solution.

Yagiura et al. (2004) proposed a tabu search algorithm with a sophisticated neighbourhood called the *chained shift* neighbourhood for the multi-resource generalised assignment problem which assigns n jobs to m agents at a minimum cost while satisfying multi-resource constraints for each agent. The large neighbourhood (a *chained shift* neighbourhood) is obtained from a sequence of shift moves. For

example, let us consider a job J_i , $i \in \{1,2,3\}$ and an agent A_j , $j \in \{1,2,3\}$. A *chained shift* move of jobs J_1 , J_2 and J_3 in this example is created through ejection and insertion operations where the operations can be stated as “the ejection of job J_1 from agent A_1 and an insertion of job J_2 into agent A_1 . The ejection of job J_2 from agent A_2 and an insertion of job J_3 into agent A_2 , and an ejection of job J_3 from agent A_3 and insertion of job J_1 into agent A_3 ”. This illustrates the shift moves of length three. The created neighbourhood is searched effectively via the improvement graph. The computational results showed that the method is effective, especially on very difficult instances in this domain.

Previous research shows good and encouraging results for most of the applications discussed above. This indicates the merit of a very large scale neighbourhood search within the context of combinatorial optimisation problems with an exponentially large number of neighbours. Some discussion on the method and a comprehensive survey of a very large scale neighbourhood techniques are highlighted in Ahuja et al. (2000, 2002c). Other related research using a very large scale neighbourhood search can be found in Ahuja et al. (2005a, 2005b).

4.3 Modelling the Examination Timetabling Problem

To investigate solution procedures for examination timetabling which are based upon the methodologies of Ahuja et al. (2001a), the examination timetabling problem is addressed as a partitioning problem.

4.3.1 Partitioning the Problem

Let $\{S_t: t \in \{1, \dots, T\}, S_t \subseteq B\}$ (where B is the set of all N examinations) denote a feasible partition of examinations that are to be scheduled at timeslot t . The partition $\{S_t\}$ divides the set B such that $\cup_{t \in \{1, \dots, T\}} S_t = B$ and $S_q \cap S_t = \emptyset$ for all $q, t \in \{1, \dots, T\}$, $q \neq t$. Following standard terminology, the partition subsets will be referred to as *cells*.

4.3.2 Cyclic Exchange Neighbourhood Structure

Consider a set $\{e_1, e_2, \dots, e_L\}$ of examinations (where $L \leq N$), each of which belongs to a different cell S_t . A neighbourhood structure is created by transferring single examinations among the cells. This can be best demonstrated by presenting a small illustrative example assuming that there are three timeslots ($t \in \{1, 2, 3\}$) and seven examinations (denoted by $e_1, \dots, e_7$). By employing the notations described above, the example is partitioned into three cells, S_t ($t \in \{1, 2, 3\}$). Suppose that the three cells are: $S_1 = \{e_2, e_3, e_7\}$, $S_2 = \{e_4, e_6\}$ and $S_3 = \{e_1, e_5\}$ (see Figure 4.1). To form a neighbourhood structure using a cyclic exchange operation, an examination (say e_3) is selected from S_1 and inserted into S_2 , e_6 moves from S_2 to S_3 and finally e_1 moves from S_3 to S_1 , thus completing a cycle of changes (see Figure 4.2). A cyclic exchange is represented as $e_3 \rightarrow e_6 \rightarrow e_1 \rightarrow e_3$ where e_3 , e_6 and e_1 all belong to different cells. This notation ($e_3 \rightarrow e_6 \rightarrow e_1 \rightarrow e_3$) can be read as “ e_3 goes to e_6 's old cell, e_6 goes to e_1 's old cell and e_1 goes to e_3 's old cell”. After the cyclic exchange S_1 , S_2 and S_3 have been transformed into S'_1 , S'_2 and S'_3 where $S'_1 = \{e_1, e_2, e_7\}$, $S'_2 = \{e_3, e_4\}$ and $S'_3 = \{e_5, e_6\}$ (see Figure 4.3). The number of examinations in each cell remains unchanged. Of course, the represented neighbourhood can be very large. A cell is called feasible if the hard constraints are not violated. Path exchange is defined similarly to cyclic exchange but without exchanging back to the first cell i.e. no examination from the last cell (S_3 in this case) is inserted into the first cell (S_1 in this case).

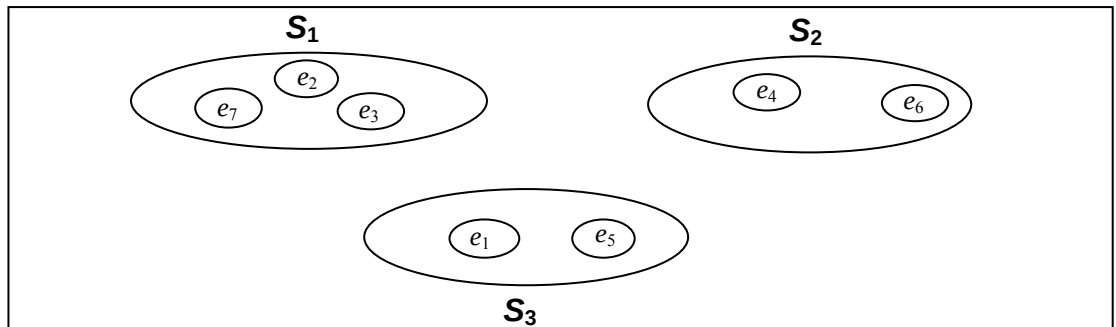


Figure 4.1. Cells before a cyclic exchange operation

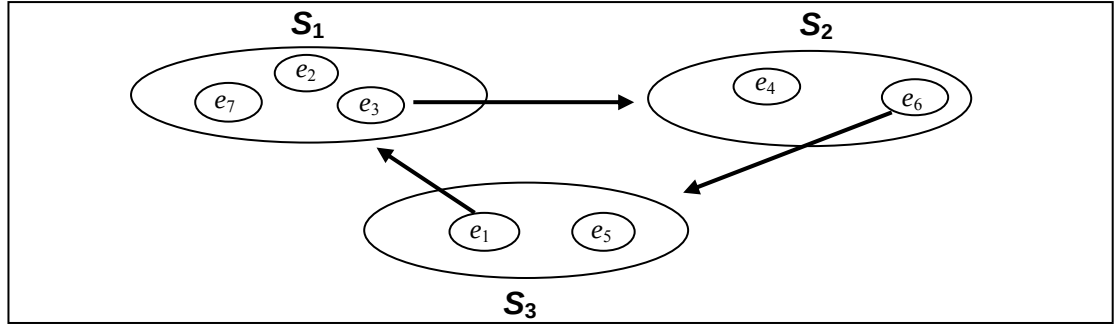


Figure 4.2. A cyclic exchange operation takes place

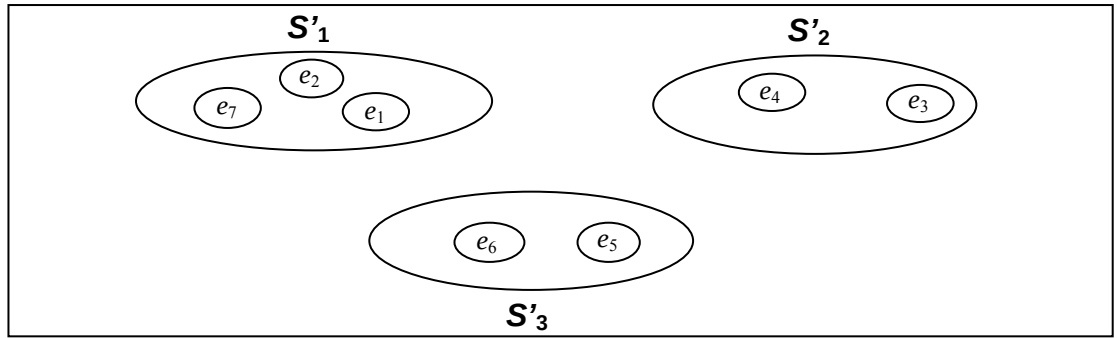


Figure 4.3. Cells after a cyclic exchange operation

4.3.3 Improvement Graph

The improvement graph was introduced in Thompson and Orlin (1989) and was first explored in Thompson and Psaraftis (1993). The improvement graph $G = (V, A)$ is a directed graph comprising of the vertex set V and the arc set A with a cost assigned to each arc $(i, j) \in A$. Each vertex i in V corresponds to an examination i in N and $A = \{(i, j) \mid i, j \in V\}$. An arc $(i, j) \in A$ signifies that the examination i moves from its current cell to the cell that contains examination j and the ejection of examination j from its current cell is feasible.

The improvement graph for the cyclic exchange neighbourhood is defined with respect to the feasible cell S_t , ($t \in \{1, \dots, T\}$). Taking into account the same simple illustrative example as the one presented in Section 4.3.2, the improvement graph is constructed by considering every pair of examinations (e_r, e_z) where $r, z \in \{1, \dots, N\}$. The directed arc (e_r, e_z) to G is added if and only if (i) the examinations e_r and e_z belong to different cells and (ii) the cell $\{e_r\} \cup \{\text{the old cell which } e_z \text{ belonged to}\} \setminus \{e_z\}$ is feasible. For instance, the directed arc $(7, 1)$ as shown in Figure 4.4, signifies

that examination e_7 leaves its current cell (S_1) and moves to the cell containing examination e_1 (S_3 in this case) and at the same time examination e_1 leaves S_3 . Examination e_7 is referred as an “*inserted examination*”, e_1 as an “*ejected examination*”, and S_3 as the “*old cell of the ejected examination*”.

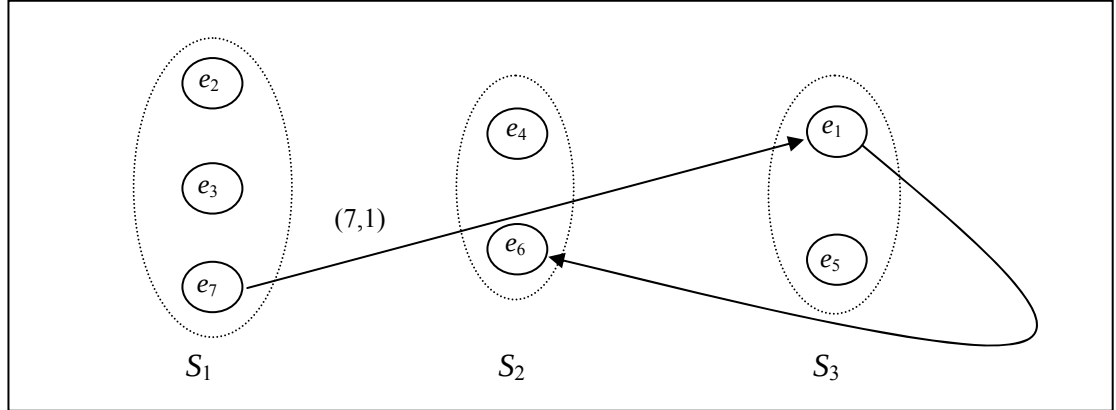


Figure 4.4. The directed arc (7, 1) of inserting examination e_7 and ejecting examination e_1

Assume that the *conflict matrix*, C_{ij} where $i, j \in \{1, \dots, 7\}$ for the example in Section 4.3.2 is that represented by Table 4.1 and that the feasible initial solution is given in Table 4.2. Note that the specifications of the conflict and solution matrices are described in Chapter 3.

Table 4.1. *Conflict matrix of the example in Section 4.3.2*

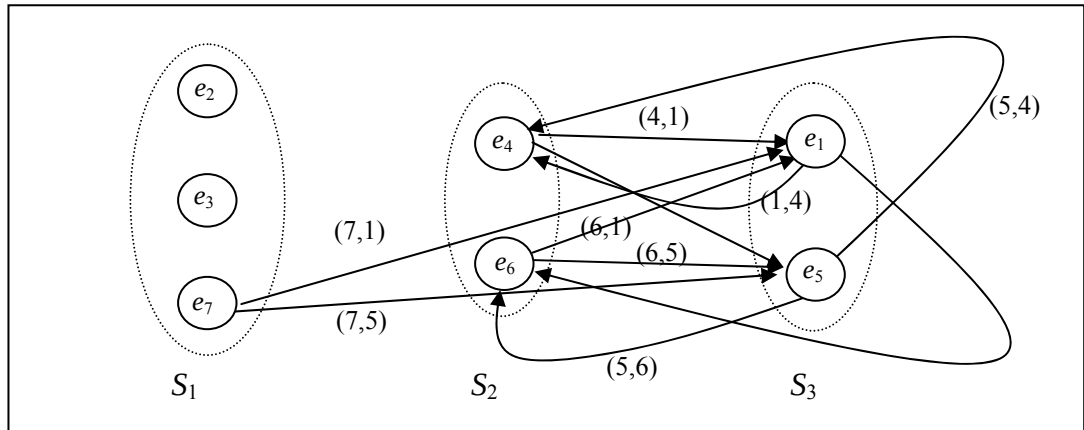
			e_1	e_2	e_3	e_4	e_5	e_6	e_7
e_2	7	3	0	0	0	2	1	3	0
e_3	4	2	2	0	0	1	0	0	0
e_4	3	2	0	2	1	0	0	0	0
e_6	5	2	0	3	0	0	0	0	1
e_1	3	1	0	0	2	0	0	0	0
e_5	2	1	0	1	0	0	0	0	0
e_7	2	1	0	0	0	0	0	1	0

Table 4.1 shows that examination e_1 (as an example) is conflicting with examination e_3 and that 2 students take both examinations.

Table 4.2. Example of the feasible initial solution

Examination	e_1	e_2	e_3	e_4	e_5	e_6	e_7
Timeslot	t_3	t_1	t_1	t_2	t_3	t_2	t_1

Table 4.2 shows a feasible initial solution where examination e_1 is scheduled in timeslot t_3 and examination e_7 is scheduled in timeslot t_1 . To construct an improvement graph, select one examination from one cell. Let us say that the selected examination is e_7 from cell S_1 . The directed arc is created by inserting examination e_7 from cell S_1 to a new cell (say S_3) and at the same time ejecting one examination from S_3 (say e_1) and maintaining the feasibility (i.e. it does not violate the hard constraints represented in Equations (3.4) and (3.8)). Figure 4.4 shows the directed arc created by inserting examination e_7 and ejecting examination e_1 . This is coded as (7,1). The new cells (after the inserted and ejected operations) are still feasible i.e. even though examination e_7 is now in cell S_3 , it does not have any students that sit both examinations e_7 and e_5 (which is also in cell S_3). We assume that the room capacity requirement in Equation (3.8) is always satisfied. The possible directed arcs in the improvement graph, for the example in Section 4.3.2, are shown in Figure 4.5.

**Figure 4.5. The improvement graph of the example in Section 4.3.2**

From Equation (3.1), it is easy to see that the total cost of Equations (3.1) and (3.5) can be defined in terms of cell $S_t (t \in 1, \dots, T)$ as follows:

$$\frac{\sum_{t=1}^T \sum_{i \in S_t} F_1(i)}{2M} \quad (4.1)$$

for the uncapacitated examination timetabling problem and

$$\frac{\sum_{t=1}^T \sum_{i \in S_t} F_2(i)}{2} \quad (4.2)$$

for the capacitated examination timetabling problem, respectively. After the cyclic exchanges, the cost for the directed arc is defined as: cost of (*“inserted examination”*) $\cup$ (*“old cell of the ejected examination”*) $\setminus$ (*“ejected examination”*) – cost of (*“old cell of the ejected examination”*). This search algorithm does not always solve the negative cost cycle to optimality. It only finds an approximate solution. The details are discussed in Section 4.4.

4.4 Search Algorithm

Figure 4.6 illustrates the pseudo-code used in these experiments.

```

Set the initial solution Sol by employing saturation degree (see
Brelaz 1979);
Calculate initial cost function value, f(Sol);
Solbest ← Sol;
Create partition;
Define neighbourhood structures and construct the improvement
graph G;
do while (not termination criterion)
    Find a negative cost partition-disjoint cycle for G using the
    modified shortest-path label correcting algorithm to obtain
    new solution Sol*;
    Calculate the quality of a new solution Sol*, f(Sol*);
    If ((f(Sol*) ≤ f(Solbest))
        Sol ← Sol*;
        Solbest ← Sol*;
    else
        Calculate the difference between old and new solution,
        δ = f(Sol*) - f(Sol);
        Generate RandNum, a random number in [0,1];
        if (RandNum < e-δ)
            Sol ← Sol*;
        Recreate partition;
        Define neighbourhood structure and update G (see Figure 4.8);
end while;

```

Figure 4.6. The pseudo-code for the large neighbourhood search algorithm applied to the examination timetabling problem

The large neighbourhood search algorithm starts with a feasible initial solution which is generated by a saturation degree graph colouring heuristic (see Brelaz, 1979) that is known to be able to quickly generate reasonably good solutions. Then the cells are created based on timeslots. The improvement graph is constructed once. To allow a creation of paths using only the insertion/ejection moves and to increase the computation speed, the last examination is kept in its current cell (i.e. no examination is inserted from the last cell to the first cell). In the *do-while* loop, the modified shortest path label-correcting algorithm adapted from Ahuja et al. (1993) is implemented in order to find the negative cost partition-disjoint cycles for the improvement graph until the termination criterion is met (the pseudo-code of the modified shortest path label-correcting algorithm is presented in Figure 4.7 and followed by a detailed description). The termination criterion is set to 1,000,000 iterations. The modified shortest path label-correcting algorithm is run several times with a different source examination (origin node), since the success of finding a valid cycle (or path) is related to the source examination from which the search is initiated. The basic idea of the modified shortest path label-correcting algorithm is to find a shortest distance from one examination as a source examination to other examinations in the improvement graph.

Some initial tests to understand the behaviour of the modified shortest path label-correcting algorithm by using different categories of the source examination were performed while trying to find the shortest path in the improvement graph. Since the initial solution consists of examinations with timeslots assigned to them and is sorted in descending order based on the number of clashes, such solutions are sorted into three categories. The cut off points for each category are based on a percentage. The first 33.3% is considered to be a high clashes category, the second 33.3% is considered to be a medium clashes category and the last 33.3% is considered to be a low clashes category. The source examinations are chosen randomly from each category. The initial tests indicate that the source examinations that were being selected from the medium clashes category were the most suitable when compared to the other two categories because:

- i. If the source examination is taken from the high clashes group, it restricts further moves to look for negative cost partition-disjoint cycles because this examination

does not have many directed arcs (connected edges) to other examinations in the improvement graph.

- ii. If the source examination is taken from the low clashes group, the examinations with fewer clashes are selected to be moved thus yielding less impact on the objective function.

Note that the selected source examinations from the medium clashes category should have (an) out going directed arc(s) connected to other examinations in the improvement graph in order to find the shortest distance between them. For example (from Figure 4.5), assume that examinations e_2 , e_3 and e_6 are categorised under the medium clashes category. Examinations e_2 and e_3 are not selected as source examinations since they do not have an out directed arc as examination e_6 . So in this example, e_6 is selected as the source examination because it has a directed arc “going to” the other cells. Once the negative cost partition-disjoint cycles using the modified shortest path label-correcting algorithm are obtained, the quality of the new solution, $f(Sol^*)$ is recalculated, and compared with the quality of the best solution, $f(Sol_{best})$. If there is an improvement (including no change in the cost function), $f(Sol^*) \leq f(Sol_{best})$, then the new solution, Sol^* is accepted and the best solution, Sol_{best} , is set with the new solution, Sol^* . If the cost is equal (zero improvement) then the new solution is accepted. This is because the new solution might be different from the old solution even though the cost function is producing the same result (i.e. the improvement is zero). In order to escape form local optima, a worse solution is accepted using the exponential monte carlo acceptance probability (see Ayob and Kendall, 2003) which is quite similar to the acceptance criterion in a simulated annealing approach. Ayob and Kendall (2003) have shown that exponential monte carlo performs well in the application of component placement sequencing for a multi-head placement machine. In this work, the new solution Sol^* is accepted if the generated random number in $[0,1]$, $RandNum$, is less than the probability which is computed by $e^{-\delta}$ where δ is the difference between the cost of the old and new solutions (i.e. $\delta = f(Sol^*) - f(Sol)$). The exponential monte carlo will exponentially increase the acceptance probability if δ is small. Note that $e^{-\delta f(Sol)}$ is not chosen because, in this case, the very worse solution is likely to be accepted if the value of $f(Sol)$ is too large. This would then make it difficult for the search to converge. The

pseudo-code for the modified shortest path label-correcting algorithm implemented in this research is shown in Figure 4.7. The details of the algorithm can be described as follows: The shortest path label-correcting algorithm maintains a set of distance labels, $dist(\cdot)$. This algorithm begins by setting the distance label for the source examination, s , to zero, $dist(s) = 0$. We maintain a predecessor index, $pred(\cdot)$. For each examination j , a predecessor index, $pred(j)$ records an examination prior to examination j in the current directed path of length $dist(j)$. The predecessor for the source examination, s , is set to zero, $pred(s) = 0$. The distance label for other examinations, $dist(j)$ is set to ∞ . We also maintain a first-in first-out list (called FIFOLIST) of all examinations j where the directed arc (i,j) in the improvement graph violates the condition, $dist(j) > dist(i) + \text{cost of the directed arc } (i,j)$. Note that the calculation for the cost of a directed arc is discussed in Section 4.3.3. Also note that the source examination, s , is added into the FIFOLIST. If the FIFOLIST is empty, a solution is obtained. We then use the predecessor indices to trace the shortest path from an examination j back to the source examinations s . Otherwise, the first examination, i , in the FIFOLIST is removed. For each directed arc (i,j) in the improvement graph, we examine if this directed arc violates the condition mentioned above. If it does, we then update the distance label for examination j , $dist(j)$. We repeat the process until the FIFOLIST is empty.

```

dist(s) := 0 and pred(s):=0;
dist(j) := ∞ for each examination j ∈ N \ {s};
FIFOLIST := {s};
do while FIFOLIST ≠ NULL
    remove an examination i from FIFOLIST;
    for each directed arc (i,j) in G do
        if dist(j) > dist(i) + cost of directed arc (i,j) then
            dist(j) = dist(i) + cost of directed arc(i,j);
            pred(j) = i;
            if j ∉ FIFOLIST then
                add examination j to FIFOLIST;
                apply "loop detection" strategy to check if
                the distance label of examination j is
                continuing to decrease for a certain number
                of counters, then terminate;
            end if;
        end if;
    end for;
end while;

```

Figure 4.7. The pseudo-code for the modified shortest path label-correcting algorithm

This paragraph describes the adaptation of the algorithm from the standard modified shortest path label-correcting algorithm by Ahuja et al. (1993) which can only be applied to an improvement graph that does not contain any negative cycle. In the modified shortest path label-correcting algorithm (in Figure 4.7), a “loop detection” strategy is applied which can detect the presence of a negative cycle. The standard modified shortest path label-correcting algorithm will keep decreasing distance labels indefinitely and will never terminate if an improvement graph contains a negative cycle. Certainly, the “loop detection” strategy will check if the distance label of examination j continues to decrease for a certain number of counters with the same predecessor i and, if so, the computation will be terminated. Thus, the negative cycle can be obtained by tracing the predecessor indices after which the cost of the solution will be recalculated. The accepted move might be a valid path or a valid cycle that has been traced from the predecessor indices. If the cost of a valid path is less than the cost of a valid cycle, then a valid path is accepted (the best improvement in terms of lower cost will be accepted). Otherwise a valid cycle is accepted.

```
Determine the cells that are (or not) involved in cyclic (or
path) exchanges called AffectedCells (or NonAffectedCells);
Determine the number of AffectedCells called
NumberOfAffectedCells;
Keep the directed arcs that are not connected to / from the
AffectedCells called OriginalArcs;
Case 1:
    repeat
        Generate the directed arcs for every examination from
        AffectedCells to NonAffectedCells cells called
        NewArcs1;
        Calculate the costs for the NewArcs1;
    until (NumberOfAffectedCells)
Case 2:
    repeat
        Generate the directed arcs for every examination from
        NonAffectedCells to AffectedCells called NewArcs2;
        Calculate the costs for the NewArcs2;
    until T where T is the number of available timeslots
Combine OriginalArcs, NewArcs1 and NewArcs2 to form the
improvement graph;
```

Figure 4.8. The pseudo-code for updating the improvement graph

The improvement graph, G , is updated after performing cyclic (or path) exchanges. The process of updating the improvement graph can be described as follows. The cells and the number of cells that play a part in the cyclic (or path) exchanges are

called the *AffectedCells* and the *NumberOfAffectedCells*, respectively. The cells that are not involved in the cyclic (or path) exchanges are called the *NonAffectedCells*. The directed arcs in the current improvement graph that are not connected to or from the *AffectedCells* are assigned to a set called *OriginalArcs*. Then the new directed arcs are generated in two different cases. The first case is where the set of directed arcs for every examination from the *AffectedCells* to the *NonAffectedCells* are generated. This is referred to as *NewArcs1*. Then the cost for *NewArcs1* is calculated. This process in Case 1 (see Figure 4.8) is repeated for all the *AffectedCells*. In the second case (which is the opposite of the first case), the set of directed arcs for every examination from the *NonAffectedCells* to *AffectedCells* are generated, which is referred to as *NewArcs2*. The costs for the *NewArcs2* are calculated. The process in Case 2 (see Figure 4.8) is repeated for all the cells. Of course, the costs for the directed arcs (a,b) and (b,a) (see Figure 4.5 for an example) are different from each other. That is the reason why the directed arcs in two different cases are generated. The combination of *OriginalArcs*, *NewArcs1* and *NewArcs2* will update the improvement graph.

4.5 Experiments and Results

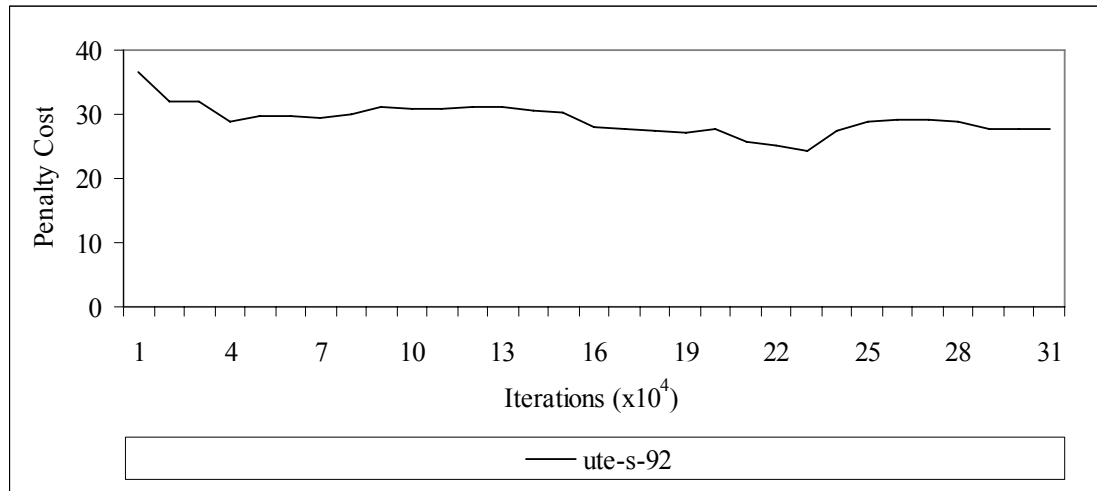
The algorithm was tested on twelve and five of the datasets for the uncapacitated and capacitated problems, respectively. The experiments were run for twelve hours for each of the datasets. Note that examination timetabling is a problem that is usually tackled several months before the schedule is required. An overnight run for examination timetabling is perfectly acceptable in a real world environment. This is a scheduling problem where the time taken to solve the problem is not critical.

4.5.1 The Uncapacitated Problem

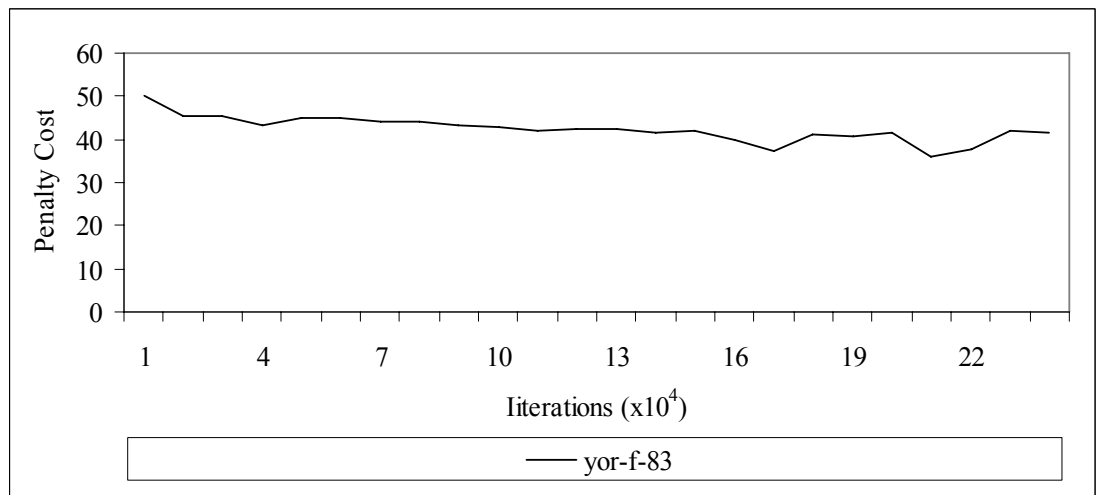
The first series of experiments that is carried out in this section considers the proximity between two examinations taken by a student as a measure of quality (once the hard constraints are satisfied). However, room capacity is not considered in this first series of experiments. The objective is to minimise the objective function (Equation (3.1)) as presented in Chapter 3. The best results from the literature and our results are presented in Table 4.3. The best results are highlighted in bold font.

We can see that our algorithm is able to obtain the best result in the literature on *yor-f-83* problem.

Figures 4.9 and 4.10 show the behaviour of the large neighbourhood search algorithm when applied to two of the datasets i.e. *ute-s-92* and *yor-f-83*.



Figures 4.9. The behaviour the of large neighbourhood search algorithm on *ute-s-92*



Figures 4.10. The behaviour of the large neighbourhood search algorithm *yor-f-83*

Table 4.3. Results on the uncapacitated problem using proximity cost

Datasets	Carter et al. (1996)	Di Gaspero and Schaerf (2001)	Caramia et al. (2001)	Burke and Newall (2003)	Merlot et al. (2003)	Kendall and Hussin (2005a)	Asmuni et al. (2005a)	White et al. (2004)	Burke et al. (2004b)	Burke et al. (2007)	Burke et al. (2006c)	Large neighbourhood search approach
<i>car-f-92</i>	6.2	5.2	6.0	4.10	4.3	4.67	4.56	4.63	4.4	5.36	4.6	4.4
<i>car-s-91</i>	7.1	6.2	6.6	4.65	5.1	5.37	5.29	5.73	4.8	4.53	4.0	5.2
<i>ear-f-83</i>	36.4	45.7	29.3	37.05	35.1	40.18	37.02	45.8	35.4	37.92	32.8	34.9
<i>hec-s-92</i>	10.8	12.4	9.2	11.54	10.6	11.86	11.78	12.9	10.8	12.25	10.0	10.3
<i>kfu-s-93</i>	14.0	18.0	13.8	13.90	13.5	15.84	15.81	17.1	13.7	15.2	13.0	13.5
<i>lse-f-91</i>	10.5	15.5	9.6	10.82	10.5	-	12.09	14.7	10.4	11.33	10.0	10.2
<i>rye-f-92</i>	7.3	-	6.8	-	8.4	-	10.35	11.6	8.9	-	-	8.7
<i>sta-f-83</i>	161.5	160.8	158.2	168.73	157.3	157.38	160.42	158	159.1	158.19	159.9	159.2
<i>tre-s-92</i>	9.6	10.0	9.4	8.35	8.4	8.39	8.67	8.94	8.3	8.92	7.9	8.4
<i>uta-s-92</i>	3.5	4.2	3.5	3.20	3.5	-	3.57	4.44	3.4	3.88	3.2	3.6
<i>ute-s-92</i>	25.8	29.0	24.4	25.83	25.1	27.60	27.78	29.0	25.7	28.01	24.8	26.0
<i>yor-f-83</i>	41.7	41.0	36.2	37.28	37.4	-	40.66	42.3	36.7	41.37	37.28	36.2

In Figures 4.9 and 4.10, the x-axis represents the number of iterations (where one iteration corresponds to an entire run of the search process) while the y-axis represents the average penalty cost per student. Every point (in the graphs) corresponds to the average penalty cost per student at a number of iterations of a separate solution. These graphs show how the large neighbourhood search algorithm explores the search space. The curve moves up and down because worse solutions are accepted with a certain probability in order to escape from local optima. The analysis of the diagrams shows that at the beginning of the search, the slope of the curves is relatively steep which indicates a high improvement in the quality of the solutions. The longer the search times, the slower the improvement of the solutions.

4.5.2 The Capacitated Problem

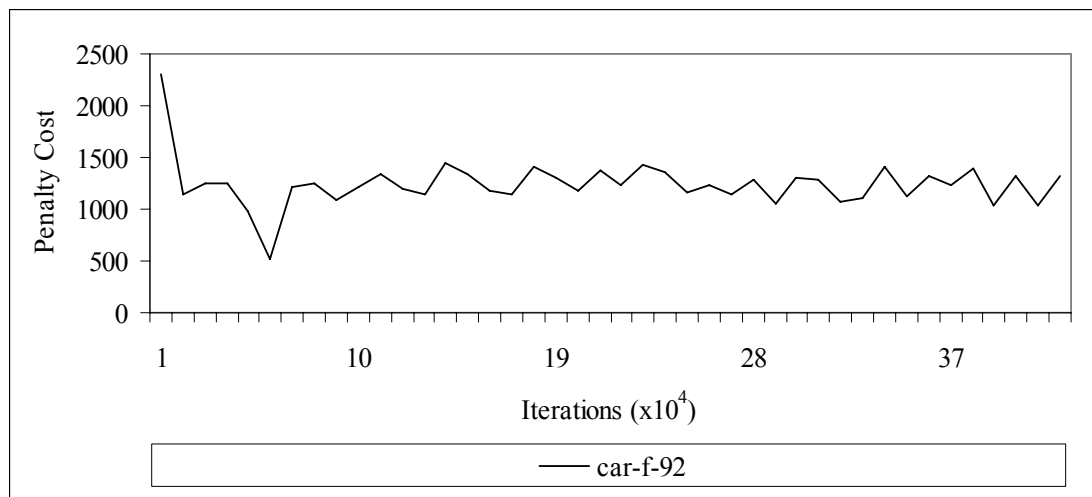
The second series of experiments is described in this section. It deals with the capacitated problem where the constraint of room capacity is taken into account in addition to the clash-free requirement. The performance of the large neighbourhood search algorithm is evaluated on five capacitated benchmarks. The computational results from the large neighbourhood search algorithm are summarised in Table 4.4 along with some other results from the literature.

Table 4.4. Results on the capacitated problem

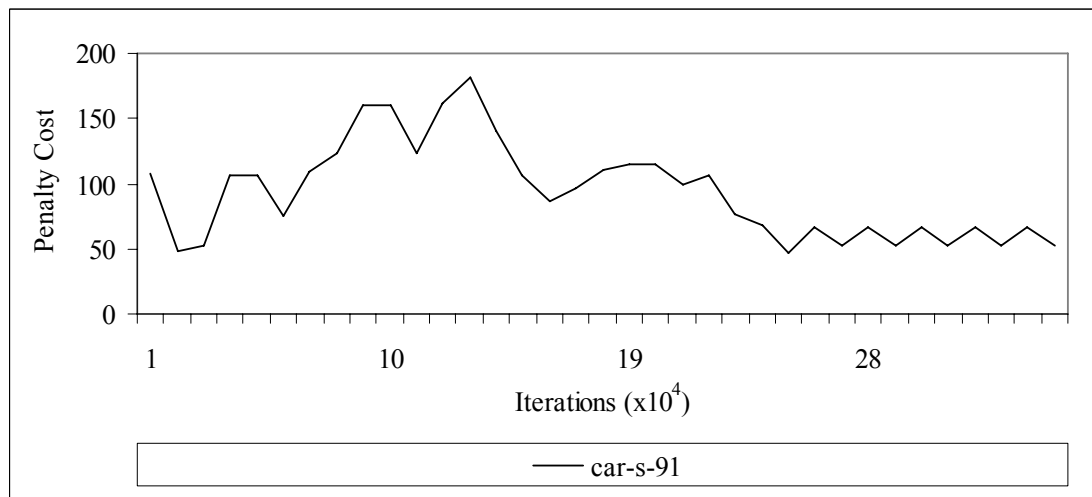
Datasets	Large neighbourhood search	Di Gaspero and Schaerf (2001)	Burke et al. (1996b)	Merlot et al. (2003)	Caramia et al. (2001)
<i>car-f-92</i>	525	424	331	158	268
<i>car-s-91</i>	47	88	81	31	74
<i>kfu-s-93</i>	206	512	974	247	912
<i>tre-s-92</i>	4	4	3	0	2
<i>uta-s-92</i>	310	554	772	334	680

Once again the best results are presented in bold. The large neighbourhood search algorithm obtained the best result for *kfu-s-93* and *uta-s-92* with 82.15% and 51.18% improvement respectively. It had 16.60% and 7.19% improvement (with respect to the next best result presented in Merlot et al., 2003). The main competitor for the capacitated problem is the algorithm of Merlot et al. (2003). In the uncapacitated problem, Merlot et al. (2003) did not obtain any best results but here it provides the

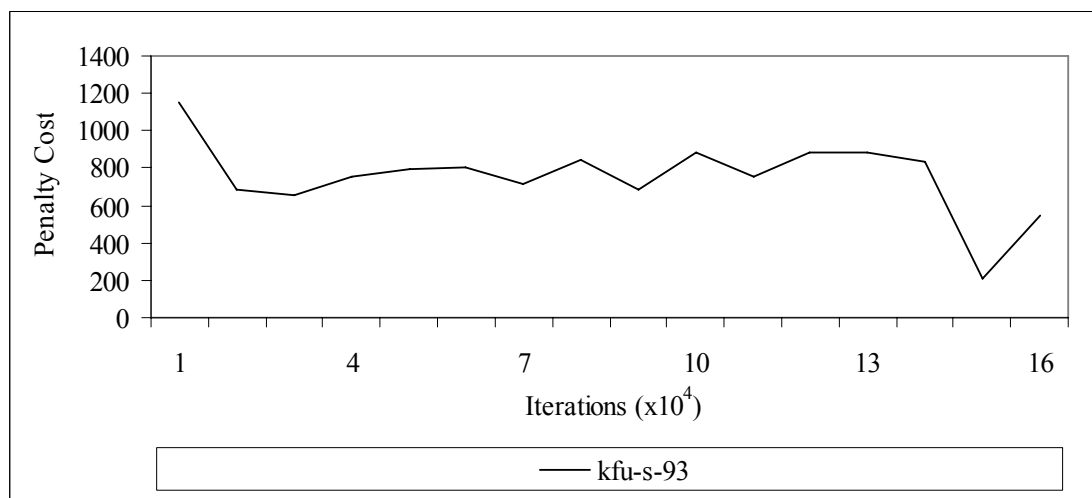
best solution in three out of five datasets. The large neighbourhood search algorithm gets the best results for two of the datasets. The results are better than the other results published in the literature except on the *car-f-92* dataset, where the large neighbourhood search algorithm performs quite poorly. Figures 4.11 to 4.15 show the behaviour of the large neighbourhood search algorithm applied to these benchmarks. A similar representation for both axes as presented in Section 4.5.1 is used. The graphs demonstrate how the large neighbourhood search algorithm explores the search space. Worse moves are accepted with a certain probability in order to allow the search to explore the search space beyond the local optima. This can be seen from the figures by noting that the curves move rapidly up and down the graph. However, accepting worse moves can lead to cycling where the move is repeated between some sets of solutions. It is believed that this causes poor results on certain instances (e.g. *car-f-92*). It is also believed that the results obtained can be related to the value of the *conflict matrix* density (see Table 3.9). The higher the value of the *conflict matrix* density, the higher the number of examinations that conflict with each other. Thus there might be fewer (and more sparsely distributed) solution points in the solution space. This is indicated from the results obtained for the *car-f-92* and *car-s-91* datasets. On the other hand, the algorithm applied to the *kfu-s-93* dataset does not get trapped in a cycle. This may be because there are more solution points in the solution space since the value of the *conflict matrix* density is lower compared to the other datasets used in this experiment. So there is evidence to suggest the need for the mechanism to escape from the cycle and to jump the barrier from one solution point to another in order to obtain a better solution. Such a mechanism is discussed in Chapter 5.



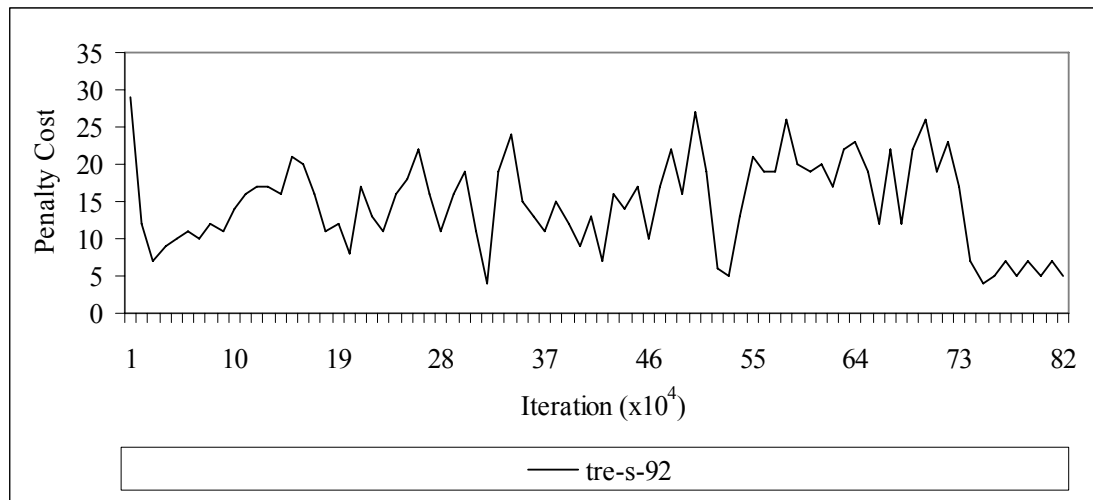
Figures 4.11. The behaviour of the large neighbourhood search algorithm on *car-f-92*



Figures 4.12. The behaviour of the large neighbourhood search algorithm on *car-s-91*



Figures 4.13. The behaviour of the large neighbourhood search algorithm on *kfu-s-93*



Figures 4.14. The behaviour of the large neighbourhood search algorithm on *tre-s-92*

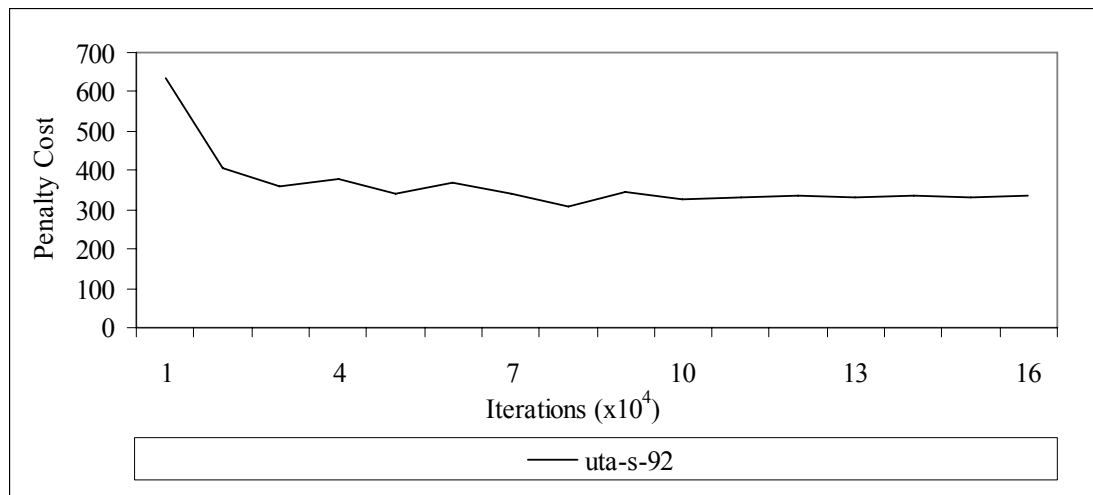


Figure 4.15. The behaviour of the large neighbourhood search algorithm on *uta-s-92*

4.6 Summary

In this chapter, an improvement procedure is modelled for the examination timetabling problem as a variant of a very large-scale neighbourhood search structure through a cyclic exchange operation. This chapter investigated the search ideas proposed by Ahuja et al. (2001a) for the capacitated minimum cost spanning tree problem. This is a new procedure in the timetabling arena and it represents an approach that outperforms the current state of the art on several benchmark problems.

It is demonstrated that the solution quality is dependent on the neighbourhood structure and the search approach. The key feature of the approach in this chapter is the combination of a very large neighbourhood structure (based on the concept of an improvement graph) and the technique of identifying improvement moves by solving negative cost partition-disjoint cycles (or paths) using a modified shortest path label-correcting algorithm. It is also shown that a cyclic exchange operation is far superior to simply employing two exchanges when defining a neighbourhood structure for examination timetabling. However, the drawback of the system is that a relatively long running time is needed to identify the improvement moves in the improvement graph that is proportional to the number of examinations. Note that in real world situations, examination timetabling is an off line problem, and the processing time is not very critical. The next chapter presents a hybridisation technique between the large neighbourhood search approach and tabu search and is applied to the capacitated examination timetabling problem. The work presented in this chapter was published in (Abdullah et al. 2006a).

Chapter 5

A Tabu-based Large Neighbourhood Search for the Capacitated Examination Timetabling Problem

5.1 Introduction

As we have seen, large neighbourhood search algorithms can be effective for solving timetabling problems. This chapter presents a tabu-based large neighbourhood search methodology in which the improvement moves are kept in a tabu list for a certain number of iterations. Note that the description of the large neighbourhood search was discussed in Chapter 4. This chapter has drawn upon Ahuja-Orlin's methodology and incorporated it with tabu lists to develop an effective examination timetabling solution scheme which is evaluated on capacitated examination problem benchmark datasets from the literature. The objective is to see how the hybridisation of a large neighbourhood algorithm, a tabu list and adaptive memory can enhance the solution quality in capacitated examination timetabling. The approach is compared against other methodologies that have appeared in the literature over recent years. The computational experiments indicate that the approach described here produces the best known results on a number of these benchmark problems. The work presented in this chapter is accepted for publication in the Journal of the Operational Research Society (Abdullah et al. 2006b).

This chapter is organised as follows: The solution search strategy that was employed is presented in Section 5.2. Experimental results are evaluated and discussed in Section 5.3 followed by some concluding remarks in Section 5.4.

5.2 A Tabu-based Large Neighbourhood Search

Figure 5.1 illustrates the pseudo-code that represents the approach used for the work in this chapter. The algorithm starts with a feasible initial solution, Sol . Cells are created based on the timeslots. The cyclic exchange neighbourhood structures are defined and the improvement graph is constructed once. The modified shortest path label-correcting algorithm (Ahuja et al. 1993) is repeatedly implemented in the *do-while* loop using different source examinations to find the negative cost partition-disjoint cycle. Note that the details about the cyclic exchange neighbourhood structures, an improvement graph and a modified shortest path label-correcting algorithm have been discussed in Chapter 4. Once the negative cost partition-disjoint cycles are found, the quality of the new solution, $f(Sol^*)$ is calculated and then compared with the quality of the best solution, $f(Sol_{best})$. The new solution, Sol^* , is accepted if there is an improvement in the solution quality. A worse solution is accepted with a certain probability using the exponential monte carlo acceptance criterion (as discussed in Chapter 4). The examination(s) that are involved in the cycle (or path) exchange is (are) added to the tabu list. If the solution is rejected by the exponential monte carlo acceptance criterion for a certain number of iterations which is equal to the *not_improving_constant* (in the pseudo-code in Figure 5.1), then the algorithm will terminate.

```

Obtain a feasible initial solution Sol;
Calculate initial cost function value, f(Sol);
Solbest ← Sol;
Create partition;
Define neighbourhood structures and construct the improvement
graph, G;
Set not improving counter ← 0;
do while (not termination criterion)
    Find negative cost partition-disjoint graph cycles for G
    using the modified shortest path label-correcting algorithm
    to obtain new solution Sol*;
    Calculate the quality of a new solution f(Sol*);
    if (f(Sol*) ≤ f(Solbest))
        Sol ← Sol*;
        Solbest ← Sol*;
        not improving counter ← 0;
    else
        Calculate the difference between old and new solution,
        δ = f(Sol*) - f(Sol);
        Generate RandNum, a random number in [0,1];
        if (RandNum < e-δ)
            Sol ← Sol*;
            not improving counter ← 0;
        else
            increase not improving counter by 1;
            if not improving counter == not_improving_constant
                exit
        Add profitable or worse moves (i.e. examinations that
        involved in cycle (or path) exchanges) to the tabu list;
        Identify the cell for each examination that involved in a
        cycle (or path) exchange, called as AffectedCells;
        Identify the cells that are not involved in the cyclic
        (path) exchanges, called as NonAffectedCells;
        Calculate the number of AffectedCells, called
        NumberOfAffectedCells;
        Recreate partition from updated solution;
        Define neighbourhood structure and update G concerning tabu
        list (see Figure 5.2);
    end while;

```

Figure 5.1. The pseudo-code for the tabu-based large neighbourhood search applied to the examination timetabling problem

This part of the search strategy is similar to the one used in Chapter 4. The tabu-based large neighbourhood search uses only short term memory. The examinations that have been involved in a cycle (or path) exchange are added to the tabu list (as our tabu restriction), denoted as $TabuList_i$ ($i \in \{1, \dots, N\}$), and are not allowed to be part of any exchange for a certain number of iterations (which is equal to the tabu tenure, TT) so that the algorithm can look at the possibility of other examinations to

be considered in performing a cyclic exchange and updating an improvement graph that may drive the search into a new region and increase the likelihood of finding better profitable exchanges with respect to the objective function. In this work, the tabu-based large neighbourhood search does not apply any aspiration criterion since the objective here is to determine which tabu tenure produces the best solutions for this problem. The tabu tenure is set whenever a tabu restriction is satisfied and it is decreased after each iteration until it reaches zero. All tabu examinations will change to non tabu status when the tabu tenure is zero. In these experiments, the tabu tenure is set to be 2, 4 and 6. The determination of these values was based upon initial experiments.

The improvement graph is updated after the cyclic (or path) exchange is performed. The cells which play a part in the process (referred to as the *AffectedCells*) can be determined by consulting the cyclic (or path) exchanges. Note that *NonAffectedCells* refers to the cells that are not implied in the cyclic (or path) exchanges. The cost for all the directed arcs that is not connected to or from the *AffectedCells* remains unchanged. These arcs are referred to as the *OriginalArcs*. This enables the algorithm to save the time taken to create and calculate the cost for *OriginalArcs*. Then the new directed arcs, referred to as *TNewDirArcs1* in case 1, are created (see Figure 5.2) by inserting an examination (from the *AffectedCells*) to the *NonAffectedCells* and ejecting one examination (the ejected examination should be different from the examinations in the *TabuList*). Calculate the cost for the new directed arcs. The other new directed arcs in case 2 are referred to as *TNewDirArcs2* (see Figure 5.2) and are created like the *TNewDirArcs1* in case 1, but in the reverse way i.e. an examination (which is not in the *TabuList*) is inserted from *NonAffectedCells* to the *AffectedCells* and at the same time we eject another examination from *AffectedCells* and this ejected examination should not be the one in the *TabuList*. Then the cost for the *TNewDirArcs2* is calculated. The combination of the *OriginalArcs*, *TNewDirArcs1* and *TNewDirArc2* will form a new improvement graph *G*. Figure 5.2 shows the pseudo-code for updating the improvement graph in this work.

```

Determine the cells that are (or not) involved in cyclic (or
path) exchanges called AffectedCells (or NonAffectedCells);
Determine the number of AffectedCells called
NumberOfAffectedCells;
Keep the directed arcs that are not connected to / from the
AffectedCells called OriginalArcs;
Case 1:
  repeat
    Generate the directed arcs for every pair of
    examinations from AffectedCells to NonAffectedCells
    iff any of these pair of examinations are not in the
    TabuList called TNewDirArcs1;
    Calculate the costs for the TNewDirArcs1;
  until (NumberOfAffectedCells)

Case 2:
  repeat
    Generate the directed arcs for every pair of
    examinations from NonAffectedCells to AffectedCells
    iff any of these pair of examinations are not in the
    TabuList called TNewDirArcs2 ;
    Calculate the costs for the NewArcs2;
  until T
Combine OriginalArcs, TNewDirArcs1 and TNewDirArcs2 to form a
new G;

```

Figure 5.2. The pseudo-code for updating the improvement graph in the tabu-based large neighbourhood search approach

5.3 Experiments and Results

The algorithm was tested on standard benchmark problems from the capacitated problem. We ran the experiment overnight (which takes approximately twelve hours) for each of the datasets.

The hybridised tabu search large neighbourhood method described in this chapter was run on six datasets. For each dataset, experiments were carried out with tabu tenures of 2, 4 and 6. Table 5.1 shows the computational results of this algorithm compared to other published results for these benchmark datasets. The best results are shown in bold.

Table 5.1. Results on the capacitated problem using tabu-based large neighbourhood search algorithm

Datasets	Tabu-based large neighbourhood search approach			Abdullah et al. (2006a) TT=0	Merlot et al. (2003)	Di Gaspero and Schaerf (2001)	Caramia et al. (2001)	Burke et al. (1996b)
	TT=2	TT=4	TT=6					
<i>car-f-92</i>	278	284	314	525	158	242	268	331
<i>car-s-91</i>	37	48	72	47	31	88	74	81
<i>kfu-s-93</i>	548	616	569	206	247	512	912	974
<i>tre-s-92</i>	0	0	1	4	0	4	2	3
<i>uta-s-92</i>	300	300	346	310	334	554	680	772
<i>nott-94</i>	18	21	32	-	2	11	44	53

The objective here is to demonstrate that the tabu-based large neighbourhood search methodology is able to produce good results for capacitated examination timetabling problems. The method produces the best known result in the literature on two of the six problems (and it ties with Merlot et al. 2003 on *tre-s-92*).

We are particularly interested to compare our results with the results in Chapter 4 (Abdullah et al. 2006a). That method did not employ a tabu list (i.e. the tabu tenure is set to zero). Since the tabu search based method is superior in four out of five problems, there is evidence to suggest that a tabu list could be used to guide the search process. In the tabu-based large neighbourhood search, a tabu restriction is applied where examinations will be tabu if examinations in cyclic exchanges have been accepted to update the current solution. The examination(s) will remain tabu for a number of iterations which is equal to the tabu tenure. The examinations in a cyclic exchange are made tabu with the aim of directing the search to other parts of the search space (in the improvement graph). The best results for four of the datasets (this work does not consider the *nott-94* dataset in this comparison) is when the value of TT is 2 (ties when TT is 4) and for the other dataset it is when TT is zero (from Abdullah et al. 2006a). It is interesting to note that for the *kfu-s-93* dataset, the best result is obtained when the value of TT is zero and that this problem instance has the lowest *conflict matrix* density (0.06). The lower *conflict matrix* density signifies that fewer examinations are conflicting with each other. This implies that this problem might have more feasible solution points in our search space. So, a tabu list, possibly,

is not needed to direct the search to other parts of the search space in the improvement graph. The higher the value of TT, the longer the examinations will remain in the tabulist. This limits the search space. It is noticeable that a higher value of TT could make the solution considerably worse and thus more difficult to improve.

Figures 5.3 and 5.4 show the performance of the algorithm using different values of TT on two datasets (*tre-s-92* and *kfu-s-93*).

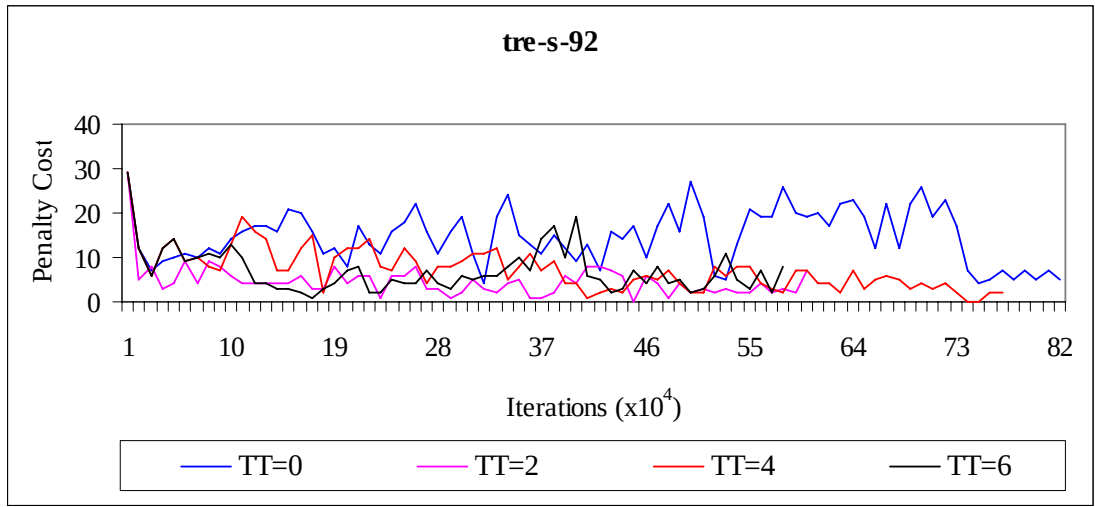


Figure 5.3. The performance of the tabu-based large neighbourhood search algorithm with different TT on the *tre-s-92* dataset

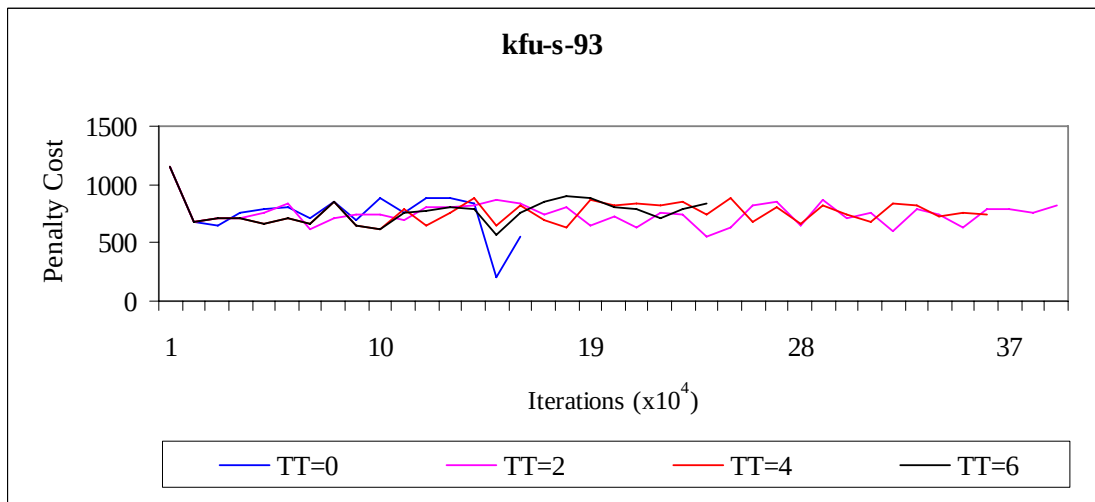


Figure 5.4. The performance of the tabu-based large neighbourhood search algorithm with different TT on the *kfu-s-93* dataset

This graph demonstrates how our algorithm explores the search space in the improvement graph. The x-axis represents the number of iterations while the y-axis represents the average penalty cost per student. The curves move up and down because at every iteration, the best improving solution (or worse solution with some probability) is accepted. In Abdullah et al. (2006a), the solutions are trapped in a cycle. The use of the tabu list in this algorithm helps to avoid cycling during the search process. To show the effect of using the tabu-based large neighbourhood search in avoiding cycling in the search process, the graph (Figure 5.5) of the solution for the *car-f-92* dataset and the solution taken from Abdullah et al. (2006a), in which we consider the value of TT to be zero, are presented. The tabu-based large neighbourhood search can find solutions (in most cases) with better local optima.

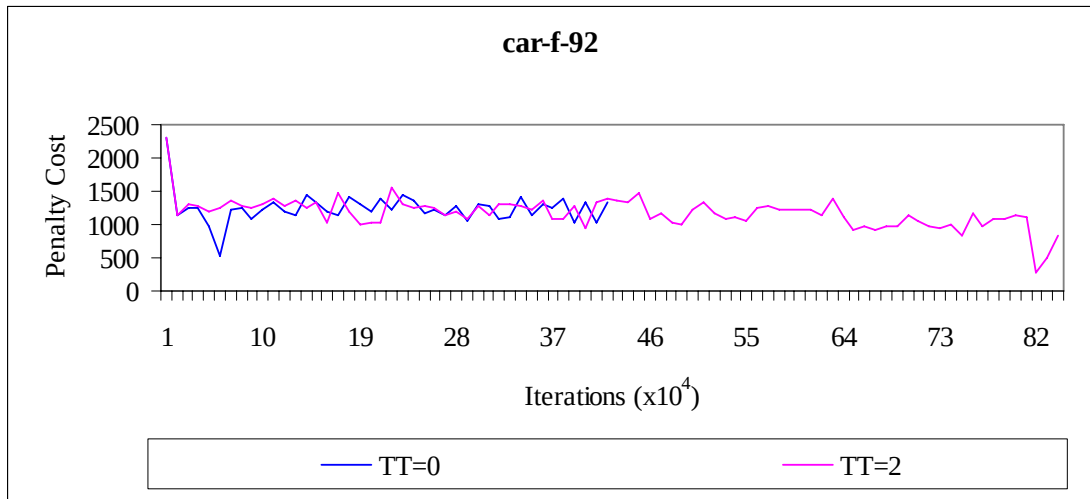


Figure 5.5. The behaviour of *car-f-92* dataset using tabu-based large neighbourhood search algorithm

5.4 Summary

The hybrid tabu-based large neighbourhood search presented in this chapter uses a tabu list to direct the search to other parts of the improvement graph and to avoid cycling during the search process. The experiments carried out showed that the algorithm performed competitively with the best published results in the literature. Indeed, it is able to provide the best known results on two of the six standard benchmark problems in this area. This shows that the hybridisation of the meta-heuristic and network flow optimisation technique is a very promising technique for

tackling the capacitated examination timetabling problem. Since the neighbourhood structure generated through the cyclic exchanges operation highlighted in this paper increases exponentially with the size of the input, we can easily see why the drawback of the algorithm is the long time needed to find a valid cycle in the improvement graph. The next chapter will discuss the implementation of a multi-start two phase approach where the large neighbourhood search approach is used in the first phase and is followed by the implementation of local search in the second phase.

Chapter 6

A Multi-start Large Neighbourhood Search Approach with Local Search Methods for the Examination Timetabling Problem

6.1 Introduction

Many of the other successful approaches in the literature represent hybridisations that often involve local search. See, for example, Burke and Newall (2003) and Merlot et al. (2003). These observations led us to explore the hybridisation of the very large neighbourhood search approach with local search methods and to propose a “multi-start” very large neighbourhood search in the first phase followed by local search methods in the second phase. The aim is to generate solutions that work well across all instances. The objectives of the chapter are to:

- Illustrate how local search can make further improvements to the solution obtained in the earlier phase.
- Compare the quality of the results obtained by a hybridisation of two different techniques (large neighbourhood search approach with a great deluge and with a simulated annealing algorithm).
- Compare the quality of the results produced from this work with other published results that used the same objective function on standard benchmark problems.
- Demonstrate that the hybridisation of the large neighbourhood search approach and the local search method represents an effective new technique for tackling examination timetabling problems.

The work presented in this chapter appears in the proceedings of the International Conference on Automated Planning and Scheduling (ICAPS-06) (Abdullah and Burke, 2006).

This chapter is organised as follows. Section 6.2 gives a brief explanation about intensification and diversification strategies. Section 6.3 discusses the two phase model of the problem. Two separate methods are discussed: a great deluge and a simulated annealing algorithm. The description of the integration of these two phases and its pseudo-code is also presented in this section. The experimental results are presented in Section 6.4. Section 6.5 concludes with a summary of the chapter.

6.2 Intensification and Diversification Strategies

The motivation for the investigation of diversification strategies is to effectively explore the solution space defined by the problem. It is particularly appropriate when better solutions can be obtained only by crossing barriers in the solution space. Various diversification strategies have been proposed in the literature. In genetic algorithms the diversifying effect can be provided by using randomisation in combining populations through crossover and mutation operators (Reeves 1995). In tabu search, the exploration process is prevented from executing the same recent moves (cycling) (Reeves 1995). Strategic oscillation is a mechanism to diversify the search by periodically changing the weights of the cost function (Costa 1994, Schaerf 1999a). White and Xie (2001) proposed a tabu relaxation i.e. reinitialising the tabu list after a number of non-improving iterations. Higgins (2001) proposed another way to diversify the search by replacing the current solution with the best solution if there is no improvement in the cost function after a certain number of iterations.

Liaw (2003) adopted an intensification strategy consisting of storing the elite solutions and restarting the search from these when applied to a two-machine preemptive open shop scheduling problem. He also applied a frequency-based memory to achieve global diversification for the same problem. In order to improve the efficiency of tabu search algorithm with short-term memory applied to a multi-agent system for integrated dynamic scheduling of steel production, Ouelhadj (2003) implemented an intensification strategy in which a tabu list was cleared and the

search was restarted from an elite solution if there was no improvement for a certain number of iterations. Bilge et al. (2004) required three parameters to define the diversification strategy in a parallel machine problem. The first parameter defines when to start the diversification phase, which is expressed in the number of non-improving iterations, the second parameter defines the length of the diversification phase and the third parameter is the tenure multiplier.

Bilge et al. (2004) described a diversification strategy using a very large tabu tenure (multiply the current tabu tenure value by a tenure multiplier after a pre-specified number of non-improving iterations). After that the short-term tabu search is restarted with the original tabu tenure, followed by an intensification strategy that restarts and carries out a search with elite solutions. These diversification and intensification strategies are applied to a tabu search algorithm for the parallel machine total tardiness problem. For more details, refer to Bilge et al. (2004).

The algorithm described and implemented in this chapter draws upon the diversification strategy proposed by Higgins (2001), where the current solution is replaced with the best solution if the number of non-improving iterations is equal to a specific length.

6.3 A “Multi-start Two-phase” Approach

The pseudo-code that is used in these experiments is presented in Figure 6.1. This approach starts with a feasible solution and only accepts moves that will maintain a feasible timetable. Any other moves are not considered. The overall aim in this approach is to minimise the objective function and to reduce the time taken to construct the improvement graph and to identify the improving moves in the improvement graph.

```

Set initial solution, Sol, by employing the saturation degree
heuristic by Brelaz (1979);
Calculate initial cost function value, f(Sol);
Solbest ← Sol;
Create partition (as discussed in Chapter 4);
Define not_improving_length;
Set number of iterations, NumOfIte;
Set not_improving_counter ← 0;
Set iteration = 0;
Select q where 1 < q ≤ T cells at random to be used in defining
the neighbourhood, where T is the maximum number of timeslots;
do while (iteration < NumOfIte)
    Define neighbourhood and construct the improvement graph, G;
    Find a negative cost partition-disjoint cycle for G using a
    modified shortest path label-correcting algorithm to obtain a
    new solution Sol*;
    Evaluate new solution f(Sol*);
    if (f(Sol*) ≤ f(Solbest))
        Sol ← Sol*;
        Solbest ← Sol*;
        Set not_improving_counter ← 0;
    else
        δ = f(Sol*) - f(Sol);
        Generate RandNum, a random number in [0,1];
        if (RandNum < e-δ)
            Sol ← Sol*;
            Recreate partition;
            Set not_improving_counter ← 0;
        else
            Increase not_improving_counter by 1;
            if (not_improving_counter == not_improving_length)
                Apply diversification strategy by replacing Sol ←
                Solbest;
                Recreate partition;
                Select q cells at random;
            Increase iteration by 1;
    end while
Run great deluge and simulated annealing algorithms, each seeded
with Solbest as an initial solution;

```

Figure 6.1. The pseudo-code for the “multi-start” approach

A “multi-start” approach is employed i.e. it reinitialises the current solution with the best solution in hand if the current solution is not improving for a certain number of iterations which is called *not_improving_length*. This “multi-start” approach represents a diversification strategy. For each iteration in the *do-while* loop, the

improvement graph is constructed after the neighbourhood structure is created. The modified shortest path label-correcting algorithm, as discussed in Chapter 4, is implemented to find the negative cost partition-disjoint cycles for the improvement graph which implies a profitable exchange if the difference between the old solution and the current solution is negative or zero.

The explanation has been discussed in Chapter 4 but it is rephrased here to present a clear understanding of the pseudo-code presented in Figure 6.1. The modified shortest path label-correcting algorithm is run several times with a different source examination. Improving solutions are always accepted. In order to escape from local optima, this algorithm will also accept a worse move with a probability from the exponential monte carlo acceptance criterion (as discussed in Chapter 4). Once the move (improvement or worse move) is accepted, the current solution is updated. The new partition is recreated from the updated solution. The sample of cells from the previous iteration is used in the next iteration if the new solution is accepted. Otherwise, a small sample of cells is reselected. The process continues until the end of the *do-while* loop.

There are two versions of this algorithm. The first one employs the great deluge algorithm in the second phase and the other one employs simulated annealing. The pseudo-code for the implementation of the great deluge algorithm is presented in Figure 6.2. We define a number of iterations as *NumOfIte* and an estimated quality of the final solution as *estimatedquality*, a decreasing rate, β (which is calculated using the Formula (6.1), adapted from Burke et al. 2004b):

$$\beta = (f(Sol) - f(estimatedquality)) / (NumOfIte) \quad (6.1)$$

The *level* (see the explanation in Section 2.7.1.6) is equal to the initial solution, $f(Sol)$ at the start and will decrease by the value β . In the *do-while* loop, a neighbour is defined by randomly selecting an examination and assigning it to a valid timeslot. The cost function value of the new neighbour is calculated using the formula as defined in Expression (3.1) in Chapter 3. A worse solution is accepted if the cost function value of the new solution, $f(Sol^*)$ is lower than the *level*. The current

solution is updated and the process continues until the number of iterations increases to the maximum number of iterations, *NumOfIte*, or until there is no improvement for a certain number of iterations, referred to as *not_improving_length_GDA* (in the pseudo-code in Figure 6.2).

```

Set initial solution as Solbest taken from large neighbourhood
search approach (Figure 6.1), Sol;
Calculate the initial cost function value, f(Sol);
Set best solution, Solbest ← Sol;
Set estimated quality of final solution, estimatedquality;
Set number of iterations, NumOfIte;
Set initial level: level ← f(Sol);
Set decreasing rate  $\beta = ((f(Sol) - \text{estimatedquality}) / (\text{NumOfIte}))$ ;
Set iteration ← 0;
Set not_improving_counter ← 0;
do while (iteration < NumOfIte)
    Define neighbourhood of Sol by randomly assigning
    examination to a valid timeslot to generate a new solution
    called Sol*;
    Calculate f(Sol*);
    if (f(Sol*) < f(Solbest))
        Sol ← Sol*;
        Solbest ← Sol*;
        not_improving_counter ← 0;
    else
        if (f(Sol*) ≤ level)
            Sol ← Sol*;
            not_improving_counter ← 0;
        else
            Increase not_improving_counter by 1;
            if (not_improving_counter == not_improving_
                length_GDA)
                exit;
        level = level -  $\beta$ ;
        Increase iteration by 1;
    end do;

```

Figure 6.2. The pseudo-code for the great deluge algorithm applied to the examination timetabling problem

The simulated annealing algorithm applied in this chapter is presented in Figure 6.3. The same parameters as those employed in Burke et al. (2004b) are used where the initial temperature T_0 is equal to 5000, the final temperature T_f is equal to 0.05 and the number of iterations, *NumOfIte* is set to be 10,000,000. At the beginning of the

search, $Temp$ is set to be T_0 . At every iteration, $Temp$ is decreased by α where α is defined as:

$$\alpha = (\log(T_0) - \log(T_f)) / NumOfIte \quad (6.2)$$

In the *do-while* loop, a neighbour is defined by randomly selecting an examination and assigning it to a randomly selected valid timeslot. A worse candidate solution is accepted if the randomly generated number is less than $e^{-\delta/T}$, where $\delta = f(Sol^*) - f(Sol)$. Then the current solution is updated. The process continues until the temperature t is less than the final temperature T_f .

```

Set initial solution as Solbest taken from large neighbourhood
search approach (Figure 6.1), Sol;
Calculate the initial cost function value, f(Sol);
Set best solution, Solbest ← Sol;
Set number of iterations, NumOfIte;
Set initial temperature T0;
Set final temperature Tf;
Set decreasing temperature rate as  $\alpha$  where  $\alpha = (\log(T_0) - \log(T_f)) / NumOfIte$ ;
Set Temp ← T0;
do while (Temp > Tf)
    Define neighbour of Sol by randomly assigning examination
    to a valid timeslot to generate a new solution called Sol*;
    Calculate f(Sol*);
    if (f(Sol*) < f(Solbest))
        Sol ← Sol*;
        Solbest ← Sol*;
    else
        Generate a random number called RandomNumber;
        if (RandomNumber ≤ e-δ/Temp) where δ = f(Sol*) - f(Sol)
            Sol ← Sol*;
        Temp = Temp - Temp * α;
end while;

```

Figure 6.3. The pseudo-code for the simulated annealing algorithm applied to the examination timetabling problem

6.4 Experiments and Results

The algorithm was tested on eleven benchmark datasets and was run for 500,000 and 10,000,000 iterations using a large neighbourhood search approach and the great deluge (and simulated annealing) algorithm, taking approximately five hours for each dataset. The results from these experiments are presented in Table 6.1. The entry in Table 6.1 in Phase 1 represents the results obtained from the large neighbourhood search. In Phase 2, the best solution obtained from Phase 1 is fed as the initial solution to the great deluge and simulated annealing algorithms. The aims of this step are: (i) to start with a good initial solution and (ii) to diversify the search from the search space that has been explored by the large neighbourhood search algorithm in Phase 1 to another part of the search space. Phase 2 shows the improved results when the large neighbourhood algorithm is applied (with respect to the initial solution) and the great deluge (GD) and simulated annealing algorithm (SA) with respect to the best solution obtained from Phase 1.

Table 6.1 shows the percentage improvement in phases 1 and 2. In phase 1, the percentage improvement is more than 10% except for the *sta-f-83* dataset. In phase 2, the great deluge and simulated annealing algorithms are able to improve the solution by more than 10% for seven and five datasets respectively. In Table 6.1, the great deluge algorithm performs better on eight of the datasets (presented in bold) while the simulated annealing method is better in three of the datasets. Based on these results, it is believed that the good quality initial solution which is fed to the simulated annealing makes the search more restricted except when the particular problem has many solutions in the search space. Experimental results show that the great deluge method outperforms the simulated annealing algorithm. This may be because of unsuitable values of the initial and final temperatures. Also, it may be due to the strength of the boundary penalty used in the great deluge algorithm.

Table 6.1. Improvement results at phase 1 and 2

Datasets	Initial solution	Large neighbourhood search (Phase 1)	Phase 1 improvement (%)	Local search (Phase 2)		Phase 2 improvement (%)	
				GD	SA	GD	SA
<i>car-f-92</i>	7.5	4.8	36.0	4.4	4.1	8.3	14.6
<i>car-s-91</i>	8.5	5.4	36.5	4.8	4.9	11.1	9.3
<i>ear-f-83</i>	52.8	40.0	24.2	36.0	36.6	10.0	8.5
<i>hec-s-92</i>	16.5	12.1	26.7	10.8	11.0	10.7	9.1
<i>kfu-s-93</i>	24.6	17.1	30.5	16.7	15.2	2.3	11.1
<i>lse-f-91</i>	19.2	13.6	29.2	13.0	11.9	4.4	12.5
<i>sta-f-83</i>	175.2	159.2	9.1	159.0	159.2	0.1	0
<i>tre-s-92</i>	12.3	9.9	19.5	8.5	8.9	14.1	10.1
<i>uta-s-92</i>	5.6	4.1	26.8	3.6	3.8	12.2	7.3
<i>ute-s-92</i>	36.6	29.6	19.1	26.0	28.3	12.2	4.4
<i>yor-f-83</i>	50.1	40.9	18.4	36.2	36.5	11.5	10.8

Table 6.2 shows the comparison of the final results obtained from this algorithm with other published results. The best results are presented in bold. It is interesting to compare the results obtained from this approach against our previous results in Chapter 4 (as in Abdullah et al. 2006a). Abdullah et al. (2006a) outperforms the “multi-start” method even though the same very large neighbourhood search approach is employed. We believed that this is the case because the “multi-start” approach limits the size of the improvement graph and this may limit the search towards a certain region of the search space. However, it reduces the time taken when compared to the method of Abdullah et al. (2006a) and is still able to obtain better results on three out of the eleven datasets (there are also ties on three datasets). The results reported in Table 6.2 show that the “multi-start” approach is able to obtain one best known result on the *yor-f-83* dataset (ties with Caramia et al. 2001 and Abdullah et al. 2006a). On the whole, the “multi-start” approach works reasonably well across all problem instances and is competitive with the published approaches. It does not produce the worst solution on a broad range of problems which illustrates the robustness of the algorithm.

Table 6.2. Comparison of results using the “multi-start two phase” approach applied to the uncapacitated examination timetabling problem using proximity cost

Datasets	Carter et al. (1996)	Di Gaspero and Schaerf (2001)	Caramia et al. (2001)	Burke and Newall (2003)	Merlot et al. (2003)	Kendall and Hussin (2005a)	Asmuni et al. (2005a)	White et al. (2004)	Burke et al. (2004b)	Burke et al. (2007)	Burke et al. (2006c)	Abdullah et al. (2006a)	Multi-start two phase approach
<i>car-f-92</i>	6.2	5.2	6.0	4.10	4.3	4.67	4.56	4.63	4.4	5.36	4.6	4.4	4.1
<i>car-s-91</i>	7.1	6.2	6.6	4.65	5.1	5.37	5.29	5.73	4.8	4.53	4.0	5.2	4.8
<i>ear-f-83</i>	36.4	45.7	29.3	37.05	35.1	40.18	37.02	45.8	35.4	37.92	32.8	34.9	36.0
<i>hec-s-92</i>	10.8	12.4	9.2	11.54	10.6	11.86	11.78	12.9	10.8	12.25	10.0	10.3	10.8
<i>kfu-s-93</i>	14.0	18.0	13.8	13.90	13.5	15.84	15.81	17.1	13.7	15.2	13.0	13.5	15.2
<i>lse-f-91</i>	10.5	15.5	9.6	10.82	10.5	-	12.09	14.7	10.4	11.33	10.0	10.2	11.9
<i>rye-f-92</i>	7.3	-	6.8	-	8.4	-	10.35	11.6	8.9	-	-	8.7	-
<i>sta-f-83</i>	161.5	160.8	158.2	168.73	157.3	157.38	160.42	158	159.1	158.19	159.9	159.2	159.0
<i>tre-s-92</i>	9.6	10.0	9.4	8.35	8.4	8.39	8.67	8.94	8.3	8.92	7.9	8.4	8.5
<i>uta-s-92</i>	3.5	4.2	3.5	3.20	3.5	-	3.57	4.44	3.4	3.88	3.2	3.6	3.6
<i>ute-s-92</i>	25.8	29.0	24.4	25.83	25.1	27.60	27.78	29.0	25.7	28.01	24.8	26.0	26.0
<i>yor-f-83</i>	41.7	41.0	36.2	37.28	37.4	-	40.66	42.3	36.7	41.37	37.28	36.2	36.2

Figures 6.4 - 6.14 show the resulting diagrams when employing a “multi-start” approach to examination timetabling problems.

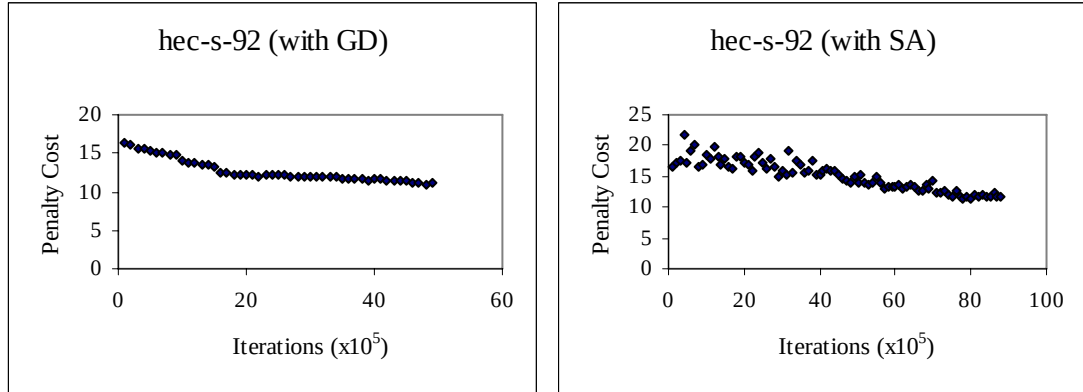


Figure 6.4. The behaviour of the “multi-start” large neighbourhood search algorithm on the *hec-s-92* dataset

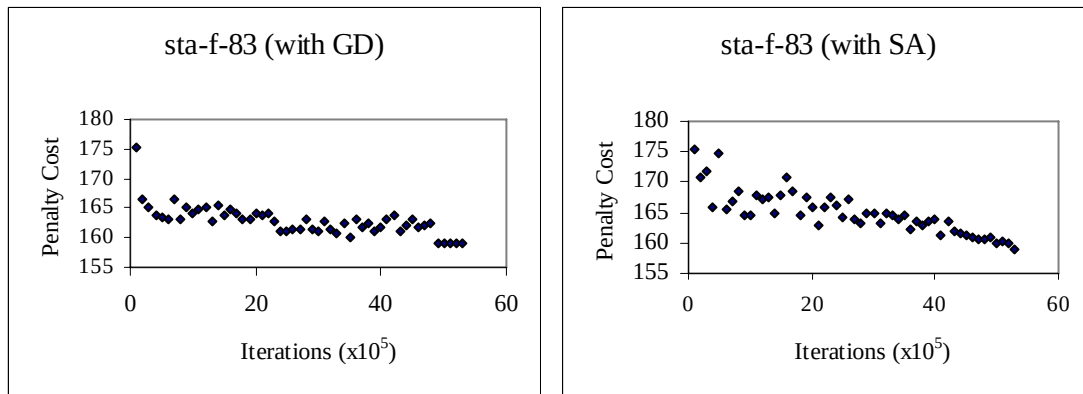


Figure 6.5. The behaviour of the “multi-start” large neighbourhood search algorithm on the *sta-f-83* dataset

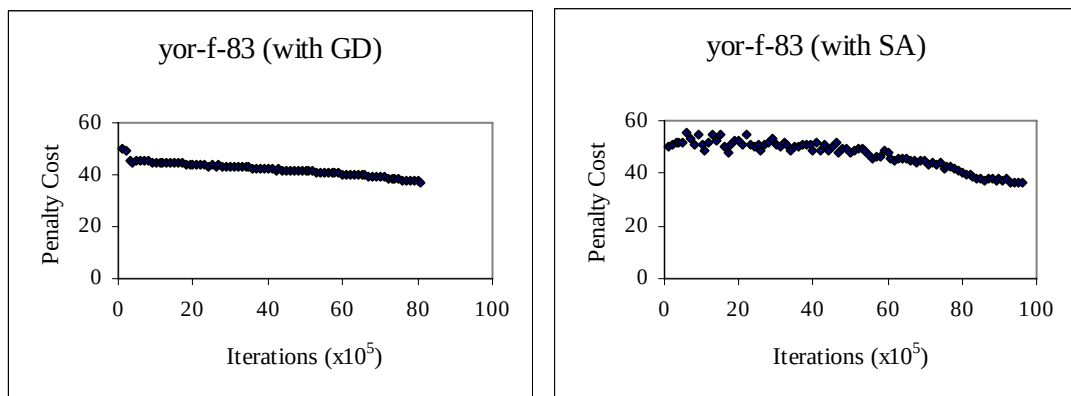


Figure 6.6. The behaviour of the “multi-start” large neighbourhood search algorithm on the *yor-f-83* dataset

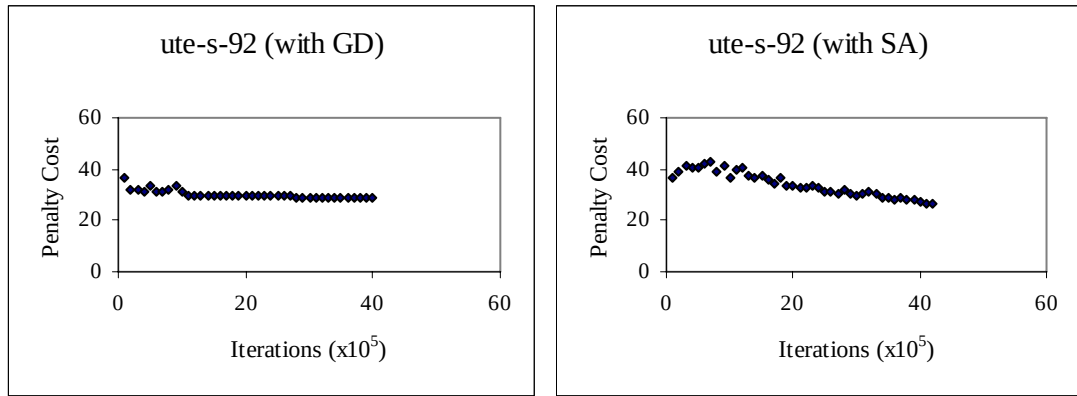


Figure 6.7. The behaviour of the “multi-start” large neighbourhood search algorithm on the *ute-s-92* dataset

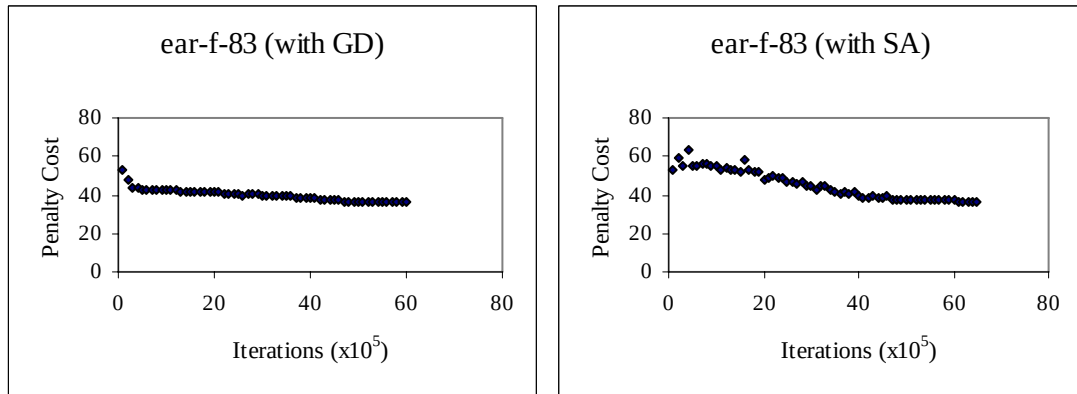


Figure 6.8. The behaviour of the “multi-start” large neighbourhood search algorithm on the *ear-f-83* dataset

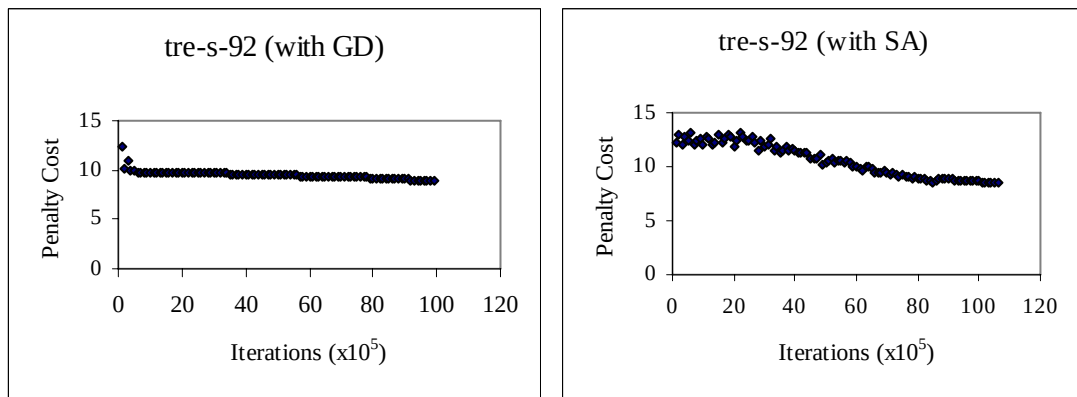


Figure 6.9. The behaviour of the “multi-start” large neighbourhood search algorithm on the *tre-s-92* dataset

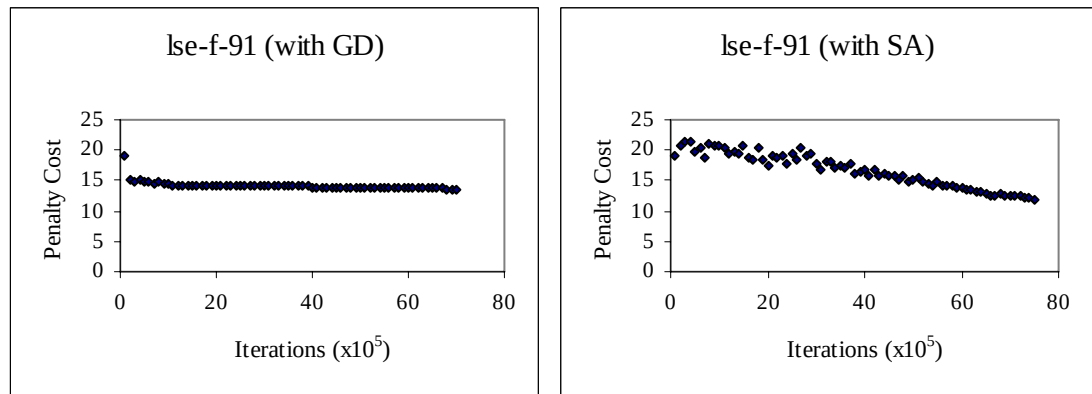


Figure 6.10. The behaviour of the “multi-start” large neighbourhood search algorithm on the *lse-f-91* dataset

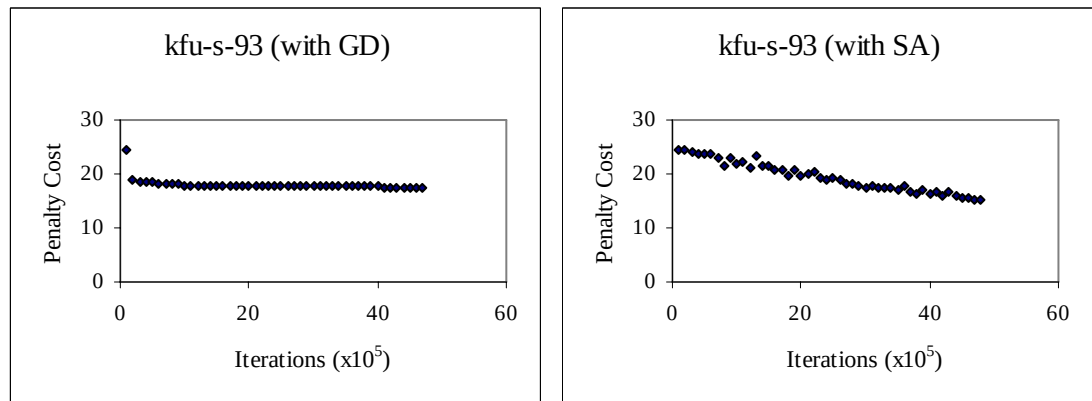


Figure 6.11. The behaviour of the “multi-start” large neighbourhood search algorithm on the *kfu-s-93* dataset

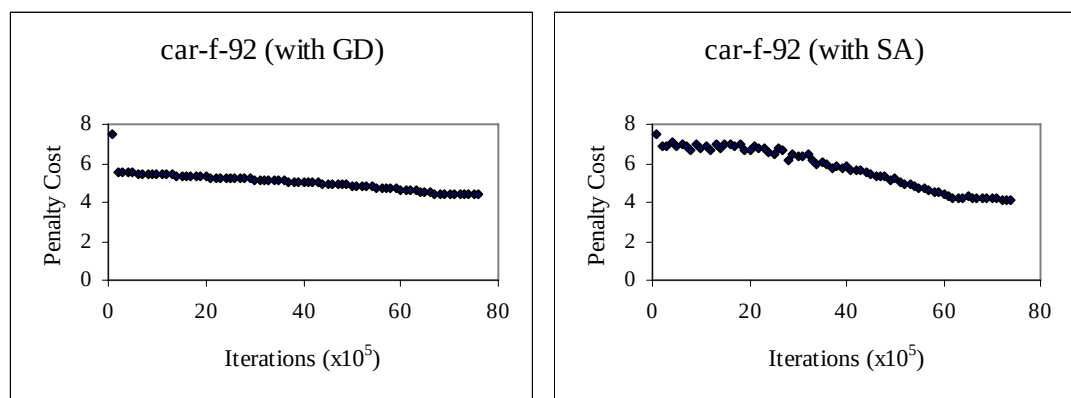


Figure 6.12. The behaviour of the “multi-start” large neighbourhood search algorithm on the *car-f-92* dataset

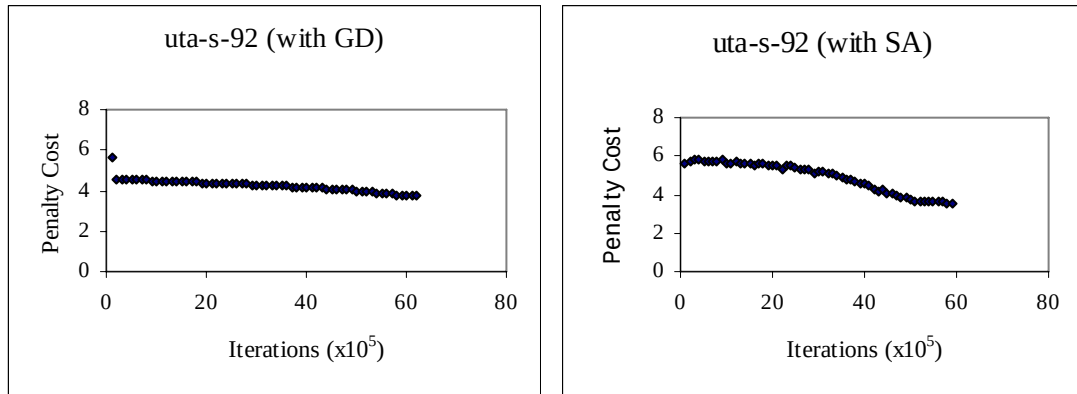


Figure 6.13. The behaviour of the “multi-start” large neighbourhood search algorithm on the *uta-s-92* dataset

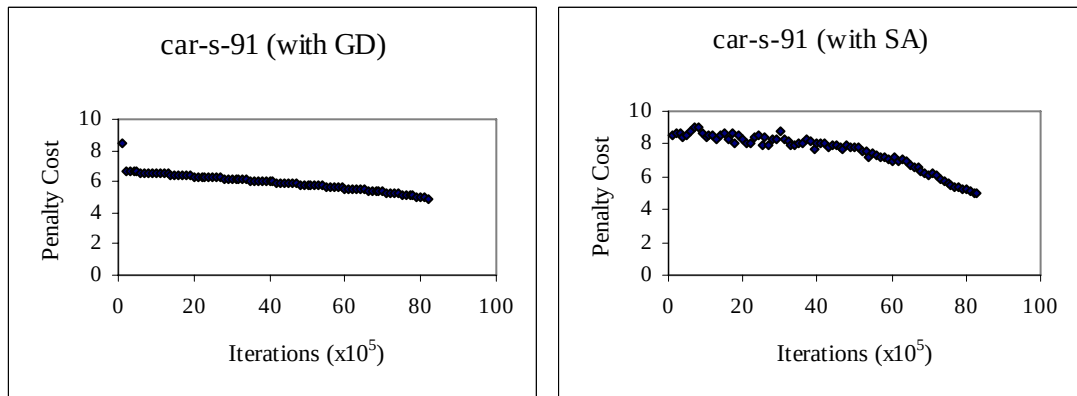


Figure 6.14. The behaviour of the “multi-start” large neighbourhood search algorithm on the *car-s-91* dataset

The distribution of points in these diagrams shows the correlation between the number of iterations and the overall solution quality. The diagrams presented in Figures 6.4 - 6.14 are very similar. An analysis of the diagrams shows that there is a trend of cost improvement as the number of iterations increases. However, as the number of iterations increases, the slope of the curves indicates a smaller decrease in the penalty cost. Figures 6.4 - 6.14 also show that the simulated annealing algorithm offers more flexibility in accepting a worse solution at the beginning of the search. It can be seen that the points in the graphs are quite scattered at the early stages of the search but the probability of accepting a worse solution is slowly lowered during the search. On the other hand, in the great deluge algorithm, the plotted graphs show that the solution points are not scattered throughout the search process.

6.5 Summary

In this chapter, a “multi-start” technique which hybridised the network flow optimisation approach with local search methods and a diversification strategy which restarts the search process (in the first phase) with a new solution (if there is no improvement in the quality of the solution after a certain number of iterations) were employed. These experiments indicate that the local search method can enhance the solution obtained from the first phase. Even though the experiments carried out in this chapter obtain only one best result, they show that the combination of the large neighbourhood search methodology with local search can produce a feasible and good quality timetable and the method is able to reduce the time taken to obtain good solutions compared to the time spent on the experiments carried out in Chapter 4. Moreover, it provides results that are consistently good across all the benchmark problems. The next chapter discusses the implementation of a local search approach to another timetabling problem domain (i.e. the university course timetabling problem).

PART III

New Course Timetabling Approaches

Chapter 7

An Investigation of the Variable Neighbourhood Search Approach for University Course Timetabling

7.1 Introduction

The course timetabling problem consists of assigning courses to a specific timeslot and room. The goal is to satisfy as many soft constraints as possible while constructing a feasible schedule. This chapter presents a variable neighbourhood search approach with an exponential monte carlo acceptance criterion. This search approach is a descent-ascent heuristic that is based on a random-descent local search. The solution returned by a local search algorithm after exploring a neighbourhood structure will be accepted based on the exponential monte carlo acceptance criterion. The approach is tested over eleven datasets from three classifications of the problem. The results demonstrate that the variable neighbourhood search approach is able to produce solutions that are competitive with state of the art techniques for the problems studied in this thesis.

A very large neighbourhood search methodology which can be effective for examination timetabling has been demonstrated earlier in this thesis. A key factor for this circumstance is the way of creating the neighbourhood structures and the search mechanism. Preliminary tests have been carried out to apply the same approach to course timetabling. Since a course timetabling problem usually has more constraints than the examination timetabling problem (based on the specification of the

university timetabling problem as presented in Chapter 3), the nodes in the generated improvement graph (note that the details on an improvement graph are presented in Chapter 4) are less connected to each other than the generated improvement graph for examination timetabling. Hence, it is not suitable to apply a modified shortest path label-correcting algorithm (also discussed in Chapter 4) to find a shortest path between source nodes and other nodes. Since the algorithm presented in Chapter 4 only deals with feasible moves, it seems clear that the method will not be suitable for course timetabling. However, a variable neighbourhood search is investigated for course timetabling in this chapter which is motivated by the goal of maintaining the importance of the neighbourhood structures.

This chapter is organised as follows: The next section describes the application of variable neighbourhood search to the course timetabling problem. It discusses local search, acceptance criteria and the neighbourhood structures that are used. The pseudo-code of the implemented algorithm is also presented in this section. The experiments and results are discussed in Section 7.3. Section 7.4 gives some brief concluding comments. The work presented in this chapter was published in the proceedings of the 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2005) (see Abdullah et al. 2005a).

7.2 Variable Neighbourhood Search (VNS) for the Course Timetabling Problem

This section discusses the application of the basic VNS (referred to hereafter as VNS-Basic) and the modified VNS with the exponential monte carlo acceptance criterion at the VNS level (referred to hereafter as VNS-EMC). The motivation for applying a monte carlo acceptance criterion is to enhance the exploration of the search space. The cooperation between VNS-EMC with a tabu search algorithm is also presented in this section which is referred to as VNS-Tabu.

7.2.1 Initial Solution: Constructive Heuristic

The initial solution is produced using a constructive heuristic which starts from an empty timetable. This feasible solution is obtained by adding or removing appropriate events (courses) from the schedule based on room availability. The aim is to first schedule the courses with the least room availabilities. We do not take into account the soft constraint violations, until the hard constraints are met. The schedule is made feasible before starting the algorithms.

7.2.2 Neighbourhood Structures within VNS

The following neighbourhood structures are used at the local search level:

- (1) Move timeslot: Take 2 timeslots (selected at random), say t_i and t_j (where $j > i$) and the timeslots are ordered $t_0, t_1, \dots, t_{44}$. Take all the courses in t_i and allocate them to t_j . Now take the courses that were in t_j and allocate them to t_{j-1} . Then allocate those that were in t_{j-1} to t_{j-2} and so on until those courses that were in t_{i+1} are allocated to t_i . Terminate the process.

Example 1:

Let us assume that the selected timeslots are t_2 and t_5 , respectively (as in Figure 7.1). The arrows show how the timeslots are reshuffled to a new position.

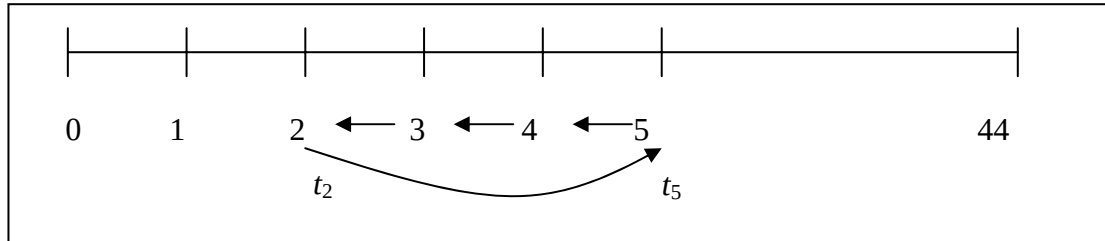


Figure 7.1. Example 1 of “move a whole timeslots” neighbourhood structure

Courses in t_2 are moved to the new position t_5 . All the courses that are scheduled in between t_2 and t_5 are afterwards reshuffled based on the following rules:

if ($t_2 < \text{current timeslot} \leq t_5$)

new position = current timeslot – 1

Example 2:

Assume that two selected timeslots are t_6 and t_3 , respectively. Note that $t_6 > t_3$ (as in Figure 7.2). The arrows show how the reshuffle process takes place.

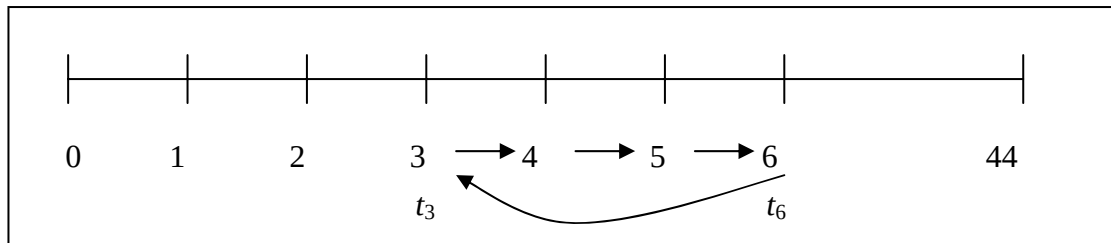


Figure 7.2. Example 2 of “move a whole timeslots” neighbourhood structure

Courses in t_3 are moved to the new position t_6 . All courses that are scheduled in the current timeslot in between t_3 and t_6 are afterwards reshuffled based on the following rules:

if ($t_6 \leq \text{current timeslot} < t_3$)

new position = current timeslot + 1

- (2) Move the highest penalty course (i.e. the course with the highest number of soft constraint violations). Take 10% of the courses at random. Then select the one with the highest penalty cost and allocate it to the timeslot (that is chosen from all of the available timeslots) which generates the lowest penalty and which does not create an infeasibility.
- (3) Move the highest penalty course from a random 30% selection of the courses to a random feasible timeslot.
- (4) Move the highest penalty course from a random 20% selection of the courses to a random feasible timeslot.
- (5) Move the highest penalty course from a random 10% selection of the courses to a random feasible timeslot.
- (6) Select a course at random and find another course at random with which it can swap timeslots.
- (7) Choose a single course at random and move to another random feasible timeslot.
- (8) Move 2 courses to random feasible timeslots.
- (9) Move 3 courses (as in (8)).

- (10) Move 4 courses (as in (8)).
- (11) Move 5 courses (as in (8)).
- (12) Select two timeslots at random and simply swap all the courses in one timeslot with all the courses in the other timeslot.

The neighbourhood structure described in (1) is implemented before the VNS approach is applied. This neighbourhood structure allows courses to be moved relatively to each other except for courses that are scheduled in the same timeslot.

7.2.3 Acceptance Criteria

In this chapter, two acceptance criteria are used. The motivation for applying acceptance criteria is to jump to other distant solution points by accepting a worse solution (i.e. case (b)). The criteria are:

- a) Descent: This accepts an improved solution only.
- b) Exponential monte carlo: This accepts an improved solution and a worse solution with a certain probability. The explanation of exponential monte carlo is discussed in Chapter 4. For more details, refer to Ayob and Kendall (2003).

7.2.4 Tabu List

One of the experiments implemented in this chapter used a tabu list to improve the efficiency of the search. The basic idea of the tabu list here is to prevent a neighbourhood structure that did not perform well recently from being chosen in future iterations, so that the search can be directed to other possible areas of the search space. In order to do so, a First In First Out (FIFO) queue of fixed length called the tabu list is employed. The list is initiated by null elements. An unperformed neighbourhood structure is added to the tabu list. For example, if the solution obtained after performing the neighbourhood structure N2 is worse than the best solution in hand (and is also rejected by the exponential monte carlo acceptance criterion) then the neighbourhood structure N2 will be added to the tabu list. Based on preliminary tests, the size of the tabu list is set to 2.

7.2.5 Stopping Criterion

In these computational experiments, the total number of iterations performed is used as a stopping criterion. Let $eval$ be the current evaluation, then the experiments stop when $eval > Max_eval$ for a given constant Max_eval . In these experiments Max_eval is set to 200,000 (as in Socha et al. 2002).

7.2.6 Local Search

A local search heuristic, which is also known as a neighbourhood search, explores the neighbourhood of the present solution by iteratively performing local changes in order to improve the quality of a solution until a local optimum (the best solution(s) in the defined neighbourhood) is found.

In this chapter, a random-descent local search that only accepts an improved solution after exploring the nearest neighbours in the defined neighbourhood structure is developed. For the approach presented in this chapter, a set of the neighbourhood structures as defined in Section 7.2.2 is applied. Hard constraints are never violated during the course of the timetabling process. Let n_k (where $k = 1, \dots, K$) be a set of predefined neighbourhood structures. Note that K is the total number of neighbourhood structures to be used in the search. Let $f(s)$ be the quality of the solution s . The local search starts by randomly generating a solution s' from the k^{th} neighbourhood. Starting from the initial solution s' , the local search sequentially visits the solution in the k^{th} neighbourhood of s' until a local optimum s'' is obtained. The solution s'' is accepted if $f(s'')$ is better than $f(s)$. This algorithm will also accept a worse solution with a probability that is generated from the exponential monte carlo acceptance criterion. Whenever a neighbourhood structure generates a better solution (or an accepted worse solution), the search starts over from the first neighbourhood (which is also the least time consuming). Otherwise, the next neighbourhood is employed.

Based on initial tests with neighbourhood ordering, it is shown that the best sequence of neighbourhood structures is to order them by increasing size. Figure 7.3 shows the pseudo-code of the approach.

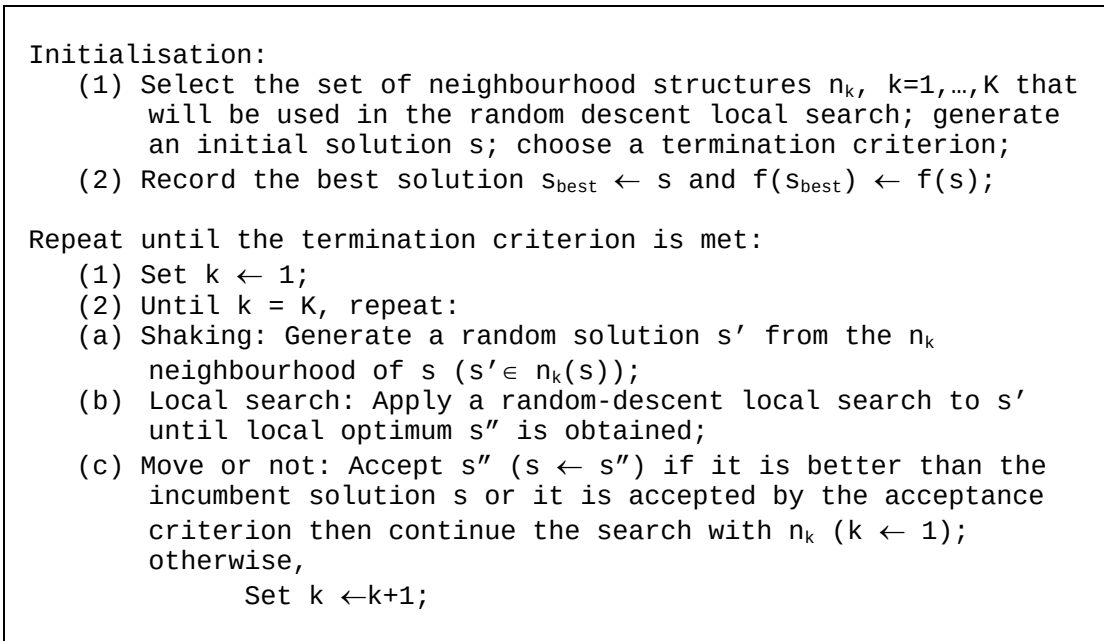


Figure 7.3. The pseudo-code of a modified VNS for the course timetabling problem

Further investigation of the performance of VNS is carried out by adding a tabu list to penalise neighbourhood structures that have not lead to an accepted new solution. This approach is called VNS-Tabu. The VNS-Tabu method slightly differs from the tabu search meta-heuristic with respect to the elements that are made tabu. In the tabu search meta-heuristic, a move is made tabu to prevent “cycling” (i.e. visiting previously visited solutions in the search space). Certainly, in the VNS-Tabu, the recently unperformed neighbourhood structures are made tabu. The pseudo-code for VNS-Tabu is given in Figure 7.4. A tabu restriction is applied where the neighbourhood structure n_k will be tabu if the value of the new solution s is greater than the value of the old solution s and the solution s'' is rejected by the exponential monte carlo acceptance criterion. The algorithm will prevent the neighbourhood structures that have not performed well recently to be chosen in the next iteration, so that the search can be directed to other possible areas of the search space. It is believed that the unperformed neighbourhood structure at the current solution search space most probably will also not perform well on the recently updated solution. The tabu tenure is set to 2 based on preliminary tests. A higher value of the tabu tenure is not appropriate in this case because the neighbourhood structures are kept in the tabu list. The higher the value of the tabu tenure, the longer the neighbourhood structures remain tabu. This would limit the number of available neighbourhood structures for

the next evaluation in the VNS approach. The neighbourhood structures will remain tabu for a number of iterations which is equal to the tabu tenure.

Initialisation:

- (1) Select the set of neighbourhood structures n_k , $k=1,\dots,K$ that will be used in the random descent local search; generate an initial solution s ; choose a termination criterion;
- (2) Record the best solution $s_{\text{best}} \leftarrow s$ and $f(s_{\text{best}}) \leftarrow f(s)$;

Repeat until the termination criterion is met:

- (1) Set $k \leftarrow 1$;
- (2) Until $k = K$, repeat:
 - (a) Shaking: Generate a random solution s' from the n_k neighbourhood of s ($s' \in n_k(s)$);
 - (b) Local search: Apply a random-descent local search to s' until local optimum s'' is obtained;
 - (c) Move or not:
 - if ($(f(s''))$ is better than incumbent solution s) or ($f(s'')$ is accepted by the acceptance criterion)) then
 - $s \leftarrow s''$;
 - set $k \leftarrow 1$;
 - while k is in the tabulist and $k < K$
 - $k \leftarrow k+1$;
 - continue the search with n_k ;
 - else
 - insert k to the tabulist;
 - set $k \leftarrow k+1$;
 - increase the tabu length by 1;
 - if tabu length $>$ tabu tenure
 - release the first neighbourhood structure from the tabu list;
 - while k is in the tabulist and $k < K$
 - $k \leftarrow k+1$;

Figure 7.4. The pseudo-code for VNS-Tabu

7.3 Experiments and Results

The proposed method was tested on the benchmark course timetabling problems presented in Socha et al. (2002). The constructive heuristic employed in this chapter is unable to produce a feasible solution for this *large* dataset. A different initialisation from Chapter 9 for the *large* dataset is used. They are grouped into 5 *small* ($N = 100$, $R = 5$, $F = 5$ and $M = 80$), 5 *medium* ($N = 400$, $R = 10$, $F = 5$ and $M = 200$) and one *large* dataset ($N = 400$, $R = 10$, $F = 10$ and $M = 400$). The details of the benchmark course timetabling problems are discussed in Chapter 3.

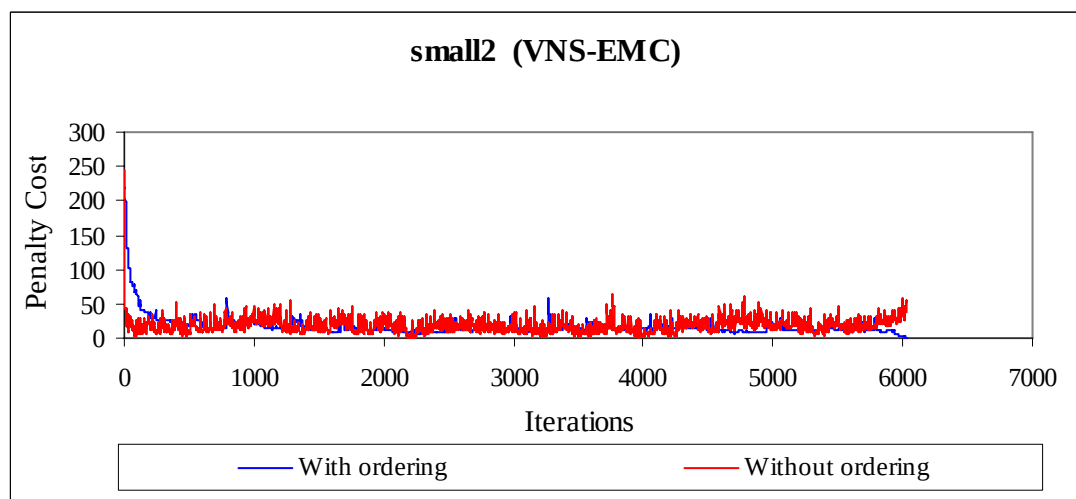
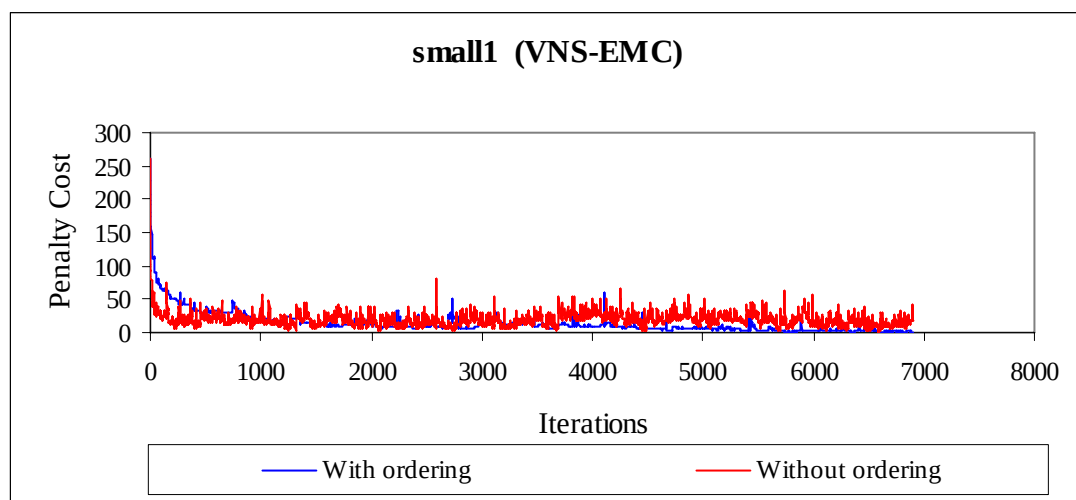
In Table 7.1, the test results on the benchmark course timetabling problems are presented for two types of variable neighbourhood search (i.e. VNS-Basic and VNS-EMC) “with and without ordering” of the neighbourhood structure. For the case “without ordering”, the VNS will continue the search with the current neighbourhood if it yields an improvement rather than go back to the first neighbourhood structure (set $k = 1$) as in “with ordering VNS” each time an improvement is found (or if the worse solution is accepted by the acceptance criterion). For instance, let $k = 5$ represent the current neighbourhood. The local optimum obtained from this neighbourhood structure will be compared to the incumbent solution. If there is an improvement in the quality of the solution, then the search will continue from $k = 5$. Otherwise, the the next neighbourhood structure ($k = 6$) is employed. This allows the next neighbourhood structure ($k = 6$ in this example) to be considered in the next search (but it does not need to tune several sequences of the neighbourhood structures in order to determine which neighbourhood structure the search should start with i.e. at $k = 1$). The motivation of this comparison is to see whether the sequence of neighbourhood structures plays a role in the VNS approach. A comparison between two VNS approaches is also made to investigate the importance of accepting worse moves in order to jump to other parts of the search space to obtain a better quality solution. The best results out of 5 runs are presented.

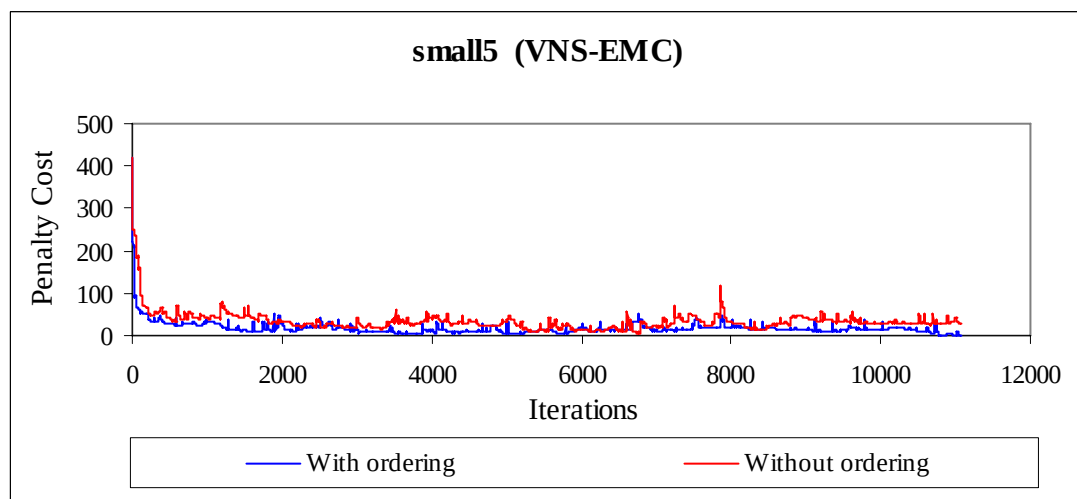
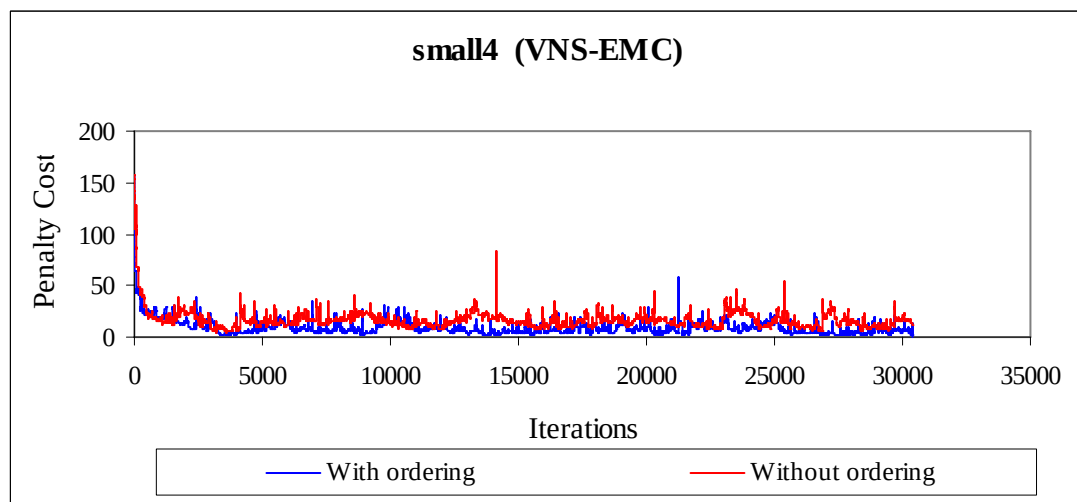
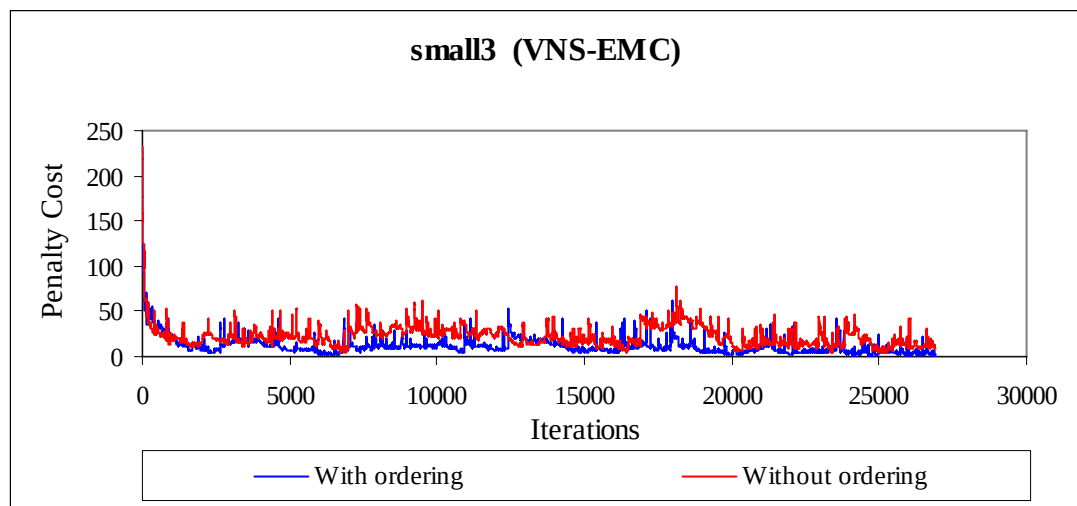
Table 7.1. Results on the course timetabling problem

Datasets	Initial solution	With ordering VNS		Without ordering VNS	
		VNS-Basic	VNS-EMC	VNS-Basic	VNS-EMC
<i>small1</i>	261	8	0	19	1
<i>small2</i>	245	12	0	7	1
<i>small3</i>	232	8	0	11	1
<i>small4</i>	158	15	0	19	0
<i>small5</i>	421	5	0	5	0
<i>medium1</i>	914	418	338	445	347
<i>medium2</i>	878	414	337	413	354
<i>medium3</i>	941	441	384	462	390
<i>medium4</i>	865	381	299	406	324
<i>medium5</i>	780	390	307	416	316
<i>large</i>	1603	992	945	996	962

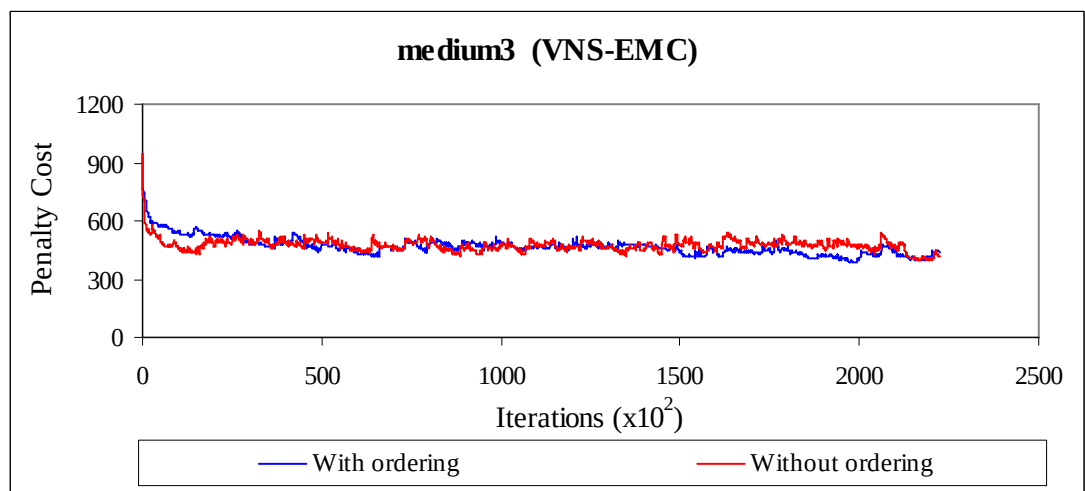
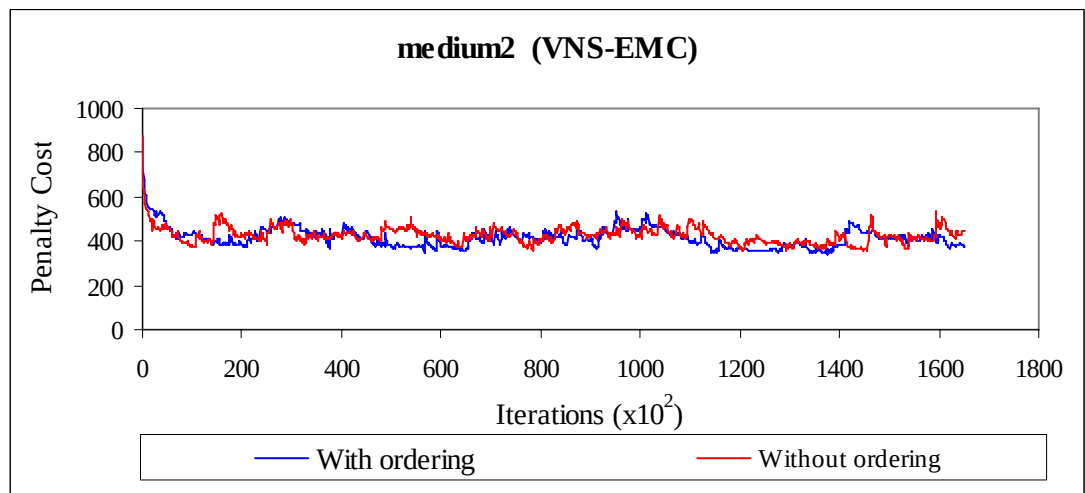
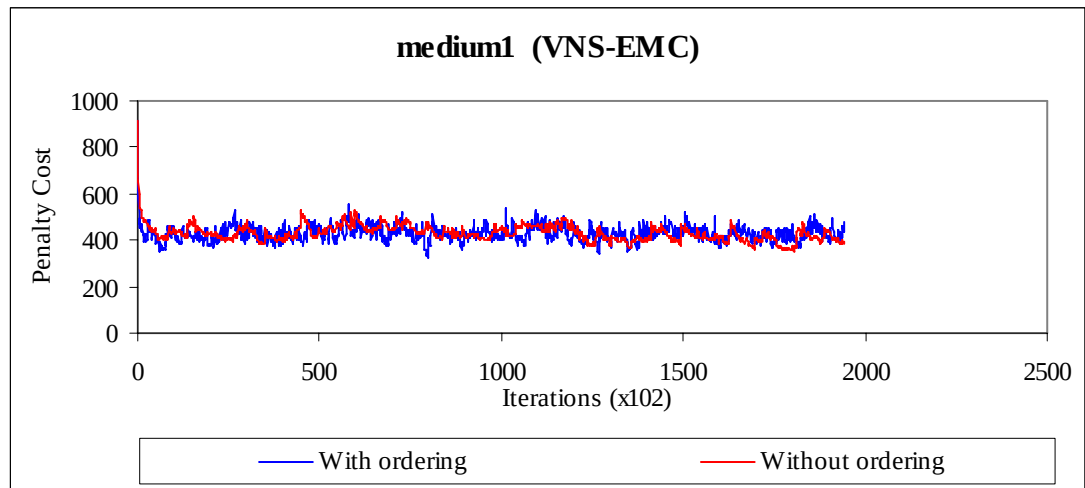
From Table 7.1, it can be seen that the “with ordering” VNS for VNS-Basic and VNS-EMC are both better than or equal to the “without ordering” VNS (except for the *medium2* dataset). This shows that ordering of the neighbourhood structure in the

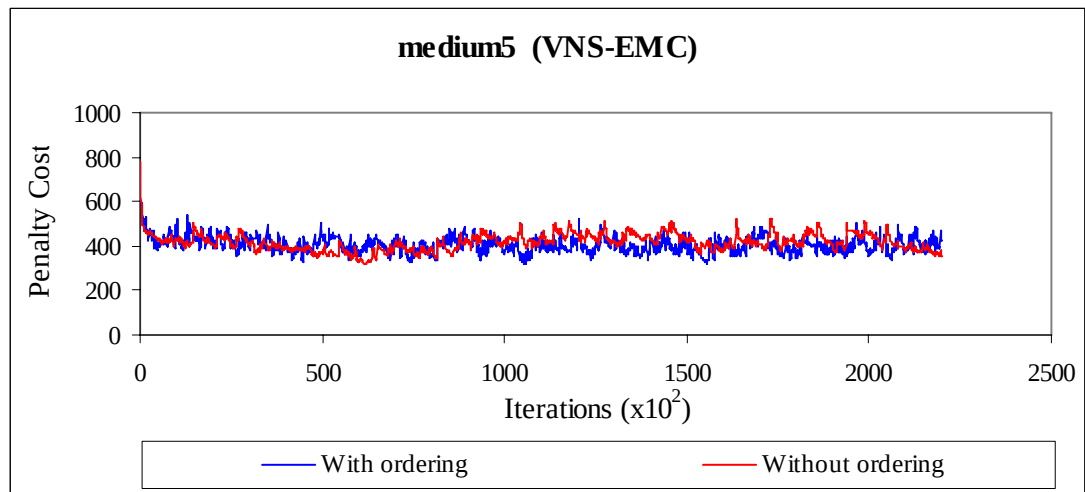
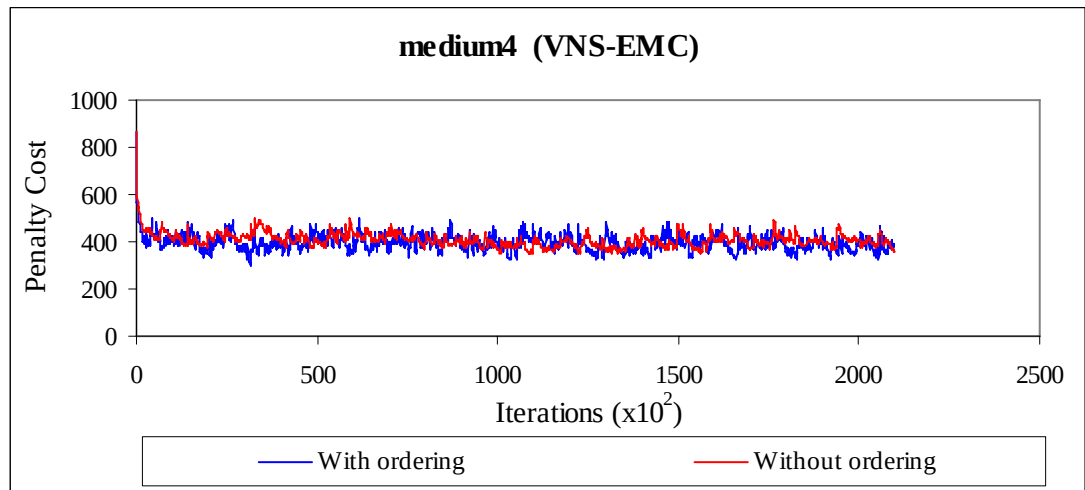
VNS approach and which neighbourhood the search will begin after obtaining a better solution (or a worse accepted solution) is of significant importance. It is believed that the search approach has a chance to explore the accepted solution using different neighbourhood structures which can lead to a different search space rather than employing the same neighbourhood structure until it is exhausted and no more improvement of the solution quality is obtained (which means that only an unimproved solution is fed to the next neighbourhood structure in the next search). The results obtained from VNS-EMC are better than VNS-Basic for both the “with and without ordering” algorithms. This shows that accepting worse solutions (as a diversification strategy) helps the VNS approach to better explore the search space as suggested by Glover and Laguna (1997). Figures 7.5a-e and 7.6a-e show the behaviour of the approach applied to the *small* and *medium* datasets on “with and without ordering” for VNS-EMC (to show the importance of ordering), respectively.





Figures 7.5a-e. The behaviour of the algorithm on the *small* datasets (VNS-EMC)





Figures 7.6a-e. The behaviour of the algorithm on the *medium* datasets (VNS-EMC)

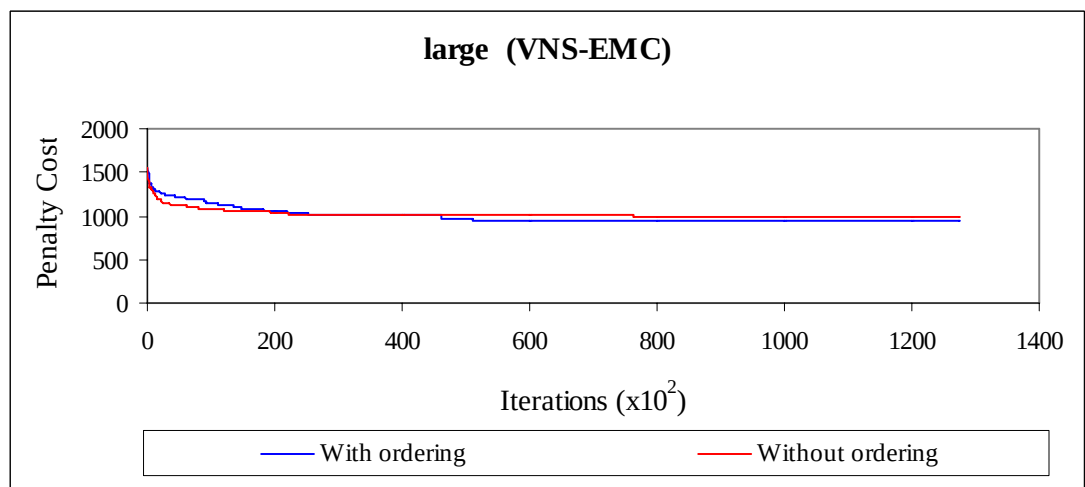
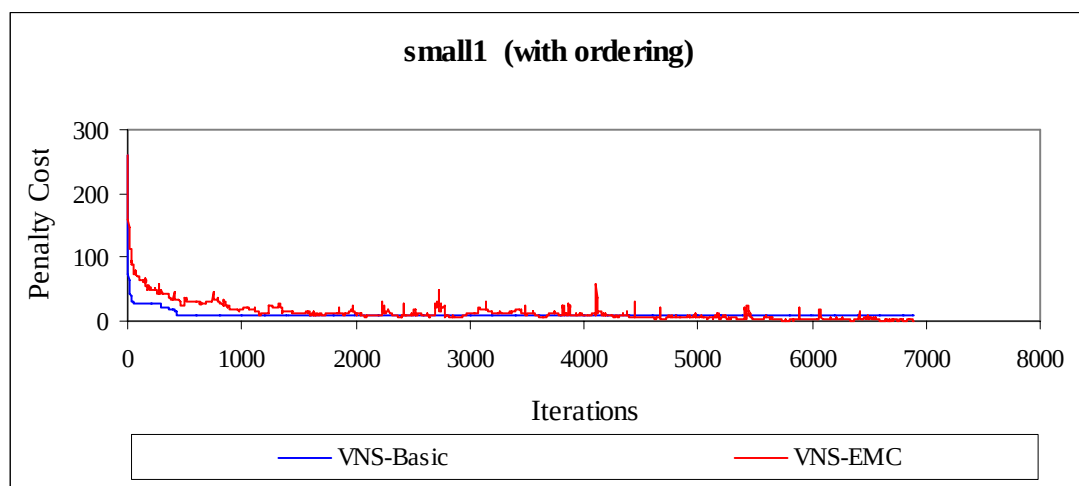
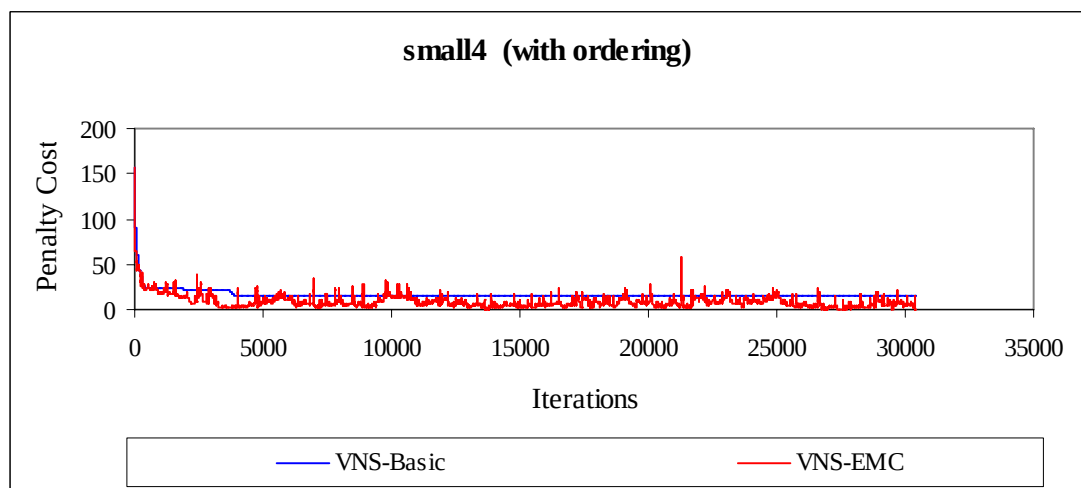
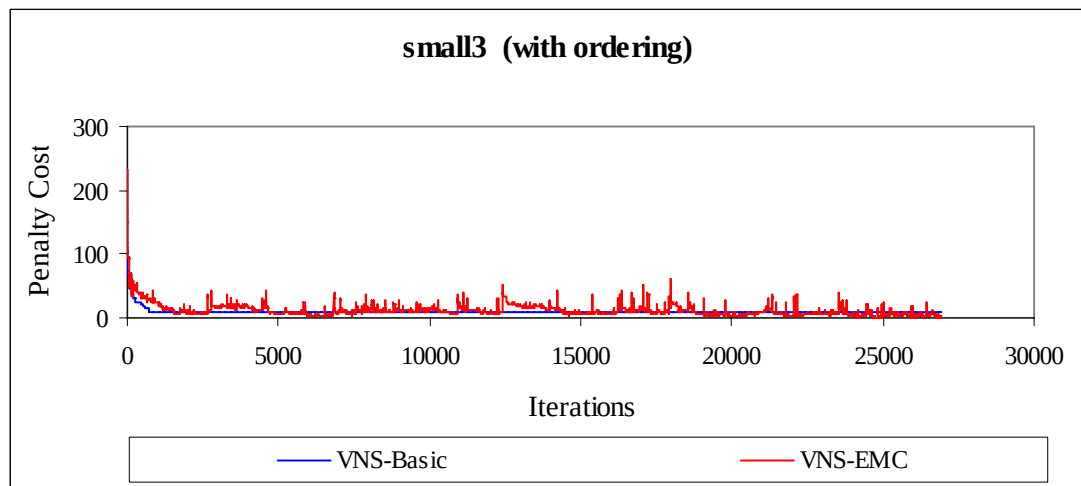
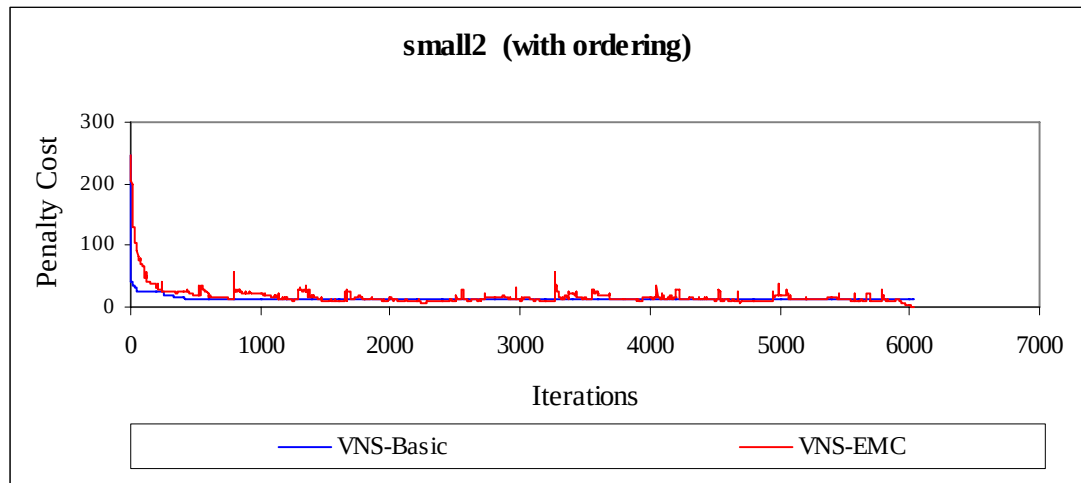


Figure 7.7. The behaviour of the algorithm on the *large* dataset (VNS-EMC)

In all the figures, the x-axis represents the number of iterations while the y-axis represents the penalty cost. Every point in the graphs corresponds to the penalty cost and the number of iterations of a separate solution. These graphs show how the algorithm explores the search space. In Figures 7.5a-e (and also in Figures 7.6a-e for VNS-EMC), the curves move up and down because worse solutions are accepted with a certain probability in order to escape from local optima. The analysis of the graphs in Figures 7.5a-e, 7.6a-e and 7.7 show that the slope of the curves is relatively steep which indicates the high quality improvement of the solutions at the beginning of the search for both the “with and without ordering” versions of VNS-EMC. The relative improvement of the solution becomes lower as the search time increases. By using the right sequence of neighbourhood structures, the “with-ordering VNS” is able to find better local optima than the “without ordering VNS”. This shows the importance of a sequence of neighbourhood structures in the VNS approach.

Figures 7.8a-e and 7.9a-e show the behaviour of the algorithm on the *small* and *medium* datasets when “with ordering” VNS-Basic and VNS-EMC are employed. The figures indicate the importance of accepting a worse solution in order to obtain a better eventual solution.





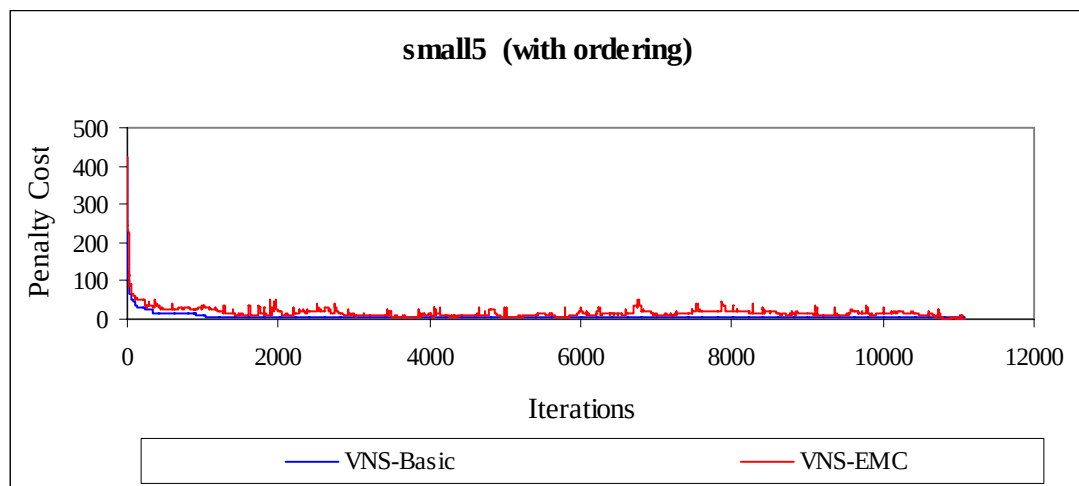
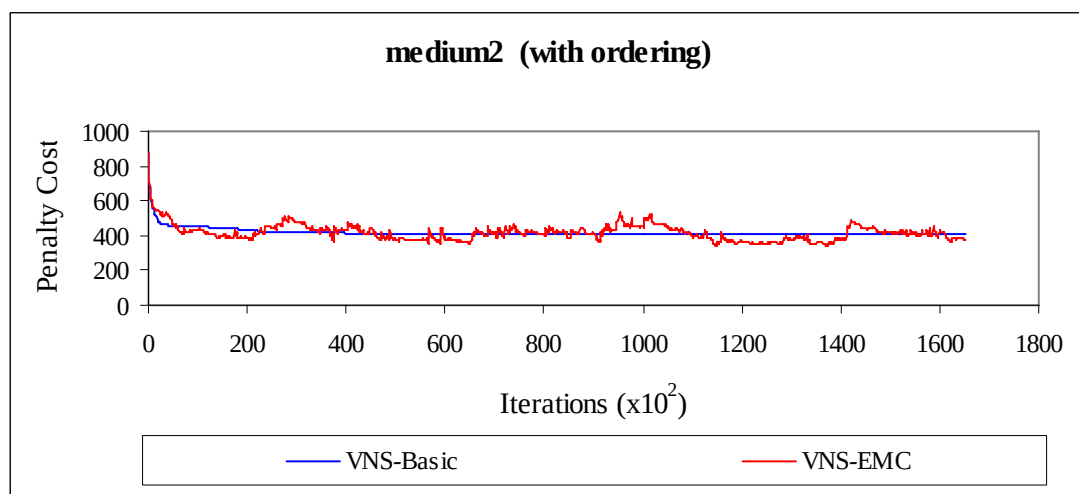
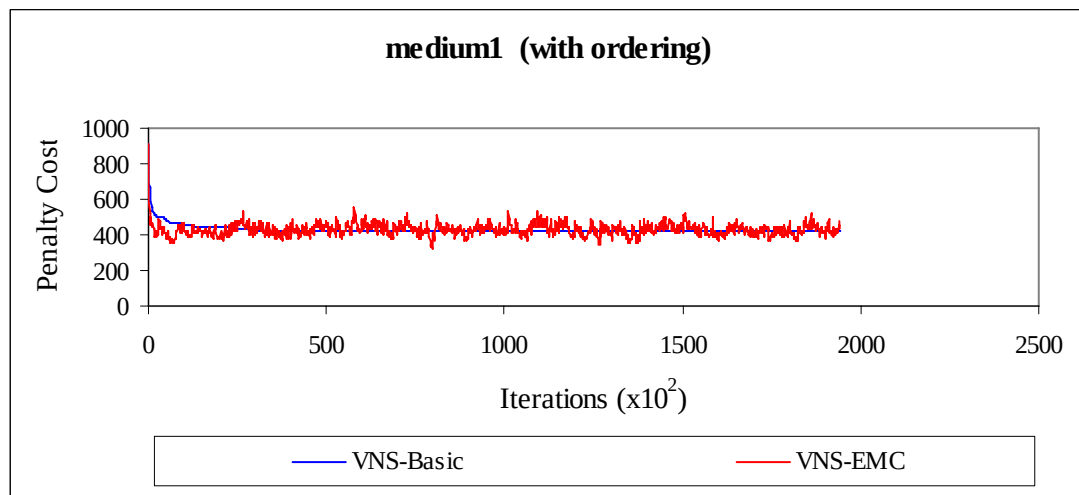


Figure 7.8a-e. The behaviour of the algorithm on the *small* datasets (with-ordering)



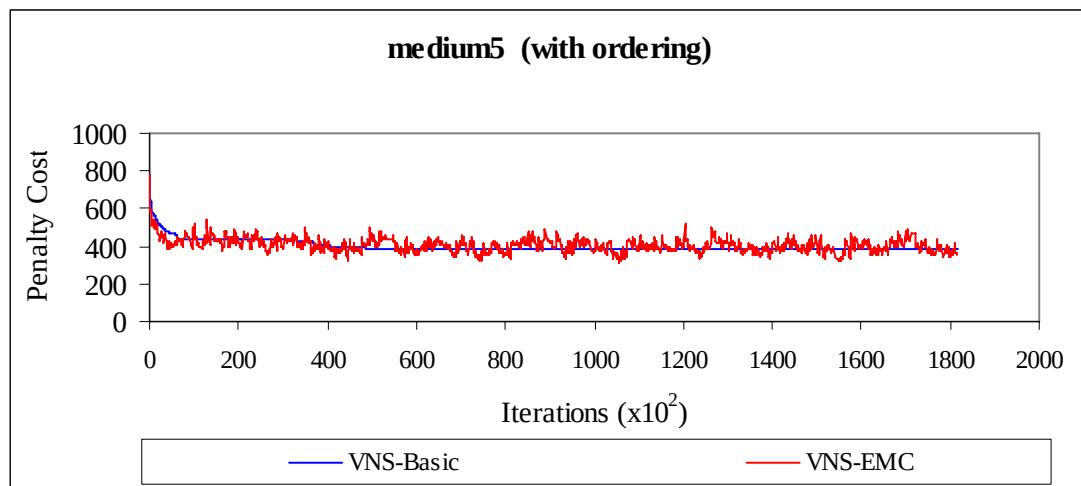
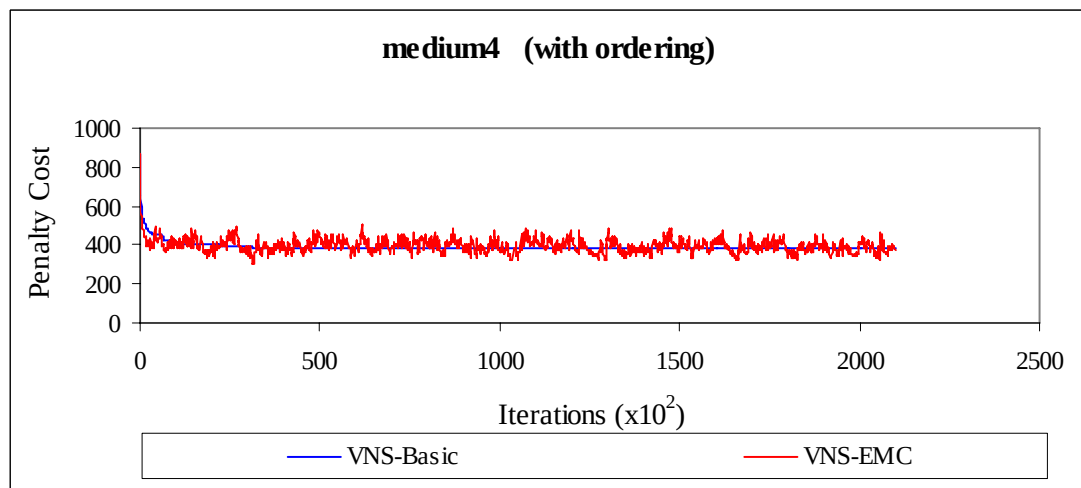
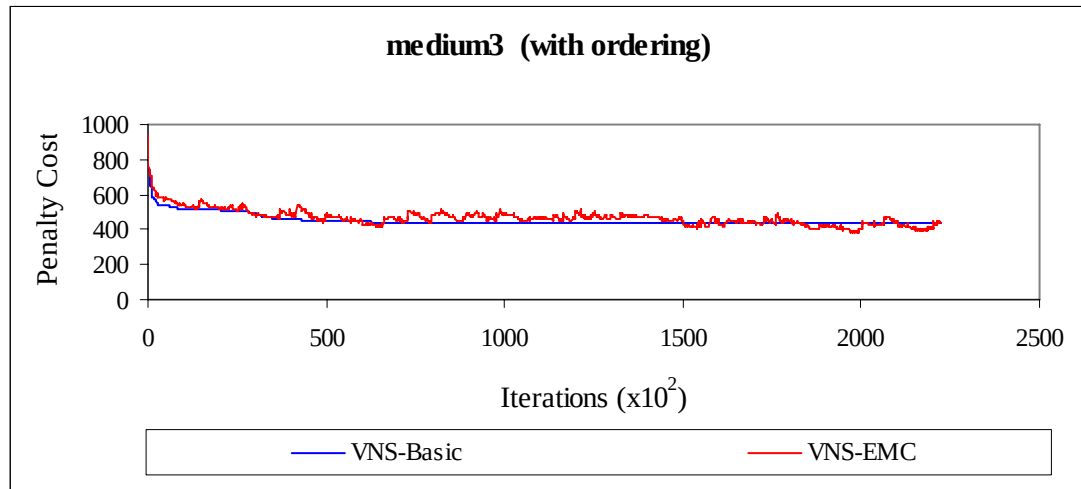


Figure 7.9a-e. The behaviour of the algorithm on the *medium* datasets (with-ordering)

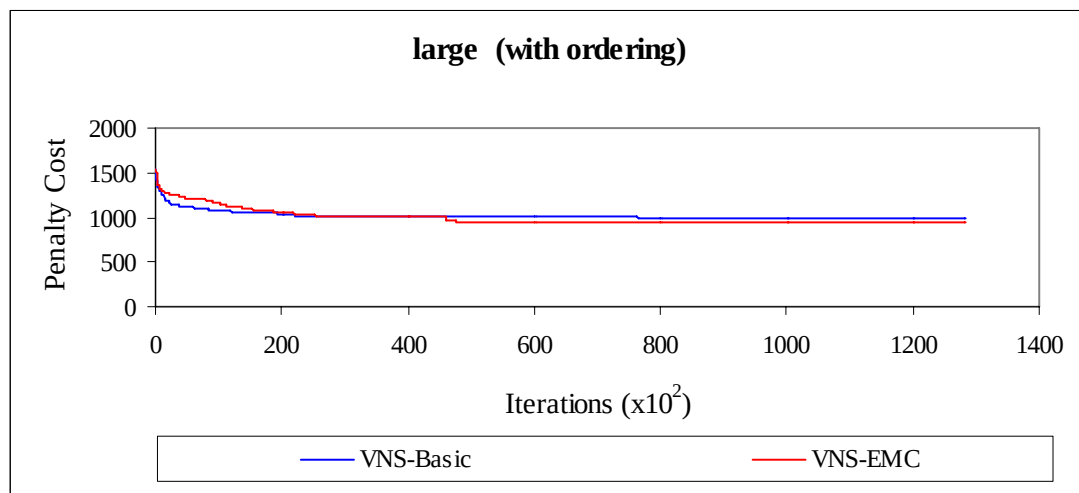


Figure 7.10. The behaviour of the algorithm on the *large* dataset (with-ordering)

The graphs in Figures 7.8a-e and 7.9a-e illustrate the same behaviour as in Figures 7.5a-e and 7.6a-e at the early stage of the search with a high improvement in the solution quality. However, VNS-Basic gets stuck in local optima after a certain number of iterations whilst, by accepting worse solutions, the VNS-EMC can obtain a better solution (i.e. zero penalty cost as shown in Figures 7.8a-e). This indicates how a diversification strategy (i.e. accepting worse solutions) efficiently helps the technique to explore the search space. This was also the case with the *large* dataset as shown in Figure 7.10.

Table 7.2 shows the comparison of this approach with other available approaches in the literature: A local search method and ant algorithm by Socha et al. (2002); a tabu-search hyper-heuristic by Burke et al. (2003a); a graph-based hyper-heuristic by Burke et al. (2007); and a fuzzy approach by Asmuni et al. (2005b). Socha et al. (2002) present the average results out of 50 runs on the *small* problems, 40 runs on the *medium* problems and 10 runs on the *large* problem. Burke et al. (2003a, 2007) and Asmuni et al. (2005b) present the best results obtained from their experiments. The term “x%Inf” in Table 7.2 indicates a percentage of runs that failed to obtain feasible solutions.

Table 7.2. Comparison results on the course timetabling problem using VNS

Datasets	VNS-EMC (Best)	Socha et al. (2002) (Average)	Socha et al. (2002) (Average)	Burke et al. (2003a) (Best)	Burke et al. (2007) (Best)	Asmuni et al. (2005b) (Best)
<i>small1</i>	0	8	1	1	6	10
<i>small2</i>	0	11	3	2	7	9
<i>small3</i>	0	8	1	0	3	7
<i>small4</i>	0	7	1	1	3	17
<i>small5</i>	0	5	0	0	4	7
<i>medium1</i>	338	199	195	146	372	243
<i>medium2</i>	326	202.5	184	173	419	325
<i>medium3</i>	384	77.5% Inf	248	267	359	249
<i>medium4</i>	299	177.5	164.5	169	348	285
<i>medium5</i>	307	100% Inf	219.5	303	171	132
<i>large</i>	945	100% Inf	851.5	80% Inf 1166	1068	1138

The best results are presented in bold. In terms of feasibility, the VNS-EMC approach is able to produce ten out of eleven feasible solutions (the solution for *large* dataset is taken from Chapter 9), whereas the local search only produced nine feasible solutions (with two infeasible solutions for the *medium5* and *large* datasets). Other approaches are able to produce feasible solutions for all datasets. It can be seen that the VNS-EMC approach produces better or equivalent results on the *small* datasets when compared against all the other methods. Indeed, it is the only approach which can get zero penalty solutions on all 5 *small* problems. For the *medium* and *large* datasets, the results are comparable to other results published in the literature except on the *medium3* and *medium5* datasets, where the VNS-EMC approach performs quite poorly. It is believed that the VNS-EMC approach performs well for the *small* datasets because they might have more feasible solution points in the search space compared to the *medium* datasets. It is also believed that the approach should be more selective in choosing the neighbourhood structures while solving the *medium* and *large* datasets.

Table 7.3 shows the comparison of the results obtained from VNS-Tabu, VNS-EMC and the best known results. It is interesting to compare the results from VNS-Tabu with the VNS-EMC results to show how the tabu list helps to reduce the penalty cost for this problem.

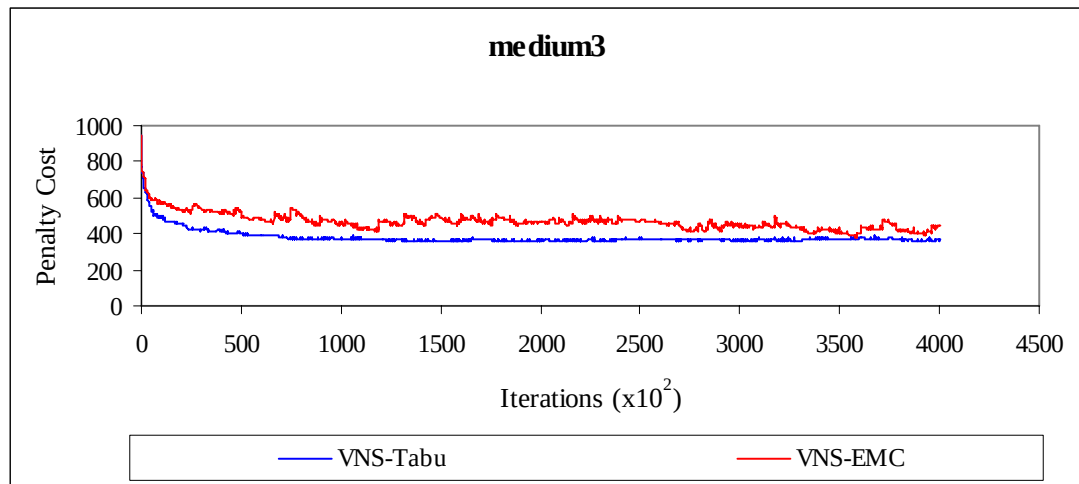
Table 7.3. The results on VNS-Tabu

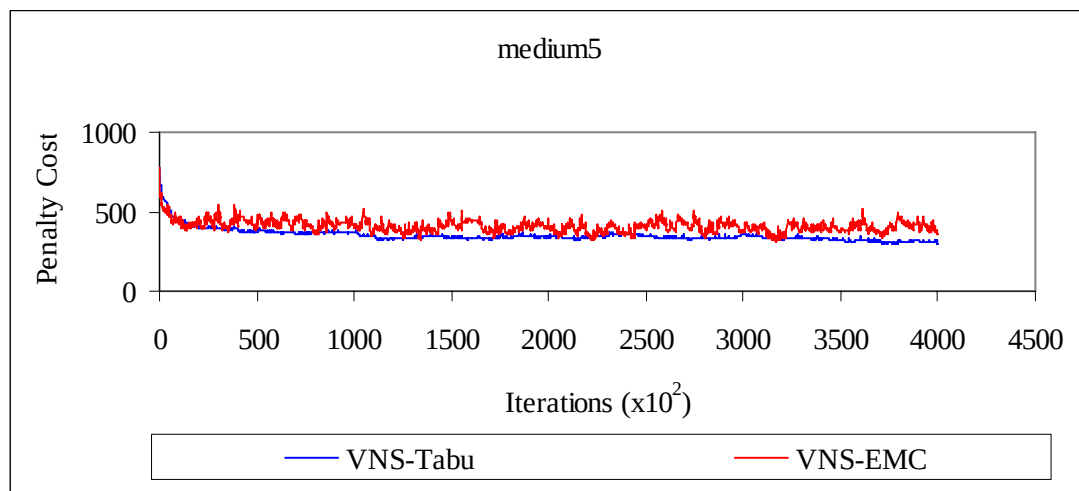
Datasets	VNS-Tabu	VNS-EMC	Best known
<i>small1</i>	0	0	1
<i>small2</i>	0	0	2
<i>small3</i>	0	0	0
<i>small4</i>	0	0	1
<i>small5</i>	0	0	0
<i>medium1</i>	317	338	146
<i>medium2</i>	313	326	173
<i>medium3</i>	357	384	248
<i>medium4</i>	247	299	164.5
<i>medium5</i>	292	307	132
<i>large</i>	926	945	851.5

The results in Table 7.3 show the best results obtained from five runs of the VNS-Tabu algorithm. It can be seen that there is no difference between VNS-Tabu and VNS-EMC in terms of the penalty cost when applied to the *small* datasets. However, there are changes in the penalty cost for the *medium* and *large* datasets (as shown in italics). The percentage improvement ($\Delta \%$) obtained by applying the VNS-Tabu compared to VNS-EMC for the *medium* and *large* dataset is computed as

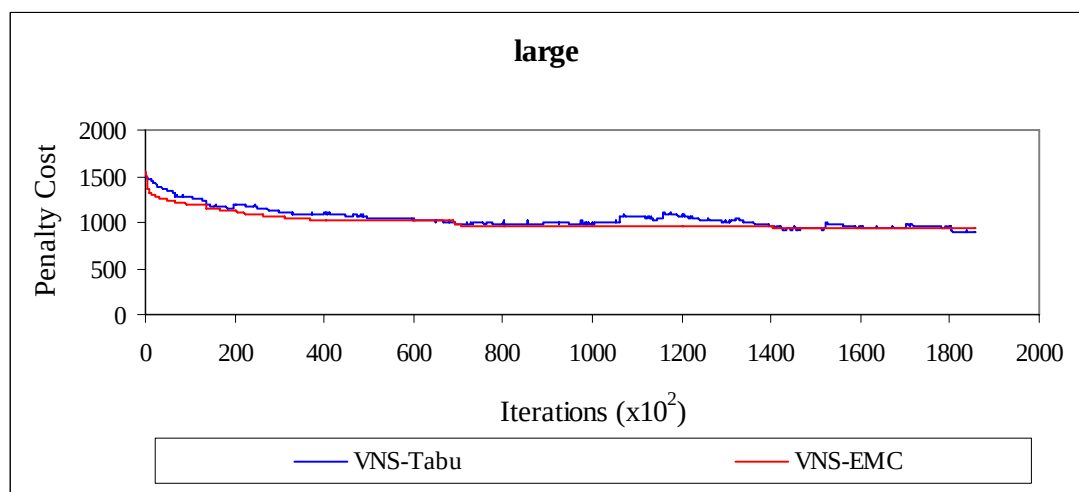
$$\Delta \% = (\text{best VNS-EMC} - \text{best VNS-Tabu}) * 100 / \text{best VNS-EMC}$$

This shows that the VNS-Tabu managed to reduce the penalty cost for the *medium* datasets by between 4.0% and 18% and for the *large* dataset by about 2%. Although VNS-Tabu cannot beat the best known results for the *medium* and *large* datasets, it is able to improve the results, especially for the *medium3* and *medium5* datasets, of the VNS-EMC approach.





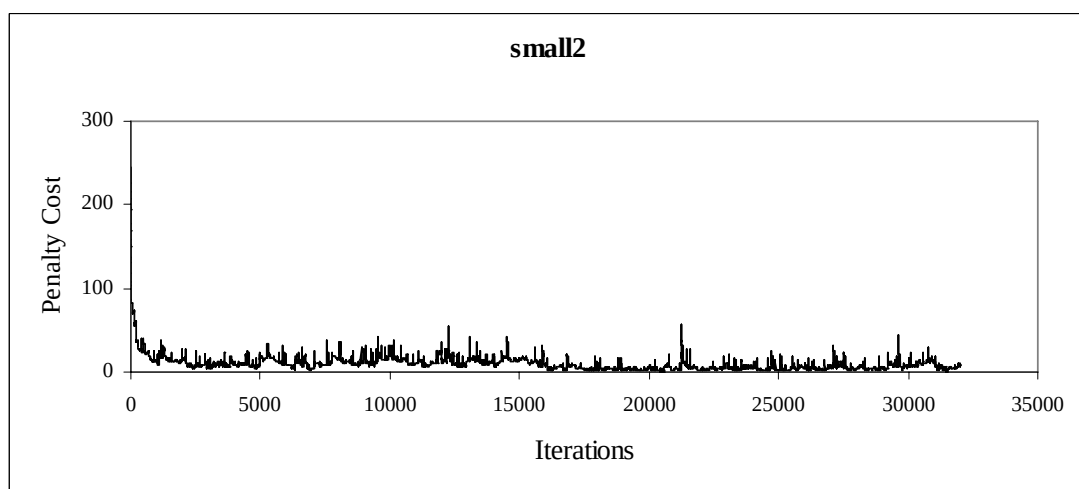
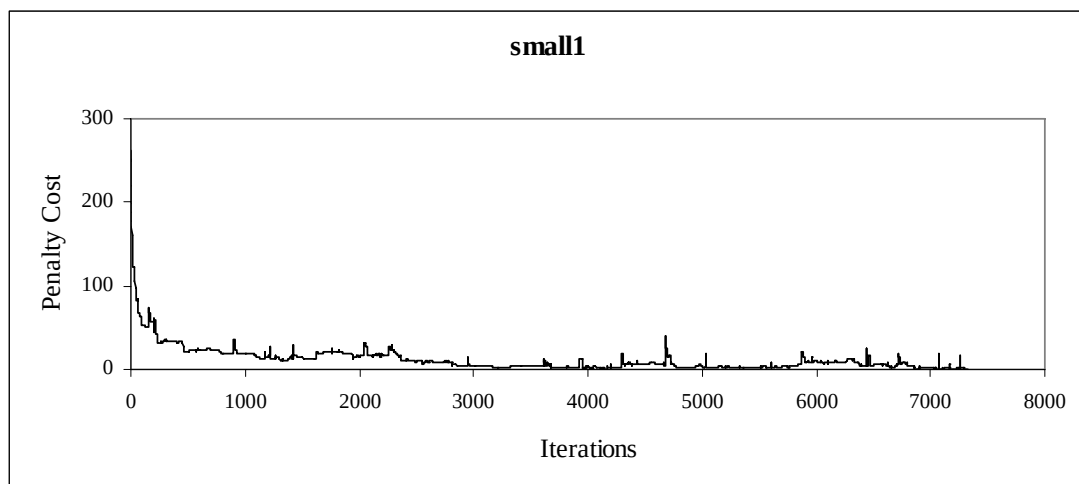
Figures 7.11a-b. The behaviour of VNS-Tabu and VNS-EMC algorithms applied to the *medium3* and *medium5* datasets, respectively

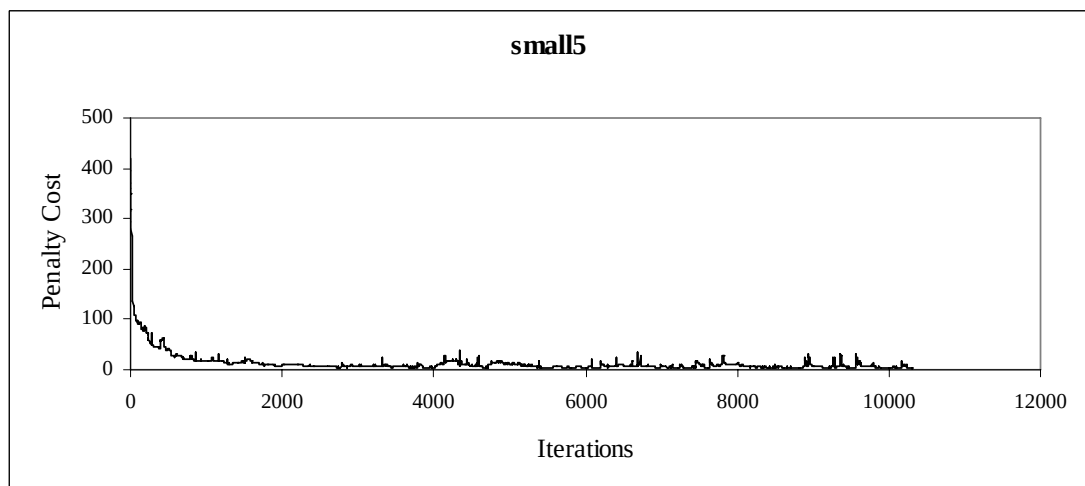
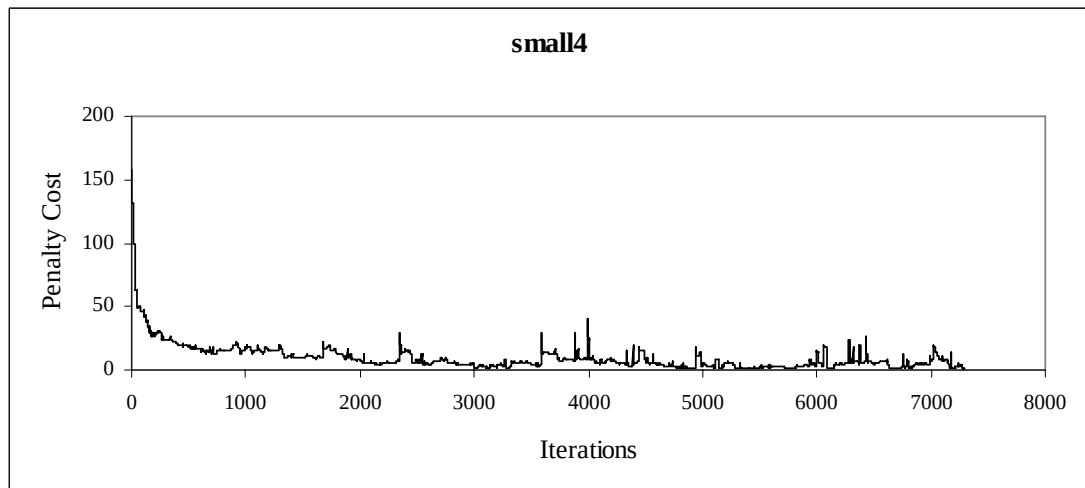
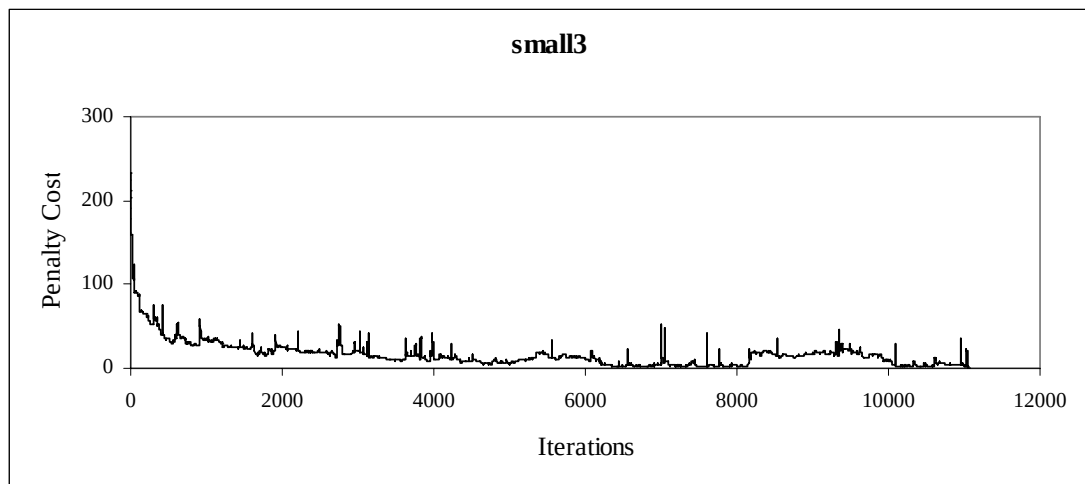


Figures 7.12. The behaviour of VNS-Tabu and VNS-EMC algorithms applied to the *large* dataset

Figures 7.11a-b and 7.12 show the behaviour of the VNS-EMC and VNS-Tabu algorithms when applied to the *medium3*, *medium5* and *large* datasets. They show that the penalty cost can be quickly reduced at the beginning of the search where there is possibly a lot of room for improvement. This becomes less pronounced towards the end of the search (as in Figures 7.5a-e). It can be seen from Figure 7.11a that VNS-Tabu is better than VNS-EMC throughout the search. However, in Figure 7.11b, VNS-EMC is slightly better than VNS-Tabu in the early part of the search. Note that, by prolonging the search process it can be seen that the pattern of the VNS-Tabu is similar for both the *medium3* and *medium5* datasets which shows that

VNS-Tabu is able to find a good solution if given more search time. Figure 7.12 shows the behaviour of VNS-Tabu applied to the *large* dataset. During the early part of the search, the VNS-EMC performs better than VNS-Tabu. However, towards the end of the search, even though the VNS-EMC does not make any improvement, VNS-Tabu is able to obtain better solutions. Prolonging the search helps the VNS-Tabu to generate more improvements. It is believed that with the help of the tabu list, the VNS-Tabu performs better than VNS-EMC and is able to find a better solution because the neighbourhood structures that are not used will not be employed in the next iteration (the algorithm will only be supplied with the neighbourhood structures that have been used recently) unlike the VNS-EMC approach. Figures 7.13a-e show the behaviour of VNS-Tabu applied to all the *small* datasets used in this experiment.

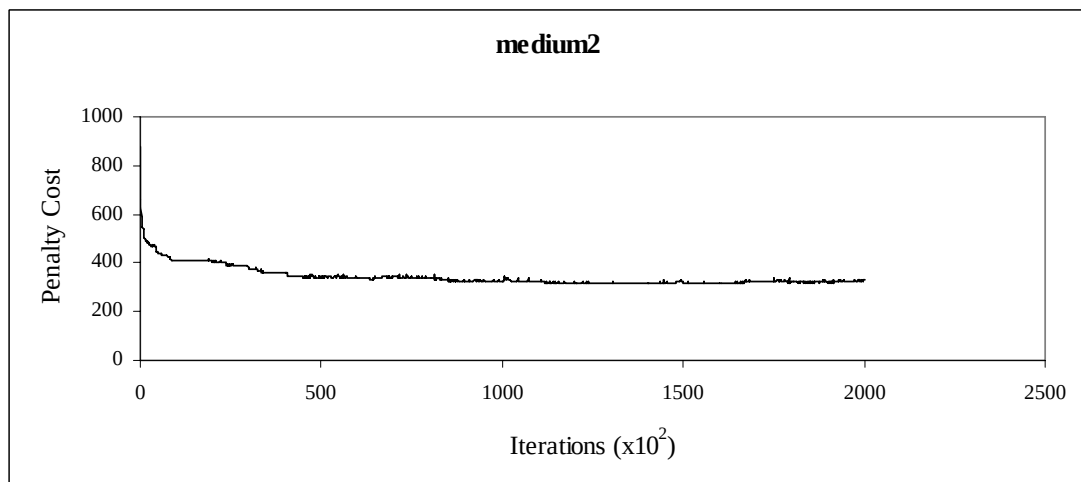
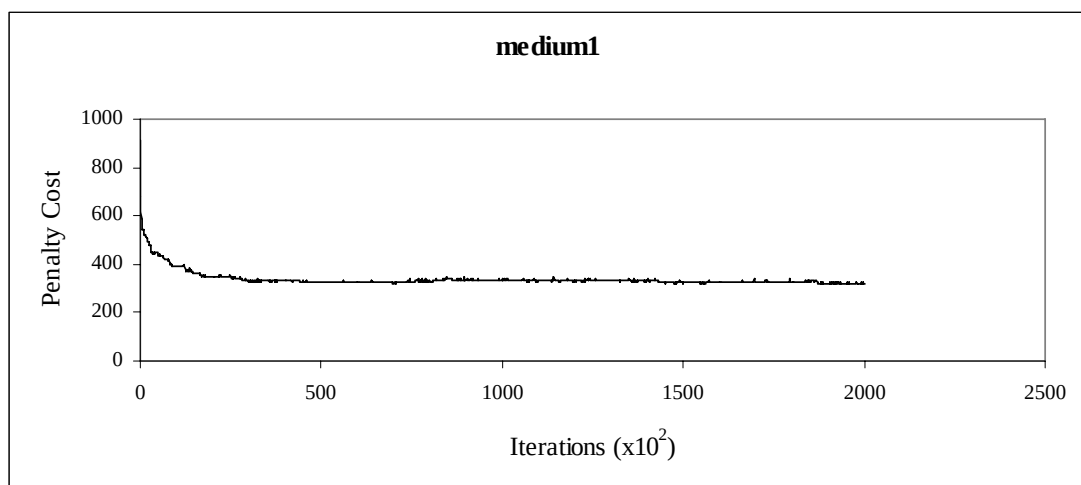


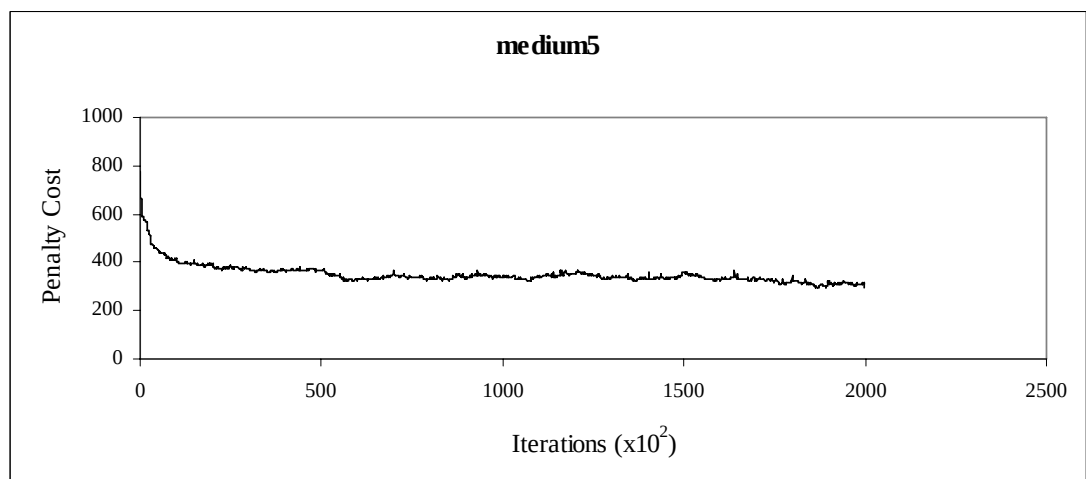
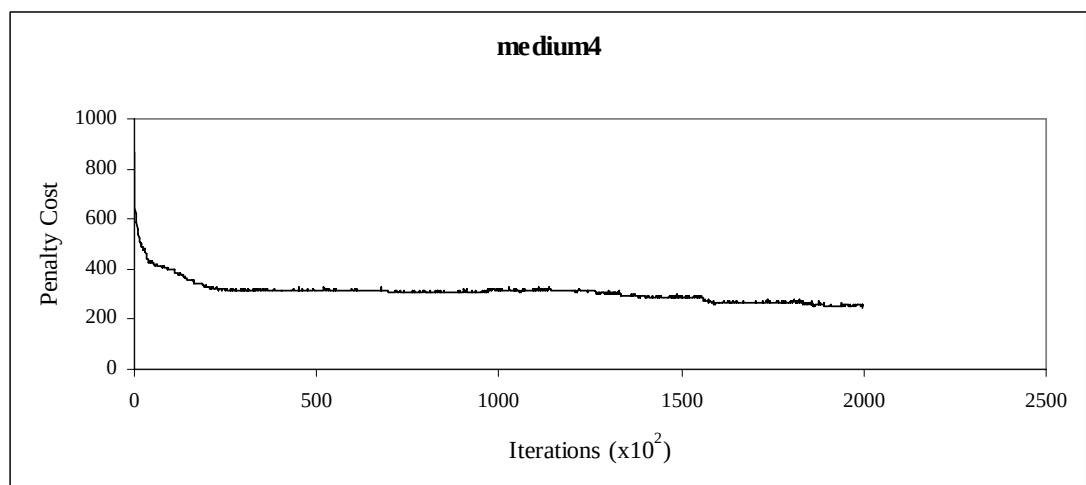
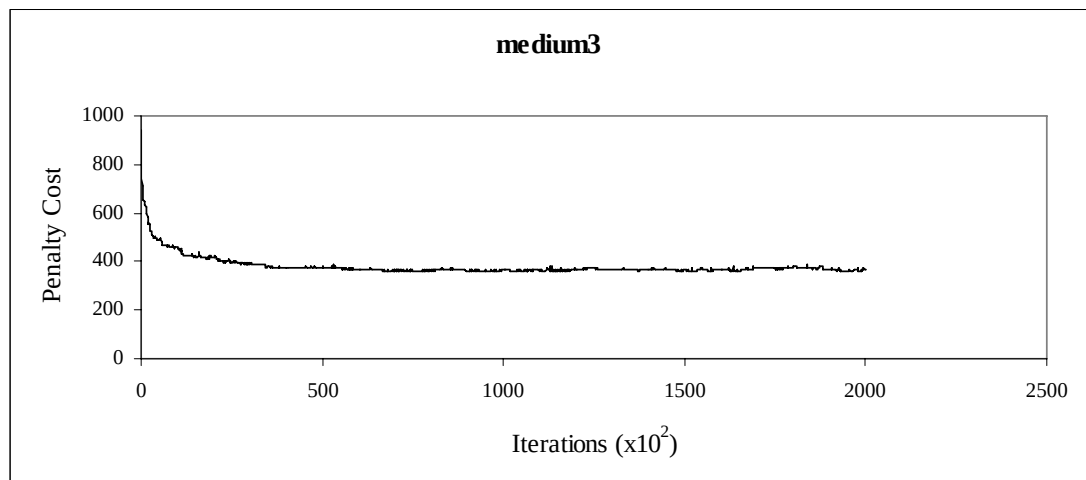


Figures 7.13a-e. The behaviour of the VNS-Tabu algorithm on the *small* datasets

Every point on a graph corresponds to the penalty cost and number of iterations. The shape of the graphs is similar for all the *small* datasets presented here which shows that the algorithm behaves in almost the same way for different problems.

Figures 7.14a-e and 7.15 show the behaviour of the VNS-Tabu algorithm applied to the *medium* and *large* datasets. All the graphs presented in Figures 7.14a-e also show a quite similar shape. This implies that the algorithm works similarly even though the complexity of the *medium* datasets is believed to be different.





Figures 7.14a-e. The behaviour of the VNS-Tabu algorithm on the *medium* datasets

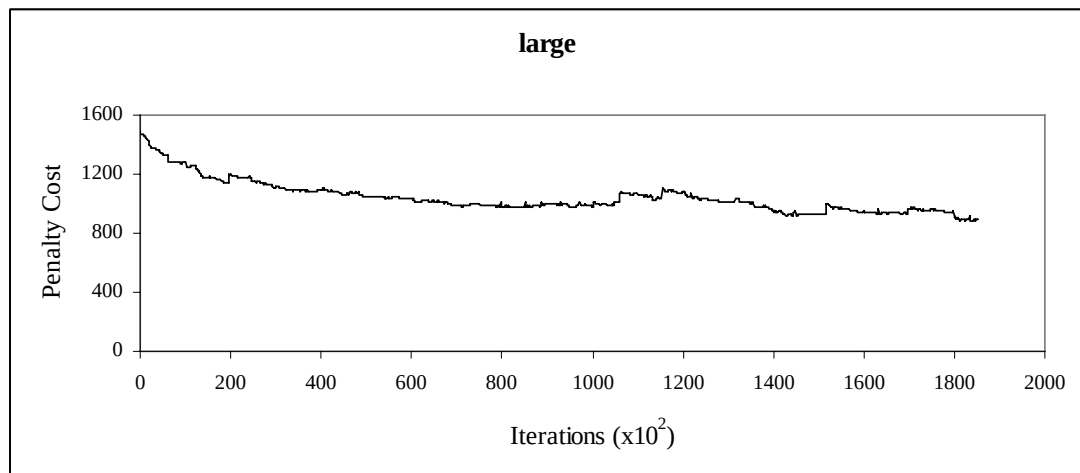


Figure 7.15. The behaviour of the VNS-Tabu algorithm on the *large* dataset

7.4 Summary

The overall goal of this chapter was to investigate a modified VNS approach for the course timetabling problem. The performance of a basic variable neighbourhood search which is referred to as VNS-Basic is compared to a modified VNS approach that uses an exponential monte carlo acceptance criterion (called VNS-EMC). Preliminary comparisons indicate that the modified VNS approach (VNS-EMC) is better than VNS-Basic and that it is competitive with other approaches in the literature applied to the same domain. The VNS-EMC approach produced solutions on five out of eleven benchmark problems that were better than or equal to those published results and, moreover, this approach is able to obtain zero penalty cost for all the *small* problems. A further investigation was carried out that incorporates the VNS-EMC with a tabu list which is referred to as VNS-Tabu. The experiments show that the VNS-Tabu performs equally well or better than VNS-EMC. The next chapter investigates local search using composite neighbourhood structures applied to the same domain.

Chapter 8

Using a Randomised Iterative Improvement Algorithm with Composite Neighbourhood Structures for University Course Timetabling

8.1 Introduction

A variable neighbourhood search is basically dealing with a number of neighbourhood structures which are ordered in a sequence. One of the key research issues is to make sure that there is an effective sequence. In Chapter 7, the sequence of neighbourhood structures is ordered by increasing size which is based on preliminary tests. If new neighbourhood structures are added, so several sequences should be tested before the right sequence is obtained. A local search algorithm (called a randomised iterative improvement algorithm) with a composite neighbourhood structure is implemented in this chapter in order to avoid a tuning process (with the aim of getting a good sequence of neighbourhood structures). This approach also offers more options to explore the search space in order to achieve a better solution. The approach is tested on the eleven course timetabling benchmark datasets that were used in Chapter 7. The results demonstrate that this approach is able to produce solutions that are better than others that appear in the literature.

This chapter is organised as follows: Section 8.2 presents a literature overview of composite neighbourhood structures. Section 8.3 illustrates the implementation of the randomised iterative improvement algorithm. The pseudo-code of the implemented algorithm is also presented in this section. Experiments and results to access the

performance of the heuristic are discussed in Section 8.4. Section 8.5 presents a summary of the chapter. The work presented in this chapter was presented at the 6th Meta-heuristic International Conference (MIC 2005) (Abdullah et al. 2005b).

8.2 Composite Neighbourhood Structures: A Literature Overview

A composite neighbourhood structure subsumes two or more neighbourhood structures. The advantage of combining several neighbourhood structures is that it helps to compensate for the insufficiency of using each type of neighbourhood structure in isolation (Grabowski and Pempera 2000 and Liaw 2003). For example, a solution space that is easily accessible by insertion moves may be difficult to reach using swap moves. Some examples of a composite neighbourhood structure from the literature applied to various domains are discussed below.

Grabowski and Pempera (2000) applied a composite neighbourhood structure to the sequencing of jobs in a production system that consists of exchanges and the insertion of elements. Gopalakrishnan et al. (2001) used three moves (swap, add and drop) in a tabu search heuristic for preventive maintenance scheduling. The decision on which move to use depends on the current state of the search. The interaction of the moves makes it possible to carry out a strategic search. The computational results show that the approach can improve the solution quality when compared to the local search heuristics presented by Gopalakrishnan et al. (1997).

Liaw (2003) also employed a composite neighbourhood structure in a tabu search approach for the two-machine preemptive open shop scheduling problem. The tabu search switches to the other neighbourhood structures (between an insertion move that moves one job from its current position to a new position and a swap move that exchanges the position of two jobs) after a number of iterations without any improvements. Computational experiments have shown that this scheme significantly improves the performance of tabu search in terms of solution quality. Ouelhadj (2003) employed a composite structure within the tabu search approach to the dynamic scheduling at a hot strip mill. The method employs three neighbourhood

schemes (swap, shift and inversion moves) alternately. Computational experiments showed that the composite structure improves the solution quality compared with tabu search using a single neighbourhood. Another example of composite neighbourhood structure was presented by Landa Silva (2003). He employed several neighbourhood structures (relocate, swap and interchange moves) in different meta-heuristics (iterative improvement, simulated annealing and tabu search) that were applied to a space allocation problem in an academic institution.

Bilge et al. (2004) used a “hybrid” neighbourhood structure in a tabu search algorithm for the parallel machine total tardiness problem. The “hybrid” structure consists of an “insert neighbourhood” with the addition of a “swap neighbourhood”. In an insert move operation, two jobs are identified and the first job is placed in the location that precedes the location of the second job. On the other hand, a swap move places each job in the location that was previously occupied by the other job.

8.3 The Randomised Iterative Improvement Algorithm

This algorithm always accepts an improved solution and a worse solution is accepted with a certain probability.

8.3.1 The Neighbourhood Structures

The different neighbourhood structures used in this algorithm can be outlined as follows:

- N1: Select a course at random and find another course at random with which to swap timeslots.
- N2: Choose a single course at random and move it to another random feasible timeslot.
- N3: Select two timeslots at random and simply swap all the courses in one timeslot with all the courses in the other timeslot.
- N4: Move timeslot (as discussed in Chapter 7).
- N5: Move the highest penalty course from a random 10% selection of the courses to a random feasible timeslot.
- N6: As N5 but with 20% of the courses.

- N7: Move the highest penalty course from a random 10% selection of the courses to a new feasible timeslot (that is chosen from all of the available timeslots) that can generate the lowest penalty cost.
- N8: As N7 but with 20% of the courses.
- N9: Select one course at random, select a timeslot at random (distinct from the one that was assigned to the selected course) and then apply the kempe chain from Thompson and Dowsland (1996a).
- N10: This is the same as N9 except that the highest penalty course from a 5% selection of the courses is chosen at random
- N11: As N10 but with 20% of the courses.

Note that most of the neighbourhood structures implemented in this chapter are similar to those used in Chapter 7. Three kempe chain move neighbourhoods (see Thompson and Dowsland 1996a) were added. The kempe chain neighbourhood involves swapping a subset of the courses in two distinct timeslots. Suppose we have timeslots t_i and t_j . A course e_1 in timeslot t_i and a new timeslot t_j are selected at random. Courses in timeslots t_i and t_j form a bipartite graph (by letting the edges denote conflicts between events). The kempe chain is defined from the initial chosen course by using the connected components of a bipartite graph. For example, consider the situation described in Figure 8.1. Here, each timeslot contains four courses.

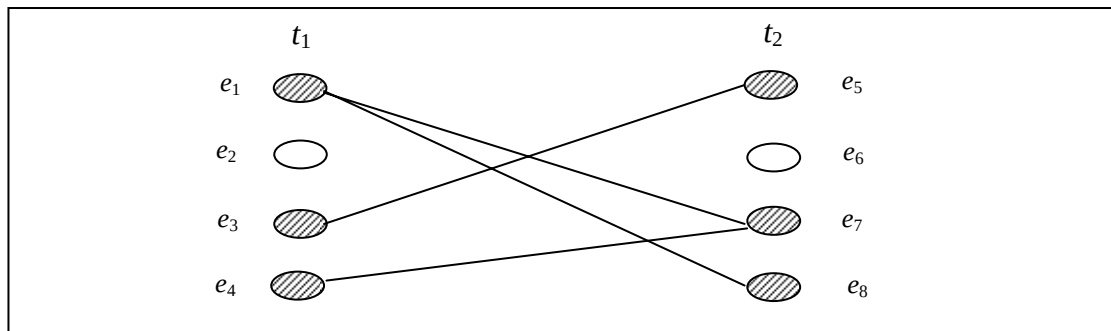


Figure 8.1. The earlier neighbourhood structure before the kempe chain move execution

All courses have clashes with other courses in the other timeslot except courses e_2 and e_6 . Assume that course e_1 is selected in the kempe chain neighbourhood to be moved from timeslot t_1 to timeslot t_2 . This means that courses e_7 and e_8 need to be

moved to timeslot t_1 and then course e_4 has to move across to timeslot t_2 to maintain feasibility. Figure 8.2 shows the result after the kempe chain operation has taken place. In order to maintain the feasibility of the solution, the room allocation of courses in each timeslot (after a kempe chain operation) cannot exceed the space available and, of course, the courses in each timeslot should be scheduled in different rooms.

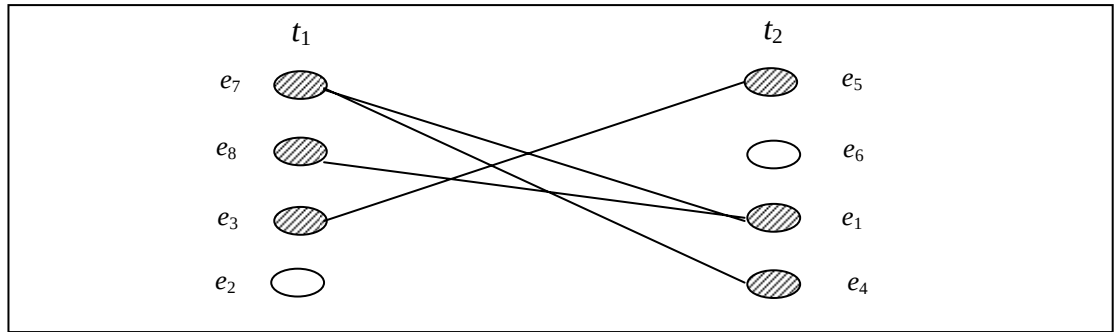


Figure 8.2. The result of a kempe chain move

3.3.2 The Algorithm

In the approach presented in this chapter, a set of neighbourhood structures such as those presented in Section 8.4.1 is applied. The hard constraints are never violated during the timetabling process. Figure 8.3 shows a schematic overview of the approach where the solution is represented as Sol and each neighbourhood structure is represented as $N1$, $N2$ and so on. The arrow shows the transition from a current solution to a new solution after a neighbourhood structure is employed.

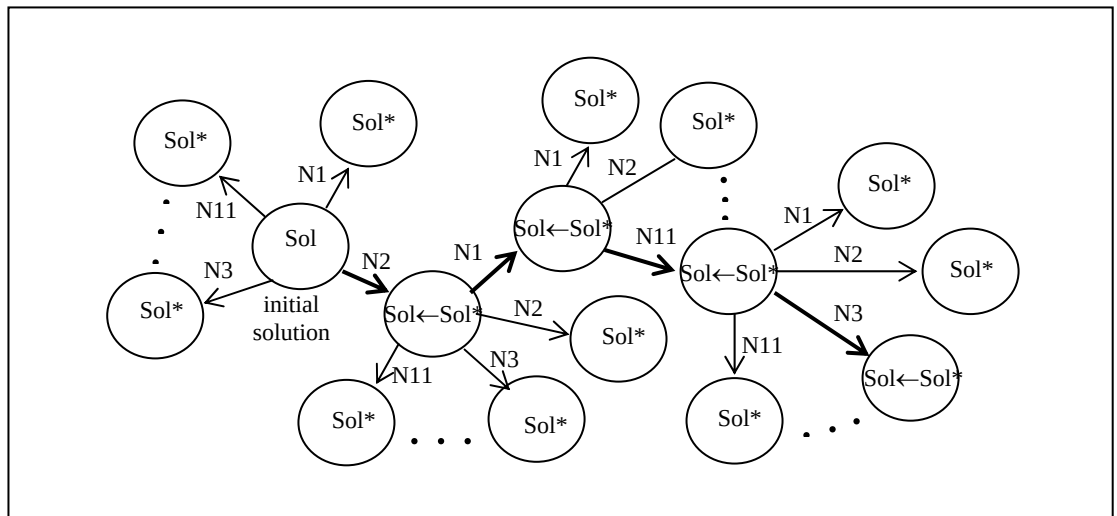


Figure 8.3. Schematic overview of the randomised iterative improvement algorithm

At the beginning of the search, a set of neighbourhood structures ($N_1, \dots, N_{11}$) is applied to the initial solution which is treated as a current solution, Sol . The new solutions obtained after employing the neighbourhood structures are represented as Sol^* . The best solution among Sol^* is selected to be a current solution. From Figure 8.3, the solution returned by the neighbourhood structure N_2 is selected. Again, a set of neighbourhood structures is applied to Sol and the solution returned by N_1 is chosen as a current solution. The process is repeated and stops when the termination criterion is met (in this work the termination criterion is set as the number of iterations i.e. 200,000 iterations). From Figure 8.3, it can be seen that a sequence of neighbourhood structures can be represented as $N_2 \rightarrow N_1 \rightarrow N_{11} \rightarrow N_3$. This sequence represents a composite neighbourhood structure being employed in the search process.

The pseudo-code for the algorithm implemented in this chapter is given in Figure 8.4.

```
Set the initial solution Sol by employing a constructive
heuristic;
Calculate initial cost for Sol, f(Sol);
Set best solution Solbest ← Sol;
do while (not termination criterion)
    for i = 1 to i= K where K is the total number of
    neighbourhood structures
        Apply neighbourhood structure i to Sol, TempSoli;
        Calculate cost for TempSoli, f(TempSoli);
    end for;
    Identify the best solution among all the TempSoli where i
    ∈ {1,...,K} call new solution Sol*;
    if (f(Sol*) < f(Solbest))
        Sol ← Sol*;
        Solbest ← Sol*;
    else
        Apply an exponential monte carlo where:
        δ = f(Sol*) - f(Sol);
        Generate RandNum, a random number in [0,1];
        if (RandNum < e-δ)
            Sol ← Sol*;
        end if
    end do
```

Figure 8.4. The pseudo-code for the randomised iterative improvement algorithm applied to the course timetabling problem

The algorithm starts with a feasible initial solution which is generated by a constructive heuristic as discussed in Chapter 7. Let K be the total number of neighbourhood structures to be used in the search (K is set to be 11 in this implementation) and $f(Sol)$ is the quality measure of the solution Sol . At the start, the best solution, Sol_{best} is set to be Sol . In a *do-while* loop each neighbourhood i where $i \in \{1, \dots, K\}$ is applied to Sol to obtain $TempSol_i$. The best solution among all the $TempSol_i$ is identified, and is set to be the new solution Sol^* . If Sol^* is better than the best solution in hand Sol_{best} , then Sol^* is accepted. Otherwise the exponential monte carlo acceptance criterion is applied. It accepts a worse solution with a certain probability. The exponential monte carlo acceptance criterion is discussed in Chapter 4.

8.4 Experiments and Results

The proposed method was tested on the benchmark course timetabling problems presented by Socha et al. (2002). The parameters for the experimental setup (i.e. the number of runs and the number of iterations) for these approaches are the same as in Chapter 7. The algorithm will terminate if the penalty cost is zero or the number of iterations is equal to 200,000. The best results out of 5 runs are presented. Again, the term “x%Inf” in Table 8.1 indicates a percentage of runs that failed to obtain feasible solutions. Table 8.1 illustrates a comparison of the approach in this chapter with other approaches in the literature.

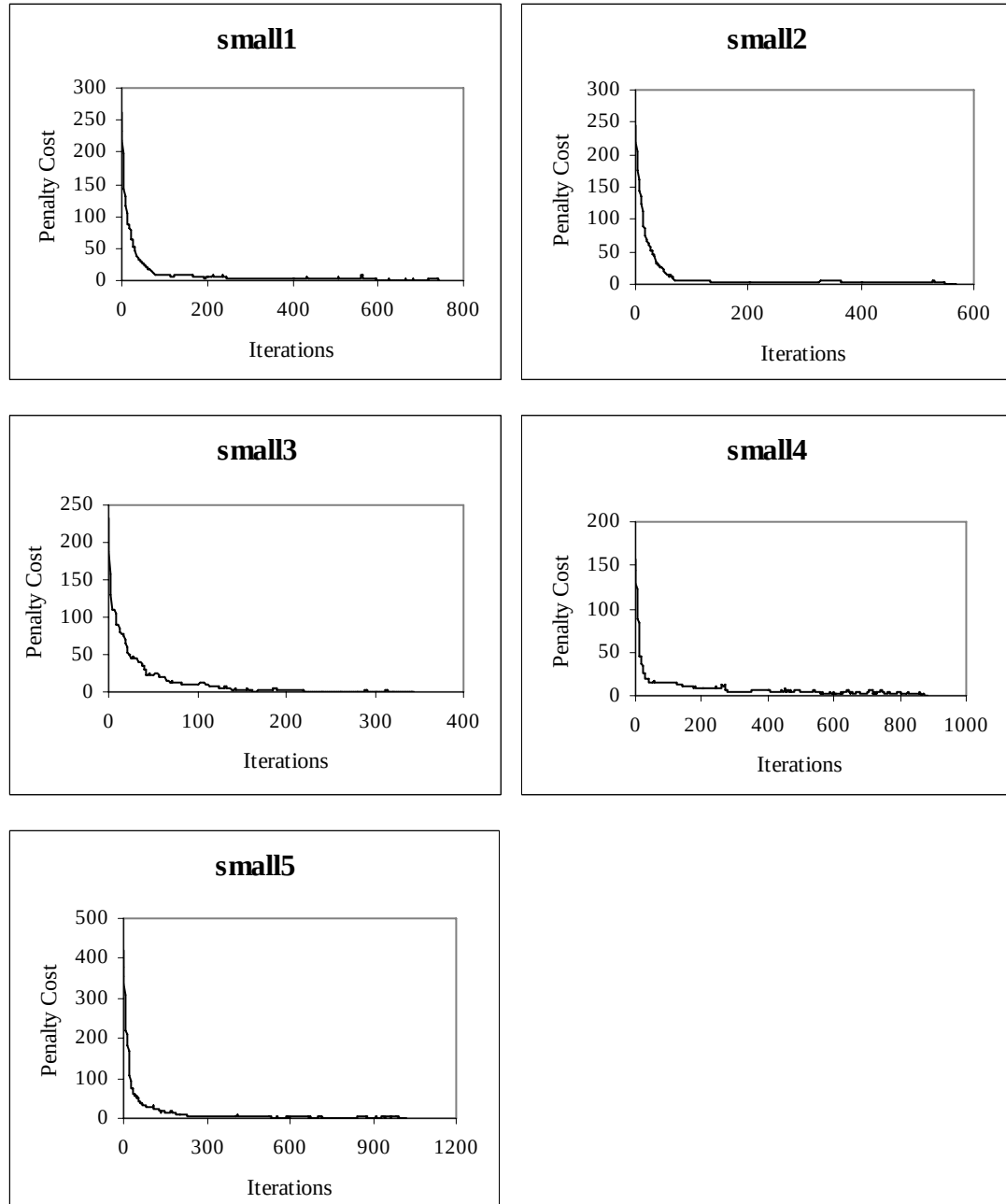
The best results are presented in bold. Note that the same initial solutions as in Chapter 7 (Abdullah et al. 2005a) are used. A different initialisation from Chapter 9 for the *large* dataset is used. It can be seen that the randomised iterative improvement algorithm is better than Abdullah et al. (2005a) on five datasets (same penalty cost for all the *small* datasets). Note that this approach is very effective on the *small* and *large* problems and quite effective on the *medium* problems. This is probably due to some neighbourhood structures being less effective for these types of problems. From this observation, it could be argued that the algorithm should intelligently select the most suitable neighbourhood structures according to the characteristics of the problems. This could be the subject of future work (see Chapter 10). It is

interesting to compare the current results with the result obtained from Chapter 7 (Abdullah et al. 2005a). We can see that the randomised iterative improvement algorithm is able to outperform the algorithm in Chapter 7. It is believed that there are more possibilities to explore the search space using the randomised iterative improvement algorithm because each neighbourhood structure is likely to explore the search space more widely from the current solution. A defined sequence of neighbourhood structures obtained from preliminary tests in Chapter 7 (Abdullah et al. 2005a) may not be the best sequence. It is clear that the results obtained in Chapter 7 (Abdullah et al. 2005a) are not as good as the results obtained in this chapter (Abdullah et al. 2005b) especially on the *medium* and *large* datasets.

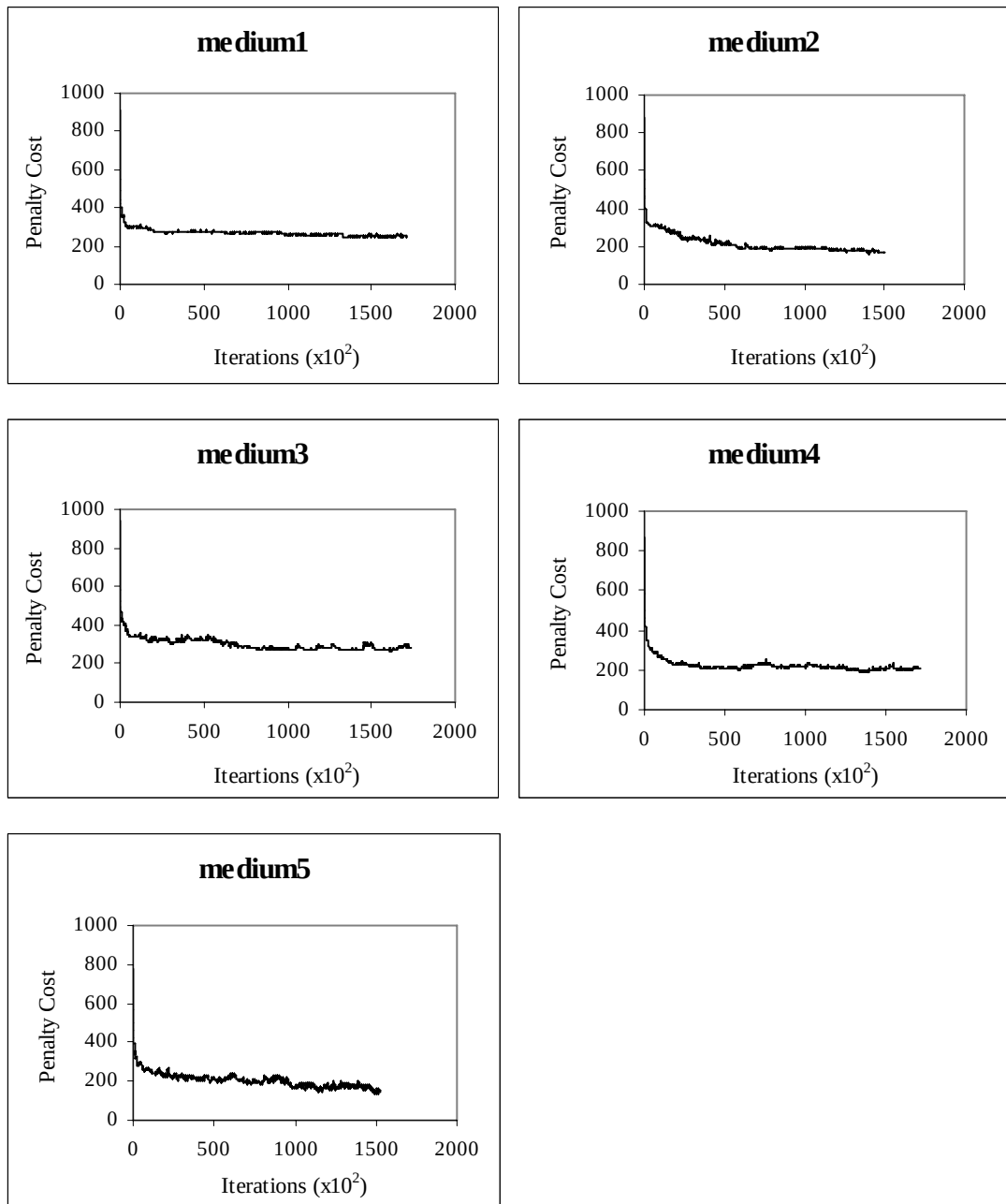
Table 8.1. Comparison results on the course timetabling problem using the randomised iterative improvement algorithm

Datasets	Initial solution	Randomised iterative improvement algorithm		Abdullah et al. 2005a (Best)	Socha et al. (2002) (Average)	Socha et al. (2002) (Average)	Burke et al. (2003a) (Best)	Burke et al. (2007) (Best)	Asmuni et al. (2005b) (Best)
		Best	Average						
<i>small1</i>	261	0	0	0	8	1	1	6	10
<i>small2</i>	245	0	0	0	11	3	2	7	9
<i>small3</i>	232	0	0	0	8	1	0	3	7
<i>small4</i>	158	0	0	0	7	1	1	3	17
<i>small5</i>	421	0	0	0	5	0	0	4	7
<i>medium1</i>	914	242	245	317	199	195	146	372	243
<i>medium2</i>	878	161	162.6	313	202.5	184	173	419	325
<i>medium3</i>	941	265	267.8	357	77.5% Inf	248	267	359	249
<i>medium4</i>	865	181	183.6	247	177.5	164.5	169	348	285
<i>medium5</i>	780	151	152.6	292	100% Inf	219.5	303	171	132
<i>large</i>	1603	757	779.8	935	100% Inf	851.5	80% Inf 1166	1068	1138

Figures 8.5a-e, 8.6a-e and 8.7 show the behaviour of the randomised iterative improvement algorithm applied to the *small*, *medium* and *large* datasets, respectively.



Figures 8.5a-e. The behaviour of the randomised iterative improvement algorithm on the *small* datasets



Figures 8.6a-e. The behaviour of the randomised iterative improvement algorithm on the *medium* datasets

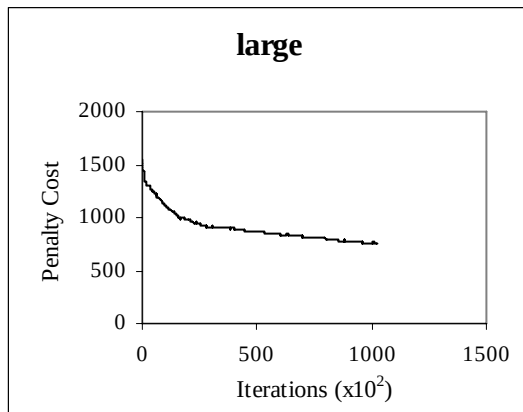


Figure 8.7. The behaviour of the randomised iterative improvement algorithm on the *large* dataset

In all the figures, the x-axis represents the number of iterations while the y-axis represents the penalty cost. The graphs illustrate the exploration of the search space. The curves move up and down because worse solutions are accepted with a certain probability in order to escape from local optima. The penalty cost can be quickly reduced at the beginning of the search where there is (possibly) a lot of room for improvement. It is believed that better solutions can be obtained in these experiments (particularly for the smaller problems) because the composite neighbourhood structures offer some flexibility for the search algorithm to explore different regions of the solution space. The graphs for the *small* datasets show that our algorithm is able to obtain zero penalty using fewer than 1500 iterations.

Figures 8.8, 8.9 and 8.10 show the frequency chart of the neighbourhood structures that have been selected to be used by the randomised iterative improvement algorithm for the *small*, *medium* and *large* datasets, respectively where the x-axis represents the datasets (except for the *large* dataset where the x-axis represents the neighbourhood structures) while the y-axis represents the frequency of the neighbourhood structures being employed throughout the search.

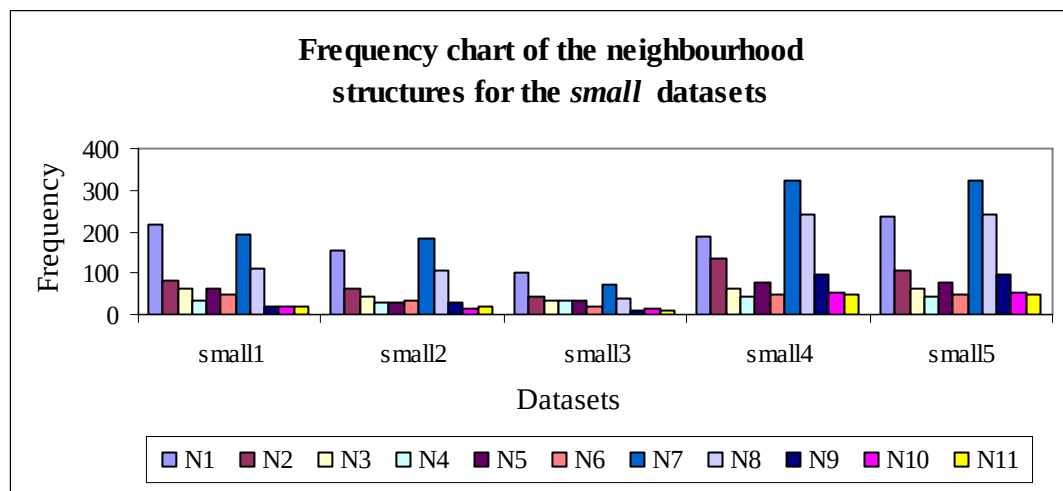


Figure 8.8. The neighbourhood structures used for the *small* datasets

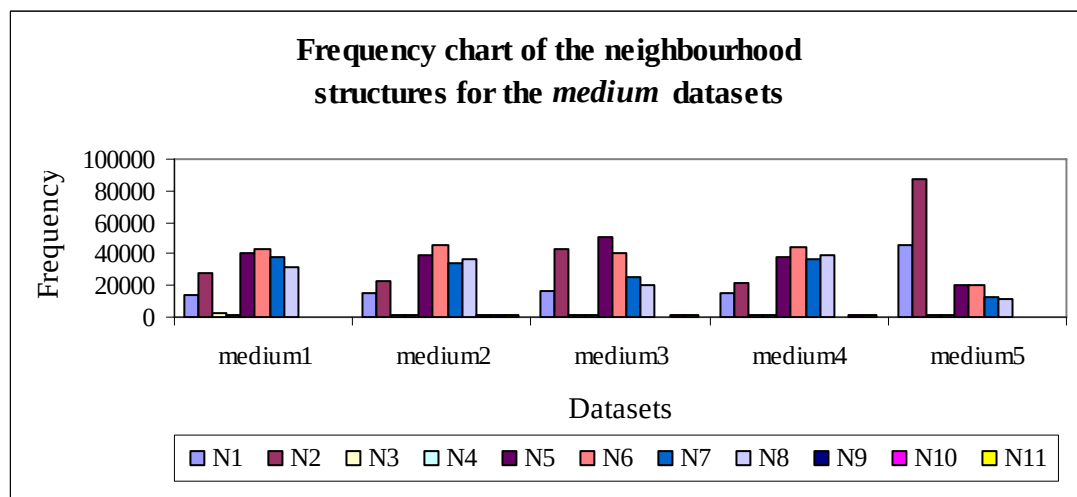


Figure 8.9. The neighbourhood structures used for the *medium* datasets

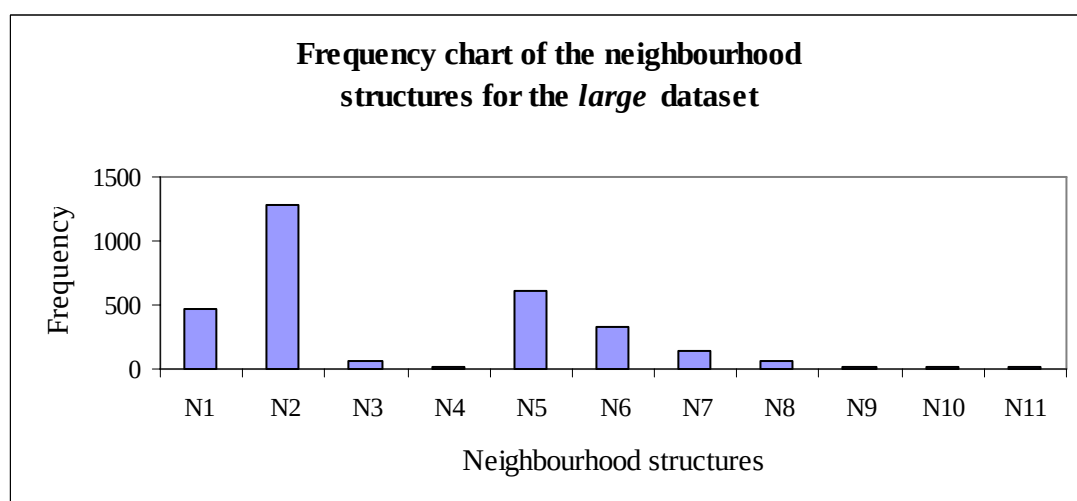


Figure 8.10. The neighbourhood structures used for the *large* dataset

It can be seen from Figure 8.7 that the neighbourhood structures N1, N2, N7 and N8 are the most effective neighbourhood structures in the randomised iterative improvement algorithm for the *small* datasets. The effective neighbourhood structures for the *medium* and *large* datasets are N1, N2, N5, N6, N7 and N8 as shown in Figures 8.9 and 8.10. This demonstrates that the set of most effective neighbourhood structures that is being supplied to the randomised iterative improvement algorithm is similar for all the datasets (i.e. N1, N2, N7 and N8). However, as the problem gets larger, there might be fewer and more sparsely distributed solution points (feasible solutions) in the solution space since many courses are conflicting with each other. There might be a need for extra neighbourhood structures (i.e. N5 and N6 in this case) to force the search algorithm to diversify its exploration of the solution space by moving from one neighbourhood structure to another.

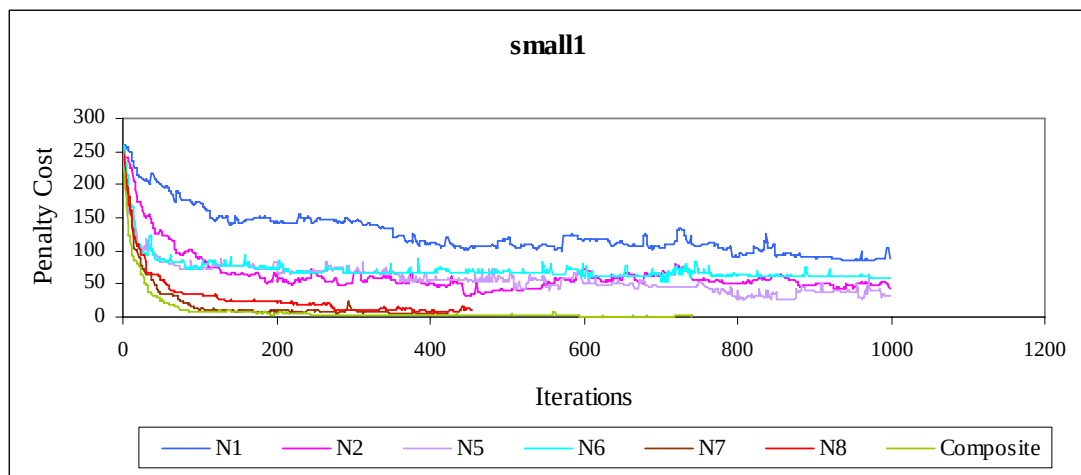
Further investigation has been carried out to check the claim that the composite neighbourhood structure performs better than the single neighbourhood structure by employing the selected neighbourhood structures separately i.e. N1, N2, N5, N6, N7 and N8 (which are the most effective neighbourhood structures used for the *small*, *medium* and *large* datasets). The diagrams in Figures 8.5a-e show that the method on the *small* datasets is able to obtain zero penalties in fewer than 1500 iterations. For the experiments that have been carried out, the number of iterations for the *small* datasets is set to be the number of iterations where the best solutions are obtained (i.e. 873, 707, 413, 1012 and 1329 iterations for *small1*, *small2*, *small3*, *small4* and *small5*, respectively). The number of iterations used for the *medium* and *large* datasets remain the same. Table 8.2 gives the comparison of the performance of variants of the randomised iterative improvement algorithm in terms of the penalty cost (objective function value). The results demonstrate that the algorithm with the composite neighbourhood structures uniformly performs the best in terms of penalty cost than single neighbourhood randomised iterative improvement algorithm variants. Note that for the *small* datasets, some single neighbourhood structures (i.e. N7 and N8) are believed to be able to produce a zero penalty solution if the search is prolonged. However, they are unable to obtain a better solution quality when the

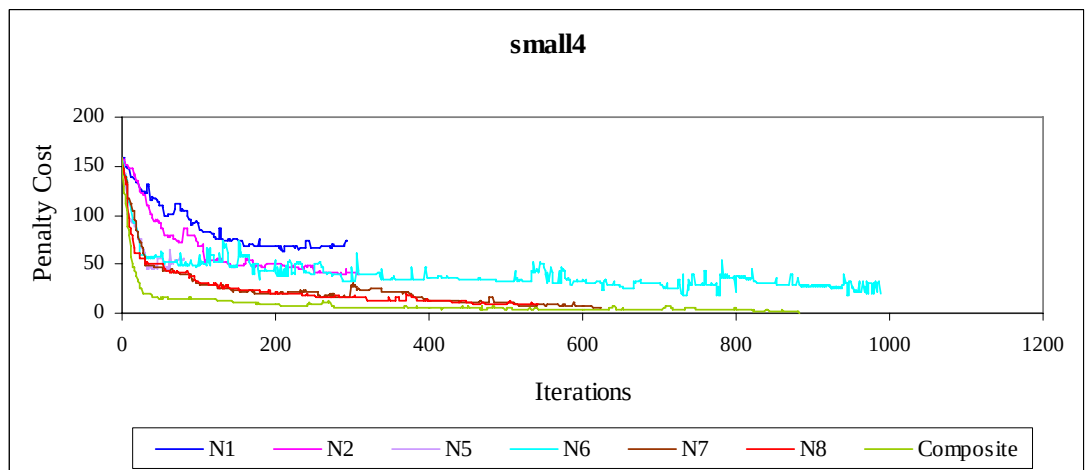
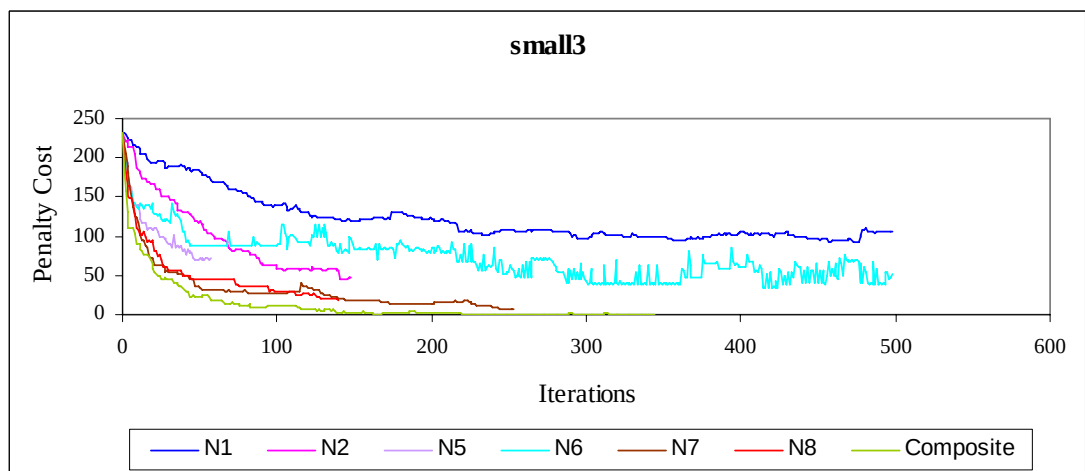
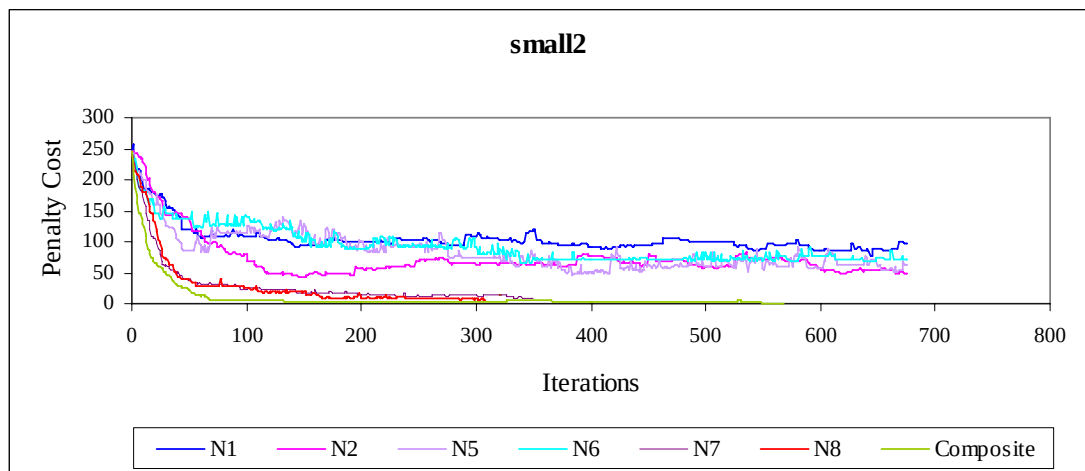
problem becomes larger and more complex (for example on the *medium* and *large* datasets).

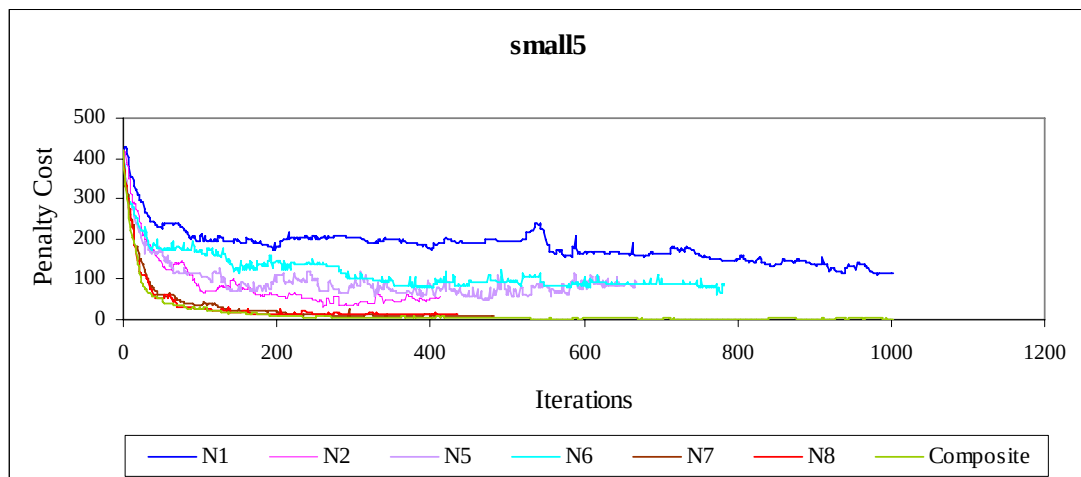
Table 8.2. Comparison of the performance of the randomised iterative improvement algorithm on single and composite neighbourhood structures

Datasets	Initial solution	Randomised iterative improvement algorithm with variant neighbourhood structures						
		N1	N2	N5	N6	N7	N8	Composite
<i>small1</i>	261	76	21	26	54	5	8	0
<i>small2</i>	245	64	27	47	59	9	6	0
<i>small3</i>	232	68	45	69	33	6	18	0
<i>small4</i>	158	63	39	44	18	5	9	0
<i>small5</i>	421	112	33	49	64	7	12	0
<i>medium1</i>	914	381	345	548	713	539	701	242
<i>medium2</i>	878	364	337	556	675	555	643	161
<i>medium3</i>	941	420	401	731	773	764	774	265
<i>medium4</i>	865	332	317	549	615	546	603	181
<i>medium5</i>	780	414	355	650	685	702	699	151
<i>large</i>	1603	1064	1054	1435	1495	1435	1485	757

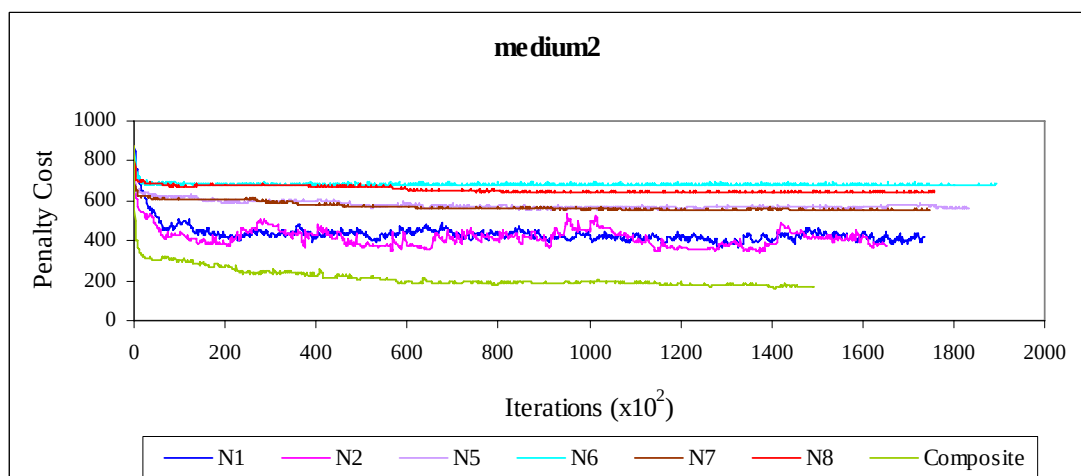
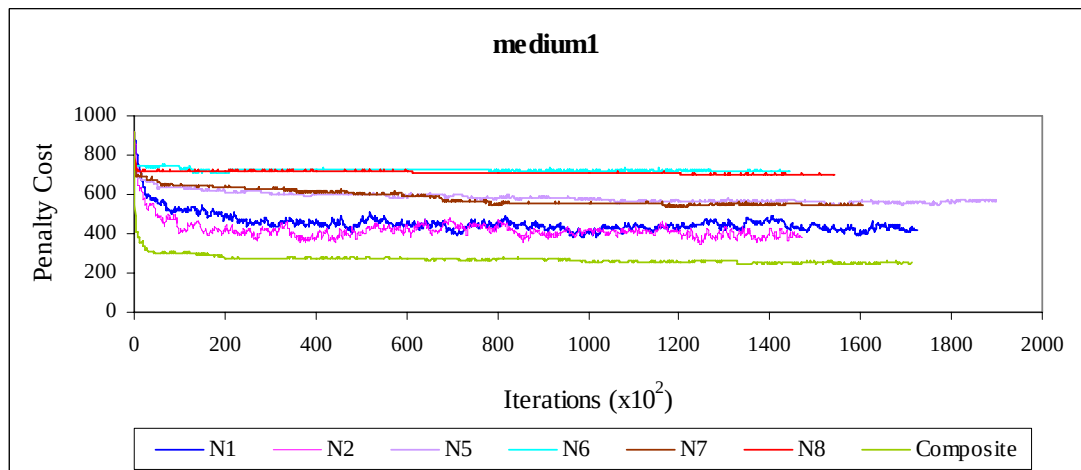
Figures 8.11a-e, 8.12a-e and 8.13 illustrate the behaviour of the randomised iterative improvement algorithm using single and composite neighbourhood structures applied to the *small*, *medium* and *large* datasets.

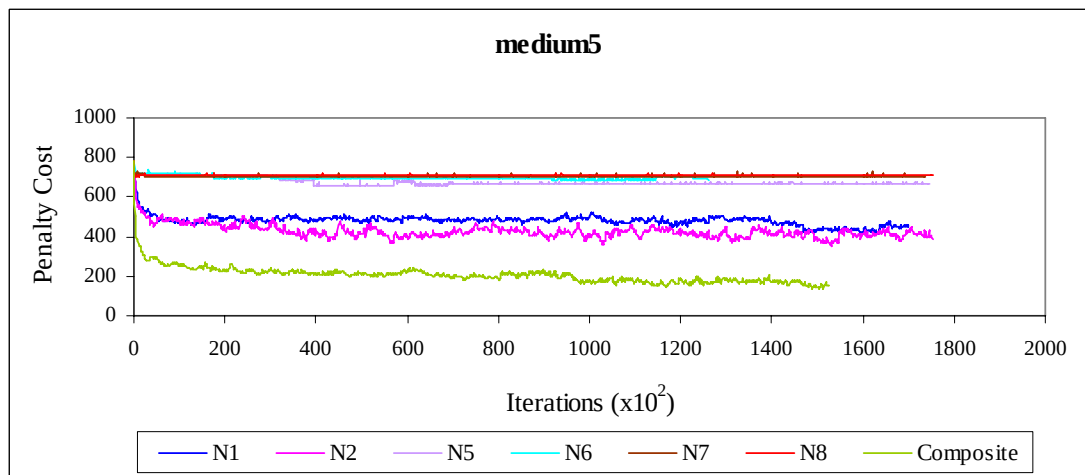
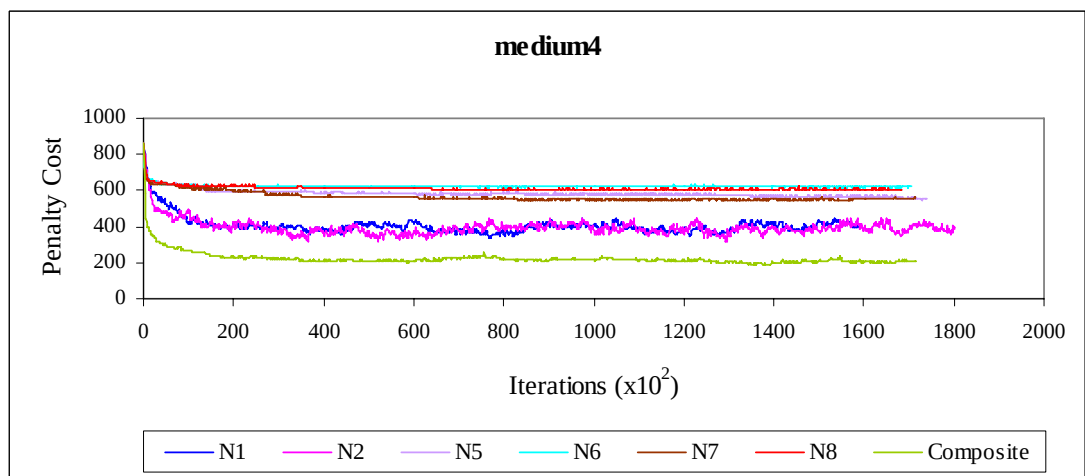
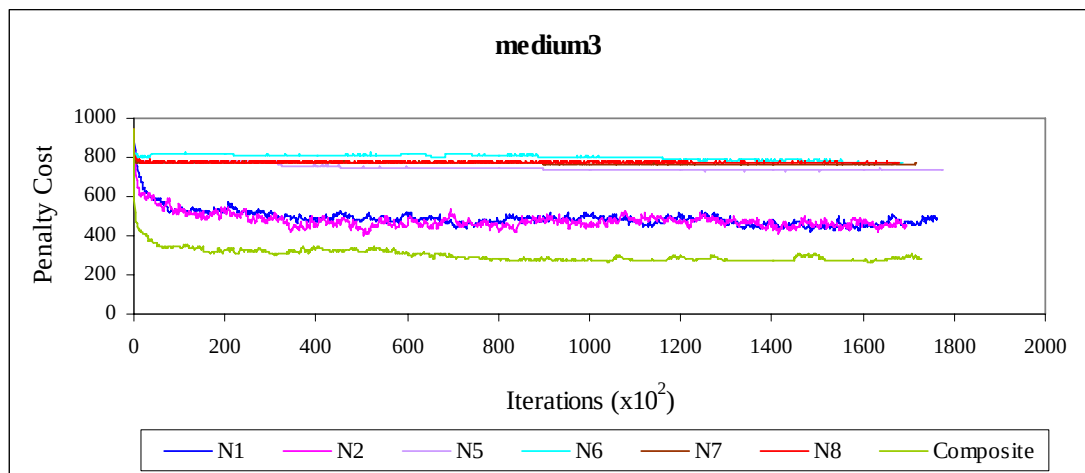






Figures 8.11a-e. The behaviour of the randomised iterative improvement algorithm using single and composite neighbourhood structures applied to the *small* datasets





Figures 8.12a-e. The behaviour of the randomised iterative improvement algorithm using single and composite neighbourhood structures applied to the *medium* datasets

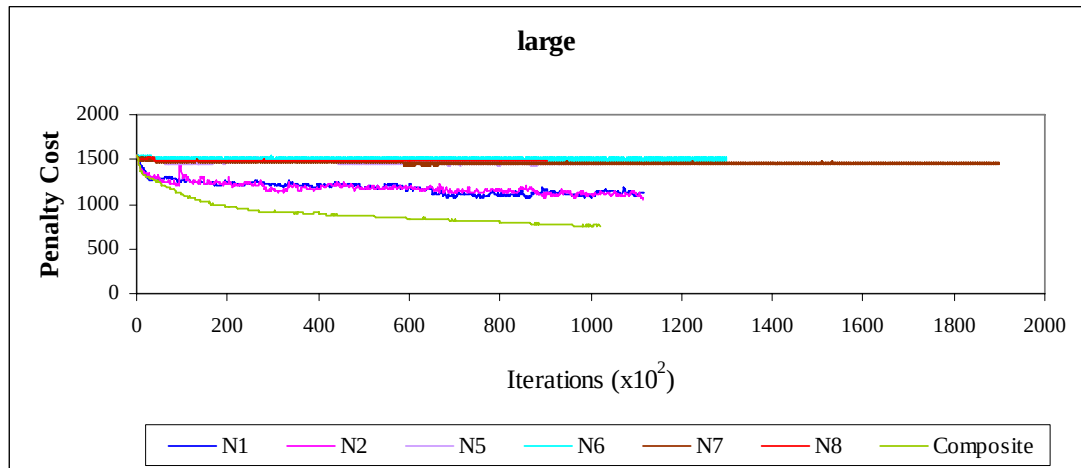


Figure 8.13. The behaviour of the randomised iterative improvement algorithm using single and composite neighbourhood structures applied to the *large* dataset

The diagrams show the convergence of the penalty cost of the randomised iterative improvement algorithm for the number of iterations required to obtain the best solutions reported in Table 8.1. It can be seen that the randomised iterative improvement algorithm with the composite neighbourhood is significantly better than other variants with a single neighbourhood in terms of solution quality given the same number of iterations.

8.5 Summary

This chapter has focused on investigating a composite neighbourhood structure with a randomised iterative improvement algorithm for the course timetabling problem. Preliminary comparisons indicate that the randomised iterative improvement algorithm is competitive with other approaches in the literature. It produces seven solutions that were better than or equal to the best results published. It is an approach that is particularly effective on smaller problems. Further experiments showed that the randomised iterative improvement algorithm with composite neighbourhood structures outperforms the single neighbourhood structure. This is believed to be the case because of the different possible ways to explore the search space from the current solution. The next chapter presents the implementation of a population-based approach with the local search method in this chapter on the course timetabling problem.

Chapter 9

A Memetic Approach for University Course Timetabling

9.1 Introduction

Memetic algorithms have proved to be powerful methodologies for solving combinatorial optimisation problems. This has particularly been the case for university timetabling problems (Burke et al. 1996b, Peachter et al. 1996). To our knowledge, memetic algorithms have not been applied to the course timetabling instances that are tackled in this thesis. This chapter aims to model a memetic algorithm and investigate how such an algorithm can improve upon the solution quality obtained by state of the art approaches from the literature.

The chapter is organised as follows: Section 9.2 describes the implementation of a memetic algorithm on the university course timetabling problem. Experimental results are presented in Section 9.3 and compared with state of the art techniques from the literature. The summary of the chapter is presented in Section 9.4.

9.2 Memetic Algorithm for the Course Timetabling

Problem

The algorithm presented here deals with a fixed length timetable. The main technique used in the algorithm is a light mutation operator followed by a randomised iterative improvement algorithm. The crossover operator is not employed in our memetic approach since our randomised iterative improvement algorithm (acting as a local search) only deals with feasible moves. The employment of a crossover operator may yield infeasible timetables that subsequently need repair mechanisms to correct infeasibilities (such as having to repair “offspring” in timetabling) (see Burke et al 1995a, Burke et al 1995b, Burke et al 1996b, Newall 1999).

9.2.1 Solution Representation

A direct representation is used for the solution. Each solution in the population is represented as a number of memes that contain information about the timeslot and room for a particular course. Figure 9.1 shows an example of the memes (courses) where e_i is course number i , $i \in \{1, \dots, N\}$ where N is a maximum number of courses. For example course e_1 is scheduled at timeslot 5 in room 2.

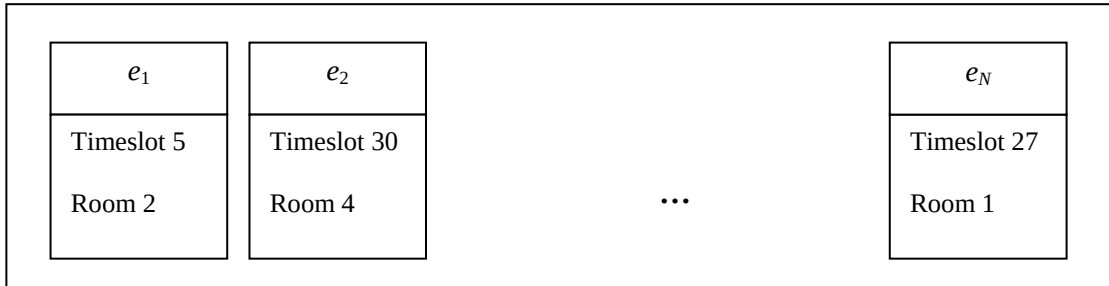


Figure 9.1. Solution representation

9.2.2 Initial Population Generation

The algorithm outlined in Figure 9.2 is used to generate large populations of random feasible timetables. The approach, which starts with an empty timetable, is similar to a random graph colouring method. A feasible solution is obtained by adding or removing appropriate events (courses) from the schedule until the hard constraints are met.

```
Generate a random ordering of courses;
  for all course
    for all available room
      for all available timeslot
        Assign room and timeslot
        if no clashes
          exit;
        end for
      end for
    end for;
  end for;
```

Figure 9.2. The pseudo-code to generate populations

9.2.3 The Evolutionary Operator: Mutation

A random mutation operator is used to execute a light mutation on 20% of the memes (courses) from 20% of the selected individuals. Courses are chosen at random from any point in the timetable and are reallocated to the earliest (after that point) feasible timeslots. The percentages of the individuals and courses were chosen based on our preliminary tests. The pseudo-code for the mutation operation is shown in Figure 9.3.

```
for 1 to 20% of the individuals
  for 1 to 20% of the courses of each individual
    Choose a course at random
    Allocate a course to a feasible timeslot
    according to the earliest possible timeslot
    and room capacity.
  end for
end for
```

Figure 9.3. The pseudo-code for the mutation operation

9.2.4 Selection

In order to maintain the population size after the application of the evolutionary operator, a roulette wheel selection is used. Each individual is allocated a segment that is proportional to its objective (fitness) value. The wheel is spun a number of times (which is equal to the population size) to select individual member of the new population. Based on our preliminary tests, the size of the population is set to 100.

9.2.5 The Local Search: Randomised Iterative Improvement Algorithm

The same algorithm as discussed in Chapter 8 is used as a local search (that is applied after the mutation takes place) in the memetic algorithm presented here.

9.2.6 The Algorithm

The schematic overview and the pseudo-code of the algorithm are presented in Figures 9.4 and 9.5, respectively. Combined, these figures represent the memetic approach used in our experiments. The algorithm begins by creating an initial population of size 100. The process creates subsequent generations by firstly selecting 20% of individuals from previous populations. Secondly, 20% of the courses from each selected individual are chosen at random to be mutated. The local search component is then employed. The best result obtained after applying this local search is kept. The solutions obtained after performing the local search are saved in the population pool and may be selected and used in the next generation where a roulette wheel is applied to select the new individual member for the next generation. The process is terminated after a predefined number of generations is reached (for the purpose of the experimentation this number was set to 30) or a timetable with a zero penalty cost is found.

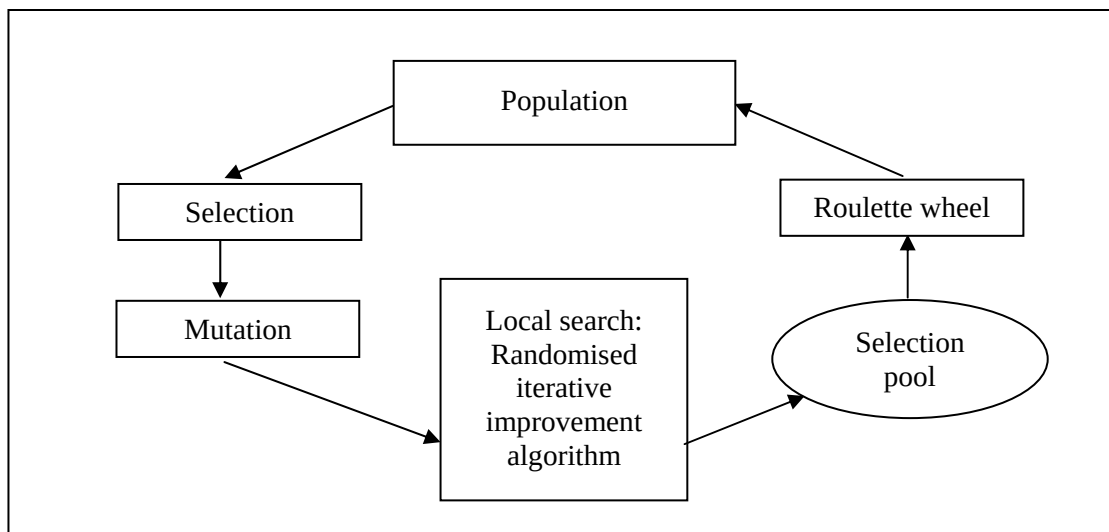


Figure 9.4. A schematic overview of the memetic algorithm applied to the course timetabling problem

```
begin
    Create population;
    do while (maximum number of generations or zero penalty
timetable is not met)
        Select individual to be mutated;
        Apply mutation;
        Apply local search;
        Select individual members of new population;
    end do;
end;
```

Figure 9.5. The pseudo-code of the memetic algorithm applied to the course timetabling problem

9.3 Experiments and Results

The algorithm has been tested on a range of standard benchmark instances as carried out in Chapters 7 and 8 (presented in Table 3.11). Experiments were undertaken to compare the effectiveness of the combination of local search with the mutation operator. The best results out of 5 runs are presented. The comparison of this algorithm with other available approaches in the literature is shown in Table 9.1. Again, the term “x%Inf” in Table 9.1 indicates a percentage of runs that failed to obtain feasible solutions.

The best results are presented in bold. It is particularly interesting to compare the results obtained here with our results from Chapter 8 (Abdullah et al. 2005b). The memetic algorithm obtained better results on *large* and all the *medium* datasets, and ties on the *small* datasets. This shows that the hybridisation between local search (that acts as an intensification strategy) and genetic operators (where the mutation operator is believed to act as a diversification strategy that helps to direct the search to different regions of the solution space) is able to produce results which are better than our previous work in Chapter 8 (Abdullah et al. 2005b) especially on the *medium* and *large* datasets. The percentage improvement obtained by applying the memetic algorithm compared with the randomised iterative improvement algorithm alone (in Chapter 8) for the *medium* dataset can be computed as

$$\text{Percentage improvement} = (\text{best randomised iterative improvement algorithm} - \text{best memetic algorithm}) * 100 / \text{best randomised iterative improvement algorithm}$$

The memetic algorithm managed to reduce the penalty cost between 7.2% and 13.9% and is able to repair the results especially for all the *medium* datasets for which two new best known results are obtained i.e. on *medium2* and *medium5*. Note that Asmuni et al. (2005b) reported the best results for the *medium5* dataset which are slightly higher than the best results obtained with the memetic algorithm. The experimental results also show that the approach works well across all datasets and is able to produce some of the best known results.

9.5 Summary

The performance of the memetic algorithm was measured on the basis of standard benchmark problems. The memetic algorithm produces some of the best results in the literature on the university course timetabling problem. By considering the hybridisation of local search and light mutation, the search can be guided towards a strong part of the solution space and hence good quality timetables can be produced. The next chapter presents some overall concluding comments about the work carried out in this thesis and it briefly highlights some possibilities for future enhancement.

Table 9.1. Comparison results on the course timetabling problem using a memetic algorithm

Datasets	Memetic algorithm		Abdullah et al. (2005b)		Abdullah et al. (2005b) (Best)	Socha et al. (2002) (Average)	Socha et al. (2002) (Average)	Burke et al. (2003a) (Best)	Burke et al. (2007) (Best)	Asmuni et al. (2005b) (Best)
	Best	Average	Best	Average						
<i>small1</i>	0	0	0	0	0	8	1	1	6	10
<i>small2</i>	0	0	0	0	0	11	3	2	7	9
<i>small3</i>	0	0	0	0	0	8	1	0	3	7
<i>small4</i>	0	0	0	0	0	7	1	1	3	17
<i>small5</i>	0	0	0	0	0	5	0	0	4	7
<i>medium1</i>	221	224.8	242	245	317	199	195	146	372	243
<i>medium2</i>	147	150.6	161	162.6	313	202.5	184	173	419	325
<i>medium3</i>	246	252	265	267.8	357	77.5% Inf	248	267	359	249
<i>medium4</i>	165	167.8	181	183.6	247	177.5	164.5	169	348	285
<i>medium5</i>	130	135.4	151	152.6	292	100% Inf	219.5	303	171	132
<i>large</i>	529	552.4	757	779.8	932	100% Inf	851.5	80% Inf 1166	1068	1138

Chapter 10

Conclusions and Future Work

This chapter sums up the major developments represented by the work reported in this thesis. Section 10.1 gives a summary of the research work that has been carried out. The scientific contributions are highlighted in Section 10.2. Section 10.3 outlines further research directions that may be undertaken. Finally, Section 10.4 lists the resulting papers that have been published and submitted for journals and conference proceedings.

10.1 Research Work Summary

The overall aim of the research presented in this thesis was to investigate how large neighbourhood search and meta-heuristic approaches could improve the state of the art of automated search methodologies for university examination and course timetabling problems. Chapter 1 motivated the work developed in this thesis by introducing the need for the development of new techniques to try to enhance the solution quality of university timetabling problems. A thorough literature review was carried out in Chapter 2 which describes the timetabling problems and methods used in solving both university examination and course timetabling problems. A detailed specification and datasets of the university timetabling problems that are used to test the performance of the heuristic approaches developed in this thesis was presented in Chapter 3.

The investigation of very large scale neighbourhood search (which is based on graph theoretical algorithms) for examination timetabling was discussed in Chapter 4. It

was shown to be a very effective method for producing high quality solutions at the expense of computational time.

A study of a hybridisation approach (between the algorithm in Chapter 4 and a tabu search method) for the capacitated examination timetabling problem was undertaken in Chapter 5. It was shown that this hybrid outperforms the very large scale neighbourhood search alone. This work lead to an exploration of a “multi-start two phase” approach which aims to enhance the solution using two different phases i.e. a large neighbourhood search approach and local search methods (Chapter 6).

In Chapters 4, 5 and 6, we clearly demonstrated the effectiveness of the very large neighbourhood approach for examination timetabling. However, initial experiments showed that the approach was not suitable for course timetabling. We had to look for alternative problem solving methodologies for this problem. With a potentially large number of different neighbourhoods which could be used within local search, a well-known meta-heuristic called a variable neighbourhood search was investigated (Chapter 7). An improvement to this algorithm was made by applying a randomised iterative improvement algorithm that employs the composite neighbourhood structures (Chapter 8). Finally, we investigated a memetic approach to the problem (Chapter 9). This involved the integration of a population-based method with a local search (where the randomised iterative improvement algorithm from Chapter 8 was used as a local search). We showed that the approach in Chapter 8 is effective when compared against other methods in the literature and that it can be improved by the hybrid methodology outlined in Chapter 9.

10.2 Contributions

A number of original contributions have been drawn from this thesis. They are identified below.

- For the first time, an investigation of the very large neighbourhood search for solving the examination timetabling problem (in which the examination timetabling problem is treated as a partitioning problem) is presented. It is shown

that this approach can produce solutions of better quality than the best in the literature on certain benchmark problems. However, there is a price to pay in terms of the computational time required.

- A hybridisation of the large neighbourhood search approach with tabu search is applied to the capacitated examination timetabling problem. Experimental results show that the hybrid approach outperforms a large neighbourhood search alone on this problem and that it can produce the best results in the literature on some benchmark instances.
- A “multi-start two phase” approach that hybridises the large neighbourhood search with local search methods is applied to the examination timetabling problem and is able to produce results that are consistently good across all the benchmark problems.
- A series of experiments are carried out on the variable neighbourhood search methodology for the course timetabling problem. The findings show that the ordering of the neighbourhood structures and the probability of accepting worse solutions (where a basic variable neighbourhood search by Mladenović and Hansen, 1997 only accepts better solutions) play a role in producing a higher level of solution quality.
- A simple but efficient idea (i.e. composite neighbourhood structures using a randomised iterative improvement algorithm) is presented, which produces some of the best known solutions for the university course timetabling test instances used in this thesis.
- A combination of a genetic algorithm with a local search is developed in Chapter 9 and applied to the university course timetabling problem. This algorithm is able to obtain best known results on eight out of eleven benchmark datasets.

10.3 Future Work

Whilst this thesis presents several new methodologies for the improvement of the state of the art of automated university timetabling and the results achieved represent the best in the literature on several benchmark problem instances, there are several research questions that have been generated. Some of these are identified below.

10.3.1 Improving the Presented Approaches

In Chapter 4, the work concentrates on applying a large neighbourhood search to the university examination timetabling problem. The drawback of this approach is the amount of time needed to identify profitable exchanges. This is proportional to the number of examinations. It is noted that in real world situations, examination timetabling is an off line problem, and the processing time is usually not very critical. If an examination timetabling scenario requires results very quickly, then the method presented in this chapter would not be the most appropriate in the timetabling literature. However, if it is reasonable to run the system overnight (and this would be the case in many real world scenarios), then this approach can produce high quality results on the standard benchmark problems and would be a highly appropriate methodology to employ. Having said this it would, of course, improve the method if it could be employed more quickly. This could be done by shortening the time taken to generate an improvement graph (maybe by using different types of moves) or by shortening the required time to identify the improvement moves in the improvement graph. In addition, a rigorous analysis on the performance of the hybridisation between a large neighbourhood search approach and a tabu search (in Chapter 5) should be carried out by varying the tabu tenure during the search process in order to identify an appropriate balance between intensification and diversification in the improvement graph where the tabu tenure is increased if there is evidence for repetitions of solutions (higher diversification is needed) while it is decreased if there are no improvements (intensification should be encouraged). It may also be worthwhile to explore the application of a tabu relaxation i.e. reinitialising the tabu list after a number of non-improving iterations and restarting the search from an elite solution.

The success of a local search technique when applied to a given problem is dependent on the technique itself and upon the neighbourhood employed during the search. The work carried out in this thesis also impacts upon the neighbourhood variation (in Chapter 7). However, the proposed approach is very successful on one group of problems but is far less successful on another group of problems. Further effort would be needed in enhancing the neighbourhood structures and improving the reliability of the algorithm by intelligently selecting a subset of neighbourhood structures from a much larger pool of neighbourhoods possibly through an evolving process offered by genetic algorithms. This might improve the quality of the solutions. Additionally, it may be worth investigating the correlation between different initialisation techniques. In particular, it could be promising to consider constructive heuristic techniques (to generate an initial solution) and improvement algorithms (possibly a VNS approach and an iterated improvement algorithm with composite neighbourhood structures) to see if different initialisations would help to obtain better solutions after the improvement algorithm has taken place.

10.3.2 Hybridisation or Multi-start Approaches

It is certainly the case that the proposed approaches carried out in this thesis are open for extension and hybridisations. For example, further exploration of the multi-start approach as presented in Chapter 6 as a hybrid method in the context of university timetabling could be a promising line of future research. Furthermore, hybridisation approaches, for example, between a great deluge or tabu search with a randomised iterative improvement algorithm as discussed in Chapter 8 could be investigated.

10.3.3 Applications of the Presented Approaches to Real-life Problems

Although this thesis focuses on large neighbourhood search and meta-heuristic approaches applied to the standard benchmark problems in the literature (see Tables 3.9 - 3.11), it is worth investigating how these approaches might work on a variety of real-life problems with more constraints. For example, the application of the course timetabling problem in this thesis deals with the assignment of courses to a timeslot and room, so it is interesting if the approaches can tackle, for instance, the assignment of teachers to courses, splitting courses within a week and considering teacher preferences. In addition, possibly the idea of a multi-objective approach can

also be included in the context of real-life problems. This would consider a range of different objectives with respect to the quality of the solutions.

10.4 Dissemination

The research work carried out in this thesis has been disseminated in international journals and refereed conference proceedings. A list of papers that are published / submitted is provided below.

- S. Abdullah, S. Ahmadi, E.K. Burke and M. Dror. Investigating Ahuja-Orlin's Large Neighbourhood Search Approach for Examination Timetabling. Accepted for publication in OR Spectrum, to appear 2006 (Abdullah et al. 2006a).
- S. Abdullah, S. Ahmadi, E.K. Burke, M. Dror and B. McCollum. A Tabu based Large Neighbourhood Search Methodology for the Capacitated Examination Timetabling Problem. Accepted for publication in the Journal of the Operational Research Society, to appear 2006 (Abdullah et al. 2006b).
- S. Abdullah, S. Ahmadi, E.K. Burke and M. Dror. Applying Ahuja-Orlin's Large Neighbourhood for Constructing Examination Timetabling Solutions, Extended Abstract. In: The Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT V), Pittsburgh, PA, August 18th-20th, pages 413-419, 2004 (Abdullah et al. 2004).
- S. Abdullah, E.K. Burke and B. McCollum. An Investigation of Variable Neighbourhood Search for Course Timetabling. In: The Proceedings of the 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2005), New York, USA, July 18th-21st, pages 413-427, 2005 (Abdullah et al. 2005a).
- S. Abdullah, E.K. Burke and B. McCollum. Using a Randomised Iterative Improvement Algorithm with Composite Neighbourhood Structures for University Course Timetabling. In: Electronic Proceedings of the 6th Meta-

heuristics International Conference (MIC 05), Vienna, Austria, August 22nd-26th, 2005 (Abdullah et al. 2005b).

- S. Abdullah and E.K. Burke. A Multi-start Large Neighbourhood Search Approach with Local Search Methods for Examination Timetabling. Accepted as a short paper in the International Conference on Automated Planning and Scheduling (ICAPS-06) (Abdullah and Burke 2006).

REFERENCES

A. Aamondt. and E. Plaza. (1994). Case-based reasoning: Foundational issues, methodological variations and system approaches. *AI Communications*, 7(1), pp 39-59.

S. Abdullah and E.K. Burke. (2006). A Multi-start large neighbourhood search approach with local search methods for examination timetabling. *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling (ICAPS 2006)*, Cumbria, UK, pp 334-337.

S. Abdullah, S. Ahmadi, E.K. Burke and M. Dror. (2004). Applying Ahuja-Orlin's large neighbourhood for constructing examination timetabling solutions. *Extended Abstract, The Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT V)*, Pittsburgh, USA, pp 413-419.

S. Abdullah, E.K. Burke and B. McCollum. (2005a) An investigation of a variable neighbourhood search approach for course timetabling. *The Proceedings of the 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2005)*, New York, USA, pp 413-427.

S. Abdullah, E.K. Burke and B. McCollum. (2005b). Using a randomised iterative improvement algorithm with composite neighbourhood structures for university course timetabling. *Electronic Proceedings of the 6th Meta-heuristics International Conference (MIC 05)*, Vienna, Austria.

S. Abdullah, S. Ahmadi, E.K. Burke and M. Dror. (to appear 2006a). Investigating Ahuja-Orlin's large neighbourhood search approach for examination timetabling. Accepted for publication in *OR Spectrum*.

S. Abdullah , S. Ahmadi, E.K. Burke, M. Dror and B. McCollum. (to appear 2006b). A tabu-based large neighbourhood search approach for the capacitated examination

timetabling problem. Accepted for publication in the Journal of the Operational Research Society.

D. Abrasom. (1991). Constructing school timetables using simulated annealing: Sequential and parallel algorithms. *Management Science*, 37(1), pp 98-113.

R. Agarwal, R.K. Ahuja, G. Laporte, and Z.J. Shen. (2003). A composite very large neighbourhood search algorithm for the vehicle routing problem. *Handbook of Scheduling: Algorithms, Models and Performance Analysis*. (Editor: J.Y-T. Leung), Chapman & Hall/ CRC, pp 49-01 to 49-23.

R.K. Ahuja and H. Hamacher. (2005). A network flow algorithm to minimise beam-on time for unconstrained multileaf collimator problems in cancer radiation therapy. *Networks*, 45, pp 36-41.

R.K. Ahuja, L.T. Magnanti and J.B. Orlin. (1993). *Network flows: Theory, algorithms and applications*. ISBN 1000499012, Prentice Hall, New Jersey, pp 133-165.

R.K. Ahuja, J.B. Orlin and D. Sharma D. (2000). Very large scale neighbourhood search. *International Transactions in Operational Research*, 7, pp 301-317.

R.K. Ahuja, J.B. Orlin, and D. Sharma. (2001a). Multi-exchange neighbourhood search algorithms for the capacitated minimum spanning tree problem. *Mathematical Programming*, 91, pp 71-97.

R.K. Ahuja, K.C. Jha, J.B. Orlin, and D. Sharma. (2002a). Very large scale neighbourhood search algorithm for the quadratic assignment problem. Submitted to *INFORMS Journal on Computing*.

R.K. Ahuja, J.B. Orlin, S. Pallottino, M.P. Scaparra and M.G. Scutella. (2002b). A multi-exchange heuristic for the single source capacitated facility location problem. *Management Science*, 50, pp 749-760.

R.K. Ahuja, O. Ergun, J.B. Orlin and A.P. Punnen. (2002c). A survey of very large scale neighbourhood search techniques. *Discrete Applied Mathematics*, 123, pp 75-102.

R.K. Ahuja, J.B. Orlin and D. Sharma. (2003a). A composite very large scale neighbourhood search for the capacitated minimum spanning tree problem. *Operations Research Letters*, 31, pp 185-194.

R.K. Ahuja, A. Kumar, K.C. Jha and J.B. Orlin. (2003b). Exact and heuristic for the weapon-target assignment problem. Submitted to *Operations Research*.

R.K. Ahuja, J. Goodstein, J. Liu, A. Mukherjee, and J.B. Orlin. (2004a). Solving the combined through-fleet assignment model with time windows using neighborhood search. *Networks*, 44, pp 160-171.

R.K. Ahuja, K.C. Jha and J. Liu. (2004b). New approaches for solving the railroad blocking problem. Submitted to *Operations Research*.

R.K. Ahuja, J. Liu, J.B. Orlin, D.Sharma and L.A. Shughart. (2005a). Solving real-life locomotive scheduling algorithms. To appear in *Transportation Science*.

R.K. Ahuja, W. Huang, H.E. Romeijn and D.R. Morales. (2005b). A heuristic approach to the multi-period single sourcing problem with production and inventory capacities and perishability constraints. To appear in *INFORMS Journal on Computing*.

R. Alvarez-Valdes, E. Crespo and J.M. Tamarit. (2000). Assigning students sections using tabu search. *Annals of Operations Research*, 96, pp 1-16.

R. Alvarez-Valdes, E. Crespo and J.M. Tamarit. (2002). Design and implementation of a course scheduling systems using tabu search: Production, manufacturing and logistics. *European Journal of Operational Research*, 137, pp 512-523.

M. Amintoosi and J. Haddadnia. (2005). Feature selection in a fuzzy student sectioning algorithm. In: The Practice and Theory of Automated Timetabling V: Selected Papers from 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT V), Pittsburg, USA, Lecture Notes in Computer Science 3616, Springer-Verlag, (Editors: E.K. Burke and M. Trick), pp 147-160.

H. Asmuni, E.K. Burke and J.M. Garibaldi. (2004). A comparison of fuzzy and non-fuzzy ordering heuristics for examination timetabling. The Proceedings of the 5th International Conference on Recent Advances in Soft Computing 2004 (RASC2004), Nottingham, UK, pp 288-293.

H. Asmuni, E.K. Burke, J.M. Garibaldi and B. McCollum. (2005a). Fuzzy multiple ordering criteria for examination timetabling. The Practice and Theory of Automated Timetabling V: Selected Papers from 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT V), Pittsburg, USA, Lecture Notes in Computer Science 3616, Springer-Verlag, (Editors: E.K. Burke and M. Trick), pp 334-353.

H. Asmuni, E.K. Burke and J.M. Garibaldi. (2005b). Fuzzy multiple heuristic ordering for course timetabling. The Proceedings of the 5th United Kingdom Workshop on Computational Intelligence (UKCI05), London, UK, pp 302-309.

A.S. Asratian and D. de Werra. (2002). A generalized class–teacher model for some timetabling problems. European Journal of Operational Research, 143(3), pp 531-542.

C. Avanthay, A. Hertz and N. Zufferey. (2003). A variable neighbourhood search for graph colouring. European Journal of Operational Research, 151, pp 379-388.

M. Ayob and G. Kendall. (2003). A monte carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine. The Proceedings of the International Conference on Intelligent Technologies (InTech'03), Chiang Mai, Thailand, pp 132-141.

M.A. Badri. (1996). A two-stage multi-objective scheduling model for faculty-course-time assignments. *European Journal of Operational Research*, 94, pp 16-28.

N. Balakrishnan, A. Lucena and R.T. Wong. (1992). Scheduling examinations to reduce second order conflicts. *Computers and Operations Research*, 19, pp 353-361.

V.A. Bardadym. (1996). Computer-aided school and university timetabling: a new wave. *The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT I)*, Edinburgh, UK, Lecture Notes in Computer Science 1153, Springer-Verlag. (Editors: E.K. Burke and P. Ross), pp 22-45.

Ü. Bilge, F. Kiraç, M. Kurtulan and P. Pekgün. (2004). A tabu search algorithm for the parallel machine total tardiness problem. *Computers and Operations Research*, 31(3), pp 397-414.

C. Blum and A. Roli. (2003). Meta-heuristics in combinatorial optimisation: Overview and conceptual comparison. *ACM Computing Survey*, 35(3), pp 268-308.

C. Blum, S. Correia, M. Dorigo, B. Paechter, O. Rossi-Doria and M. Snoek. (2002). A GA evolving instructions for a timetable builder. *The Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling*, Gent, Belgium, pp 120-123.

P. Boizumault, Y. Delon and L. Peridy. (1996). Constraint logic programming for examination timetabling. *The Journal of Logic Programming*, 26, pp 217-233.

S.C. Brailsford, C.N. Potts and B.M. Smith. (1996). Constraint satisfaction problems: Algorithms and applications. *European Journal of Operational Research*, 119, pp 557-581.

D. Brelaz. (1979). New methods to colour the vertices of a graph. *Communication of ACM*, 22, pp 251-156.

B. Bullnheimer. (1998). An examination scheduling model to maximize student's study time. The Practice and Theory of Automated Timetabling II: Selected Papers from 2nd International Conference on the Practice and Theory of Automated Timetabling (PATAT II), Toronto, Canada, Lecture Notes in Computer Science 1408, Springer-Verlag. (Editors: E.K. Burke and M. Carter), pp 78-91.

E.K. Burke and M. Carter. (editors), (1998). The Practice and Theory of Automated Timetabling II: Selected Papers from 2nd International Conference on the Practice and Theory of Automated Timetabling (PATAT II), Toronto, Canada, Lecture Notes in Computer Science 1408, Springer-Verlag.

E.K. Burke and P. De. Causmaecker. (editors), (2003). The Practice and Theory of Automated Timetabling IV: Selected Papers from 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT IV), Gent, Belgium, Lecture Notes in Computer Science 2740, Springer-Verlag.

E.K. Burke and W. Erben. (editors), (2001). The Practice and Theory of Automated Timetabling III: Selected Papers from 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT III), Konstanz, Germany, Lecture Notes in Computer Science 2079, Springer-Verlag.

E.K. Burke and G. Kendall. (editors), (2005). Introductory Tutorials in Optimisation, Decision Support and Search Methodology, Kluwer.

E.K. Burke and J.D. Landa Silva. (2004). The design of memetic algorithms for scheduling and timetabling problems. Recent Advances in Memetic Algorithms, Studies in Fuzziness and Soft Computing, (Editors: N. Krasnogor, W. Hart and J.Smith), Springer, 166, pp 289-312.

E.K. Burke and J.P. Newall. (1999). A multi-stage evolutionary algorithm for the timetable problem. IEEE Transactions on Evolutionary Computation, 3.1, pp 63-74.

E.K. Burke and J.P. Newall. (2003). Enhancing timetable solutions with local search methods. *The Practice and Theory of Automated Timetabling IV: Selected Papers from 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT IV)*, Gent, Belgium, Lecture Notes in Computer Science 2740, Springer-Verlag. (Editors: E.K. Burke and P. De Causmaecker), pp 195-206.

E.K. Burke and J.P. Newall. (2004). Solving examination timetabling problems through adaptation of heuristic orderings. *Annals of Operations Research*, 129, pp 107-134.

E.K. Burke and S. Petrovic. (2002). Recent research directions in automated timetabling. *European Journal of Operational Research*, 140(2), pp 266-280.

E.K. Burke and P. Ross. (1996). Editors: *The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT I)*, Edinburgh, UK, Lecture Notes in Computer Science 1153, Springer-Verlag.

E.K. Burke and M. Trick. (2005). Editors: *The Practice and Theory of Automated Timetabling V: Selected Papers from 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT V)*, Pittsburg, USA, Lecture Notes in Computer Science 3616, Springer-Verlag.

E.K. Burke, D.G. Elliman and R.F. Weare. (1993). Extensions to a university exams timetabling system. *The Proceeding of the IJCA-93 Workshop on Knowledge-based Production, Planning, Scheduling and Control*, Chambery, France, pp 42-48.

E.K. Burke, D.G. Elliman and R.F. Weare. (1994a). A university timetabling system based on graph colouring and constraint manipulation. *Journal of Research on Computing in Education*, 27(1), pp 1-18.

E.K. Burke, D.G. Elliman and R.F. Weare. (1994b). A genetic algorithm for university timetabling. The Proceedings of the Artificial Intelligence and Simulation of Behaviour (AISB) Workshop on Evolutionary Computing, University of Leeds, UK.

E.K. Burke, D.G. Elliman and R.F. Weare. (1994c). A genetic algorithm based university timetabling system. The Proceedings of the 2nd East-West International Conference on Computer Technologies in Education Crimea, Ukraine, 1, pp 35-40.

E.K. Burke, D.G. Elliman and R.F. Weare. (1995a). A hybrid genetic algorithm for highly constrained timetabling problems. The Proceedings of the 6th International Conference on Genetic Algorithms (ICGA'95), Pittsburgh, USA, pp 605-610.

E.K. Burke, D.G. Elliman, P.H. Ford and R.F. Weare. (1995b). Specialised recombinative operators for the timetabling problem. The Proceedings of the Artificial Intelligence and Simulation of Behaviour (AISB) Workshop on Evolutionary Computing, University of Sheffield, UK, Lecture Notes in Computer Science 993, Springer-Verlag, pp 75-85.

E.K. Burke, D.G. Elliman, P.H. Ford and R.F. Weare. (1996a). Examination timetabling in British universities - A survey. The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT I), Edinburgh, UK, Lecture Notes in Computer Science 1153, Springer-Verlag. (Editors: E.K. Burke and P. Ross), pp 76-92.

E.K. Burke, J.P. Newall and R.F. Weare. (1996b). A memetic algorithm for university exam timetabling. The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT I), Edinburgh, UK, Lecture Notes in Computer Science 1153, Springer-Verlag. (Editors: E.K. Burke and P. Ross), pp 241-250.

E.K. Burke, K. Jackson, J.H. Kingston and R.F. Weare. (1997). Automated university timetabling: the state of the art. *The Computer Journal*, 40(9), pp 565-571.

E.K. Burke, J.P. Newall and R.F. Weare. (1998a). A simple heuristically guided search for the timetable problem. *The Proceedings of the International ICSC Symposium on Engineering of Intelligent System (EIS'98)*, Canada/Switzerland: ICSC/Academic Press, pp 574-579.

E.K. Burke, J.P. Newall and R.F. Weare. (1998b). Initialisation strategies and diversity in evolutionary timetabling. *Evolutionary Computation*, 6(1), pp 81-103.

E.K. Burke, B. MacCarthy, S. Petrovic and R. Qu. (2000). Structured cases in CBR - re-using and adapting cases for timetabling problems. *Knowledge-based System*, 13(2-3), pp 159-165.

E.K. Burke, Y. Bykov and S. Petrovic. (2001a). A multi-criteria approach to examination timetabling. *The Practice and Theory of Automated Timetabling III: Selected Papers from 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT III)*, Konstanz, Germany, *Lecture Notes in Computer Science* 2079, Springer-Verlag. (Editors: E.K. Burke and W. Erben), pp 118-131.

E.K. Burke, B. MacCarthy, S. Petrovic and R. Qu. (2001b). Case-based reasoning in course timetabling: an attribute graph approach. *The Proceeding of the 4th International Conference on Case-based Reasoning (ICCBR-2001)*, Springer Lecture Notes in Artificial Intelligence 2080, pp 90-104.

E.K. Burke, G. Kendall and E. Soubeiga. (2003a). A tabu-search hyper-heuristic for timetabling and rostering. *Journal of Heuristics*, 9(6), pp 451-470.

E.K. Burke, A.J. Eckersley, B. McCollum, S. Petrovic and R. Qu. (2003b). Similarity measures for exam timetabling problems. *The Proceeding of the 1st Multidisciplinary*

International Conference on Scheduling: Theory and Applications (MISTA 2003), Nottingham, UK, 1, pp 120-136.

E.K. Burke, E. Hart, G. Kendall, J.P. Newall, P. Ross and S. Schulenburg. (2003c). Hyper-heuristics: An emerging direction in modern search technology. Chapter 16 in Handbook of Meta-Heuristics, (Editors: F. Glover and G. Kochenberger), Kluwer, pp 457-474.

E.K. Burke, Y. Bykov, J. Newall and S. Petrovic. (2003d). A time-predefined approach to course timetabling. Yugoslav Journal of Operations Research (YUJOR), 13(2), pp 139-151.

E.K. Burke, J. Kingston and D. de Werra. (2004a). Applications to timetabling. Handbook of Graph Theory, (Editors: J. Gross and J. Yellen), Chapman Hall/CRC Press, pp 445-474.

E.K. Burke, Y. Bykov, J.P. Newall and S. Petrovic. (2004b). A time-predefined local search approach to exam timetabling problem. IIE Transactions, 36(6), pp 509-528.

E.K. Burke, A.J. Eckersley, B. McCollum, S. Petrovic and R. Qu. (2004c). Analysing similarity in exam timetabling. The Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT V), Pittsburg, USA, pp 89-106.

E.K. Burke, A.J. Eckersley, S. Petrovic and R. Qu. (2004d). A case-based heuristic selection prototype system for exam timetabling problems, Extended Abstract. The Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT V), Pittsburgh, USA, pp 447-450.

E.K. Burke, B. MacCarthy, S. Petrovic and R. Qu. (2006a). Multiple-retrieval case-based reasoning for course timetabling problems. Journal of the Operational Research Society, 57(2), pp 148-262.

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

E.K. Burke, A.J. Eckersley, B. McCollum, S. Petrovic and R. Qu. (2006c). Hybrid variable neighbourhood approaches to university exam timetabling. Technical Report NOTTCS-TR-2006-2, School of Computer Science and IT, University of Nottingham, United Kingdom.

E.K. Burke, A. Meisels, S. Petrovic and R. Qu. (2007). A graph-based hyper-heuristic for timetabling problems. *European Journal of Operational Research*, 176, pp 177-192.

M. Caramia, P. Dell’Olmo and G.F. Italiano. (2001). New algorithms for examination timetabling. *Algorithms Engineering 4th International Workshop, Proceedings WAE 2001, Saarbrücken, Germany, Lecture Notes in Computer Science 1982, Springer-Verlag*, pp 230-241.

M.P. Carrasco and M.V. Pato. (2001). A multi-objective genetic algorithm for the class/teacher timetabling problem. *The Practice and Theory of Automated Timetabling III: Selected Papers from 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT III), Konstanz, Germany, Lecture Notes in Computer Science 2079, Springer-Verlag*. (Editors: E.K. Burke and W. Erben 2001), pp 3-17.

M.W. Carter. (1986). A survey of practical applications of examination timetabling. *Operations Research*, 34, pp 193-202.

M.W. Carter. (2001). A comprehensive course timetabling and student scheduling system at the University of Waterloo. *The Practice and Theory of Automated Timetabling III: Selected Papers from 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT III), Konstanz, Germany, Lecture Notes in Computer Science 2079, Springer-Verlag*. (Editors: E.K. Burke and W. Erben), pp 64-82.

M.W. Carter and D.G. Johnson. (2001). Extended clique initialisation in examination timetabling. *Journal of the Operational Research Society*, 52, pp 538-544.

M.W. Carter and G. Laporte. (1996). Recent developments in practical examination timetabling. *The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT I)*, Edinburgh, UK, Lecture Notes in Computer Science 1153, Springer-Verlag. (Editors: E.K. Burke and P. Ross), pp 3-21.

M.W. Carter and G. Laporte. (1998). Recent developments in practical course timetabling. *The Practice and Theory of Automated Timetabling II: Selected Papers from 2nd International Conference on the Practice and Theory of Automated Timetabling (PATAT II)*, Toronto, Canada, Lecture Notes in Computer Science 1408, Springer-Verlag. (Editors: E.K. Burke and M. Carter), pp 3-19.

M.W. Carter, G. Laporte and J.W. Chinneck. (1994). A general examination scheduling system. *Interfaces*, 24(3), pp 109-120.

M.W. Carter, G. Laporte and S. Lee. (1996). Examination timetabling: Algorithmic strategies and applications. *Journal of the Operational Research Society*, 47(3), pp 373-383.

S. Casey and J. Thompson. (2003). GRASping the examination scheduling problem. *The Practice and Theory of Automated Timetabling IV: Selected Papers from 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT IV)*, Gent, Belgium, Lecture Notes in Computer Science 2740, Springer-Verlag. (Editors: E.K. Burke and P. De Causmaecker), pp 232-244.

C. Cheng, L. Kang, N. Leung and G.M. White. (1996). Investigations of a constraint logic programming approach to university timetabling. In *The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT I)*, Edinburgh, UK, Lecture

Notes in Computer Science 1153, Springer-Verlag. (Editors: E.K. Burke and P. Ross), pp 112-129.

S.C. Chu and H.L. Fang. (1999). Genetic algorithms vs tabu search in timetable scheduling. The Proceedings of the 3rd International Conference on Knowledge-based Intelligent Information Engineering System, Adelaide, Australia, pp 492-495.

A. Colorni, M. Dorigo and V. Maniezzo. (1998). Meta-heuristics for high school timetabling. Computational Optimisation and Applications, 9, pp 275-298.

D. Corne, H.L. Fang and C. Mellish. (1993). Solving the modular exam scheduling problem with genetic algorithms. The Proceedings of the 6th International Conference of Industrial and Engineering Applications of AI and Expert System, pp 370-373.

D. Corne, P. Ross and H.L. Fang. (1994). Fast practical evolutionary timetabling. The Proceedings of Evolutionary Computing AISB Workshop, Lecture Notes in Computer Science 865, Springer-Verlag, pp 251-263.

D. Costa. (1994). A tabu search for computing an operational timetable. European Journal of Operational Research, 76, pp 98-110.

D. Costa and A. Hertz. (1997). Ant can colour graphs. Journal of the Operational Research Society, 48, pp 295-305.

P. Côté, T. Wong and R. Sabourin. (2005). A hybrid multi-objective evolutionary algorithm for the uncapacitated exam proximity problem. The Practice and Theory of Automated Timetabling V: Selected Papers from 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT V), Pittsburg, USA, Lecture Notes in Computer Science 3616, Springer-Verlag. (Editors: E.K. Burke and M. Trick), pp 294-312.

P. David. (1998). A constraint-based approach for examination timetabling using local repair technique. The Practice and Theory of Automated Timetabling II:

Selected Papers from 2nd International Conference on the Practice and Theory of Automated Timetabling (PATAT II), Toronto, Canada, Lecture Notes in Computer Science 1408, Springer-Verlag. (Editors: E.K. Burke and M. Carter), pp 169-186.

S. Deris, S. Omatu, H. Ohta and P. Saad. (1999). Incorporating constraint propagation in genetic algorithm for university timetable planning. *Engineering Application of Artificial Intelligence*, 12, pp 241-253.

S. Deris, S. Omatu and H. Ohta. (2000). Timetable planning using the constraint-based reasoning. *Computers & Operations Research*, 27(9), pp 819-840.

S. Desroches, G. Laporte and J.M. Rousseau. (1978). HOREX: A computer program for the construction of examination. *INFOR*, 16(3), pp 294-298.

D. de Werra. (1985). An introduction to timetabling. *European Journal of Operational Research*, 19, pp 151-162.

D. de Werra. (1996a). Extensions of colouring models for scheduling purposes. *European Journal of Operational Research*, 92, pp 474-492.

D. de Werra. (1996b). Some combinatorial models for course scheduling. *The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT I)*, Edinburgh, UK, Lecture Notes in Computer Science 1153, Springer-Verlag. (Editors: E.K. Burke and P. Ross), pp 296-308.

D. de Werra. (1997a). Restricted colouring models for timetabling. *Discrete Mathematics*, 165/166, pp 161-170.

D. de Werra. (1997b). Theory and methodology: The combinatorics of timetabling. *European Journal of Operational Research*, 96, pp 504-513.

D. de Werra, A.S. Asratian and S. Durand. (2002). Complexity models of some special types of timetabling problems. *Journal of Scheduling*, 5, pp 171-183.

L. Di Gaspero. (2002). Recolor, shake and kick: A recipe for the examination timetabling problem, Extended Abstract. The Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT IV), Gent, Belgium, pp 404-407.

L. Di Gaspero and A. Schaerf. (2001). Tabu search techniques for examination timetabling. *The Practice and Theory of Automated Timetabling III: Selected Papers from 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT III)*, Konstanz, Germany, Lecture Notes in Computer Science 2079, Springer-Verlag. (Editors: E.K. Burke and W. Erben), pp 104-117.

M. Dorigo and G. Di Caro. (1999). The ant colony optimisation meta-heuristic. *New Ideas in Optimisation* (Editors: D. Corne, M. Dorigo & F. Glover), McGraw-Hill, pp 11-32.

M. Dorigo and T. Stützle. (2003). The ant colony optimisation meta-heuristic: algorithms, applications, and advances. *Handbook of Meta-heuristics*, International Series in Operations Research and Management Science, (Editors: F. Glover and G. Kochenberger), Kluwer, pp 251-285.

M. Dorigo and T. Stützle. (2004). *Ant colony optimisation*. MIT Press, ISBN 0-262-04219-3.

M. Dorigo, V. Maniezzo and A. Colomi. (1996). The ant system: Optimisation by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics*, 26, pp 29-41.

K.A. Dowsland. (1998). Off-the-peg or made-to measure? Timetabling and scheduling with SA and TS. *The Practice and Theory of Automated Timetabling II: Selected Papers from 2nd International Conference on the Practice and Theory of*

Automated Timetabling (PATAT II), Toronto, Canada, Lecture Notes in Computer Science 1408, Springer-Verlag. (Editors: E.K. Burke and M. Carter), pp 37-52.

K.A. Dowsland and J.M. Thompson. (2005). Ant colony optimisation for the examination scheduling problem. *Journal of the Operational Research Society*, 56, pp 426-438.

G. Dueck. (1993). New optimisation heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104, pp 86-92.

G. Dueck and T. Scheuer. (1990). Threshold accepting: A general purpose optimisation appearing superior to simulated annealing. *Journal of Computational Physics*, 90(1), pp 161-175.

M.A.S. Elmohamed, P. Coddington and G. Fox. (1998). A comparison of annealing techniques for academic course scheduling. *The Practice and Theory of Automated Timetabling II: Selected Papers from 2nd International Conference on the Practice and Theory of Automated Timetabling (PATAT II)*, Toronto, Canada, Lecture Notes in Computer Science 1408, Springer-Verlag. (Editors: E.K. Burke and M. Carter), pp 92-112.

W. Erben. (2001). A grouping genetic algorithm for graph colouring and exam timetabling. *The Practice and Theory of Automated Timetabling IV: Selected Papers from 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT IV)*, Gent, Belgium, Lecture Notes in Computer Science 2740, Springer-Verlag. (Editors: E.K. Burke and P. De Causmaecker), pp 132-156.

W. Erben and J. Keppler. (1996). A genetic algorithm solving a weekly course-timetabling problem. *The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT I)*, Edinburgh, UK, Lecture Notes in Computer Science 1153, Springer-Verlag. (Editors: E.K. Burke and P. Ross), pp 198-211.

A. Ergül. (1996). GA-based examination scheduling experience at Middle East Technical University. The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT I), Edinburgh, UK, Lecture Notes in Computer Science 1153, Springer-Verlag. (Editors: E.K. Burke and P. Ross), pp 212 – 226.

F. Focacci, F. Laburthe and A.Lodi. (2003). Local search and constraint programming. Handbook of Meta-heuristics, (Editors: F. Glover and G. Kochenberger), Chapter 13, Kluwer Academic Publishers, pp 369-403.

E. Foxley and K. Lockyer. (1968). The construction of examination timetables by computer. The Computer Journal, 11, pp 264-268.

P. Galliner and J.K. Hao. (1999). Hybrid evolutionary algorithms for graph colouring. Journal of Combinatorial Optimisation, 3, pp 379-397.

M. Gendreau and J. Potvin (2005). Introductory Tutorials in Optimisation, Decision Support and Search Methodology, Chapter 6: Tabu search. Kluwer. (Editors: E.K. Burke and G. Kendall).

F. Glover. (1986). Future paths for integer programming and links to artificial intelligence. Computers & Operation Research, 13(5), pp 533-549.

F. Glover and M. Laguna. (1997). Tabu search. Kluwer Academic Publisher, ISBN 0-7923-8187-4.

D.E. Goldberg. (1989). Genetic algorithms in search, optimisation and machine learning. Addison Wesley.

M. Gopalakrishnan, S.L. Ahire and D.M. Miller. (1997). Maximising the effectiveness of a preventive maintenance system: An adaptive modeling approach. Management Science, 43(6), pp 827-840.

- M. Gopalakrishnan, S. Mohan and Z. He. (2001). A tabu search heuristic for preventive maintenance scheduling. *Computers & Industrial Engineering*, 40, pp 149-160.
- J. Grabowski and J. Pempera. (2000). Sequencing of jobs in some production system: Theory and methodology. *European Journal of Operational Research*, 125, pp 535-550.
- P. Hansen and N. Mladenović. (2001). Variable neighbourhood search. *European Journal of Operational Research*, 130, pp 449-467.
- W.E. Hart, N. Krosnigor and J.E. Smith. (2004). *Studies in fuzziness and soft computing: Recent advances in memetic algorithms*. Springer, pp 3-27.
- A. Hertz. (1991). Tabu search for large scale timetabling problems. *European Journal of Operational Research*, 54, pp 39-47.
- A. Hertz. (1992). Finding a feasible course schedule using tabu search. *Discrete Applied Mathematics*, 35, pp 255-270.
- A. Hertz and D. de Werra. (1990). The tabu search meta-heuristic: How we used it. *Annals of Mathematics and Artificial Intelligence*, 1, pp 111-121.
- M. Henz and J. Würtz. (1996). Using Oz for college timetabling. *The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT I)*, Edinburgh, UK, Lecture Notes in Computer Science 1153, Springer-Verlag. (Editors: E.K. Burke and P. Ross), pp 162-177.
- A.J. Higgins. (2001). A dynamic tabu search for large-scale generalised assignment problems. *Computers and Operations Research*, 28(10), pp 1039-1048.

J.H. Holland. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.

H. Inshibuchi, T. Yoshida and T. Murata. (2003). Balance between genetic search and local search in memetic algorithms for multi-objective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation*, 7(2), pp 204-223.

D. Johnson. (1990). Timetabling university examinations. *Journal of the Operational Research Society*, 41(1), pp 39-47.

D.E. Joslin and D.P. Clements. (1999). Squeaky wheel optimisation. *Journal of Artificial Intelligence Research*, 10, pp 353-373.

L. Kang and G.M. White. (1992). A logic approach to a resolution of constraints in timetabling. *European Journal of Operational Research*, 61, pp 306-317.

G. Kendall and N. Hussin. (2005a). An investigation of a tabu search based hyper-heuristic for examination timetabling. *Multidisciplinary Scheduling: Theory and Applications*, (Editors: G. Kendall, E.K. Burke, S. Petrovic and M. Gendreau), Springer, pp 309-328.

G. Kendall and N. Hussin. (2005b). Tabu search hyper-heuristic approach to the examination timetabling problem at MARA University of Technology. *The Practice and Theory of Automated Timetabling V: Selected Papers from 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT V)*, Pittsburg, USA, Lecture Notes in Computer Science 3616, Springer-Verlag. (Editors: E.K. Burke and M. Trick), pp 270-293.

S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi. (1983). Optimisation by simulated annealing. *Science*, 220, pp 671-380.

J.L. Kolodner. (1993). *Case-based reasoning*. Morgan Kaufman Publishers, Inc. San Mateo.

J.L. Kolodner and D. Leake. (1996). A tutorial introduction to case-based reasoning. *Case-Based Reasoning: Experiences, Lessons and Future Directions*, (Editor: D. Leake), AAAI Press/The MIT Press, pp 31-65.

S.C. Kong and L.F. Kwok. (1999). A conceptual model of knowledge-based timetabling system. *Knowledge-based System*, 12, pp 81-93.

P. Kostuch. (2005). The university course timetabling problem with a three-phase approach. *The Practice and Theory of Automated Timetabling V: Selected Papers from 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT V)*, Pittsburg, USA, Lecture Notes in Computer Science 3616, Springer-Verlag. (Editors: E.K. Burke and M. Trick), pp 109-125.

G. Konstantinow and C. Coakley. (2004). Use of genetic algorithms in reactive scheduling for course timetable adjustments, Extended Abstract. *The Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT V)*, Pittsburg, USA, pp 521-522.

N. Krasnogor and J.E. Smith. (2005). A tutorial for competent memetic algorithms: model, taxonomy and design issues. *IEEE Transactions on Evolutionary Computation*, 9(5), pp 474- 488.

G. Lajos. (1996). Complete university modular timetabling using constraint logic programming. *The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT I)*, Edinburgh, UK, Lecture Notes in Computer Science 1153, Springer-Verlag. (Editors: E.K. Burke and P. Ross), pp 146 – 161.

J.D. Landa Silva. (2003). Meta-heuristic and multi-objective approaches for space allocation. PhD Thesis, School of Computer Science and IT, University of Nottingham, United Kingdom.

J.D. Landa Silva, E.K. Burke and S. Petrovic. (2004). An introduction to multi-objective meta-heuristics for scheduling and timetabling. Multiple Objective Meta-heuristics, (Editors: X. Gandibleux, M. Sevaux, K. Sorensen and V. T'kindt), Lecture Notes in Economic and Mathematical Systems 535, Springer-Verlag, pp 91-129.

G. Laporte. and S. Desroches. (1986), The problem of assigning students to course section in a large engineering school. Computers and Operations Research, 13, pp 387-394.

W. Legierski. (2003). Search strategy for constraint-based class-teacher timetabling. The Practice and Theory of Automated Timetabling IV: Selected Papers from 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT IV), Gent, Belgium, Lecture Notes in Computer Science 2740, Springer-Verlag. (Editors: E.K. Burke and P. De Causmaecker), pp 247-261.

R. Lewis and B. Paechter. (2004). New crossover operators for timetabling with evolutionary algorithms. The 5th International Conference on Recent Advances in Soft Computing (RASC 2004), Nottingham, UK, 5, pp 189-195.

R. Lewis and B. Paechter. (2005). Application of the grouping genetic algorithm to university course timetabling. The 5th European Conference in Evolutionary Computation in Combinatorial Optimisation (EvoCop 2005), Lausanne, Switzerland, Lecture Notes in Computer Science 3448, pp 144-153.

C.F. Liaw. (2003). An efficient tabu search approach for the two-machine preemptive open shop scheduling problem. Computers & Operations Research, 30(14), pp 2081-2095.

Meta-heuristics Network. Available at <http://www.metaheuristics.net/>. [Accessed 18th August 2005].

L.T.G. Merlot, N. Boland, B.D. Hughes and P.J. Stuckey. (2003). A hybrid algorithm for the examination timetabling problem. The Practice and Theory of Automated Timetabling IV: Selected Papers from 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT IV), Gent, Belgium, Lecture Notes in Computer Science 2740, Springer-Verlag. (Editors: E.K. Burke and P. De Causmaecker), pp 207-231.

N. Mladenović and P. Hansen. (1997). Variable neighbourhood search. Computers and Operations Research, 24(11), pp 1097-1100.

P. Moscato. (1999). Memetic algorithms: A short introduction. New Ideas in Optimisation, (Editors: D. Corne, M. Dorigo and F. Glover), McGraw-Hill, pp 219-234.

P. Moscato. (2002). Memetic algorithms. Handbook of Applied Optimisation, (Editors: P.M. Pardalos and M.G.C. Resende), Oxford University Press, New York, pp 157-167.

G.A. Neufeld and J. Tartar. (1974). Graph colouring conditions for the existence of solutions to the timetable problem. Communications of the ACM, 17(8), pp 450-453.

J.P. Newall. (1999). Hybrid methods for automated timetabling. PhD Thesis, School of Computer Science and IT, University of Nottingham, United Kingdom.

K. Nonobe and T. Ibaraki. (1998). A tabu search approach to the constraint satisfaction problem as a general problem solver. European Journal of Operational Research, 106, pp 599-623.

W.P.M. Nuijten, G.M. Kunnen, E.L.H. Aarts and F.P.M. Dignum. (1994). Examination timetabling: A case study for constraint satisfaction. The Proceedings of the ECAI'94 Workshop on Constraint Satisfaction Issues Raised by Practical Applications, pp 11-19.

D. Ouelhadj. (2003). A multi-agent system for the integrated dynamic scheduling of steel production. PhD Thesis, School of Computer Science and IT, University of Nottingham, United Kingdom.

I.H. Osman and G. Laporte. (1996). Meta-heuristics: A bibliography. *Annals of Operations Research*, 63, pp 513-623.

L. Pacquete and C.M. Fonseca. (2001). A study of examination timetabling with multi-objective evolutionary algorithm. *The Proceedings of the 4th Meta-heuristics International Conference (MIC 2001)*, pp 149-154.

L. Pacquete and T. Stützle. (2002). Empirical analysis of tabu search for the lexicographic optimisation of the examination timetabling problem. *The Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT IV)*, Gent, Belgium, pp 413-420.

B. Paechter, A. Cumming, M.G. Norman and H. Luchian. (1996). Extensions to a memetic timetabling system. *The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT I)*, Edinburgh, UK, *Lecture Notes in Computer Science* 1153, Springer-Verlag. (Editors: E.K. Burke and P. Ross), pp 251-265.

B. Paechter, R.C. Rankin and A. Cumming. (1998). Improving a lecture timetabling system for university-wide use. *The Practice and Theory of Automated Timetabling II: Selected Papers from 2nd International Conference on the Practice and Theory of Automated Timetabling (PATAT II)*, Toronto, Canada, *Lecture Notes in Computer Science* 1408, Springer-Verlag. (Editors: E.K. Burke and M. Carter), pp 156-165.

F.V. Partovi and B. Arinze. (1995). A knowledge-based approach to the faculty course assignment problem. *Socio-Economics Planning Science*, 29(3), pp 245-256.

S. Petrovic and E.K. Burke. (2004). University timetabling. Handbook of Scheduling: Algorithms, Models and Performance Analysis, Chapter 45, (Editor: J. Leung), CRC Press.

S. Petrovic and Y. Bykov. (2003). A multi-objective optimisation technique for exam timetabling based on trajectories. The Practice and Theory of Automated Timetabling IV: Selected Papers from 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT IV), Gent, Belgium, Lecture Notes in Computer Science 2740, Springer-Verlag. (Editors: E.K. Burke and P. De Causmaecker), pp 181-194.

S. Petrovic and R. Qu. (2002). Case-based reasoning as a heuristic selector in a hyper-heuristic for course timetabling. The Proceedings of the 6th International Conference on Knowledge-Based Intelligent Information & Engineering Systems and Applied Technologies (KES 2002), Milan, Italy, 82, pp 336–340.

S. Petrovic, Y. Yang and M. Dror. (2004). Use of case-based reasoning in solving examination timetabling problems. Submitted to the European Journal of Operational Research. School of Computer Science and IT, Technical Report NOTTCS-TR-2004-6, University of Nottingham.

S. Petrovic, Y. Yang and M. Dror. (2005). Case-based initialisation for examination timetabling. Multidisciplinary Scheduling: Theory and Applications, (Editors: G. Kendall, E.K. Burke, S. Petrovic and M. Gendreau), Springer, pp 289-308.

S. Petrovic, V. Patel and Y. Yang. (2005). University timetabling with fuzzy constraints. The Practice and Theory of Automated Timetabling V: Selected Papers from 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT V), Pittsburg, USA, Lecture Notes in Computer Science 3616, Springer-Verlag. (Editors: E.K. Burke and M. Trick), pp 313-333.

D.T. Pham and D. Karaboga. (2000). Intelligent Optimisation Techniques, ISBN 1-85233-028-7, Springer-Verlag.

R. Qu. (2002). Case-based reasoning for course timetabling problems. PhD Thesis, School of Computer Science and IT, University of Nottingham, United Kingdom.

R. Qu and E.K. Burke. (2005). Hybrid variable neighbourhood hyper-heuristics for exam timetabling problems. Electronic Proceedings of the 6th Meta-heuristics International Conference (MIC 05), Vienna, Austria.

C.R. Reeves. (1995). Modern Heuristic Techniques for Combinatorial Problems, (Editor), McGraw-Hill.

D.C. Rich. (1996). A smart genetic algorithm for university timetabling. The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT I), Edinburgh, UK, Lecture Notes in Computer Science 1153, Springer-Verlag. (Editors: E.K. Burke and P. Ross), pp 181-197.

B.P. Romero. (1982). Examination scheduling in a large engineering school: A computer assisted participative procedure. Interfaces, 12, pp 17-23.

P. Ross and D. Corne. (1995). Comparing genetic algorithm, simulated annealing and stochastic hill climbing of timetabling problems. The Proceedings of the 1995 AISB Workshop on Evolutionary Computing, Lecture Notes in Computer science 993, Springer-Verlag, pp 94-102.

P. Ross, D. Corne and H.L. Fang. (1994). Improving evolutionary timetabling with delta evolution and directed mutation. The Proceedings of the Conference on Parallel Problem Solving from Nature III, Lecture Notes in Computer Science 866, Springer-Verlag, pp 556-565.

P. Ross, E. Hart and D. Corne. (1998). Some observations about GA based timetabling. The Practice and Theory of Automated Timetabling II: Selected Papers from 2nd International Conference on the Practice and Theory of Automated

Timetabling (PATAT II), Toronto, Canada, Lecture Notes in Computer Science 1408, Springer-Verlag. (Editors: E.K. Burke and M. Carter), pp 115-129.

O. Rossi-Doria, C. Blum, J. Knowles, M. Samples, K. Socha and B. Paechter. (2002). A local search for the timetabling, Extended Abstract. The Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling (PATAT IV), Gent, Belgium, pp 124-127.

O. Rossi-Doria, M. Samples, M. Birattari, M. Chiarandini, M. Dorigo, L.M. Gambardella, J. Knowles, M. Manfrin, M. Mastrolilli, B. Paechter, L. Paquete and T. Stützle. (2003). A comparison of the performance of different meta-heuristics on the timetabling problem. The Practice and Theory of Automated Timetabling IV: Selected Papers from 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT IV), Gent, Belgium, Lecture Notes in Computer Science 2740, Springer-Verlag. (Editors: E.K. Burke and P. De Causmaecker), pp 329-354.

H. Rudová and K. Murray. (2003). University course timetabling with soft constraints. The Practice and Theory of Automated Timetabling IV: Selected Papers from 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT IV), Gent, Belgium, Lecture Notes in Computer Science 2740, Springer-Verlag. (Editors: E.K. Burke and P. De Causmaecker), pp 310-328.

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

A. Schaerf. (1999b). Local search techniques for high-school timetabling problems. IEEE Transactions on Systems, Man, and Cybernetics, 29(4), pp 368-377.

S.M. Selim. (1988). Split vertices in vertex colouring and their application in developing a solution to the faculty timetable problem. The Computer Journal, 30(1), pp 76-82.

K. Socha, J. Knowles and M. Samples. (2002). A max-min ant system for the university course timetabling problem. The Proceedings of the 3rd International Workshop on Ant Algorithms (ANTS 2002), Lecture Notes in Computer science 2463, Springer-Verlag, pp 1-13.

K. Socha, M. Samples and M. Manfrin M. (2003). Ant algorithm for the university course timetabling problem with regard to the state-of-the-art. The Proceedings of the 3rd European Workshop on Evolutionary Computation in Combinatorial Optimisation, Essex, UK, Lecture Notes in Computer science 2611, Springer-Verlag, pp 334-345.

H. Terashima-Marín. (1998). Combining of GAs and CSP strategies for solving the examination timetabling problem. PhD Thesis, Instituto Tecnológico y de Estudios Superiores de Monterrey, 1998.

H. Terashima-Marín, P. Ross and M. Valenzuela-Rendón. (1999a). Clique-based crossover for solving the timetabling problem with GAs. Proceedings of the Congress on Evolutionary Computation. Washington, D.C., pp 1200-1206.

H. Terashima-Marín, P. Ross and M. Valenzuela-Rendón.. (1999b). Evolution of constraint satisfaction strategies in examination timetabling. Proceedings of the Genetic and Evolutionary Conference, Orlando, Florida, Morgan Kaufmann pp 635-642.

J.M. Thompson and K.A. Dowsland. (1996a). Variants of simulated annealing for the examination timetabling problem. Annals of Operations Research, 63, pp 105-128.

J.M. Thompson and K.A. Dowsland. (1996b). General cooling schedule for a simulated annealing based timetabling system. The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT I), Edinburgh, UK, Lecture

Notes in Computer Science 1153, Springer-Verlag. (Editors: E.K. Burke and P. Ross), pp 345-363.

J.M. Thompson and K.A. Dowsland. (1998). A robust simulated annealing based examination timetabling system. *Computers Operations Research*, 25, pp 637-648.

P.M. Thompson and J.B. Orlin. (1989). The theory of cyclic transfer. Working paper No. OR 200-89, Operations Research Centre, MIT, Cambridge, MA.

P.M. Thompson and H.N. Psaraftis. (1993). Cyclic transfer algorithms for multivehicle routing and scheduling problems. *Operations Research*, 41, pp 935-946.

H. Ueda, D. Ouchi, K. Takahashi and T. Miyahara. (2001). A co-evolving timeslot/room assignment genetic algorithm technique for university timetabling. *The Practice and Theory of Automated Timetabling III: Selected Papers from 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT III), Konstanz, Germany, Lecture Notes in Computer Science 2079, Springer-Verlag. (Editors: E.K. Burke and W. Erben), pp 48-63.*

S. Voß, S. Martello, I.H. Osman and C. Roucairol. (1999). Editors: *Meta-heuristics-Advances and Trends in Local Search Paradigms for Optimisation*, Kluwer Academics Publishers, Dordrecht, The Netherlands.

D.J.A. Welsh and M.B. Powell. (1967). An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10(1), pp 85-86.

G.M. White and P.W. Chan. (1979). Towards the construction of optimal examination schedules. *INFOR*, 17(3), pp 219-229.

G.M. White and M. Haddad. (1983). A heuristic method for optimising examination timetabling schedules which have day and night course. *Computers and Education*, 7, pp 235-238.

G.M White and B.S Xie. (2001). Examination timetables and tabu search with longer-term memory. The Practice and Theory of Automated Timetabling III: Selected Papers from 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT III), Konstanz, Germany, Lecture Notes in Computer Science 2079, Springer-Verlag. (Editors: E.K. Burke and W. Erben), pp 85-103.

G.M. White and J. Zhang. (1998). Generating complete university timetables by combining tabu search with constraint logic. The Practice and Theory of Automated Timetabling II: Selected Papers from 2nd International Conference on the Practice and Theory of Automated Timetabling (PATAT II), Toronto, Canada, Lecture Notes in Computer Science 1408, Springer-Verlag. (Editors: E.K. Burke and M. Carter), pp 187-198.

G.M White, B.S Xie and S. Zonjic. (2004). Using tabu search with longer-term memory and relaxation to create examination timetables. European Journal of Operational Research, pp 153, 80-91.

V. Wiers. (1997). A review of the applicability of OR and AI scheduling techniques in practice. Omega, 25(2), pp 145-153.

A. Wren. (1996). Scheduling, timetabling and rostering – A special relationship? The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT I), Edinburgh, UK, Lecture Notes in Computer Science 1153, Springer-Verlag. (Editors: E.K. Burke and P. Ross), pp 46-75.

M. Wright. (2001). Subcost-guided search – Experiments with timetabling problems. Journal of Heuristics, 7, pp 251-260.

M. Yagiura, S. Iwasaki, T. Ibaraki and F. Glover. (2004). A very large-scale neighbourhood search algorithm for the multi-resource generalised assignment problem. Discrete Optimisation, 1, pp 87-98.

Y. Yang and S. Petrovic. (2005). A novel similarity measure for heuristic selection in examination timetabling. *The Practice and Theory of Automated Timetabling V: Selected Papers from 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT V)*, Pittsburg, USA, Lecture Notes in Computer Science 3616, Springer-Verlag. (Editors: E.K. Burke and M. Trick), pp 247-269.

K. Zervoudakis and P. Stamatopoulos. (2001). A generic object-oriented constraint-based model for university course timetabling. *The Practice and Theory of Automated Timetabling III: Selected Papers from 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT III)*, Konstanz, Germany, Lecture Notes in Computer Science 2079, Springer-Verlag. (Editors: E.K. Burke and W. Erben), pp 28-47.