Fuzzy Methodologies for Automated University Timetabling Solution Construction and Evaluation

by Hishammuddin Asmuni, MSc

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

April 2008

To my loving family - Asmah Yunos, Irfan Fikri and Ainul Nadhirah.

Contents

List of Figures v

List of Tables viii

Abstract xiii

Acknowledgement xv

1 Introduction 1

1.1 Background and Motivation . . . 1
1.2 Aims and Scope . . . 2
1.3 Overview of this Thesis . . . 4

I Background 6

2 Review of the State of the Art 7

2.1 Description of the Timetabling Problem . . . 7
2.1.1 Introduction . . . 7
2.1.2 University Examination Timetabling . . . 9
2.1.3 University Course Timetabling . . . 13
2.2 Previous Research in University Timetabling . . . 16
2.2.1 The General Framework . . . 16
2.2.2 Sequential Constructive Approaches . . . 17
2.2.3 Iterative Improvement Methods . . . 24
2.3 Evaluation of Timetable Quality . . . 32
2.3.1 Data Sets and Problem Descriptions . . . 33
2.3.2 Existing Evaluation Functions . . . 37
2.3.3 Multi-objective and Multi-criteria Approaches . . . 40
2.4 The Need for Fuzzy Techniques in Timetabling . . . 43

2.5 Fuzzy Techniques in Timetabling 44
2.6 Generalisation of Problem Solving Approaches 47
2.7 Chapter Summary 50

3 Theory of Fuzzy Sets and Fuzzy Systems 51
3.1 Introduction 51
3.1.1 Fuzzy Sets and Membership Functions 52
3.1.2 Linguistic Variables, Values and Rules 54
3.1.3 Fuzzy Operators 57
3.1.4 Fuzzy Hedges 58
3.1.5 Defuzzification Methods 59
3.1.6 Overview of Fuzzy Systems 62
3.2 Chapter Summary 67

II Fuzzy Construction 68

4 Fuzzy Multiple Heuristic Orderings for Examination Timetabling 69
4.1 Introduction 69
4.2 The Basic Sequential Constructive Algorithm 70
4.3 Why Fuzzy Multiple Heuristic Orderings? 74
4.4 The Fuzzy Multiple Heuristic Ordering 76
4.4.1 Fuzzy Modeling 77
4.4.2 Experiments and Results 88
4.5 Consistency of the Different Heuristic Ordering 99
4.5.1 Experimental Results 99
4.5.2 Performance Analysis and Discussions 103
4.6 Chapter Summary 111

5 Comparison of Fuzzy and Non-Fuzzy Multiple Heuristic Ordering 113
5.1 Introduction 113
5.2 Extension to Three Heuristic Ordering 114
5.2.1 Algorithmic Changes to Reduce Computational Time 114
5.2.2 Experiments with Revised Algorithm 119
5.2.3 Experimental Results 121
5.2.4 Discussion of Results 131
5.3 Chapter Summary 141

6 Generalisation of the Fuzzy Multiple Heuristic Ordering **142**
6.1 Introduction 142
6.2 Application to Course Timetabling 143
6.2.1 Problem Definition 144
6.2.2 Experimental Results 146
6.2.3 Discussion of Results 148
6.3 Alternative Combinations of Heuristic Orderings 150
6.3.1 Experimental Results 155
6.3.2 Discussion of Results 161
6.4 Alternative Approaches to Tuning the Fuzzy System 165
6.4.1 Tuning Fuzzy Rules with Fixed Membership Functions 165
6.4.2 Randomly Generated Fuzzy Models 167
6.4.3 Testings and Results 172
6.5 Chapter Summary 177

III Fuzzy Evaluation **179**

7 A Novel Fuzzy Approach to Evaluate the Examination Timetabling **180**
7.1 Introduction 181
7.2 Assessing Timetable Quality 181
7.2.1 Disadvantages/Drawbacks of Current Evaluation Functions 181
7.2.2 The Proposed Fuzzy Evaluation Function 183
7.2.3 Input Normalisation 185
7.3 Preliminary Investigations 188
7.3.1 Experiments Setup 188
7.3.2 Experimental Results 190
7.3.3 Discussion 196
7.4 Chapter Summary 201

8 Determination of Boundary Settings **203**
8.1 Introduction 203
8.2 Approximate Boundaries using Weighting Factors 205
8.2.1 Approximate Boundaries for *Average Penalty* 205
8.2.2 Approximate Boundaries for *Highest Penalty* 214
8.3 Algorithmic Determination of the Lower Boundary 215
8.3.1 *Brute Force Lower Limit Approximation Algorithm* 216

8.3.2 *Greedy Lower Limit Approximation Algorithm* . . . 217
8.3.3 Comparison of Lower Limit Algorithms . . . 220
8.3.4 Algorithmic Derivation of Boundaries . . . 223
8.4 Evaluation of Boundary Settings . . . 224
8.4.1 Methods . . . 224
8.4.2 Results . . . 225
8.4.3 Discussion . . . 225
8.5 Review of Previously Published Results . . . 237
8.6 Chapter Summary . . . 239

9 Conclusions and Future Work **242**
9.1 Summary of Contributions . . . 243
9.1.1 Fuzzy Construction of Timetables . . . 243
9.1.2 Fuzzy Evaluation of Timetables . . . 245
9.2 Future Research . . . 246
9.3 Dissemination . . . 248
9.3.1 Journal Paper . . . 248
9.3.2 Conference Papers . . . 249
9.3.3 Abstract . . . 250

A Analysis of Modified Algorithms **251**

B Crisp Values for the 35 Solutions **261**

Bibliography **278**

List of Figures

2.1 Phases in constructing solutions for timetabling problem 17

3.1 Membership function for the set of cold temperatures, defined as $cold = \{x|x \leq 10\}$ 52
3.2 Membership function for the fuzzy set $cold = \{x \mid x \textit{ is less than about } 10\}$ 53
3.3 Membership functions for the linguistic variable 'temperature' 55
3.4 Some common membership functions 56
3.5 Fuzzy sets operations (adapted from Negnevitsky (2002, Chap. 4)) 57
3.6 Apply hedge 'very' onto linguistic value 'warm' 59
3.7 The Centre of Gravity (COG) method of defuzzification 60
3.8 The Mean of Maxima (MOM) method of defuzzification 61
3.9 The Smallest of Maxima (SOM) and The Largest of Maxima (LOM) methods of defuzzification 61
3.10 The Bisector of Area (BOA) method of defuzzification 62
3.11 Components of fuzzy system 63
3.12 Characteristic of linguistic variables 64
3.13 The fuzzified value for both input linguistic variables 65
3.14 Evaluation of rules fulfillment (firing levels) 66
3.15 Aggregation of rules 66
3.16 Defuzzification of final shape 67

4.1 A general framework for producing timetabling solutions 71
4.2 Pseudo code for the '*rescheduling procedure*' used if 'skipped' exams exist 72
4.3 Example of examinations ordered by various combinations of heuristics . 75
4.4 The steps involved in a fuzzy version of *Process 1* (from Fig. 4.1) 78
4.5 A nine-rule Mamdani inference process 81
4.6 Membership functions for *Fixed Fuzzy LD+LE Model* 84
4.7 The membership function for tuned fuzzy model 87

4.8 Range of possible membership functions 87
4.9 Best fuzzy model for data sets *CAR-F-92*, *CAR-S-91*, *EAR-F-83*, *HEC-S-92*, *KFU-S-93* and *LSE-F-91* 93
4.10 Best fuzzy model for data sets *RYE-F-92*, *STA-F-83*, *TRE-S-92*, *UTA-S-92*, *UTE-S-92* and *YOR-F-83* 94

5.1 The modified algorithm 118
5.2 Pseudo code for the new '*rescheduling procedure*' 119

6.1 Possible combinations of two heuristic orderings 152
6.2 Possible combinations of three heuristic orderings 153
6.3 Pseudo-code for *Tuned Fuzzy Rules 1* 168
6.4 Pseudo-code for *Tuned Fuzzy Rules 2* 169
6.5 An example of defining a random fuzzy model 171
6.6 Pseudo-code for *Random Model 1* 173
6.7 Pseudo-code for *Random Model 2* 174

7.1 Gaussian membership function for $\mu(x_k, \sigma_k)$ 185
7.2 Membership functions for input and output variables 186
7.3 Fuzzy rules for *Fuzzy Evaluation Function* 187
7.4 Indicative illustrations of the range of normalised inputs and associated output obtained for the *LSE-F-91* and *TRE-S-92* data sets 192
7.5 Firing level for **Rule 1** with different normalised input values 199
7.6 A graphical comparison of the effect of the two boundary settings for *UTA-S-92* 200

8.1 Illustration of boundary coverage concept 204
8.2 Boundary coverage using weighted factors 205
8.3 Pseudo code for approximation of maximum total penalty, P_{max} 207
8.4 A graphical illustrations of P_{max} calculations for *LSE-F-91* 208
8.5 Pseudo code for *BFLLAA* for three enroled exams 218
8.6 Illustrative example of BFLLAA for 3 enroled exams 219
8.7 Pseudo code for *BFLLAA* for four enroled exams (continue from Figure 8.5)220
8.8 Pseudo code for *GLLAA* 221
8.9 A comparison of rankings produced by the five boundary settings used for *CAR-F-92* 226

8.10 A comparison of rankings produced by the five boundary settings used for *CAR-S-91* 227
8.11 A comparison of rankings produced by the five boundary settings used for *EAR-F-83* 228
8.12 A comparison of rankings produced by the five boundary settings used for *HEC-S-92* 229
8.13 A comparison of rankings produced by the five boundary settings used for *KFU-S-93* 229
8.14 A comparison of rankings produced by the five boundary settings used for *LSE-F-91* 230
8.15 A comparison of rankings produced by the five boundary settings used for *RYE-F-92* 230
8.16 A comparison of rankings produced by the five boundary settings used for *STA-F-83* 231
8.17 A comparison of rankings produced by the five boundary settings used for *TRE-S-92* 231
8.18 A comparison of rankings produced by the five boundary settings used for *UTA-S-92* 232
8.19 A comparison of rankings produced by the five boundary settings used for *UTE-S-92* 232
8.20 A comparison of rankings produced by the five boundary settings used for *YOR-F-83* 233
8.21 Illustrative of minimum exams that cause penalty 236

List of Tables

2.1 Examination timetabling problem characteristics. The twelve problem instances that used in this research are highlighted in bold font. 36

2.2 The capacitated problem specifications 39

3.1 Examples of hedges (taken from Negnevitsky (2002, Chap. 4)). For the graphical representation, the thicker line is the new shape when the hedge act on the linguistic value. 58

4.1 Minimum and maximum values for heuristic *LD*, *LE*, *SD* and *WLD* for each data set. The minimum and maximum values for heuristic *LCD* is similar to *LD*. 79

4.2 Fuzzy rule set for *Fixed Fuzzy LD+LE Model* 85

4.3 The fuzzy rule set for the *Fuzzy LD+LE Model* 89

4.4 The fuzzy rule set for the *Fuzzy SD+LE Model* 89

4.5 The fuzzy rule set for the *Fuzzy SD+LD Model* 89

4.6 Experimental results for single and fuzzy multiple heuristic orderings . . 91

4.7 Values for *cp* parameters 92

4.8 Comparison of *cp* parameters combinations for the best fuzzy model and the second best fuzzy model 95

4.9 Results comparison 100

4.10 The penalty costs obtained by the different heuristic orderings on each of the twelve benchmark data sets. In each case the best result, the worst result, the average result and the standard deviation obtained over 30 repeated runs are given. 101

4.11 The number of skipped exams obtained due to the fact that there was no valid time slot available in the first attempt to assign the exam into the time slots — i.e. the number of exams in the skipped list after *Process 5* 103

4.12 The number of iterations of the '*rescheduling procedure*' required for each data set . . . 104
4.13 A comparison of the computational time (in seconds) required to construct the solutions for each heuristic ordering methods for each data set . . . 105

5.1 A comparison of the results obtained by the different algorithms on the *CAR-S-91*, *KFU-S-93* and *UTA-S-92* data sets . . . 116
5.2 Fuzzy rule set for *Fuzzy LD+LE+SD Model* . . . 121
5.3 Values for weighted factors identified in the tuning process . . . 122
5.4 Values for *cp* parameters obtained from the fuzzy tuning process . . . 122
5.5 The penalty costs obtained by the different non-fuzzy heuristic orderings on each of the twelve benchmark data sets . . . 124
5.6 The penalty costs obtained by the different fuzzy multiple heuristic orderings on each of the twelve benchmark data sets . . . 127
5.7 The Best Fuzzy, Best Linear and t-Test (Two-Sample Assuming Unequal Variances) Result of the twelve benchmark data sets . . . 128
5.8 The number of '*rescheduling procedure*' required for non-fuzzy heuristic orderings for each data set . . . 129
5.9 The number of '*rescheduling procedure*' required for fuzzy multiple heuristic orderings for each data set . . . 130
5.10 A comparison of the computational time (in seconds) required to construct the solutions for non-fuzzy heuristic ordering for each data set . . . 132
5.11 A comparison of the computational time (in seconds) required to construct the solutions for fuzzy multiple heuristic orderings for each data set . . . 133
5.12 A comparison of the average computational time required for *Fuzzy SD+LE Model* when old algorithm (*ALG1.0*) were run in two different computers. Values in the second column are extracted from Table 4.13, and values in the third column are extracted from Table A.2. For each data set, the percentage improvement is shown in the fourth column. . . . 134
5.13 A comparison of the average computational time required for *Fuzzy SD+LE Model* when the new algorithm (*ALG2.0*) and the old algorithm (*ALG1.0*) were run in two different computers. Values in the second column are extracted from Table 4.13, and values in the third column are extracted from Table 5.11. For each data set, the percentage improvement is shown in the fourth column. . . . 135

5.14 Comparison of results for single heuristic orderings using two different algorithms . . . 137
5.15 Comparison of results for two heuristic orderings used simultaneously when two different algorithms were applied . . . 138
5.16 A comparing of results obtained herein with results published by other researchers . . . 140
5.17 A comparison of results obtained using different *constructive* approaches 141

6.1 Course timetabling problem characteristics . . . 145
6.2 Comparison of solution quality between *Single Heuristic Ordering* and *Fuzzy Multiple Heuristic Ordering* . . . 147
6.3 Comparison of number of iterations of '*rescheduling procedure*' required to produce the solutions shown in Table 6.2 . . . 148
6.4 Comparison of solution quality with other results in literature . . . 149
6.5 Fuzzy rule set when combining two heuristic orderings (with *SD* as one of the variable) . . . 153
6.6 Fuzzy rule set when combining two heuristic orderings (without *SD*) . . . 154
6.7 Fuzzy rule set when combining three heuristic orderings (with *SD* as one of the variable) . . . 154
6.8 Fuzzy rule set when combining three heuristic orderings (without *SD*) . . 154
6.9 A comparison of results for five *Single Heuristic Ordering* with last time slot selection . . . 157
6.10 A comparison of results for five *Single Heuristic Ordering* with random time slots selection . . . 158
6.11 Experimental results for two heuristic orderings applied simultaneously . 159
6.12 Experimental results for three heuristic orderings applied simultaneously 160
6.13 A comparison of 'best' penalty cost for the five single heuristic orderings. The lowest value are highlighted with bold font. . . . 163
6.14 A comparison of average penalty cost for the five single heuristic orderings. The lowest value are highlighted with bold font. . . . 163
6.15 A comparison of 'best' results obtained in Chapter 4, Chapter 5 and this Chapter . . . 164
6.16 Fuzzy rule set for *Fuzzy LD+LE+SD Model* . . . 166
6.17 Integer codes assigned to fuzzy model parameters . . . 170
6.18 A comparison of results for *Fuzzy LD+SD+LE Model* when utilising fixed and tuned fuzzy rules . . . 175

6.19 A comparison of results for tuning fuzzy model randomly 176

7.1 Minimum and maximum values for *Average Penalty* and *Highest Penalty* obtained from the 35 timetable solutions for each data set 191
7.2 A comparison of the results obtained using the two alternative forms of the normalisation process for data sets *CAR-F-92*, *CAR-S-91*, *EAR-F-83* and *HEC-S-92* 193
7.3 A comparison of the results obtained using the two alternative forms of the normalisation process for data sets *KFU-S-93*, *LSE-F-91*, *RYE-F-92* and *STA-F-83* 194
7.4 A comparison of the results obtained using the two alternative forms of the normalisation process for data sets *TRE-S-92*, *UTA-S-92*, *UTE-S-92* and *YOR-F-83* 195
7.5 Range of timetable quality 198

8.1 The 'best' and 'worst' timetable solutions known 210
8.2 Approximate boundaries derived by considering all students 211
8.3 Distribution of students enroled for a particular number of exams 212
8.4 Approximate boundaries derived by excluding students with only one exam 213
8.5 A comparison of the range of boundary settings for *average penalty* . . . 213
8.6 The boundary settings for *highest penalty* 215
8.7 *lowerLimit* values for *average penalty* calculated using *BFLLAA* and *GLLAA* 222
8.8 Boundary settings for *average penalty* using *BFLLAA lowerLimit* 224
8.9 A comparison of the results of fuzzy evaluation obtained by using the approximate boundary settings based on the three new methods introduced in this Chapter 226
8.10 Analysis of students involved in calculating the lower limit proximity cost 237
8.11 A comparison between lower limit with the 'best' results in literature . . 239

A.1 Analysis of Changes in Algorithm for *Tuned Fuzzy LD+LE Model* 251
A.2 Analysis of Changes in Algorithm for *Tuned Fuzzy LD+LE Model* 256

B.1 Crisp values of *average penalty* and *highest penalty* for *CAR-F-92*, *CAR-S-91* and *EAR-F-83*. 262
B.2 Crisp values of *average penalty* and *highest penalty* for *HEC-S-92*, *KFU-S-93* and *LSE-F-91*. 263

B.3 Crisp values of *average penalty* and *highest penalty* for *RYE-F-92*, *STA-F-83* and *TRE-S-92*. 264
B.4 Crisp values of *average penalty* and *highest penalty* for *UTA-S-92*, *UTE-S-92* and *YOR-F-83*. 265

Abstract

This thesis presents an investigation into the use of fuzzy methodologies for University timetabling problems. The first area of investigation is the use of fuzzy techniques to combine multiple heuristic orderings within the construction of timetables. Different combinations of multiple heuristic ordering were examined, considering five graph-based heuristic orderings - *Largest Degree*, *Saturation Degree*, *Largest Enrolment*, *Largest Coloured Degree* and *Weighted Largest Degree*. The initial development utilised only two heuristic orderings simultaneously and subsequent development went on to incorporate three heuristic orderings simultaneously. A central hypothesis of this thesis is that this approach provides a more realistic scheme for measuring the difficulty of assigning events to time slots than the use of a single heuristic alone. Experimental results demonstrated that the fuzzy multiple heuristic orderings (with parameter tuning) outperformed all of the single heuristic orderings and non-fuzzy linear weighting factors. Comprehensive analysis has provided some key insights regarding the implementation of multiple heuristic orderings.

Producing examination timetables automatically has been the subject of much research. It is generally the case that a number of alternative solutions that satisfy all the hard criteria are possible. Indeed, there are usually a very large number of such feasible solutions. Some method is required to permit the overall quality of different solutions to be quantified, in order to allow them to be compared, so that the 'best' may be selected. In response to that demand, the second area of investigation of this thesis is concerned with a new evaluation function for examination timetabling problems. A novel approach, in which fuzzy methods are used to evaluate the end solution quality, separate from the objective functions used in solution generation, represents a significant addition to the literature.

The proposed fuzzy evaluation function provides a mechanism to allow an overall decision in evaluating the quality of a timetable solution to be made

based on common sense rules that encapsulate the notion that the timetable solution quality increases as both the *average penalty* and the *highest penalty* decrease. New algorithms to calculate what is loosely termed the 'lower limits' and 'upper limits' of the proximity cost function for any problem instance are also presented. These limits may be used to provide a good indication of how good any timetable solution is. Furthermore, there may be an association between the proposed 'lower limit' and the formal lower bound. This is the first time that lower limits (other than zero) have been established for proximity cost evaluation of timetable solutions.

Acknowledgements

I would like to express my gratitude to my supervisors, Dr. Jonathan Garibaldi, Professor Edmund Burke and Dr. Barry McCollum, for their expertise, encouragement, support, and valuable advice during my research.

I would also like to thank the Ministry of Science, Technology and Innovation (MOSTI) of Malaysia and the Universiti Teknologi Malaysia (UTM) for the doctoral scholarship and other financial support throughout the course of my PhD study.

Special thanks to my lovely wife Asmah Binti Yunos, my son Irfan Fikri, my daughter Ainul Nadhirah, my parents, my brother and sisters for their love, patience and understanding throughout the study.

To all ASAP Group members, thank you for the friendly working environment in the group.

Chapter 1

Introduction

1.1 Background and Motivation

The problem of timetabling examinations and courses is of much interest and concern to academic institutions. The basic problem is to allocate a time slot and a room for all events (exams, lectures, seminars, tutorials) within a limited number of permitted time slots and rooms in order to find a feasible timetable. This assignment process is subject to 'hard' constraints which *must* be satisfied in order to get a feasible timetable. An example of such constraint is that no student is required to attend two events at the same time.

In addition, it is also important to build a *good quality* lecture timetable that considers not only the administration requirements, but also takes into account lecturers' and students' preferences. It is generally desirable (but not essential) to satisfy these preferences and, as such, they are termed 'soft' constraints.

As this task is time consuming and tedious to carry out manually, much effort has been directed over the last few decades to generate timetables automatically. With a large number of events needing to be assigned to resources (time slots and rooms) and a list of constraints (both hard and soft) needing to be addressed, there are a large number of potential solutions to this problem. Furthermore, the process of generating timetables is complex, with a number of key decision points. Two major decision points are how to construct feasible solutions and how to evaluate their effectiveness (essentially, how to decide which of several alternative solutions is 'the best'). Many factors need to be considered in both these key decision areas, with much information being available. To date, there has been relatively little research into how the available information can be combined, with the goal of achieving better solutions.

Since Zadeh introduction the notion of fuzzy sets in 1965 (Zadeh, 1965), fuzzy methodologies have been widely utilised in a number of decision support contexts. Indeed, fuzzy methodologies have made significant impact in many areas, including consumer technologies such as fuzzy logic auto-focus digital cameras and fuzzy washing machines. It has been shown that such fuzzy approaches can be successful in combining multiple sources of information (Zimmermann, 1996). The motivation for the work presented in this thesis is to investigate whether the use of fuzzy methodologies could be of benefit in automating the decision making process in the construction and evaluation of solutions to the examination timetabling problem. Although the main focus of this thesis is on examination timetabling, the solution construction technique was also applied to course timetabling.

1.2 Aims and Scope

The first area of investigation, described in Chapters 4 to 6, is an exploration of how fuzzy techniques can be employed to combine multiple heuristics within the construction of timetables. During the process of construction, the order in which exams are assigned to time slots has been shown to have a major effect on the eventual solution. An assessment of how difficult it is to place a given exam into a timetable (in effect, some measure of how hard it is to satisfy the constraints relevant to the particular exam) is often used to guide the order of placement. The usual strategy is to place the most difficult exams first, on the basis that it is better to leave the easier exams until later in the process when there are fewer time slots remaining. There are many different criteria that may be used when assessing this difficulty.

A common approach has been to employ graph based heuristics (a heuristic is an approximate rule or a 'rule-of-thumb' (Burke and Kendall, 2005, Chap. 1)) to provide a quantitative indication of difficulty. This measure is then used to determine the order in which the exams are assigned into the timetable and, hence, are referred to as 'heuristic orderings'. Examples of such heuristics are the number of other exams in conflict with the given exam, the number of students enroled on each exam, etc. Detailed descriptions of these heuristic orderings are given in Section 2.2.2. In this thesis, for the first time, fuzzy methodologies are used to combine multiple heuristics *simultaneously* in order to provide a measure of the difficulty of placing each exam. This measure is then used to order (rank) the exams for assignment. Various combinations of heuristics are investigated in the construction process. To investigate the wider applicability of this novel fuzzy approach, the techniques were also applied to the domain of course timetabling.

The second major area of investigation, described in Chapters 7 and 8, is the use of fuzzy methodologies in the evaluation of the quality of timetable solutions. It is generally the case that a number of alternative solutions that satisfy all the hard criteria are possible. Indeed, there is usually a very large number of such feasible solutions. Some method is required to permit the overall quality of different solutions to be quantified, in order to allow them to be compared, so that the 'best' may be selected. In principle, a range of different measures of quality might be used to evaluate how well a given solution satisfies the various soft constraints. Such a measure is termed an 'objective function' which can be used either to evaluate a range of solutions manually, or can be used in an automated process to determine the best solution. Again, in principle, a number of alternative objectives can be combined into a single objective function or can be kept separate in a multi-objective framework. The trade-offs between different objectives underpin the motivation for studying multi-objective methods. In this thesis, fuzzy methodologies are employed to evaluate the quality of solutions using a number of identified key criteria, *after* a variety of alternative solutions have been produced.

There are a number of objectives that were addressed in order to accomplish the primary aim of the research which can be outlined as follows:

1. to investigate the use of fuzzy techniques to combine, initially, two heuristics simultaneously in ordering events in examination timetabling;
2. to compare the fuzzy combination of heuristics with a non-fuzzy approach;
3. to expand the investigation to consider three heuristics simultaneously;
4. to investigate the wider applicability of the technique through its application to course timetabling;
5. to explore the use of fuzzy techniques in the evaluation of constructed solutions; and
6. to establish the boundaries of the fuzzy evaluation method in order to determine how good a solution actually is.

1.3 Overview of this Thesis

The remaining Chapters of this thesis are divided into three parts. Part I describes the timetabling problem in general, distinguishing examination and course timetabling, and goes on to describe the current state of research in examination timetabling and the basics of fuzzy set theory. In Part II (which covers Chapters 4 to 6) the implementation of fuzzy approaches in constructing solutions to examination timetabling is described. Part III (which covers Chapter 7 and 8) presents a novel fuzzy approach to evaluate the quality of timetables. The individual Chapters of this thesis are summarised below.

Chapter 2 provides a description of educational timetabling problems and presents a review of different algorithms and approaches developed in attempting to automate the generation of solutions to University timetabling problems. The examination timetabling benchmark data sets that are used in this research are also described together with a description of objective functions currently used in the evaluation of timetable solution quality. Chapter 3 provides a description of fuzzy set theory and fuzzy reasoning. This is a self-contained Chapter that provides the material necessary for understanding the basic features of the fuzzy techniques used in this thesis. This self-contained Chapter is intended for readers who are not familiar with fuzzy methodologies.

In Chapter 4, a new fuzzy approach that uses two heuristic orderings simultaneously to measure the difficulty of assigning exams into time slots is developed and tested on the benchmark data sets. The aim of this initial study was to investigate the effects of using multiple heuristic ordering as compared to a single heuristic ordering. Chapter 5 presents a comparison of fuzzy and non-fuzzy multiple heuristic ordering approaches. The technique implemented in Chapter 4 is further enhanced to include the use of three heuristic orderings simultaneously. In Chapter 6, a generalisation of the technique is investigated. First, the suitability of fuzzy multiple heuristic ordering in course timetabling is assessed. Then, an exploration was carried out of all possible combinations of orderings using either two or three heuristics simultaneously, from a set of five heuristics. Finally, a range of methods to tune the fuzzy models utilised in these techniques were investigated.

Chapter 7 presents a new fuzzy evaluation function for examination timetabling, based on both how good the constructed timetable is as a whole and on how good the solution is for individual students. In Chapter 8, two algorithms for determining lower boundaries of the quality of solutions based on the underlying structure of the problem are presented. Finally, Chapter 9 provides some concluding remarks and suggestions for future research that arise from the work presented in this thesis.

Part I

Background

Chapter 2

Review of the State of the Art

2.1 Description of the Timetabling Problem

2.1.1 Introduction

The Oxford Advanced Learner's Dictionary defines a timetable as 'a list showing the times at which particular events will happen'. Wren (1996) described timetabling as a special type of scheduling. He defined timetabling as follows:

> *"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."*

Since the early 1960's, an enormous number of research papers reporting work on timetabling problems have appeared in the literature. After more than 40 years, research in this area is still very active and new research directions are continuing to emerge. Examples of recent papers can be found in the series of Proceedings of the International Conference on the Practice and Theory of Automated Timetabling (PATAT) (Burke and Carter, 1998; Burke and Causmaecker, 2003; Burke and Erben, 2001; Burke and Ross, 1996; Burke and Rudová, 2006; Burke and Trick, 2005). Overviews and surveys can be found in papers by Bardadym (1996); Burke and Petrovic (2002); Burke *et al.* (1997); Carter (1986); Carter and Laporte (1996); de Werra (1985); Petrovic and Burke (2004); Qu *et al.* (2006); Schaerf (1999); Schmidt and Strohlein (1980).

With regard to educational timetabling, generally, the problems can be classified into three types (Schaerf, 1999; Schaerf and Di Gaspero, 2001), each with their own specific characteristics and constraints:

School timetabling These are problems that are concerned with assigning the weekly lessons in schools. The aim is to assign a set of teachers to a set of classes (groups of students) for a set of lessons (subjects to be taught) in a set of time periods, while at the same time satisfying a set of constraints. There are many variations to the basic problem. For example, in junior (lower) schools, sometimes a single teacher remains in the same room with the same class for the whole day, teaching a variety of subjects in the various time periods, whereas in secondary (high) schools, teachers may remain in the same room while different classes are taught different lessons (in the same subject) throughout the day, or a teacher may move between rooms for different lessons. Examples of hard constraints are that no teacher may teach in two different rooms in the same time period and that no classes can can have different lessons at the same time. Soft constraints may cover issues such as rest periods for teachers (these may also be hard constraints), teachers preferences for certain rooms and / or specific timing of certain lessons. Further examples of constraints are listed by Costa (1994) and Loo *et al.* (1985). As school timetabling is out of the scope of this thesis, no further mention will be made of it.

University examination timetabling This problem is concerned with assigning a set of (course or subject specific) examinations, each of which a group of students is enroled in, to a given set of time slots. A typical hard constraint is that no student can be timetabled to sit more than one exam at the same time. Further details of the problem specification and examples of hard and soft constraints are given in the following Section.

University course timetabling This timetabling problem is concerned with assigning courses and associated events to time slots, groups of students and lecturers in such a way that no conflict occurs in any period, and the number of students assigned to a room is no more than the maximum room capacity. For more details, see Section 2.1.3 below.

Basically, these three types of timetabling problems share the same general characteristic of the need to assign events to time slots while minimising the constraint violations. However, key differences remain between these problems. For example, in the examination timetabling problem, groups of students can sometimes be brought together into one room (to take different exams at the same time). This is *not* the case in course timetabling. See Carter and Laporte (1998) for a more detailed description of the differences between school and university course timetabling. For a full description of the differences between university examination and course timetabling, see McCollum (2006).

2.1.2 University Examination Timetabling

Carter *et al.* (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."*

Examination timetabling is essentially the problem of allocating exams to a limited number of time periods in such a way that none of the specified hard constraints are violated. A timetable which satisfies all hard constraints is often called a *feasible* timetable. In addition to the hard constraints, there are often many soft constraints whose satisfaction is desirable (but not essential). The set of constraints which need to be satisfied is usually very different from one institution to another, as reported by Burke *et al.* (1996a). Examples of common hard constraints are:

- the requirement to avoid any student being timetabled for two different exams at the same time;
- no unscheduled exam(s) exist at the end of the timetabling process;
- room capacity should be able to accommodate all students who are enroled for the exam(s) scheduled in the particular room(s).

In practice, each institution usually has a different way of evaluating the quality of a feasible timetable. In many cases, the measure of quality is calculated based upon a penalty function which represents the degree to which the soft constraints are satisfied. Example of soft constraints are as follows:

- Exam X shall be scheduled before/after exam Y.
- Avoid students having to sit exams in consecutive time slots.
- Exams with a large number of students should be scheduled earlier in the timetable.
- Only certain time slots and/or rooms may be available for particular exams.
- Exams with questions in common should be scheduled in the same time slot.

More details of constraints for examination timetabling are listed by Burke *et al.* (1996a), and Di Gaspero and Schaerf (2001).

Several models and formulations for timetabling problems have been presented by various researchers. There is a well known analogy between a basic version of the timetabling problem (with no soft constraints) and the graph colouring problem (Burke *et al.*, 1994c, 2004b; Carter *et al.*, 1996; de Werra, 1985; Welsh and Powell, 1967). In the graph colouring problem, the goal is to find the minimum number of colours which can be used to colour the graph vertices in such a way that none of the connected adjacent vertices are coloured with the same colour. This minimum number of colours is known as the 'chromatic number' of a graph. The timetabling problem, in its simplest form (without soft constraints), can be represented as a graph colouring problem, in which the nodes represent the exams, colours represent the time slots and the edges represent the conflict between exams (Burke *et al.*, 2004b). Hence, if the examination timetabling problem is considered as a graph colouring problem, the aim is to find the minimum number of time slots which are able to accommodate all the exams without any clashes.

By analysing the student enrolment list, the exams that conflict (for example, exams that have at least one common student) can be identified. Cole (1964) represented the conflicting exams using the 'incompatibility table', while Broder (1964) used the term 'conflict matrix' for the same thing. The conflicting exams are represented by a conflict matrix, $C = [c_{ij}]_{NxN}$ where $i, j \in \{1, ..., N\}$ (N is the number of exams). Element c_{ij} denotes the number of students enroled for both exam i and exam j. When a non-weighted graph is employed, it is also possible to use $c_{ij} = 1$ if there is conflict between exam i and exam j; $c_{ij} = 0$ otherwise. It is a symmetrical matrix, i.e. element $c_{ij} = c_{ji}$. For diagonal cells (i.e. $i = j$), each cell either contains the number of students enroled for the particular exam (c_{ij} = number of students for exam i) or the cell contains zero ($c_{ij} = 0$) to denote that there is no conflict. Either is acceptable, depending on how the information stored in the conflict matrix is utilised. Essentially, several pieces of information can be generated from the conflict matrix that are related to graph theory. The number of exams in conflict for an exam is equivalent to the node degree. Node degree values are utilised when heuristic orderings (e.g. *Largest Degree*, *Largest Coloured Degree* and *Weighted Largest Degree*, see Section 2.2.2) are employed to order the exams by difficulty when constructing solutions. It is also possible to use the diagonal cell values for the heuristic ordering *Largest Enrolment* if c_{ij} is not set to zero when $i = j$.

Mathematical models also exists for the examination timetabling problem. Consider the following notations (adopted from Marín (1998)):

- N is the number of exams
- P is the number of time slots available
- $T = [t_{np}]_{NxP}$ is the matrix which represents assignments of the exams into time slots where $n \in \{1, ..., N\}$ and $p \in \{1, ..., P\}$.

$$t_{np} = \begin{cases} 1, & \text{if exam } n \text{ is scheduled in time slot } p \\ 0, & \text{otherwise} \end{cases}$$

- Z_{np} is the cost of scheduling exam n in time slot p
- $Y_{nm} = 1$ if exam n clashes with exam m (i.e. exam n and m have common students), and 0 otherwise

For any timetable solution construction, the objective is to minimise

$$\sum_{n=1}^{N} \sum_{p=1}^{P} Z_{np}\, t_{np} \tag{2.1}$$

subject to:

$$\sum_{p=1}^{P} t_{np} = 1, \; where\ n \in \{1, ..., N\} \tag{2.2}$$

$$\sum_{n=1}^{N-1} \sum_{m=n+1}^{N} \sum_{p=1}^{P} t_{np}\, t_{mp}\, Y_{nm} = 0 \tag{2.3}$$

Equation 2.2 denotes that all exams must be scheduled by the end of the timetabling process. Equation 2.3 denotes the requirement that a student is not able to attend more than one exam at the same time. As there are many different criteria that can be included when evaluating the timetable quality, the definition of Z_{np} is dependent on which criteria are to be used for the particular educational institution. In the context of the benchmark data sets used in this research, the descriptions of evaluation functions which have been applied to the benchmark data sets are given in Section 2.3.2.

2.1.3 University Course Timetabling

In this thesis, the main focus is on constructing and evaluating solutions to the university examination timetabling problem. The course timetabling problem is only considered in Section 6.2 in the context of exploring the generality of the proposed fuzzy approach to timetable construction. Hence, only a very brief definition of the problem and survey of particularly relevant literature is given here.

A general overview of course timetabling can be found in the paper by Carter and Laporte (1998). They 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."*

Note that, although course timetabling is sometimes also referred to as class-teacher timetabling, e.g. Burke *et al.* (2004b), the term course timetabling is preferred in this thesis. A complete formal description of the problem can be found in Burke *et al.* (2004b).

As stated earlier, in university course timetabling, a set of courses and associated events is assigned to a set of rooms and time periods within a week and, at the same time, students and teachers are assigned to the courses so that the appropriate lessons can take place, subject to a variety of hard and soft constraints. In 2002, Paechter introduced a course timetabling problem instance generator as part of an 'International Timetabling Competition'[1], organised by the Metaheuristics Network and sponsored by the PATAT

[1] http://www.metaheuristics.org/index.php?main=4&sub=44

International Conference series. The objective of the International Timetabling Competition was to create feasible weekly class timetables for a university, in which a number of hard constraints were satisfied, while minimising the number of soft constraints broken. The instance generator was used to produce simplified, but realistic, problem instances, all of which had at least one perfect solution (a solution with no constraint violations, hard or soft). The competition used the following hard constraints:

1. No student is required to attend more than one course at the same time
2. A course can only be scheduled to a room which satisfies the features required by the course
3. A course can only be scheduled to a room which has enough room to accommodate all students registered for it
4. Only one course can be scheduled in one room at any time slot

Solutions which do not violate any of the hard constraints are defined as feasible solutions. Besides these (and other possible) hard constraints, there are a variety of soft constraints which have been proposed and used. The following soft constraints are defined for the data set generation in the competition:

1. No student should be scheduled to attend only one course on a day
2. No course should be scheduled at the last time slot of the day for any student
3. No student should be scheduled to attend more than two courses consecutively in any one day

By definition, it is not compulsory to satisfy the soft constraints for any given problem. Usually, some form of penalty function is used to measure the degree to which the soft constraints are satisfied. Although there is no universally accepted method, often the numbers of students for which each constraint is not satisfied are simply summed.

Automated approaches to course timetabling have been studied over the last thirty years. A comprehensive survey of the early approaches can be found in Carter and Laporte (1998). Other surveys of university timetabling that cover both examination and course problems include Burke *et al.* (1997), Burke *et al.* (2004a), Carter (2001), Schaerf (1999) and Wren (1996). The set of twenty problem instances introduced for the competition itself (three more instances were also generated, to be used as 'unseen' tests) have also been used by a number of authors as a benchmark data set. The competition was won by Kostuch (2005) utilising a 'three-phase approach' featuring simulated annealing, which obtained the best results on 13 out of the 20 problem instances. Burke *et al.* (2003a) also entered the competition, using an approach based on the Great Deluge Algorithm (see Section 2.2.3.1), which obtained the best results on the remaining 7 of the 20 problem instances. Other approaches used in the competition included those based on simulated annealing, a hybrid local search method and several variations of tabu-search. Since the close of the competition, the same twenty data sets have subsequently been used by other authors including Chiarandini *et al.* (2006).

Paechter's test instance generator was used by Socha *et al.* (2002) to generate eleven problem instances of various sizes. They compared a local search method and an Ant Colony Optimisation algorithm on these eleven problem instances and showed that the Ant Colony Optimisation algorithm achieved better performance. The same eleven problem instances have subsequently been used by other authors as a means of comparison (and are also used in this thesis, as described in Section 6.2). Burke, Kendall and Soubeiga (2003c)) introduced a hyper-heuristic (see Section 2.6) that utilised tabu-search in an attempt to raise the level of generality of automated timetabling systems, and the system was used to solve both these course timetabling problem instances and nurse scheduling problems. Burke *et al.* (2007) developed a graph-based hyper-heuristic approach which used a sequence of heuristic orderings to construct the initial solution and then applied steepest descent local search to improve the solution. These data sets were also used by Abdullah *et al.* (2005) who employed a variable neighbourhood search with a fixed length tabu list used to penalise the unperformed neighbourhood structures. Following on from this, Abdullah *et al.* (2006c) applied a randomised iterative improvement method featuring composite neighbourhood structures to the test instances. Finally, real-world data sets have also been used in case studies of university course timetabling by other authors including Avella and Vasil'Ev (2005), Daskalaki *et al.* (2004), Dimopoulou and Miliotis (2004) and Santiago-Mozos *et al.* (2005).

Despite the fact that the problem of timetabling university courses is very different from timetabling university examinations, some authors have blurred the distinctions and/or have applied the same techniques to solve both problems (McCollum, 2006). Hence, in the remainder of the Chapter, in which the main focus is on examination timetabling, some occasional references to key results in course timetabling will be found.

2.2 Previous Research in University Timetabling

2.2.1 The General Framework

Hertz (1991) stated that approaches developed to solve timetabling problems usually consist of two phases. Figure 2.1 shows a general framework for finding solutions to timetabling problems. Normally, in Phase 1, a solution is (or solutions are) constructed by using a sequential construction algorithm. The constructed solutions can be feasible or infeasible. If a solution is infeasible, it can be 'corrected' during the iterative improvement phase (Phase 2).

In Phase 2, the initial solution is modified in order to improve the solution while ensuring the feasibility of the solution. The improvements can be implemented by using any search algorithm such as Genetic Algorithms (Holland, 1992), Tabu Search (Glover, 1986), Simulated Annealing (Kirkpatrick *et al.*, 1983) or the *Great Deluge Algorithm* (Dueck, 1993) (to name just a few approaches). In Section 2.2.3, a brief description

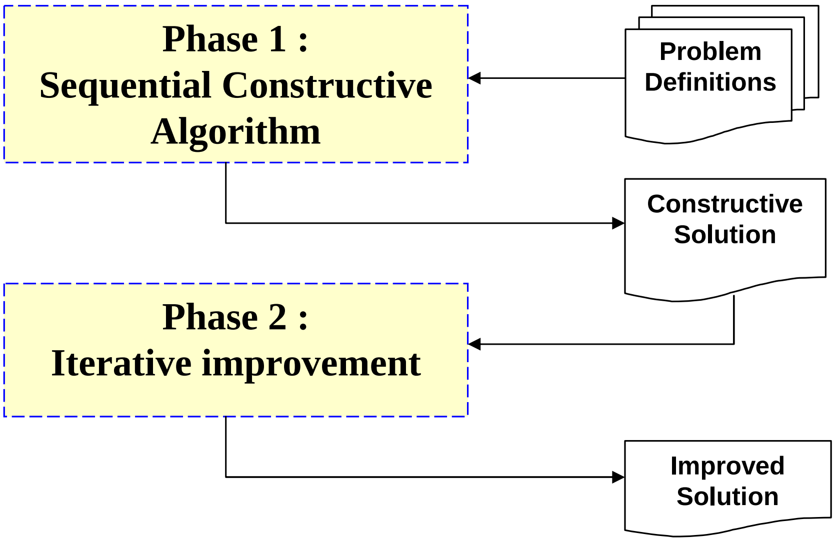


Figure 2.1: Phases in constructing solutions for timetabling problem

of various search algorithms and approaches applied to university timetabling problems is presented.

In the first part of this research, the focus is on the construction process, as constructing feasible solutions is a difficult task especially for large, real-world timetabling problems (Hertz, 1991). A detailed explanation of the construction algorithm employed in this research is outlined in Section 4.2.

2.2.2 Sequential Constructive Approaches

The use of a sequential constructive algorithm is amongst the earliest approaches used to tackle the examination timetabling problem in an automated way (Broder, 1964; Cole, 1964; Foxley and Lockyer, 1968). In this approach, the concept of 'failed first' was implemented. The basic idea was to first schedule the exams that might cause problems if they were to be left to later in the scheduling process. By doing so, the overall aim was to avoid the assignment of exams to time slots which might later lead to an infeasible solution. An infeasible solution is reached when at least one exam remains unscheduled. In many cases this is because exams placed earlier have invalidated all the potentially valid time slots. In such a situation, a different ordering may enable a feasible solution to be found.

Approaches which order the events prior to assignment to a period have been discussed by several authors including Boizumault *et al.* (1996), Brailsford *et al.* (1999), Burke and Newall (2004), Burke, Kingston and de Werra (2004b), Burke and Petrovic (2002), and Carter *et al.* (1996). In the context of the exam timetabling benchmark data sets used in this research (described in Section 2.3.1), this sequencing strategy has been implemented by Carter *et al.* (1996), Burke and Newall (2004), and Burke *et al.* (2007). Usually, the unscheduled exams are ordered in a sequence that represents how difficult it is judged that they will be to place in the timetable (most difficult first). A number of commonly used strategies have been adopted from the graph colouring problem. Many studies employ graph theory to calculate the 'difficulty to schedule an exam'. The following list describes the most common graph colouring based heuristic orderings used in timetabling research:

Largest Degree (LD) First. Exams are ranked in descending order by the number of exams in conflict — i.e. priority is given to exams with the greatest number of exams in conflict.

Largest Enrolment (LE) First. Exams are ranked in descending order by the number of students enroled in each of the exams — i.e. exams with the highest number of students are given the highest priority.

Least Saturation Degree (SD) First. Exams are ranked in increasing order by the number of valid time slots remaining in the timetable for each exam — priority is given to exams with fewer time slots available.

Largest Coloured Degree (LCD) First. This heuristic is based on *LD*. For this heuristic, only exams which have been already assigned to the schedule are considered as the exams which will cause conflict.

Weighted Largest Degree (WLD) First. This heuristic is also based on *LD*. Besides the number of exams in conflict, the total number of students involved in the conflict are taken into account as well.

In general, heuristic orderings are divided into two categories: *static* and *dynamic*. *Static* heuristic orderings are predetermined before the start of the assignment process and their values remain the same throughout the process. For the heuristic orderings described above, *LD*, *LE* and *WLD* are categorized as *static* heuristic orderings. The number of exams in conflict with each exam and the number of students enroled for each exam only need to be calculated once by analysing the specific problem structure. On the other hand, *SD* and *LCD* are considered to be a *dynamic* heuristic orderings because the number of valid time slots available for unscheduled exams and the number of exams assigned to time slots may change each time an exam is assigned to a valid time slot; in which case, the unscheduled exams need to be reordered.

In 1961, Appleby *et al.* implemented graph colouring techniques in the preparation of school timetables. Since then, the use of graph based heuristic orderings have been extended to other types of timetabling problem. *LD* was the most widely used heuristic ordering in earlier research into examination timetabling (Broder, 1964; Cole, 1964; Welsh and Powell, 1967). Wood (1968) utilised the heuristic orderings *LE* and *LD*. In his approach, exams were selected starting with those that require the room with the largest capacity. These exams were then ordered decreasingly by the number of exams in conflict. The same process was applied to the second largest room and so on. Johnson (1990) also combined heuristic orderings *LE* and *LD*, but he considered them simultaneously through the simple linear combination of *LD*, with *LE* multiplied by a weighted factor w_{LE} that was varied.

The use of *Saturation Degree* was first presented by Brělaz (1979) for the graph colouring problem. Brělaz suggested that a vertex with the smallest number of colours that can be used to colour the vertex is the most difficult vertex to be coloured. Mehta (1981) implemented the heuristic *SD* in order to satisfy the requirement made by the registrar that all the exams must be scheduled in twelve time slots. However, in order to satisfy the requirement that no student should have his/her exams scheduled at the same time, they found that the minimum number of time slots required was thirteen. Therefore, in order to minimise the number of students who had two exams at the same time and to spread out each student's schedule, preprocessing of the collected data (e.g. grouping several exams as one exam) and adjustment of the sequence of time slots was required.

Kiaer and Yellen (1992) modeled a course timetabling problem as a weighted graph. Weights for edges were not based on the number of students who registered for the two connected vertices (conflicting courses), but the edges were assigned weights, 1, n or n^2, where n was the number of courses. Five heuristics based on weighted graph parameters were employed to select which courses were to be scheduled next. One of the heuristics was similar to the heuristic *SD*, but they called the heuristic '*forbidden degree*'. Their heuristic algorithm showed promising results when tested on randomly generated weighted graphs (80 graphs with 50 and 100 vertices respectively, and 100 graphs with 20 vertices). They also observed that their heuristic '*forbidden degree*' was more efficient for problems with a higher number of average vertices. When applied to real problems, the solutions produced by applying various heuristics outperformed the solution generated manually by the administration.

Laporte and Desroches (1984), and Carter *et al.* (1996) investigated four different types of graph based heuristic orderings to rank the exams in decreasing/increasing order to estimate how difficult it was to schedule each of the exams. They considered *Largest Degree*, *Saturation Degree*, *Weighted Largest Degree* and *Largest Enrolment*. These heuristics were used individually to order the exams. Then, the exams were selected sequentially and assigned to a time slot that satisfied all the specified constraints. In

Carter *et al.*'s approach, their algorithm first found the maximum clique of conflicting examinations. A clique of exams is a group of exams in which each exam conflicts with every other exam. The size of the maximum clique can also be used to determine the minimum number of time slots required to schedule all the exams for particular problem instances (Gendreau *et al.*, 1993). Exams grouped in the maximum clique were first assigned to different time slots, and then the heuristic ordering was applied to the remaining exams. Carter *et al.* tested the approach on ten random problems and thirteen real problems. Carter and Johnson (1999, 2001) investigated further the use of cliques for examination timetabling.

Casey and Thompson (2003) investigated the efficiency of these four heuristic orderings (i.e. *Largest Degree*, *Saturation Degree*, *Weighted Largest Degree* and *Largest Enrolment*) in constructing the initial solutions in the first phase of their *Greedy Randomised Adaptive Search Procedure (GRASP)* algorithm. Roulette wheel selection was employed to choose the next exam to be scheduled from the top n exams in the exam ordering, where the appropriate value for n was experimentally determined to be between 2 and 6 depending on the total number of exams in the problem instance. The selected exam was then scheduled into the first time slot that satisfied all the hard constraints. Foxley and Lockyer (1968) ordered the exams by a 'priority formula' that used all the known facts concerning the examinations. They also allowed a manual special priority setting to override other soft constraints. For example, they set a special priority for final year papers.

Although the aim of sequentially processing the ordered events (by certain criteria or heuristics) is to make sure all events can be scheduled by the end of the construction phase of the timetabling process, it is not always the case that all events are assigned at the first attempt. In addition to this, there are commonly used strategies to select which time slot an event is to be assigned to. This can also have a significant effect on the timetable construction process. Some common strategies mentioned in the literature are as follows:

- Use the first or the last valid time slot.
- Choose a valid time slot at random.
- Use the time slot that will cause the least penalty cost.
- Use the time slot that will minimise the number of unused seats.

In the case where a feasible timetable is not achievable during construction, various approaches can be applied. Usually, reshuffling the earlier scheduled events is performed. In Carter *et al.* (1996) and, Laporte and Desroches (1984), if no clash free time slot was found, 'backtracking' was implemented. In order to make a time slot available, the time slot with the minimum number of conflicting scheduled exams that needed to be 'bumped back' was chosen. They used *minimum disruption cost* (the cost of reshuffling the conflicting exams from the selected time slot into another valid time slot and inserting the current unscheduled exam into the selected time slot) to identify which exam was to be moved. All conflicting exams were either moved to the different time slot with the least penalty cost (while maintaining feasibility) or returned to the unscheduled exams list. For the purpose of avoiding an infinite loop, the number of times an exam could be returned in this manner was limited to three. This process was continued until all the exams were scheduled and a feasible solution produced. A similar backtracking approach was applied by Casey and Thompson (2003).

Another approach proposed in Burke and Newall (2004) applied an adaptive heuristic technique in which the exam list was initially ordered by a particular heuristic. This heuristic could then be altered to take into account the penalty that placed exams imposed upon the timetable. Their work was motivated by the *Squeaky Wheel* approach introduced by Joslin and Clements (1999).

Some researchers have implemented heuristic ordering in the process of splitting events into independent sets. The events are split into groups in such a way that no events in conflict are grouped together. The groups of event are then assigned to time slots with the objective of minimising the violation of certain soft constraints. One of the earliest papers that applied this approach to the graph colouring problem was published by Wood (1969). He presented a comparison of two grouping approaches for graph colouring problems. The first was based on the graph heuristic *LD*, while in the second pairs of objects were grouped based on their similarity. A similarity matrix was generated based on the information obtained from the conflict matrix. As defined by Wood, "if vertices i and j are not connected, the similarity is the number of other vertices k which are connected to both i and j". Experiments on real timetabling problems showed that the similarity matrix approach obtained better results in two out of the six problems, while the results for the remaining four of the problems were equal to the results produced when the graph based heuristic *LD* was employed. However, when tested on randomly generated data sets, it was observed that, overall (about 75% of the cases), the graph based heuristic *LD* produced better results compared to the similarity matrix approach.

Desroches *et al.* (1978) presented an automated examination timetabling system called *HOREX* employed by the *l'Ecole Polytechnique de Montréal*. The authors experimented with five heuristic orderings in the selection of exams to be placed into non conflicted groups. These heuristic orderings included two random approaches, ordering by *Largest Enrolment* and two other approaches that were developed based on the number of exams in conflict (no detailed description was given). Burke *et al.* (1994c) used the degree of a vertex (graph based heuristic *LD*) to determine which exams could be grouped together in the same time slot. In each group, exams were ordered increasingly by the number of students enroled. In turn, exams were assigned to rooms with the aim of minimizing the number of unused seats. It was, of course, possible to have more than one exam in one room.

In summary, there has been much research into different heuristic orderings. Carter *et al.* (1996) indicated that it is not easy to determine which heuristic ordering is the most appropriate for any given problem in hand. In addition, other work (e.g. Burke and Newall, 2004) has suggested that adaptively changing the heuristic ordering during construction can produce better solutions compared to only using one heuristic ordering throughout the process. A study by Burke *et al.* (1998b) also suggested that the use of heuristic ordering for creating the initial solutions of an evolutionary algorithm for timetabling problems could substantially improve performance. A common theme of these observations is that different heuristics may be beneficial in different circumstances during construction. This observation lead to the conjecture that considering a *combination* of different heuristics *simultaneously* might lead to a further improvement in solution quality.

2.2.3 Iterative Improvement Methods

As stated earlier, having constructed a solution in Phase 1, the solution is then often improved in Phase 2. The process is almost invariably an iterative process in which the solution is modified at each step in order to (hopefully) improve the quality of the solution. The most common approach is to utilise metaheuristic optimisation methods for this iterative improvement. An excellent general review of metaheuristic approaches in combinatorial optimisation can be found in Blum and Roli (2003). The aim of any heuristic search technique is to provide an efficient way of iteratively exploring the search space of a given problem. However, most methods will get trapped in local optima. The main aim of a metaheuristic technique is to escape from local optima and thus hopefully produce better solutions. Voβ *et al.* (1999) defined a metaheuristic as follows:

> *"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."*

Glover and Laguna (1997) defined the term as follows:

> *"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."*

While Osman and Kelly (1996) defined it as:

> *"A meta-heuristic is an iterative generation process which guides a subordinate heuristic ..."*

Blum and Roli also quoted several definitions of metaheuristics given by several researchers and outlined the basic characteristics of metaheuristics. This Section concentrates on the metaheuristic approaches for educational timetabling.

2.2.3.1 Simulated Annealing and the Great Deluge Algorithm

Simulated Annealing (Kirkpatrick *et al.*, 1983) has been successfully applied to the examination timetabling problem by Thompson and Dowsland (1996, 1998). They focused on developing a robust *Simulated Annealing* approach in which the cooling schedule was determined in an automated way and adapted depending on the problem instances and objective functions defined for the given problem instances. Bullnheimer (1997) focused on the use of *Simulated Annealing* in small scale examination timetabling problems and, particularly, on breaking down one larger real-world problem instance into several smaller sub-problems. Burke *et al.* (2004a) also investigated the use of *Simulated Annealing* in examination timetabling in a comparison with the *Great Deluge Algorithm* (see below).

An approach that works in a very similar manner to *Simulated Annealing* is known as *Great Deluge Algorithm* (Dueck, 1993). In comparison with the temperature parameter used by *Simulated Annealing*, *Great Deluge Algorithm* uses two parameters that are, perhaps, more meaningful to the user, namely the amount of computational time required and an estimate of the quality of the desired solution. The advantage of the *Great Deluge Algorithm* is that, as these parameters are more meaningful, the algorithm is easier for the inexperienced user to apply and less parameter tuning is required. The application of the *Great Deluge Algorithm* in examination timetabling problems has been investigated by Burke *et al.* (2004a) and, Burke and Bykov (2006). A comparison of *Great Deluge Algorithm* and *Simulated Annealing* applied to Carter *et al.*'s examination benchmark data sets, as reported by Burke *et al.* (2004a), and Abdullah and Burke (2006), showed that, overall, *Great Deluge Algorithm* produced better results than *Simulated Annealing*, although *Great Deluge Algorithm* did not produce better results in all cases.

2.2.3.2 *Tabu Search*

Tabu Search (Glover, 1986) has also been successfully applied to a wide range of educational timetabling problems. With various problems definitions to deal with, a variant of *Tabu Search* setups was employed to solve examination timetabling problems and course timetabling problems. Di Gaspero and Schaerf (2001) investigated a family of *Tabu Search* algorithms and applied the algorithms to the examination timetabling benchmark data sets. The experimental results showed that "*The most effective algorithm makes use of a shifting penalty mechanism, a variable-size tabu list, a dynamic neighbourhood selection, and a heuristic initial state.*". In 2003, Di Gaspero and Schaerf further enhanced the algorithm by incorporating a set of multi-neighbourhood strategies to improve the performance of a local search method. The algorithm was applied to the course timetabling problem, and the experimental results demonstrated that the enhanced algorithm produced much better results compared to the old algorithm.

For course timetabling, a comparison between a combination of constraint logic with *Tabu Search*, constraint logic alone and *Tabu Search* alone was studied by White and Zhang (1998). By employing constraint logic to construct the initial solution and then applying *Tabu Search* to improve the initial solutions, they showed that better solutions were produced compared to using constraint logic alone. They also showed that the timetable could be constructed in much shorter time compared to employing tabu search alone. White and Xie (2001) developed a *Tabu Search* algorithm which they called OTTABU. Both types of adaptive memory (namely recency-based short term memory and frequency-based longer term memory) were employed in order to avoid cycling with the aim of improving the quality of solution(s). They applied a four phase system to construct examination timetables for the University of Ottawa. It was found that better timetables were produced compared to the timetable obtained without longer term memory. They also applied the algorithm to two of Carter's proximity cost benchmark data sets (namely *CAR-F-92* and *UTA-S-92*). It was observed that the algorithm demonstrated the same improvements (i.e. using longer term memory produced better timetables) when applied to different problems (i.e. different problem instances and different penalty cost). Later, White *et al.* (2004) extended this research by applying the approach to twelve of Carter's proximity cost benchmark data sets (see Table 2.1) and comparing the solutions to the results published by other researchers. The paper states that the performance of the new algorithm was 'favourable'.

2.2.3.3 Evolutionary Algorithms

Evolutionary Algorithms (EA) are motivated by the process of natural evolution (Holland, 1992). The main feature of EAs is that they are population based. That is, a number of solutions are maintained within the algorithm and new solutions are produced by combining or changing the solutions in the current population with the aim of producing better solutions. Amongst the popular EAs are *Genetic Algorithm*, *Memetic*

Algorithm and Ant Colony Optimisation algorithms.

Burke *et al.* (1994a,b) investigated GAs with a direct representation scheme that considered both time slot allocations and room assignment for university timetabling. They considered problems with non-fixed timetable lengths and their method only accepts feasible timetables. Burke *et al.* (1995a) used GAs for examination timetabling with the objective of minimising the number of time slots required. They compared eight selection heuristics for their uniform crossover operators. Two of the heuristics were based on graph colouring heuristics (*LD* and *LCD*); one was a random heuristic; and the remaining were specially designed heuristics that highlighted the two constraints that needed to be addressed (i.e. the number of time slots and the spread of the conflicting exams) either individually or combined. These heuristic crossover operators were developed with the aim of avoiding infeasible timetables being produced during the recombination process. The experimental results showed that good quality timetables might be produced by integrating heuristics in crossover operators. Similar heuristic crossover operators were successfully implemented by Burke *et al.* (1995b) for another set of more difficult timetabling problems.

Ross *et al.* (1998) discussed the effectiveness of direct representation for GA implementations in exam timetabling problems. They suggested that a GA is more suitable for finding a good algorithm instead of directly searching for the solutions for a particular problem. Erben (2001) pointed out that the poor performance of GAs (compared to conventional heuristic approaches) in graph colouring problems was primarily because of the inappropriate selection of solution encoding schemes. As an alternative to direct representation, Erben applied grouping GAs for graph colouring problems and exam timetabling problems. In grouping GAs, a group of non-connected nodes (for graph colouring problems) or a group of non-conflicting exams (for exam timetabling problems) is treated as one *gene*. The chromosome length represented the graph chromatic number for graph colouring problems or the number of time slots for exam timetabling problems. In order to generate feasible solutions, the hard and soft constraints need to be incorporated in the crossover operator and mutation operator. Quite encouraging results were obtained when this approach was applied to one of the capacitated problems of Carter *et al.*'s benchmark data set (specifically the *TRE-S-92* problem instance).

2.2.3.4 Memetic Algorithms

The term '*Memetic Algorithm*' was introduced by Moscato (1989) in a Technical Report which described a heuristic which used "Simulated Annealing for local search with a competitive and cooperative game between agents, interspersed with the use of a crossover operator". Later, Moscato and Norman (1992) went on to explain a similar approach that utilised local search within a GA implementation. In Burke *et al.* (1996b), instead of using a crossover operator, a hill climbing local search was performed after each mutation operation. Two types of mutation operator were proposed, termed 'light' and 'heavy' mutation. A comparison with approaches that merely relied on multi-start random descent local search showed that this approach obtained better results for the Nottingham capacitated examination timetabling problem. However, they also observed that further tests on more highly constrained problems (i.e. Carter *et al.*'s capacitated examination timetabling benchmark problems) showed that this approach was outperformed by their previous approach presented in Burke *et al.* (1995b). Despite this, motivated by these quite promising results, an extended version of the approach was outlined by Burke and Newall (1999). Although the focus of the paper was on heuristic decomposition of the timetable problem, the results also showed that incorporating GAs with heuristic techniques and local search approaches obtained better results than using the standard GA alone. More detailed descriptions on the design of Memetic Algorithms for timetabling problems can be found in Burke and Landa Silva (2004).

2.2.3.5 Ant Colony Optimisation

Ant Colony Optimisation is another population based approach, introduced by Dorigo *et al.* (1996). Initially, applications of Ant Colony Optimisation were focused on the Traveling Salesman Problem (Dorigo and Gambardella, 1997; Dorigo *et al.*, 1996). A study by Costa and Hertz (1997) investigated the use of Ant Colony Optimisation in graph colouring problems in which they called their ant system ANTCOL. Costa and Hertz stated that their results are quite satisfactory although their results did not match the best results reported in the literature. Inspired by the findings, Socha *et al.* (2002) used an Ant Colony Optimisation algorithm to construct course timetables. The Ant Colony Optimisation approach was compared to other local search techniques and it was found that Ant Colony Optimisation produced the best solutions (see Section 6.2.3 for details of the results obtained).

The basic ANTCOL developed by Costa and Hertz (1997) was then modified and improved by Dowsland and Thompson (2005). Instead of implementing the ant algorithm to random graphs (as implemented by Costa and Hertz (1997)), Dowsland and Thompson applied their improved ANTCOL to Carter *et al.*'s examination timetabling benchmark data sets with the aim of findings the minimum number of time periods required to produce clash free timetables. Overall, the improved ANTCOL has produced competitive results compared to the results obtained using the sequential constructive algorithm developed by Carter *et al.* (1996) and Merlot *et al.* (2003) (see hybrid approach below).

Eley (2006) investigated the use of two ant colony approaches namely MMAS-ET that based on Max-Min Ant System (MMAS) (as applied by Socha *et al.* (2002) to examination timetabling) and ANTCOL-ET which is a modified version of ANTCOL (that originally used by Costa and Hertz (1997) to solve graph colouring problem) to the Carter *et al.*'s proximity cost benchmark problems. For both MMAS-ET and ANTCOL-

ET approaches, additional hill climber is incorporated. The experimental results show that in average the ANTCOL-ET with hill climber approach has produced better results compared to ANTCOL-ET without hill climber, MMAS-ET with hill climber and MMAS-ET without hill climber. In comparison the best results in literature, Eley's results are comparable to the results obtained by other approaches.

2.2.3.6 Hybrid Approaches

More recently, there has been much research into hybridised methods which draw on two or more of the techniques mentioned above. In an implementation of the *GRASP* algorithm in examination timetabling, Casey and Thompson (2003) observed that better solutions could be produced by combining a limited form of *Simulated Annealing* with Kempe chain neighbourhoods (Thompson and Dowsland, 1996) and a memory function that avoided exams sharing the same time slot as in the previous iteration during the improvement phase. Azimi (2005) developed a hybrid heuristic based on a combination of *Tabu Search* and *Ant Colony Optimisation*. The author selected these two metaheuristic approaches to be combined based on an earlier analysis comparing four metaheuristic approaches including *Simulated Annealing*, *Genetic Algorithms*, *Tabu Search* and *Ant Colony Optimisation*. The analysis was carried out with ten randomly generated examination timetabling data sets, and the well known proximity cost penalty function was used to evaluate the timetable solutions. Azimi introduced three different approaches to combine *Tabu Search* and *Ant Colony Optimisation* metaheuristics, and the results showed that all the three hybrid heuristics outperformed all the metaheuristics applied individually.

The hybrid approach developed by Caramia *et al.* (2001) has produced several best known results for the Carter *et al.*'s proximity cost benchmark problems for several data sets (see Table 4.9). Their algorithm start with a greedy constructive heuristic and followed with an optimiser in the attempts to spread out the students' schedule. Caramia *et al.* also applied their algorithm to the capacitated problem (see Section 2.3.2.2). Merlot *et al.* (2003) applied three-stage hybrid approach to a real-world examination timetabling problem (i.e. for University of Melbourne) and the Carter *et al.*'s examination timetabling benchmark data sets (graph colouring problem, uncapacitated problem and capacitated problem). In the first stage, initial feasible solution is constructed using a constraint programming technique. The initial solution is improved using Simulated Annealing in stage two, and in the final stage the solution is further improved by implementing a hill climbing method. In comparison to the best results in literature for the benchmark data sets, they obtained competitive results for the graph colouring problem and uncapacitated problem, while for the the capacitated problem they produced best results for several problem instances. A study that investigated the hybridisation of large neighbourhood search and *Tabu Search* was presented in Abdullah *et al.* (2006b). Experimental results showed that their solutions for the capacitated problem were competitive to the best solutions reported in the literature; they obtained best results for two out of the six data sets.

Rahoual and Saad (2006) carried out work in which *Genetic Algorithms* and *Tabu Search* were hybridised in an attempt to produce solutions for benchmark and real world university course timetabling problems, with quite promising results. They produced comparable results for the benchmark data sets, while for the real world data sets the solutions were produced in just one hour compared to the three to four weeks required to prepare solutions manually.

2.3 Evaluation of Timetable Quality

The quality of a given timetable can be evaluated by measuring to what degree the specified constraints are satisfied. Usually, the main concern is the satisfaction of all the hard constraints. However, it is also very important to minimise the violation of the soft constraints, because, in many cases, the quality of the constructed timetable is evaluated by measuring the fulfillment of these constraints. In practice, the constraints imposed by various academic institutions can be very different (Burke *et al.*, 1996a). Such variations make the timetabling problem more challenging. Due to the complexity of the problems, algorithms or approaches that have been successfully applied to one problem may not perform well when applied to different timetabling problem instances. The variety of constraints may also require different formulations of objective functions or evaluation functions.

In this Section, the discussion will concentrate on evaluation functions that have been applied to the examination timetabling benchmark data sets introduced by Carter *et al.* (1996). A detailed description of these examination timetabling problems is now provided. Note that, as course timetabling only features in the first part of Chapter 6, a detailed description of the problem instances used for course timetabling is given in Section 6.2.1.

2.3.1 Data Sets and Problem Descriptions

In 1996, Carter *et al.* introduced a set of examination timetabling benchmark data. This benchmark data set collection consisted of thirteen problem instances. Originally this data came from real university examination timetabling problems. As such, these data sets varied considerably in terms of resource availability and constraints specified. For the sake of generality, these data sets were then simplified so that only the following constraints were represented:

Hard constraint The constructed timetable must be conflict free in that no student can be scheduled for two different exams at the same time.

Soft constraint The solution should attempt to minimise the number of exams assigned in adjacent time slots in such a way as to reduce the number of students sitting exams in close proximity.

Unfortunately, over time, these original thirteen problem instances have become slightly modified due to a number of factors, and these modifications have, in effect, meant that the problem instances differ. Qu *et al.* (2006) have recently completed a thorough re-classification of all problem instances which have appeared in published work. They discovered that there are now, effectively, twenty one different problem instances. The complete list of these twenty one instances, with the different characteristics and their various levels of complexity, and with Qu *et al.*'s proposed new naming convention, are shown in Table 2.1.

For each data instance, two files are supplied — a student data file (with a '.stu' file extension) and course data file (with a '.crs' file extension). Detailed list of the exams enrolled on by each student are stored in the student data file, while the total number of students enrolled for each exam is stored in the course data file. It is to be expected that the total enrolment for all exams represented in both the student data file and the course data file are equal for each data instance. However, three of the data instances with suffix "II" in Table 2.1 (car92 II, car91 II and pur93 II) have conflicts in the number of enrolments (i.e. the total number of exam enrolments presented in the two data files is different). For these three instances, the seventh column of the table presents the total number of enrolments in the student data file and in the course data file, respectively.

The details contained in the student data file (alone) actually provide enough information in order for the scheduler to construct a feasible timetable (without referring to the course data file). Examples of the information available are the conflicting exams, the total number of students enrolled for each exam and the number of exams enrolled by each student. This information can be used to measure the density of conflicting exams for each problem instance. The *conflict density* values shown in column eight of the Table indicates the density of conflicting exams. To calculate the conflict density, a conflict matrix C is defined in which each element c_{ij} is one if exam i conflicts with exam j (at least one student is enrolled for both exam i and j), or zero otherwise. The *conflict density* is calculated by summing the number of other exams that each exam is conflicted with (i.e. the elements of the conflict matrix for which $c_{ij} = 1$), and dividing by the total number of elements in the conflict matrix. Note that, for $pur93\ II$, the conflict density value is marked with '-'. The inconsistency of the data files for this problem instance requires further data conversion in order to make the conflict density calculation possible. As this problem instance is not used in this research, no attempt has been made to calculate its conflict density.

In this research, only twelve out of the thirteen original data sets were used (as highlighted in bold text in the Table). The remaining data set (*PUR-S-93*) has almost four times the number of exams compared to any of the other data sets but with a very low conflict density. The low conflict density means that the problem is loosely constrained and so, in effect, is relatively easy to solve. Initial experimental tests confirmed that a prohibitive amount of time would have been required to create fuzzy models for this data set. This, taken in conjunction with its large size, means that excessive computational time would have been devoted for little gain. Thus it was excluded from comparison in the large number of experiments undertaken as part of this research.

Table 2.1: Examination timetabling problem characteristics. The twelve problem instances that used in this research are highlighted in bold font.

Data Set	Name used in this thesis	Institution	Number of time slots (P)	Number of exams (N)	Number of students (S)	Number of enrolments	Conflict density
car92 I	***CAR-F-92***	**Carleton University, Ottawa**	**32**	**543**	**18419**	**55522**	**0.1377**
car92 II	-	Carleton University, Ottawa	32	543	18419	55189/55522	0.1368
car91 I	***CAR-S-91***	**Carleton University, Ottawa**	**35**	**682**	**16925**	**56877**	**0.1282**
car91 II	-	Carleton University, Ottawa	35	682	16925	56242/56877	0.1260
ear83 I	***EAR-F-83***	**Earl Haig Collegiate Institute, Toronto**	**24**	**190**	**1125**	**8109**	**0.2655**
ear83 II	-	Earl Haig Collegiate Institute, Toronto	24	189	1108	8014	0.2719
hec92 I	***HEC-S-92***	**Ecole des Hautes Etudes Commerciales, Montreal**	**18**	**81**	**2823**	**10632**	**0.4155**
hec92 II	-	Ecole des Hautes Etudes Commerciales, Montreal	18	80	2823	10625	0.4222
kfu93	***KFU-S-93***	**King Fahd University, Dharan**	**20**	**461**	**5349**	**25113**	**0.0555**
lse91	***LSE-F-91***	**London School of Economics**	**18**	**381**	**2726**	**10918**	**0.0624**
pur93 I	-	Purdue University, Indiana	42	2419	30032	120681	0.0295
pur93 II	-	Purdue University, Indiana	42	2419	30032	120688/120686	-
rye92	***RYE-F-92***	**Ryerson University, Toronto**	**23**	**486**	**11483**	**45051**	**0.0751**
sta83 I	***STA-F-83***	**St. Andrew's Junior High School, Toronto**	**13**	**139**	**611**	**5751**	**0.1430**
sta83 II	-	St. Andrew's Junior High School, Toronto	13	138	549	5689	0.1924
tre92	***TRE-S-92***	**Trent University, Peterborough, Ontario**	**23**	**261**	**4360**	**14901**	**0.1800**
uta92 I	***UTA-S-92***	**Faculty of Arts & Science, University of Toronto**	**35**	**622**	**21266**	**58979**	**0.1254**
uta92 II	-	Faculty of Arts & Science, University of Toronto	35	638	21329	59144	0.1214
ute92	***UTE-S-92***	**Faculty of Engineering, University of Toronto**	**10**	**184**	**2750**	**11793**	**0.0845**
yor83 I	***YOR-F-83***	**York Mills Collegiate Institute, Toronto**	**21**	**181**	**941**	**6034**	**0.2873**
yor83 II	-	York Mills Collegiate Institute, Toronto	21	180	919	6012	0.3041

2.3.2 Existing Evaluation Functions

In the context of Carter *et al.*'s benchmark data sets, several different evaluation functions have been introduced in order to measure the quality of the timetable solution. In addition to the commonly used function that evaluates only the proximity cost (see Section 2.3.2.1 for details), other evaluation functions have been derived based on the satisfaction of other soft constraints, such as minimising the number of consecutive exams in one day or overnight (Burke and Newall, 1999; Burke *et al.*, 1996b) and assigning large exams to early time slots (Petrovic *et al.*, 2005).

2.3.2.1 The Uncapacitated Problem

The proximity cost function was the original evaluation function used to measure the quality of timetables for Carter *et al.*'s benchmarks (Carter *et al.*, 1996). Besides the need to construct a clash-free timetable, it is also required to schedule the exams within the maximum number of time slots given. This evaluation function is motivated by the goal of spreading out each individual student's examination timetable. In the implementation of the proximity cost, it is assumed that the timetable solution satisfies the defined hard constraint — i.e. no student can attend more than one exam at the same time. In addition, the solution must be developed in such a way that it will promote the spreading out of each student's exams so that students have as much time as possible between exams. If two exams scheduled for a particular student are t time slots apart, a penalty weight is set to $w_t = 2^{5-t}$ where $t \in \{1, 2, 3, 4, 5\}$. The weight, w_t, is multiplied by the number of students that sit both the scheduled exams. The average penalty per student is calculated by dividing the total penalty by the total number of students. This function was originally implemented by Carter *et al.* (1996) and has been widely adopted by many subsequent researchers in this area. The maximum number of time slots for each data set is predefined and fixed, but no limitation in terms of capacity per time slot is set. Thus, this is usually termed the 'uncapacitated problem'. Consecutive exams either in the same day or overnight are treated the same, and there is no consideration of weekends or other actual gaps between logically consecutive days. The following formulation represents this proximity function (adapted from Burke *et al.* (2004a)):

$$\frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} s_{ij} w_{|p_j - p_i|}}{S} \tag{2.4}$$

where N is the number of exams, s_{ij} is the number of students enroled in both exam i and j, p_i is the time slot where exam i is scheduled, and S is the total number of students; subject to $1 \leq |p_j - p_i| \leq 5$.

2.3.2.2 The Capacitated Problem

Burke *et al.* (1996b) devised a new evaluation function that took into account the maximum capacity allowed in each time slot. In addition to the clash-free timetable requirements, an additional hard constraint was defined to specify that the total number of students timetabled for a particular time slot must not exceed the maximum number of students allowed. Of the twenty one data sets shown in Table 2.1, only five data sets are usually used with this evaluation function. Table 2.2 shows the restrictions applied to the five data sets. Note that the number of time slots are different from the list shown in Table 2.1. Furthermore, in this problem the timetable is arranged to have three time slots on weekdays and only one morning slot on Saturday. This is termed the 'capacitated problem'.

The objective is to minimise the number of students who have to sit two exams in the same day without any gap between the two exams. The problem is formulated as follows:

$$\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left[\sum_{p=1}^{P-1} t_{ip} t_{j(p+1)} c_{ij} + t_{ip} t_{j(p-1)} c_{ij} \right] \tag{2.5}$$

Table 2.2: The capacitated problem specifications

Data Set	Number of time slots (P)	Max Students Per Time Slot (X)
CAR-F-92	31	2000
CAR-S-91	51	1550
KFU-S-93	20	1955
TRE-S-92	35	655
UTA-S-92	38	2800

subject to constraints specify in Equations 2.2 and 2.3; and

$$\sum_{j=i+1}^{N} t_{ip}s_i \leq X_p, \forall p \in \{1, ..., P\} \tag{2.6}$$

where X_p is the maximum number of seats available in time slot p. Equation 2.6 specifies that the total number of students who are enroled for all exams timetabled in any period must not exceed the maximum number of seats available.

Burke and Newall (1999) extended the previous evaluation function by defining different weights for two consecutive exams in the same day and two exams in consecutive overnight time slots. The evaluation function was:

$$\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left(\left[\sum_{p=1}^{P} t_{ip}t_{j(p+1)}c_{ij}d_{p(p+1)} + t_{ip}t_{j(p-1)}c_{ij}d_{p(p-1)} \right] + 5000t_{i(P+1)} \right) \tag{2.7}$$

where

$$d_{pq} = \begin{cases} 3, & \text{if periods } p \text{ and } q \text{ are on the same day} \\ 1, & \text{if periods } p \text{ and } q \text{ are on an adjacent day} \\ 0, & \text{otherwise} \end{cases} \tag{2.8}$$

subject to constraints specify in Equations 2.2, 2.3 and 2.6.

Noted that incomplete solutions are acceptable in this evaluation function. Unscheduled exams (if any) are assigned to the $(P+1)^{th}$ period and a penalty of 5000 is given for each unscheduled exam. For Equations 2.7 and 2.8, care must be taken to take into account the gap due to the weekend break (any two conflicting exams that are timetabled on Saturday and Monday respectively must be given zero penalty). Burke *et al.* (2004a) presented an efficient way to deal with this weekend break issue. Finally, although not the capacitated problem, Petrovic *et al.* (2005) employed fuzzy methodologies to create a novel objective function based on two criteria, one of which was based on the size of exams — see Section 2.5 for details.

2.3.3 Multi-objective and Multi-criteria Approaches

Considering all the evaluation functions discussed above (particularly Equations 2.5 and 2.7), it is obvious that, in each case, the timetable quality is evaluated using a single objective function which represents the summation of a weighted combination of measures of soft constraint violation. In the context of the benchmark data sets problem instances that are considered here, it seems that these evaluation functions are sufficient to serve the purpose of providing comparative measures of the performance of the various approaches that have been developed.

However, in real world timetabling problems, more constraints must be taken into account in the timetable construction and optimisation process. Some of the constraints have a certain level of preference defined by the administration (of the organisation) and, in most cases, these constraints impose conflicting objectives. Coello (2006) defined a multi-objective optimisation problem (MOP) as

> *" a problem which has two or more objectives that we need to optimize simultaneously. It is important to mention that there might be constraints imposed on the objectives. It is also important to emphasize that it is normally the case that the objectives of the* MOP *are in conflict with each other."*

For example, consider two constraints. The first constraint is that an exam with a large number of students should be scheduled in the earlier time slots of the overall timetable, while the second constraint is that no student should be scheduled for two exams in consecutive time slots. In the situation where most of the students are enroled on the same subjects, improving the satisfaction of the first constraint (one objective) will inevitably have to compromise the second constraint (another objective). Therefore, a more flexible way of measuring the timetable quality is required for multi-objective approaches. To quote McCollum (2006)

> *"More work is required on how the quality of solutions are measured. The challenge for researchers is the provision of a solution where the user understands the trade offs between the original objectives."*

An excellent introduction to multi-objective optimisation approaches for scheduling and timetabling is presented by Landa Silva *et al.* (2004). The terms 'multi-objective' and 'multi-criteria' sometimes appear to be used interchangeably. One definition of the difference has been given by Hwang and Yoon (1981). They defined the term 'multi-objective' to refer to dealing with more than one decision factor in the *design* or *creation* stage of a process, 'multi-attribute' to refer to dealing with more than one decision factor in the *evaluation* stage of a process, and 'multi-criteria' to be a term which meant either multi-objective or multi-attribute. As an example of less strict use of the terms, although the following papers attempt to solve a similar problem, Burke *et al.* (2001a) used the term 'multi-criteria', while in Petrovic and Bykov (2003) the term 'multi-objective' was used. These two papers deal with examination timetabling problems in which nine different criteria (objectives) need to be optimised. The overall aim was to minimise the violation of each of the constraints. In order to tackle the problem, Burke *et al.* (2001a) applied a 'compromise programming approach' in which the distance between the current solution and an ideal point (where all the criteria were satisfied) was used to measure the solution's quality. Petrovic and Bykov (2003) presented a more transparent method in which the user can express their preferences. Guided by the reference solution specified by the user, a trajectory was drawn from the origin (initial solution) to a reference point and a local search using the *Great Deluge Algorithm* (Burke *et al.*, 2004a; Dueck, 1993) was performed to move the point along the specified trajectory in order to search for solutions that improved on one or more of the criteria. Other approaches that may be considered multi-objective were presented by (Arani and Lotfi, 1989; Lotfi and Cerveny, 1991).

The fuzzy multiple heuristic ordering method described in Chapters 4 to 6 and the multi-attribute fuzzy evaluation of timetables described in Chapters 7 and 8 should not be confused with multi-objective approaches to examination timetabling, such as those described in (Arani and Lotfi, 1989; Burke *et al.*, 2001a; Lotfi and Cerveny, 1991; Petrovic and Bykov, 2003). In the approaches adopted in this thesis, the problem of judging the difficulty of exams to be scheduled by using more than one heuristic ordering (multi-criteria) and the problem of selecting the most 'fair' timetable by considering two criteria are formulated as fuzzy decision problems. In contrast, in multi-objective / multi-criteria techniques, more than one criteria are kept separate (rather than being combined together). In such an approach, the concept of *pareto optimality* is usually adopted. In pareto-based evaluation, the concept of *dominance* is used to establish which solutions are considered to be better than others. A solution is said to dominate another solution if all the criteria are *better* (lower or higher depending on whether minimisation or maximisation is being considered). The *pareto front* is the set of all non-dominated solutions. In (Arani and Lotfi, 1989; Burke *et al.*, 2001a; Lotfi and Cerveny, 1991; Petrovic and Bykov, 2003), *pareto optimisation* concepts were employed to explore the solution space with the aim of minimising violations of the list of specified criteria (with the criteria kept separate). For a detailed discussion of multi-objective approaches in scheduling and timetabling, see Landa Silva *et al.* (2004).

2.4 The Need for Fuzzy Techniques in Timetabling

In everyday life situations, human always have to deal with knowledge that is uncertain, ambiguous or imprecise in nature. In other words, most of the time humans have to think and reason based on fuzzy information. Linguistic terms (everyday words) can be seen as the source of fuzziness. Words such as *fast*, *tall* and *heavy* are fuzzy. For example, we cannot define a single quantitative value to represent the term *fast*. The definition for the term is dependent on the context in which it is being used. If, for example, the term 'fast' is used to refer to a jet aeroplane, the definition obviously different than compared to the use of the term in the context of a car travelling on a motorway.

(McCollum, 2006) has declared that there is a gap between academic research in university timetabling and the practitioners who are solving real world problems. McCollum also suggested that an approach that reflected real world situations more adequately is critically desired. In attempting to solve real world timetabling problems, we must recognise that these problems are typically ill-defined and difficult to model. A common weakness of the 'traditional' technologies applied to solve timetabling problems (as described in Sections 2.2.2, 2.2.3 and 2.3.3) is that their development is based on classical reasoning and modelling techniques in which binary logic and crisp classifications are usually implemented.

The capability to use linguistic terms (words) in reasoning is one of the main strengths of fuzzy logic. (Zadeh, 1999) stated that

> *"In its traditional sense, computing involves for the most part manipulation of numbers and symbols. By contrast, humans employ mostly words in computing and reasoning, arriving at conclusions expressed as words from premises expressed in a natural language or having the form of mental perceptions. As used by humans, words have fuzzy denotations."*

Another important feature of fuzzy reasoning is its capability to handle multiple input attributes simultaneously. A survey conducted by (Burke *et al.*, 1996a) showed that the main concern of practitioners who solve real world university timetabling problems was to construct timetables that satisfy a *variety* of constraints. Furthermore, in practice, each institution has its own requirements that classify the constraints into hard and soft constraints.

In addition to the requirement that the timetable solutions must be feasible (i.e. fulfil all the hard constraints), the quality of timetable solutions is usually measured by taking into account the satisfaction degree of each of the soft constraints. When soft constraints are considered, rather than applying binary logic (i.e. satisfied or not satisfied), a constraint is satisfied to a certain degree. Although partial satisfaction of the soft constraints is (by definition) acceptable, naturally, for any timetable solution, the quality is considered to be higher if all the soft constraints have a high degree of satisfaction. However, these soft constraints often conflict with each other, meaning that maximising satisfaction of any of the constraints might degrade the satisfaction of other constraints.

These observations motivate this research towards mimicking how human timetabling experts solve real world problems. In summary, the gap between timetabling research and timetabling practice needs to be closed; fuzzy techniques provide a range of tools than can help to close this gap. Further descriptions of the fuzzy techniques used in this work are given in Chapter 3.

2.5 Fuzzy Techniques in Timetabling

Since being introduced by Zadeh (1965), fuzzy methodologies have been successfully applied in a wide range of real world applications (Pappis and Siettos, 2005, p. 466). Some specific examples in a selection of scheduling, timetabling and rostering areas are as follows. Fuzzy evaluation functions have been utilized in generator maintenance scheduling by Dahal *et al.* (1999), while Abboud *et al.* (1998) used fuzzy target gross sales (fuzzy goals) to find 'optimal' solutions of a manpower allocation problem, where several company goals and salesmen constraints needed to be considered simultaneously. Fuzzy methodologies have been investigated for other timetabling problems such as aircrew rostering by Teodorovic and Lucic (1998), driver scheduling by Li and Kwan (2003) and nurse rostering by Aufm Hofe (2001).

In the specific context of examination timetabling, fuzzy methods have been implemented for measuring the similarity of problem instances in a case based reasoning framework by Yang and Petrovic (2005). In this work, a fuzzy similarity measure was used to retrieve a good heuristic for a new problem based in comparison with previous problems that were stored in the case base. The selected heuristic was then applied to the new problem for generating an initial solution before the *Great Deluge Algorithm* was applied in the improvement stage. The results obtained indicated that the performance of the *Great Deluge Algorithm* was better when this fuzzy similarity measure was applied in the initialisation stage, compared to other initialisation approaches.

More recently, Petrovic *et al.* (2005) employed fuzzy methodologies to measure the satisfaction of various soft constraints. The authors described how they modeled two soft constraints, namely a *constraint on large exams* and a *constraint on proximity of exams* [sic], in the form of fuzzy linguistic terms. Two sets of rules were defined to derive the 'degree of satisfaction' of these constraints from more fundamental input variables. The *constraint on large exams* was derived from the size of the exam and the earliness of the time period that the exam was assigned to, while the *constraint on proximity of exams* was derived from the number of students sitting both exams and the number of time periods between the two exams. The aggregation of the fuzzy outputs (for the two soft constraints) was then used as the fitness function to evaluate the overall quality of the solution. A linear equation with weighting factors (to differentiate which soft constraint is the most important) was employed. The objective function can be formulated as follows (Petrovic *et al.*, 2005).

For a timetable T, the satisfaction degree of both constraints was aggregated as

$$F(T) = w_1 f_1(T) + w_2 f_2(T) \tag{2.9}$$

where w_1 and w_2 indicates the relative importance of the constraints.

The satisfaction degree for a large exam was formulated as

$$f_1(T) = min\{\mu_{f_n}(e_n)|n = 1, ..., N\}$$

for all exams e_n, $n = 1, \ldots, N$, where N is the total number of exams, and $\mu_{f_n}(e_n)$ is the degree of satisfaction of constraint f_n by exam e_n.

The satisfaction degree for the proximity of exams was formulated as

$$f_2(T) = \frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \mu_{f_2}(e_i, e_j)}{C}$$

where e_i and e_j, $i, j = 1..N$ are conflicted exams, while C is the total number of pairs of exams in conflict.

A memetic algorithm was then implemented to iteratively improve timetables using $F(T)$ as the objective function being optimised. They evaluated their approach on seven of the Carter *et al.* benchmark problem instances. However, as the objective function being optimised was not the usual proximity cost, they could not compare the effectiveness of their approach with other approaches. See Petrovic *et al.* (2005) for further details of the evaluation process.

Amintoosi and Haddadnia (2005) utilised the fuzzy c-means clustering algorithm to group students in a course into smaller sections. A student on one course could select several subjects that he/she wanted to enrol on. Therefore, when students were to be assigned to a group, it was required that students in the same group had the same schedule (i.e. students enroled for the same subjects). It was also required that the number of students in each section were balanced with the other sections so that the room capacity constraints could be satisfied. Their simulation results show that better sectioning of students was obtained when subjects with 'the most' and 'the fewest' enrolments were removed during the clustering process. That is, subjects with common students and the less popular subjects could be excluded when comparing the similarity of students' schedules.

It can be seen that interest in applying fuzzy methodologies to the university timetabling problem has only really gained significant interest in recent years, although its application to scheduling problems in general has been reported by Slany (1996), Slowinski and Hapke (2000), and others.

2.6 Generalisation of Problem Solving Approaches

In recent years, interest in the development of approaches that have a higher degree of generality has increased due to the potential of applying such approaches to problem instances with different characteristics or to problems in different domains. One such approach employs a knowledge-based technique known as Case Based Reasoning (*CBR*) to construct solutions for timetabling problems. The main concept of *CBR* is that similar problems can often be solved by using similar solution methods. That is, any current problems are attempted to be solved based on the knowledge acquired from previous experience of solving similar problems. For any new problems, instead of trying to solve the problems from scratch, *CBR* provides a mechanism to allow the solving of the problem from a certain point of the problem solving process. Hence, the important issue is how to measure the similarity between the new problem and the old problems stored in the knowledge database. Burke *et al.* (2000, 2001b) investigated the use of attribute graphs for representing the structure of timetabling problems. The authors demonstrated that more information about timetabling problems can be represented using these attribute graphs and, as a result, they claimed that the retrieval of the most similar cases could be performed more efficiently. An improved version of this approach was described in Burke *et al.* (2006a), in which they introduced a 'multiple retrieval technique', with the intention of applying the same approach to solve large timetabling problems. This research formed part of the PhD work reported in Qu (2002). Those studies described above focused on assessing the reusability of previous good timetable solutions or part of the timetable solutions (stored in a case base) in order to generate new timetable for new target problem instances by measuring their similarity.

The second approach that promotes the issue of generality in problem solving techniques, which has gained the attention of the timetabling community, is the so called *hyper-heuristics* approach. An excellent introduction of the *hyper-heuristics* approach can be found in Burke *et al.* (2003b). They define a hyper-heuristic as:

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

Ross (2005a) noted that the broad aim of hyper-heuristic approaches is

> *" to discover some algorithm for solving a whole range of problems that is fast, reasonably comprehensible, trustable in terms of quality and repeatability, and with good worst-case behaviour across that range of problems."*

A variety of hyper-heuristic approaches have been developed in attempts to solve university timetabling problems. *Tabu Search* based hyper-heuristics have been successfully developed for examination timetabling by (Burke *et al.*, 2003c; Kendall and Mohd Hussin, 2005a,b). *CBR* has also been investigated in the hyper-heuristic context for choosing heuristics in the construction of university timetable solutions. Such an approach was presented by Burke *et al.* (2002) and Petrovic and Qu (2002) in which *CBR* was employed to predict the appropriate heuristic for course timetabling problems. In Burke *et al.* (2006b), *Tabu Search* was integrated with a *Case Based Reasoning* technique to search for the best sequence of heuristic orderings for particular timetabling problems. They experimented with the approach using artificially generated course and examination timetabling problems. Their experimental results suggested that using permutations of different heuristic orderings in solving the problem is better than using any single heuristic ordering alone. Case based heuristic selection for university examination timetabling has also been investigated in Yang and Petrovic (2005) (see the second paragraph of Section 2.5). Rattadilok *et al.* (2005) investigated a *choice function* based hyper-heuristic that applied to course timetabling problems. In addition to the sequential *choice function* based hyper-heuristic algorithm on a single processor, they also experimented with parallel architectures in which two distributed *choice function* based hyper-heuristic approaches were developed by implementing software agent technology. The aim of applying distributed algorithms is to extend the search space coverage and to reduce the computational time in the timetable constructions. Their experimental results showed that, when distributed algorithms were used, better solutions can be generated in shorter times compared to those when the sequential *choice function* based hyper-heuristic was implemented.

Recently, Burke *et al.* (2007) proposed a new graph based hyper-heuristic approach for solving course and examination timetabling problems. Instead of using a single heuristic to find solutions for course and examination timetabling problems, a sequence of heuristics was applied. The authors used *Tabu Search* and deepest descent local search in order to find the best list of heuristics to guide the constructive algorithm in finding the 'best solution' for each problem instance. A comprehensive experimental study on hyper-heuristics that analysed the performance of combinations of heuristic selection mechanisms and move acceptance criteria is presented in Bilgin *et al.* (2006). With regards to examination timetabling problems, the results demonstrated that the combination of *choice function* heuristic selection with *Monte Carlo* acceptance criteria was better than the other hyper-heuristic combinations. Note that heuristics mentioned in this context are not limited only to *heuristic orderings* but also include other type of heuristics such as low level heuristics used to move or swap events during the timetable construction (or improvement).

2.7 Chapter Summary

This Chapter has presented a brief introduction of educational timetabling problems, with a more detailed description of examination timetabling problems. As timetabling problems are tedious tasks to solve manually, a wide variety of approaches and algorithms have been applied to timetabling problems with the aim of developing computer-based automated timetabling systems. An overview of graph based heuristics implemented in the construction of initial solutions was presented. Various heuristic and meta-heuristic approaches that have been implemented in the improvement phase were also highlighted (although it should be noted that iterative improvement is outside the scope of this thesis). Furthermore, the range of objective functions, and various multi-objective and multi-criteria approaches used in timetabling were discussed. Finally, a review of the various ways in which fuzzy methodologies have been used in the context of timetabling was presented. In the next Chapter, a background to the fuzzy techniques utilised in the remainder of this thesis is presented, for the reader unfamiliar with fuzzy systems.

Chapter 3

Theory of Fuzzy Sets and Fuzzy Systems

3.1 Introduction

In many decision making environments, it is often the case that several factors need to be taken into account simultaneously. Often, it is not known which factor(s) need to be emphasised more in order to generate a better decision. Somehow, a trade off between the various (potentially conflicting) factors must be made. The general framework of fuzzy reasoning facilitates the handling of such uncertainty. Fuzzy systems are used for representing and employing knowledge that is imprecise, uncertain, or unreliable. This Chapter will describe the general properties of fuzzy set theory.

The concept of fuzzy logic was first introduced in 1965 by Zadeh in his seminal paper on fuzzy sets (Zadeh, 1965). Since then, research on fuzzy set has expanded to cover a wide range of disciplines and applications. In the present thesis, the use of fuzzy techniques is focused only on its use in rule-based systems. Therefore, this Chapter presents a general background of the fuzzy set theory and fuzzy methodologies that are utilised within the research work. The contents have been selected to be sufficient to explain how these fuzzy techniques work. A fully detailed descriptions of the logical framework based on fuzzy sets (i.e. full fuzzy logic) is not included, as it is not utilised here. For a full description of the functioning of fuzzy systems, the interested reader is referred to Cox and O'Hagen (1998) for a simple treatment or Zimmermann (1996) for a more complete treatment.

3.1.1 Fuzzy Sets and Membership Functions

Fuzzy sets can be considered as an extension of classical or 'crisp' set theory. In classical set theory, an element x is either a member or non-member of set A. Thus, the membership $\mu_A(x)$ of x into A is given by:

$$\mu_A(x) = \begin{cases} 1, & \text{if } x \in A \\ 0, & \text{if } x \notin A \end{cases}$$

Consider room temperature as an example. One might say that "a temperature less than 10°C is cold". This statement can be represented in the form of classical set as $cold = \{x|x \leq 10\}$ and the membership function characterising this set is shown in Figure 3.1.

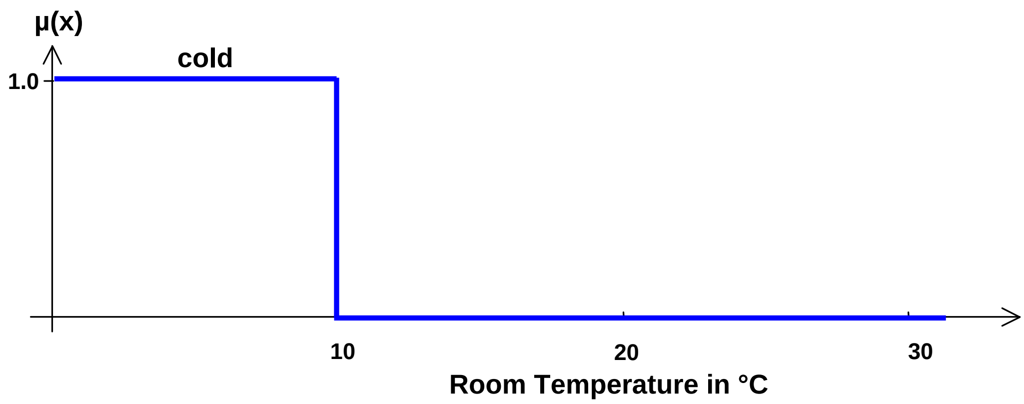


Figure 3.1: Membership function for the set of cold temperatures, defined as $cold = \{x|x \leq 10\}$

In contrast to classical set theory, the fuzzy set methodology introduced the concept of *degree* to the notion of membership. More formally, a fuzzy set A of a universe of discourse X (the range over which the variable spans) is characterised by a *membership function* $\mu_A(x) : X \rightarrow [0, 1]$ which associates with each element x of X a number $\mu_A(x)$ in the interval $[0, 1]$, with $\mu_A(x)$ representing the *grade of membership* of x in A. The precise meaning of the membership grade is not rigidly defined, but is supposed to capture the 'compatibility' of an element to the notion of the set.

Returning to the example above, an everyday statement like "a temperature below about 10℃ is considered cold" can be represented in the form of the fuzzy set shown in Figure 3.2. In comparison with classical set in which only sharp boundaries are permitted, the concept of membership degree in fuzzy sets allows fuzzy or blurred boundaries to be defined. In Figure 3.2, it can be seen that a temperature of 11℃ can also be considered as cold but with a lesser degree of membership than for 10℃ (i.e $\mu_{cold}(x = 11) = 0.85$); whereas in a classical set the degree of membership is zero (i.e. a temperature of 11℃ does not belong to the set *cold* at all). Fuzzy sets provide the tools to represent problems in everyday language, and it is this property that provides a problem solving technique that mimics the characteristics of human reasoning and decision making.

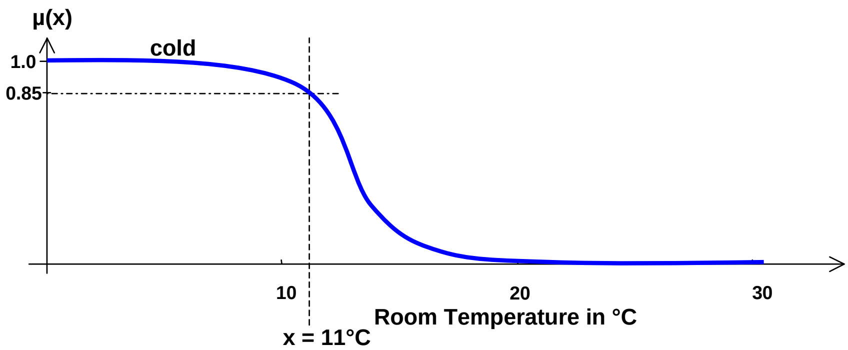


Figure 3.2: Membership function for the fuzzy set $cold = \{x \mid x \ is \ less \ than \ about \ 10\}$

3.1.2 Linguistic Variables, Values and Rules

The term 'linguistic variable' was introduced by Zadeh (1975) to refer to a variable whose values are in the form of "linguistic expressions" rather than numerical values. In the example shown in Figure 3.2, 'temperature' is a linguistic variable with a linguistic value 'cold'. Other possible linguistic values for the linguistic variable 'temperature' could include terms such as 'moderate', 'warm' and 'hot'. Each linguistic value is represented by a fuzzy set (membership function) in which the characteristic of each fuzzy set is dependent on the context of the particular problem. Although these linguistic terms are very subjective, they might be interpreted as (for example):

- 'cold' to be a temperature below about 10 ℃
- 'moderate' to be a temperature around 15 ℃
- 'warm' to be a temperature around 20 ℃
- 'hot' to be a temperature above about 25 ℃

In a universe of discourse $U = [0, 50]$, these linguistic values would be associated with fuzzy sets whose membership functions are as follows:

$$\mu_{cold}(x) = \begin{cases} 1, & \text{if } x \leq 10 \\ 1 - (x-10)/5, & \text{if } 10 < x < 15 \\ 0, & \text{otherwise} \end{cases}$$

$$\mu_{moderate}(x) = \begin{cases} 1 - |x-15|/5, & \text{if } 10 < x < 20 \\ 0, & \text{otherwise} \end{cases}$$

$$\mu_{warm}(x) = \begin{cases} 1 - |x-20|/5, & \text{if } 15 < x < 25 \\ 0, & \text{otherwise} \end{cases}$$

$$\mu_{hot}(x) = \begin{cases} 1, & \text{if } x \geq 25 \\ 1 - (x - 30)/5, & \text{if } 20 < x < 25 \\ 0, & \text{otherwise} \end{cases}$$

Graphical representations of these fuzzy sets are shown in Figure 3.3. Over the universe of discourse, the temperature T is partitioned into four fuzzy sets — *cold*, *moderate*, *warm* and *hot*. These fuzzy sets are partially overlapping. Hence, it can be seen that the room temperature of 18℃ has partial membership in both the fuzzy set *moderate* and the fuzzy set *warm*, where

$$\begin{aligned} \mu_{moderate}(x = 18) &= 0.25, \quad \text{and} \\ \mu_{warm}(x = 18) &= 0.75 \end{aligned}$$

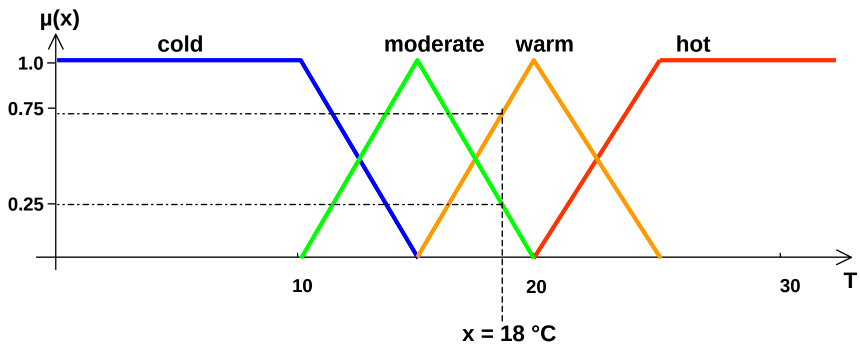
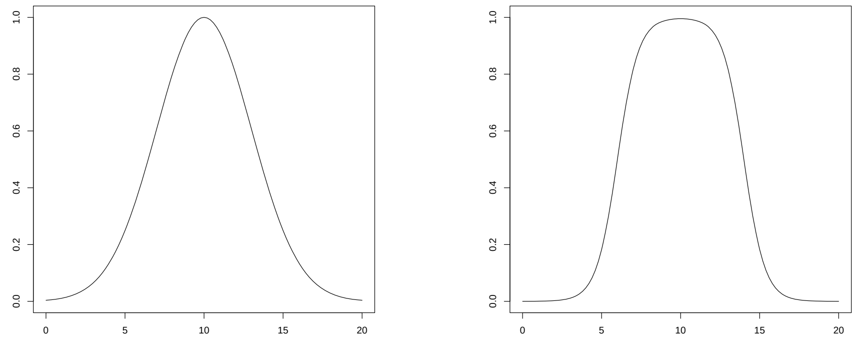


Figure 3.3: Membership functions for the linguistic variable 'temperature'

In this example, triangular and trapezoidal shape membership functions are defined. In practice, any kind of membership functions that are suitable for the problem in hand can be defined and used. Some common functions are depicted in Figure 3.4.

In order to perform inference, rules which connect input variables to output variables in 'IF ... THEN ...' form are used to describe the desired system response in terms of



(a) Gaussian (b) Sigmoid

Figure 3.4: Some common membership functions

linguistic variables (words) rather than mathematical formulae. The 'IF' part of the rule is referred to as the 'antecedent', the 'THEN' part is referred to as the 'consequent'. The number of rules depends on the number of inputs and outputs, and the desired behaviour of the system. Once the rules have been established, such a system can be viewed as a non-linear mapping from inputs to outputs.

Based on this general form of fuzzy rules, several alternative ways of defining fuzzy rules have been used for knowledge representation in fuzzy systems (Kasabov, 1996, p. 192). In this research, the standard form of Mamdani-style fuzzy rules (Mamdani and Assilian, 1975) are implemented. In Mamdani's approach, rules are of the form:

$$R_i : \text{if } (x_1 \text{ is } A_{i1}) \text{ and ... and } (x_r \text{ is } A_{ir}) \text{ then } (y\ is\ C_i) \text{ for } i = 1, 2, ..., L$$

where L is the number of rules, x_j $(j = 1, 2, 3, ..., r)$ are input variables, y is the output variable, and A_{ij} and C_i are fuzzy sets that are characterised by membership functions $A_{ij}(x_j)$ and $C_i(y)$, respectively. In the fuzzy reasoning process (a more detailed explanation is given in Section 3.1.6), each rule is evaluated in order to determined the degree of fulfillment of the rule.

3.1.3 Fuzzy Operators

The main fuzzy operations defined by Zadeh (1965) are as follows:

Let A and B be two fuzzy sets with membership functions $\mu_A(x)$ and $\mu_B(x)$ respectively. The intersection operation (which corresponds to the logical 'AND') is defined as

$$\mu_{A \cap B}(x) = min[\mu_A(x), \mu_B(x)] \tag{3.1}$$

and the union operation (which corresponds to the logical 'OR') is defined as

$$\mu_{A \cup B}(x) = max[\mu_A(x), \mu_B(x)] \tag{3.2}$$

In addition, the complement operator (which corresponds to the logical 'NOT') is defined as

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \tag{3.3}$$

A graphical representation of these operations is shown in Figure 3.5.

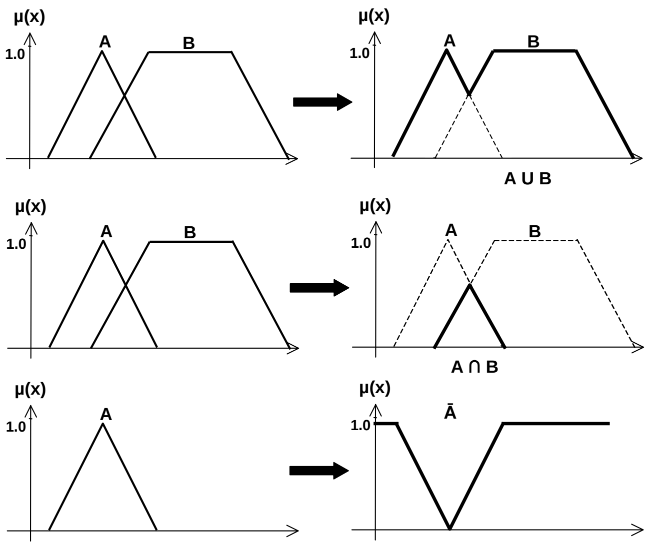


Figure 3.5: Fuzzy sets operations (adapted from Negnevitsky (2002, Chap. 4))

3.1.4 Fuzzy Hedges

In addition to the primary linguistic values (terms), it is also possible to apply the concept of fuzzy modifiers, called *hedges*. Terms such as *very*, *more or less*, and *slightly* are examples of *hedges*. Hedges are applied to linguistic values in order to modify the shape of the particular fuzzy sets. The ability to define hedges provides more flexibility in defining fuzzy statements that are closer to everyday language. In practice, the terms categorised as hedges have mathematical expressions that define their operations. Some examples of hedges with their mathematical expressions and graphical representations are shown in Table 3.1. However, the actual definition of hedges and their operations for any particular problem are, again, subjective and dependent on the desired behaviour of the fuzzy system.

Table 3.1: Examples of hedges (taken from Negnevitsky (2002, Chap. 4)). For the graphical representation, the thicker line is the new shape when the hedge act on the linguistic value.

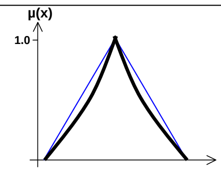
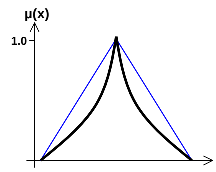
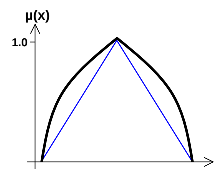
Hedge	Mathematical expression	Graphical representation
Slightly	$[\mu_A(x)]^{1.7}$	
Very	$[\mu_A(x)]^2$	
More or less	$\sqrt{\mu_A(x)}$	

Figure 3.6 depicts the application of the hedge 'very' to the linguistic value 'warm'. A room temperature of 18°C has 0.7 degree of membership in the fuzzy set 'warm', and so belongs to the fuzzy set 'very warm' with a membership degree of 0.49.

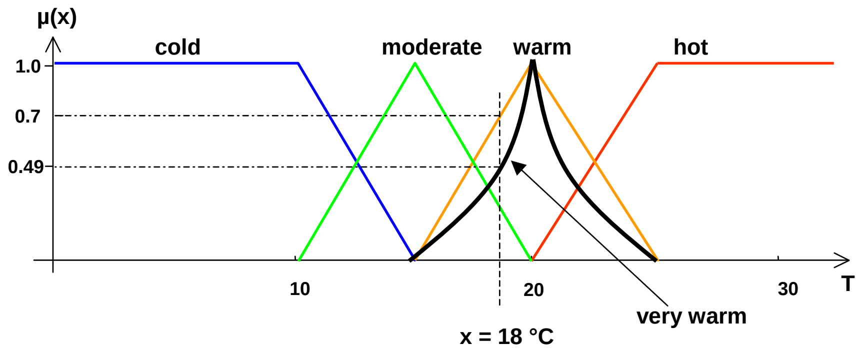


Figure 3.6: Apply hedge 'very' onto linguistic value 'warm'

3.1.5 Defuzzification Methods

The final output of a Mamdani system is one or more arbitrarily complex fuzzy sets which (usually) need to be defuzzified. Defuzzification is a mathematical process used to extract crisp output from fuzzy output set(s). This process is necessary because all fuzzy sets inferred by fuzzy inference in the fuzzy rules must be aggregated to produce one single number as the output of the fuzzy model. Various types of defuzzification have been suggested in literature (Cox and O'Hagen, 1998). The properties of the specific application being developed will determine which defuzzification method can be utilised. However, there is no systematic procedure to choose which method is the most suitable for any given application. In the following sections, the five most often used defuzzification methods are described.

3.1.5.1 Centre of Gravity (COG) Method

Probably the common form of defuzzification is termed the 'centre of gravity' method, as it is based upon the notion of finding the centroid of a planar figure. This method can be expressed mathematically as follows:

$$x^* = \frac{\int_a^b \mu(x) \cdot x dx}{\int_a^b \mu(x) dx}$$

Theoretically, the output is calculated over a continuum of points in the aggregate membership function. In practice, an approximate value can be derived by calculating it over a sample of points. The formula is given by:

$$x^* = \frac{\sum_a^b \mu(x) \cdot x}{\sum_a^b \mu(x)}$$

Figure 3.7 shows a graphical illustration of the method of finding the point representing the centre of gravity in the interval $[a, b]$ for the output fuzzy set.

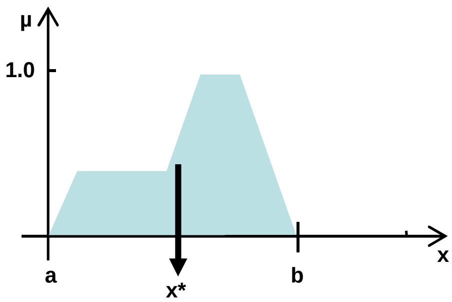


Figure 3.7: The Centre of Gravity (COG) method of defuzzification

3.1.5.2 The Mean of Maxima (MOM) Method

The Mean of Maxima method returns the average of the base-variable values at which their membership values reach the maximum. The formula is given by:

$$x^* = \sum_{j=1}^{k} \frac{x_j}{k}$$

where k is the number of discrete elements of the output fuzzy set that reach the maximum memberships. The graphical illustration of the method is shown in Figure 3.8.

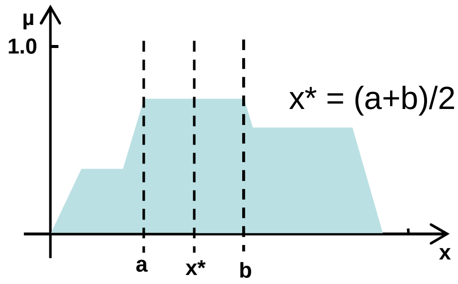


Figure 3.8: The Mean of Maxima (MOM) method of defuzzification

3.1.5.3 The Smallest of Maxima (SOM) and The Largest of Maxima (LOM) Methods

The Smallest of Maxima method, returns the smallest value of x that belongs to $[a, b]$ at which their membership values reach the maximum. Meanwhile, The Largest of Maxima method, returns the largest value of x that belongs to $[a, b]$.

A graphical illustration of these methods is shown in Figure 3.9.

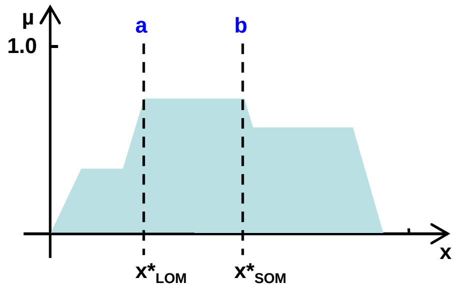


Figure 3.9: The Smallest of Maxima (SOM) and The Largest of Maxima (LOM) methods of defuzzification

3.1.5.4 The Bisector of Area (BOA) Method

The Bisector of Area (BOA) Method returns the vertical line that partitions the region into two sub-regions of equal area. This method satisfies

$$\int_{\alpha}^{x*} \mu_A(x)dx = \int_{x*}^{\beta} \mu_A(x)dx$$

where $\alpha = min\{x|x \in X\}$ and $\beta = max\{x|x \in X\}$. A graphical illustration of this method is shown in Figure 3.10.

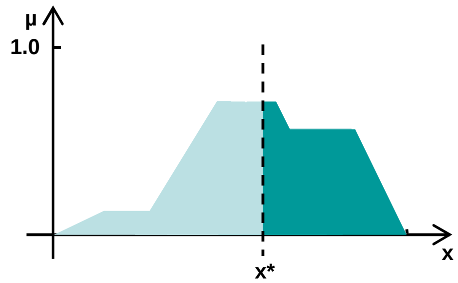


Figure 3.10: The Bisector of Area (BOA) method of defuzzification

3.1.6 Overview of Fuzzy Systems

Figure 3.11 shows the five interconnected components of a fuzzy system. The fuzzification component computes the membership grade for each crisp input variable based on the membership functions defined. The inference engine then conducts the fuzzy reasoning process by applying the appropriate fuzzy operators in order to obtain the fuzzy set to be accumulated in the output variable. The defuzzifier transforms the output fuzzy set to a crisp output by applying a specific defuzzification method.

Briefly, the main steps in fuzzy system design are as follows:

- Analyse and understand the problem in consideration.

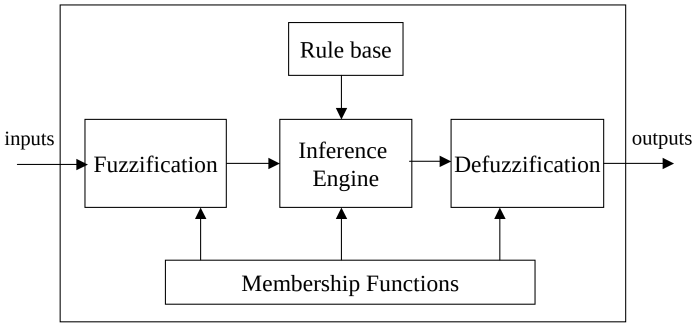


Figure 3.11: Components of fuzzy system

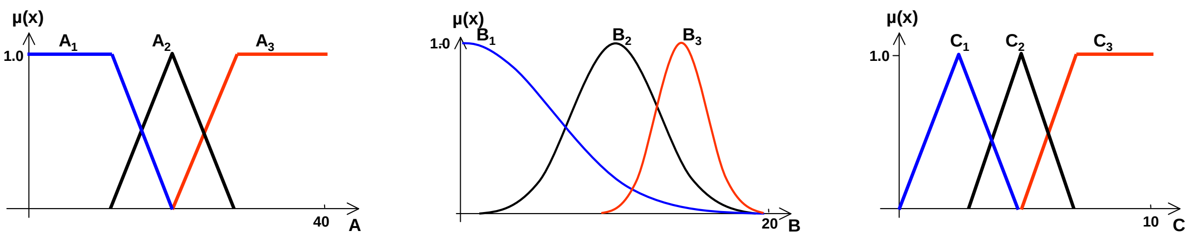
- Determine the linguistic variables (the inputs and outputs). For each linguistic variable, identify the linguistic values and define the fuzzy sets (membership functions).
- Identify and define the fuzzy rule set.
- Choose the appropriate methods for fuzzification, fuzzy inference and defuzzification.
- Evaluate the system.

If necessary, this sequence of steps is then repeated an arbitrary number of times while fine tuning the fuzzy system by modifying the fuzzy input/output sets and/or fuzzy rules.

In reality, modeling a fuzzy system is a difficult task. Finding a sufficiently good system can be viewed as a search problem in high-dimensional space, in which each point represents a rule set, the membership functions, and the evaluation function is some measure of the corresponding system behaviour. This is due to the fact that the performance of a fuzzy system is highly dependent on how the system developer defines the linguistic variables, the membership functions, fuzzy rules set and so on. No formal methods exist to determine the appropriate fuzzy model in a given context. The term 'fuzzy model' is used to mean the combination of selected linguistic variables (input and output variables), membership functions for each linguistic variable and a rule set (as the inference engine and the fuzzification methods are fixed — see below). Most of the time, the system is either built based on expert knowledge or by systematically training the system using the available data. There are many alternative ways in which this general fuzzy methodology (as shown in Figure 3.11) can be implemented in any given problem. In our implementation, the standard Mamdani style fuzzy inference was used with standard Zadeh (min-max) operators (Negnevitsky, 2002).

Consider a simple example, in order to understand how Mamdani style fuzzy inference works. This example is for a fuzzy system with two input variables and one output variable. The purpose of this example is to illustrate how the final crisp output is obtained for the particular input values.

Step 1 - Determining linguistic variables and fuzzy sets. Let the two inputs be represented as linguistic variables A and B; and the output as linguistic variable C. A_1, A_2 and A_3 are linguistic values for A; B_1, B_2 and B_3 are linguistic values for B; C_1, C_2 and C_3 are linguistic values for C with membership functions as shown in the graphical representations given in Figure 3.12.



(a) Input: A　(b) Input: B　(b) Output: C

Figure 3.12: Characteristic of linguistic variables

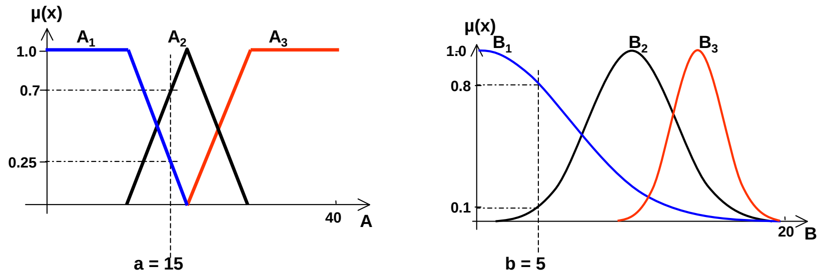
Let us define three rules as follows:

Rule 1 : IF (a is A_1) AND (b is B_1) THEN (c is C_1)

Rule 2 : IF (a is A_2) OR (b is B_2) THEN (c is C_2)

Rule 3 : IF (a is A_3) AND (b is B_3) THEN (c is C_3)

Step 2 - Fuzzification. The fuzzified values for input values $a = 15$ and $b = 5$ are shown in Figure 3.13.



(a) Input: A (b) Input: B

Figure 3.13: The fuzzified value for both input linguistic variables

Step 3 - Fuzzy Inferencing (Evaluate Rules). The firing level for each rule is determined using the min-max operator shown in Equations (3.1) and (3.2). If the *AND* operator appears in the antecedents part, the minimum fuzzified value will be selected. On the other hand, if the *OR* operator appears, the maximum fuzzified value will be selected. Figure 3.14 shows the process graphically. It can be seen that *Rule 3* is not activated because both input values (i.e. $a = 15$ and $b = 5$) have zero membership degree for the linguistic values A_3 and B_3 respectively.

Step 4 - Rules Output Aggregation. Having evaluated all the rules, the final shape of the output is determined by combining all of the activated rule consequents. The aggregation result is shown in Figure 3.15.

Step 5 - Defuzzification. *COG* method of defuzzification (as described in Section 3.1.5.1) is used to defuzzified the output fuzzy set. Figure 3.16 shows the calculated 'centre of gravity' of the final output fuzzy set for this simple example problem.

Even when created with expert knowledge, the system invariably needs to be fine tuned in order to obtain a satisfactory system performance (where 'satisfactory' may be defined in terms of how good is the fuzzy system is compared to the equivalent manual system; or perhaps in terms of whether the system behaves as previously specified; etc.). The use of search algorithms for tuning fuzzy system has been applied by many

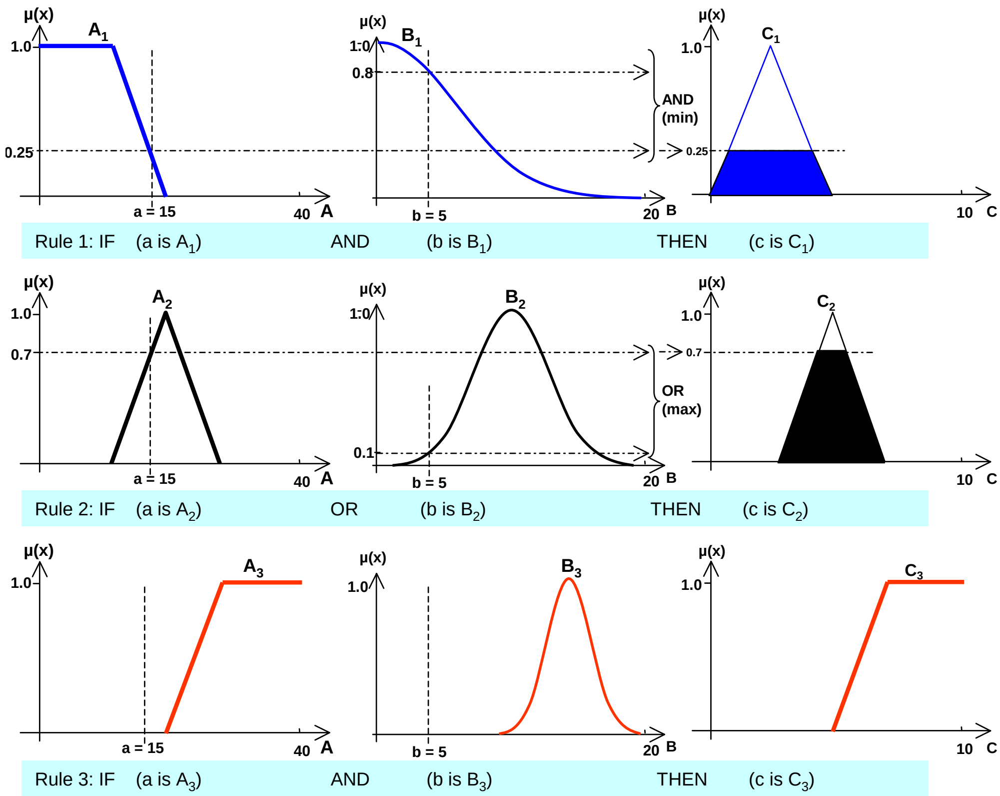


Figure 3.14: Evaluation of rules fulfillment (firing levels)

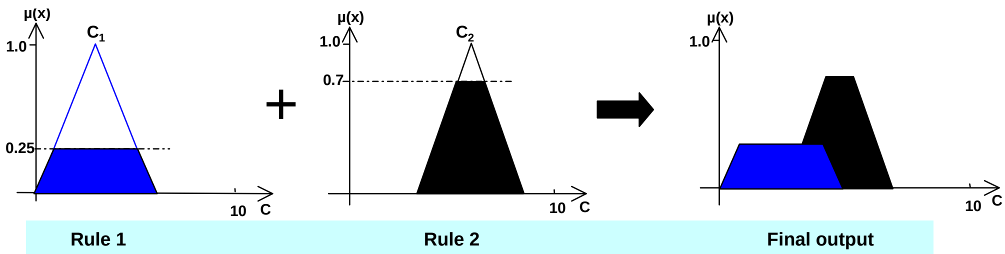


Figure 3.15: Aggregation of rules

researchers. Such methods include Genetic Algorithms (Gómez-Skarmeta and Jiménez, 1999; Setnes and Roubos, 2000; Shimojima *et al.*, 1995; Wang *et al.*, 1998) and Simulated Annealing (Garibaldi and Ifeachor, 1999).

In spite of the fact that sophisticated search techniques are often utilised in fuzzy tuning, it was outside the scope of this thesis to perform any extensive application of such methods. As the focus was to investigate the applicability of fuzzy techniques in the

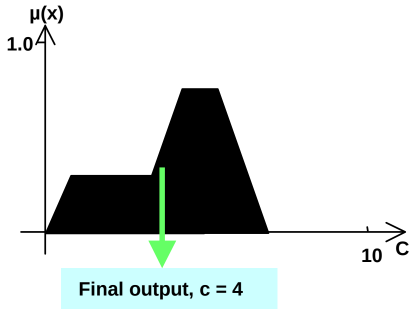


Figure 3.16: Defuzzification of final shape

university timetabling problem, a simple exhaustive search was employed for fine tuning the fuzzy system (more details of the tuning process utilised are given in Section 4.4.1.3).

3.2 Chapter Summary

This Chapter presents the basics of fuzzy set theory, fuzzy inference and fuzzy modelling. Although the presented material only covers a very small part of the huge body of fuzzy set theory and fuzzy techniques in general, its is designed to be enough for the unfamiliar reader to understand the conceptual framework of the fuzzy methodologies that are implemented in the rest of this thesis.

The very limited previous research into fuzzy techniques in timetabling problems encouraged the author to investigate alternative ways of employing fuzzy techniques to assist in finding solutions for timetabling problems. It is the author's belief that the power of fuzzy techniques may be very useful in the timetabling problem environment, in which key decisions are influenced by many subjective factors. The ability to represent the problem in natural language may provide the mechanism to investigate how human experts (timetabling officers) construct timetable solutions in the real world. Although a thoroughly exhaustive examination of fuzzy techniques in all aspects of timetabling would be a vast undertaking, clearly beyond the scope of any one thesis, this thesis sets out, for the first time, to explore these issues.

Part II

Fuzzy Construction

Chapter 4

Fuzzy Multiple Heuristic Orderings for Examination Timetabling

4.1 Introduction

This Chapter presents an initial investigation into considering multiple heuristic orderings simultaneously for measuring the difficulties of scheduling exams into time slots. As far as the author is aware this work is the first attempt to apply fuzzy techniques in considering more than one heuristic ordering to measure the difficulty of assigning exams into time slots. To allow full investigation and analysis, the scope of the preliminary study presented in this Chapter is to combine two heuristic orderings simultaneously. In further defining the problem at this stage, three out of five single heuristic orderings described in Section 2.2.2 are considered to be combined (with three alternative combinations of two heuristic orderings simultaneously). Further investigations that consider three heuristic orderings simultaneously are presented in Chapter 5. In Chapter 6, various combinations of two and three heuristic orderings are considered as the five single heuristic orderings are explored.

This Chapter is a very important and necessary first step as it serves as the foundation for the detailed analysis outlined over the following two Chapters. It can be divided into two parts. In the first part, the approach developed is described, followed by initial experimental results. In the second part, results of more extensive investigations are reported and rigorous analysis of the compared heuristics are presented. The main aims of this Chapter are as follows:

- To illustrate that fuzzy multiple heuristic ordering is more effective compared with single heuristic ordering where one heuristic is implemented individually
- To analyse the effect of using different combinations of heuristic orderings for constructing initial solutions of timetabling problems

4.2 The Basic Sequential Constructive Algorithm

The sequential construction algorithm used as part of this investigation employs a heuristic ordering in the initial construction phase. This is depicted in Figure 4.1. The sequential constructive algorithm requires the following steps:

Process 1 *Choose heuristic ordering*

In order to determine the sequence in which exams are scheduled to a valid time slot, it must be decided which heuristic ordering is to be employed. Usually, any of the heuristic orderings described earlier can be employed on their own to measure the exams' difficulty to be scheduled. In this research, an alternative approach is introduced in which two heuristic orderings are considered simultaneously to measure the exams' difficulty.

Process 2 *Calculate the difficulty of the exam to be scheduled*

Having chosen a heuristic ordering to be implemented, the calculation of the assessment of difficulty is performed and exams are ordered in a specified sequence.

Process 3-Process 5 *Sequentially assign exams to time slots*

For each exam in turn (starting with the most difficult to schedule) the following

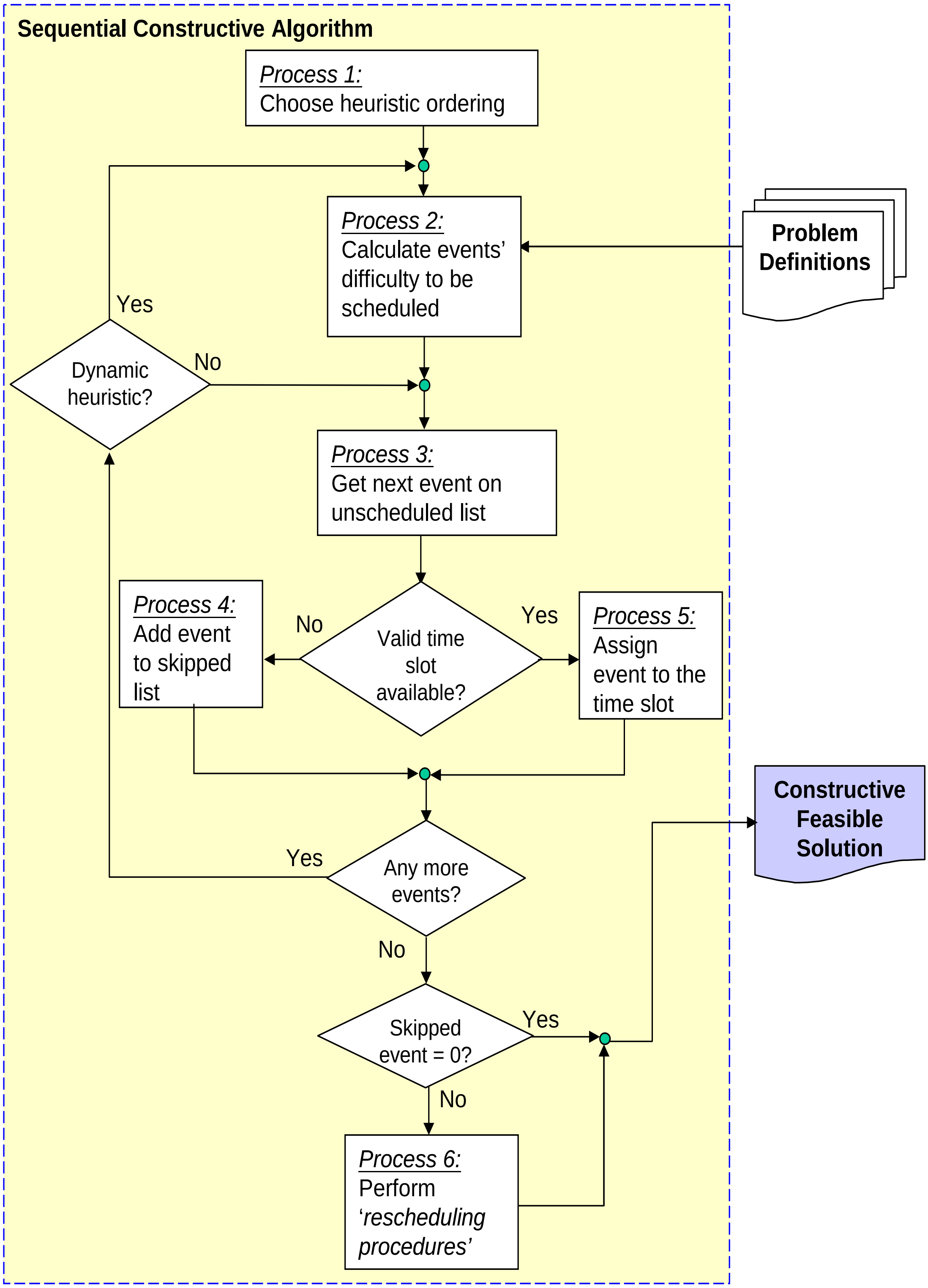


Figure 4.1: A general framework for producing timetabling solutions

sequence of events are carried out. The free time slots are examined in turn to find valid ones, and for each, the penalty is calculated that would result from placement of the exam in that slot. After examining each of the time slots, the exam is scheduled into the available slot incurring the least penalty (if two or more slots share the lowest penalty cost, the exam is scheduled into the last such time slot). If no valid time slot is available, the exam is not assigned and is recorded on a 'skipped list'. If a dynamic heuristic is being used, the remaining exams' difficulties are updated and the exams are reordered accordingly.

Process 6 *Perform a* 'rescheduling procedure'

This process is only performed when there is at least one exam that could not be scheduled because no valid time slot was available — i.e there are skipped event(s) from *Process 3*. The process for scheduling the skipped exams is shown in Figure 4.2.

```
Copy all skipped events into unscheduled events list
While there exist unscheduled events
    E* = next unscheduled event that needs to be scheduled;
    Find time slots where event E* can be inserted with minimum number of
    scheduled events needed to be removed from the time slot;
    If found more than one time slot with the same number of scheduled events
    need to be removed
        Select a time slot t randomly from the candidate list of time slots;
    End if
    While there exist events that conflict with event E* in time slot t
        Et = next conflicted event in time slot t ;
        If found another time slot with minimum penalty cost to move event Et
            Move event Et to the time slot;
        else
            Bump back event Et to unscheduled events list;
        End if
    End While
    Insert event E* to timeslot t ;
    Remove event E* from unscheduled event list;
    If dynamic ordering heuristic is in used
       Sort unscheduled events using selected heuristic ordering;
    End if
End While
```

Figure 4.2: Pseudo code for the '*rescheduling procedure*' used if 'skipped' exams exist

The steps outlined above continue until all the exams are scheduled (i.e. feasible solution is constructed). The reason for this is to make sure that the timetable produced is comparable to results published in the literature - in the context of the benchmark data sets use in this research.

The sequential construction algorithm used here is similar to the approach applied by Carter *et al.* (1996) with some modification. Basically, there are three differences between these two algorithms. The first difference is related to the initial stage of the algorithm. In the algorithm used here, the heuristic ordering is applied to *all* exams, whereas Carter *et al.*'s algorithm first finds the maximum-clique of examinations and assigns them to different time slots, and then applies heuristic ordering to the remaining exams. The second difference is in the selection of the free time slot. As with both approaches, a search is carried out to find the clash free time slot with least penalty cost in order to assign each exam to a time slot. In the algorithm used here, if several time slots are available, then the last available time slot in the list will be selected. (It was found that the choice of assigning exams to the last available time slot or the first available time slot made little difference, as the main purpose of this was simply to spread out the student's timetable.) In contrast, Carter *et al.* chose the first clash free time slot found in which to assign the exam. Thirdly, for reshuffling a scheduled exam, a time slot is randomly selected from the list of time slots with the minimum number of scheduled exams that needed to be 'bumped back', whereas Carter *et al.* used a *minimum disruption cost* to break any ties.

Although the main purpose of the '*rescheduling procedure*' is to make sure all exams can be scheduled into time slots, it is not guarantee that this procedure can be applied to construct a feasible timetable for any timetabling problem. No thorough experimentation was performed to test the reliability of the function in term of it applicability to other timetabling problems. In fact, throughout this research (especially for membership functions tuning), the maximum number of iterations allowed for '*rescheduling proce-*

dure' have been set. Meaning that, for any combination of *cp* parameters, the scheduling process will be terminated if the number iterations of '*rescheduling procedure*' is exceeded the predefined number of iterations allowed. Later on, in the discussion of the experiment results, it can be seen that (see Tables 4.11 and 4.12) the performance of the '*rescheduling procedure*' is dependent on the heuristic (or combination of heuristics) applied to measure the difficulty of scheduling the exams. Therefore, it might be possible that depending on the complexity of the timetabling problem instances, the '*rescheduling procedure*' could cycle for ever without a feasible solution ever being reached.

4.3 Why Fuzzy Multiple Heuristic Orderings?

As mentioned earlier in Chapter 3, making decisions in multiple attributes environment is not an easy task. When making a decision based on more than one attribute, the problem lies in deciding which attribute should be emphasized in order to obtain the best decision. Often it is difficult to resolve conflicting attributes. Consider the example shown in Figure 4.3. In this example, there are ten exams ($e1$, $e2$, $e3$, $e4$, $e5$, $e6$, $e7$, $e8$, $e9$, $e10$) with the given *LD* and *LE* values. Figure 4.3(a) shows the ten exams in an unordered list, Figures 4.3(b) to (f) show the results of using different heuristic orderings to order the ten exams. It can be seen that when two different heuristic orderings are used individually, the orderings are substantially different (see Figure 4.3(b) and Figure 4.3(c)).

It is interesting to note that if both heuristic orderings are used as a pair (e.g. use *LD* as the main attribute and *LE* to break any tie, or vice versa — see Figure 4.3(d)), the ordering is almost the same as that produced when only the main attribute used on its own. This can be observed if we compare Figure 4.3(b) with Figure 4.3(d); and Figure 4.3(c) with Figure 4.3(e).

Another way to use both attributes to handle such multiple attribute decision making is simply to multiply the value of each attribute by a weighting factor and summate (i.e.

form a simple linear combination). In this example, the formulation is:

$$weight(e_j) = w_l LD_j + w_e LE_j$$

where $j = 1, 2, ...n$; n is the number of exams; and w_l and w_e are the weighting factors (any real number) for LD and LE respectively. Using a simple combination to represent the relative importance of both attributes can result in quite a different ordering (see Figure 4.3(f) where weights $w_l = 0.5$ and $w_e = 0.6$ have been used - these values were arbitrarily chosen to illustrate the point). In effect, neither the LD nor LE attributes alone control the exam ordering; it is determined by considering both attributes simultaneously. However, the problem then becomes that of needing to search for the

Unordered exams list

exams	LD	LE
e1	30	40
e2	10	30
e3	50	20
e4	20	35
e5	39	10
e6	10	43
e7	10	20
e8	19	25
e9	27	15
e10	45	30

(a)

Ordered by LD only

exams	LD	LE
e3	50	20
e10	45	30
e5	39	10
e1	30	40
e9	27	15
e4	20	35
e8	19	25
e2	10	30
e6	10	43
e7	10	20

(b)

Ordered by LE only

exams	LD	LE
e6	10	43
e1	30	40
e4	20	35
e2	10	30
e10	45	30
e8	19	25
e7	10	20
e3	50	20
e9	27	15
e5	39	10

(c)

Ordered by LD and then LE

exams	LD	LE
e3	50	20
e10	45	30
e5	39	10
e1	30	40
e9	27	15
e4	20	35
e8	19	25
e6	10	43
e2	10	30
e7	10	20

(d)

Ordered by LE and then LD

exams	LD	LE
e6	10	43
e1	30	40
e4	20	35
e10	45	30
e2	10	30
e8	19	25
e3	50	20
e7	10	20
e9	27	15
e5	39	10

(e)

Ordered by linear combination of both attributes

exams	LD	LE	weight
e10	45	30	40.5
e1	30	40	39.0
e3	50	20	37.0
e4	20	35	31.0
e6	10	43	30.8
e5	39	10	25.5
e8	19	25	24.5
e2	10	30	23.0
e9	27	15	22.5
e7	10	20	17.0

(f)

Figure 4.3: Example of examinations ordered by various combinations of heuristics

appropriate values of w_l and w_e to be used. Johnson (1990) implemented a similar formula for constructing initial solutions to the examination timetabling problem in which he set w_l to a constant value ($w_l = 1$) while varying the w_e value. The aim of this was simply to produce a range of alternative initial solutions which were then subject to iterative improvement.

Heuristic orderings are based on assumptions. For example, an exam is more difficult to schedule if it has a 'large' number of other exams in conflict or if it has a 'small' number of valid time slots available. This, in effect, is dealing with linguistic terms, where no exact values for 'large' and 'small' have been defined. This allows for a certain amount of uncertainty when attempting to combine such heuristics. The general framework of fuzzy reasoning facilitates the handling of such uncertainty. The original hypothesis was that this problem might be one where fuzzy techniques may be of use. In essence, fuzzy methodologies allow non-linear combinations of multiple heuristics to be considered.

4.4 The Fuzzy Multiple Heuristic Ordering

This Section introduces the concept of fuzzy multiple heuristic ordering. The basic features of the *sequential constructive algorithm* used have been described in Section 2.2.2. As mentioned earlier, certain ordering strategies that have been widely studied for the timetabling problem have evolved from studying the graph colouring problem. As this is the first attempt to implement the concept of fuzzy multiple heuristic ordering, the preliminary investigation was based on three of these heuristic orderings - *LD*, *LE* and *SD*. As discussed in Section 2.2.2, these sequencing strategies have proven to be highly effective in constructing solutions for graph colouring problems and examination timetabling problems when applied on an individual basis.

4.4.1 Fuzzy Modeling

This Section presents the development of this particular fuzzy model. Considering the first three single heuristic orderings explained in Section 2.2.2, there are three alternative ways in which two single heuristic orderings can be simultaneously combined. The possible combinations are:

- *LD* and *LE*, referred to as the *Fuzzy LD+LE Model*
- *SD* and *LE*, referred to as the *Fuzzy SD+LE Model*
- *SD* and *LD*, referred to as the *Fuzzy SD+LD Model*

These three heuristic ordering combinations provide alternative ways for ordering a list of exams. Therefore, in *Process 1* (see Figure 4.1), instead of simply choosing any one of the single heuristic orderings to be implemented, the process needs to be modified/improved so that the fuzzy approach can be incorporated. Accordingly, the extended version of *Process 1* is shown in Figure 4.4. It is worth mentioning that fuzzy methodologies are *only* employed in *Process 1*; the other processes in the dotted-box of Figure 4.1 remain the same.

Fuzzy modeling can be thought of as the task of designing the fuzzy inference system specific to the particular application area. The selection of important parameters for the inference system is crucial, as the overall system behaviour is highly dependent on a large number of factors such as how the membership functions are chosen, the number of rules involved, the fuzzy operators used, and so on. As two heuristics are being combined into a single overall heuristic, a fuzzy system with two inputs and one output is developed. The input variables used are dependent on the heuristic combinations selected. Three pairs of input variables are possible, namely *LD* and *LE*, *SD* and *LE*, or *SD* and *LD*. With any pair of input variables, an output variable called *examweight* is generated. This output variable, *examweight*, represents the overall difficulty of scheduling an exam to a time slot. Each of the input and output variables are associated with

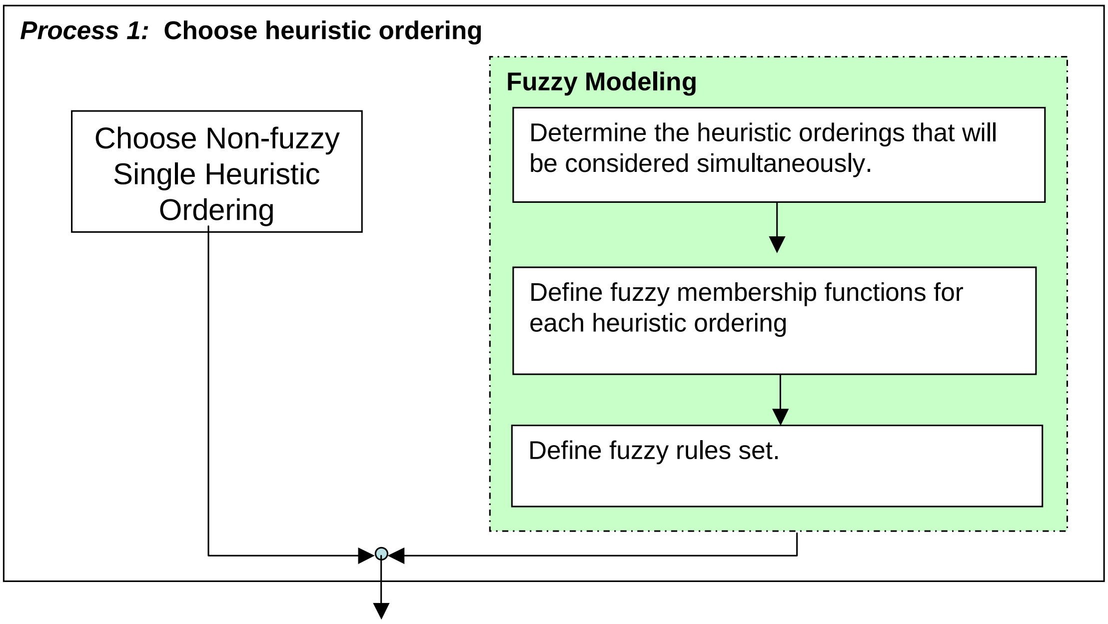


Figure 4.4: The steps involved in a fuzzy version of *Process 1* (from Fig. 4.1)

three linguistics terms: *small*, *medium* and *high*. Each linguistic term is represented by a fuzzy membership function.

Normalised Membership Functions By analysing the minimum and maximum values of each heuristic ordering (see Table 4.1), it can been seen that the values for different heuristic ordering are in widely different scales. To further complicate of the issue, for some heuristics, values between data sets are also widely different. For the purpose of maintainability (easy maintenance), it was decided to implement the membership function with the universe of the discourse (x-axis) for each fuzzy variable defined to the range between 0 and 1. This means that the actual input value needed to be transformed into a new value in the range $[0, 1]$. In general, this can be achieved using a transformation such as:

$$v' = \frac{(v - min_x)}{(max_x - min_x)}$$

where v is the actual value in the initial range $[min_x, max_x]$. In the case here, min_x, was set to zero for each of LD, LE and SD. In Table 4.1, it can be seen that, value for min in 24 cases are equal to zero, while another 24 cases are greater than zero (in the range between 1 and 22). Therefore, min_x, was set to zero for more convenient, as it doesn't make any difference. The maximum values were set by examination of the problem instance: $max_x(LD)$ was set to the largest number of conflicts found for any exam in the problem instance; $max_x(LE)$ was set to the maximum number of students enroled to any exam in the problem instance; and $max_x(SD)$ was set to the total number of time slots available in the problem instance.

This is due to the fact that, rather than recalculate the parameters for the fuzzy sets shape, it is much easier to transform the original value in the range $[min_x, max_x]$ to the new range $[0, 1]$. For example, if $v = 10$ in $[0, 20]$, the normalised value v' is 0.5 in the new range $[0, 1]$.

Table 4.1: Minimum and maximum values for heuristic LD, LE, SD and WLD for each data set. The minimum and maximum values for heuristic LCD is similar to LD.

	LD		***LE***		***SD***		***WLD***	
	min	**max**	**min**	**max**	**min**	**max**	**min**	**max**
CAR-F-92	0	381	2	1566	0	32	0	4740
CAR-S-91	0	472	2	1385	0	35	0	4718
EAR-F-83	4	134	1	232	0	24	4	1665
HEC-S-92	9	62	7	634	0	18	22	2315
KFU-S-93	0	247	1	1280	0	20	0	5089
LSE-F-91	0	134	1	382	0	18	0	1229
RYE-F-92	0	274	3	943	0	23	0	5118
STA-F-83	7	61	1	237	0	13	7	2090
TRE-S-92	0	145	1	407	0	23	0	1267
UTA-S-92	1	303	1	1314	0	35	1	4382
UTE-S-92	2	58	1	482	0	10	3	1847
YOR-F-83	7	117	1	175	0	21	7	779

4.4.1.1 An Illustrative Example

This section will illustrate the functioning of the fuzzy inference process for a nine-rule system based on two input variables, LD and LE. Each of the input and output variables were assigned three linguistic terms; fuzzy sets corresponding to meanings of *small*, *medium* and *high*, referred to as 'membership functions'. These membership functions were chosen arbitrarily to span the universe of discourse (range) of the variable. A rule set connecting the input variables (LD and LE) to a single output variable, *examweight*, was constructed. The following nine rules describe the behaviour of the system:

Rule 1: IF (LD is *small*) AND (LE is *small*) THEN (*examweight* is *very small*)
Rule 2: IF (LD is *small*) AND (LE is *medium*) THEN (*examweight* is *small*)
Rule 3: IF (LD is *small*) AND (LE is *high*) THEN (*examweight* is *medium*)
Rule 4: IF (LD is *medium*) AND (LE is *small*) THEN (*examweight* is *small*)
Rule 5: IF (LD is *medium*) AND (LE is *medium*) THEN (*examweight* is *medium*)
Rule 6: IF (LD is *medium*) AND (LE is *high*) THEN (*examweight* is *high*)
Rule 7: IF (LD is *high*) AND (LE is *small*) THEN (*examweight* is *medium*)
Rule 8: IF (LD is *high*) AND (LE is *medium*) THEN (*examweight* is *high*)
Rule 9: IF (LD is *high*) AND (LE is *high*) THEN (*examweight* is *very high*)

The first stage is to normalise the input values to lie in the range $[0, 1]$, as the universe of the discourse (x-axis) for the fuzzy variable was defined to be between 0 and 1. Figure 4.5 illustrates the inferencing of this system with arbitrarily chosen normalised values for LD and LE of 0.90 and 0.80, respectively. For each rule in turn, the fuzzy system operates as follows. Consider *Rule 6* as an example, as this rule provides a good example of firing levels for different membership functions for both input variables. The input component ('fuzzifier') computes the degree of membership for each input variable based on the membership functions defined. That is, in *Rule 6*, the degree of membership

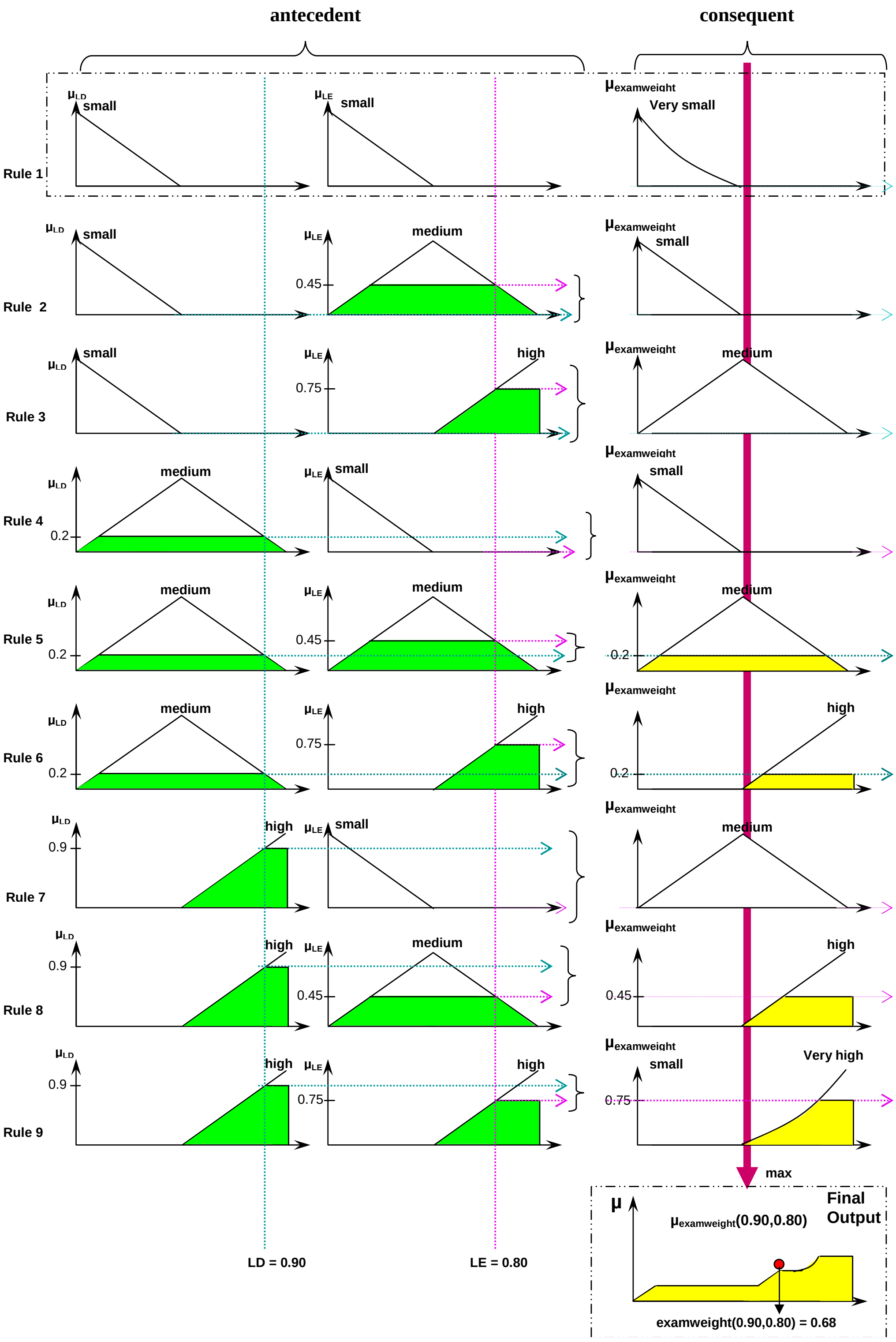


Figure 4.5: A nine-rule Mamdani inference process

is computed for LD in the fuzzy set *medium* and for LE in the fuzzy set *high*. As shown in the figure, the determined degree of memberships for each input variable are:

$$\begin{aligned} \mu_{medium}(LD = 0.90) &= 0.20, \quad \text{and} \\ \mu_{high}(LE = 0.80) &= 0.75 \end{aligned}$$

With these fuzzified values, the inference engine then computes the overall truth value of the antecedent of the rule (*Rule 6*) by applying the appropriate fuzzy operators corresponding to any connective(s) (*AND* or *OR*). In the example, the fuzzy *AND* operator is implemented as a minimum function:

$$\begin{aligned} \textbf{Rule 6} & \quad \text{IF } (LD \text{ is } \textit{medium}) \text{ AND } (LE \text{ is } \textit{high}) \\ \mu_{Rule1} &= \mu_{medium}(LD = 0.90) \ \wedge \ \mu_{high}(LE = 0.80) \\ &= min(0.20, 0.75) \\ &= 0.20 \end{aligned}$$

Next, the inference engine applies the implication operator to the rule in order to obtain the fuzzy set to be accumulated in the output variable. In this case, inferencing is implemented by truncating the output membership function at the level corresponding to the computed degree of truth of the rule's antecedent. The effect of this process can be seen in the *consequent* part of *Rule 6* in which the membership function for linguistic term *high* was truncated at the level of 0.20. The same processes are applied to all of the rules.

Finally, all the truncated output membership functions are aggregated together to form a single fuzzy subset (labeled as *Final Output* in Figure 4.5) by taking the maximum across all the consequent sets. A further step (known as 'defuzzification') is then performed if (as is usual) the final fuzzy output is to be translated into a crisp output.

Using the 'centre of gravity defuzzification', the defuzzified value for the conclusion is found (approximately):

$$\begin{aligned}
\sum_i \mu(x_i) \cdot x_i = {} & (0.15 * 0.05) + (0.2 * 0.1) + (0.2 * 0.15) + (0.2 * 0.2) + (0.2 * 0.25) \\
& + (0.2 * 0.3) + (0.2 * 0.35) + (0.2 * 0.4) + (0.2 * 0.45) + (0.2 * 0.5) \\
& + (0.2 * 0.55) + (0.35 * 0.6) + (0.4 * 0.65) + (0.45 * 0.7) + (0.5 * 0.75) \\
& + (0.65 * 0.8) + (0.7 * 0.85) + (0.75 * 0.9) + (0.75 * 0.95) + (0.75 * 1.0) \\
= {} & 5.07
\end{aligned}$$

$$\begin{aligned}
\sum_i \mu(x_i) = {} & 0.15 + 0.2 + 0.2 + 0.2 + 0.2 + 0.2 + 0.2 + 0.2 + 0.2 + 0.2 \\
& + 0.2 + 0.35 + 0.4 + 0.45 + 0.5 + 0.65 + 0.7 + 0.75 + 0.75 + 0.75 \\
= {} & 7.45
\end{aligned}$$

$$\begin{aligned}
\frac{\sum_i \mu(x_i) \cdot x_i}{\sum_i \mu(x_i)} &= \frac{5.07}{7.45} \\
&= 0.680537
\end{aligned}$$

In the example of Figure 4.5, the output for the fuzzy system (that represents how difficult the exam is to be scheduled) is 0.68 for the given inputs (i.e an exam with LD and LE of 0.90 and 0.80, respectively).

All exams in the given problem instance are evaluated using the same fuzzy system, and the sequential constructive algorithm uses the crisp output of each exam for ordering all exams. The exam with the biggest crisp value is selected to be scheduled first, and the process continues until all the exams are scheduled without violating any of the hard constraints.

4.4.1.2 Initial Fuzzy Model : *Fixed Fuzzy LD+LE Model*

In order to test how the sequential constructive algorithm would work when multiple heuristic ordering were implemented, a fixed fuzzy model that took into account multiple heuristic ordering was developed. Here, the term 'fixed' refers to the 'best' identified fuzzy model during an intiial 'trial and error' exercise. As this fuzzy model was used to test the applicability of fuzzy techniques for measuring the difficulty of scheduling the exams, no further improvements were made to the fuzzy model. Alternative fuzzy models obtained with membership functions tuning are explained in Section 4.4.1.3.

Two out of the three ordering criteria described in Section 2.2.2, namely *largest degree* (*LD*) and *largest enrolment* (*LE*) were selected as input variables. The membership functions used in this experiment are shown in Figure 4.6. The choice of these membership functions was based on 'trial and error' to test how the algorithm would work when exams were ordered with the aid of fuzzy reasoning.

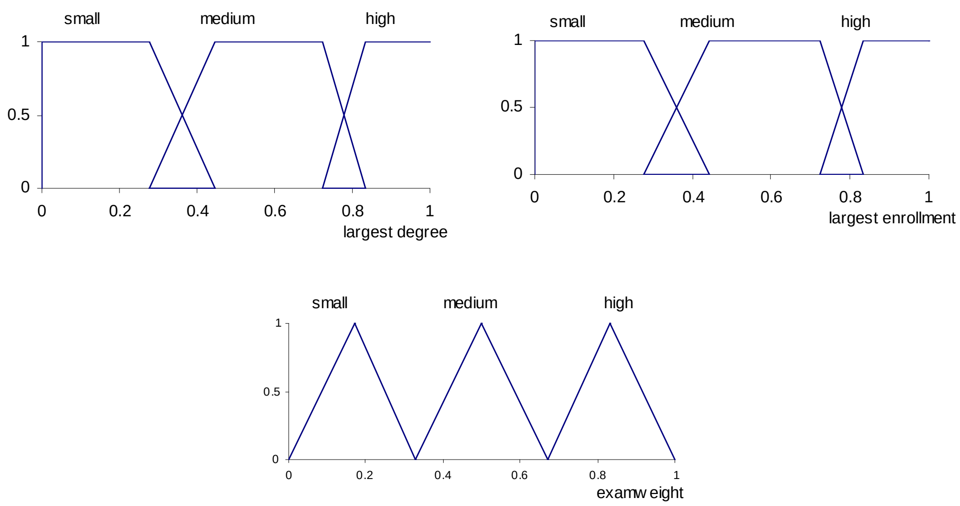


Figure 4.6: Membership functions for *Fixed Fuzzy LD+LE Model*

The fuzzy rules used in this experiment are shown in Table 4.2. For simplicity, the fuzzy rules are expressed as a linguistic matrix (see Lim *et al.* (1996)). In such a linguistic matrix, the left-most column and the first row denote the variables involved

Table 4.2: Fuzzy rule set for *Fixed Fuzzy LD+LE Model*

		LE		
		S	M	H
LD	S	VS	VS	M
	M	M	M	H
	H	S	M	VH

VS: very small
S: small
M: medium
H: high
VH: very high

in the antecedent part of the rules. The second column contains the linguistic terms applicable to the input variable shown in the first column; those in the second row correspond to the input variable shown in the first row. Each entry in the main body of the matrix denotes the linguistic values of the consequent part of a rule.

Note that, in addition to the three basic terms, the *hedge* 'very' was utilised to create two extra terms for the output variable. The 'very' hedge squares the membership grade $\mu(x)$ at each x of the fuzzy set for the term to which it is applied. Thus the membership function of the fuzzy set for 'very small' is obtained by squaring the membership function of the fuzzy set 'small'. For instance, the bottom-right entry in Table 4.2 is read as *"IF LD is high AND LE is high THEN examweight is very high"*. The same representation is also used to express the fuzzy rule sets for the tuned fuzzy model explained in the following sections. This fixed fuzzy model is presented here for the purpose of comparison with the tuned fuzzy model explained in the following section.

4.4.1.3 Extension of the Initial Fuzzy Model : Tuning Membership Functions

An extension to the *Fixed Fuzzy LD+LE Model*, a restricted form of exhaustive search was used to find the most appropriate shape for the fuzzy membership functions for each of the combination. There are very many alternatives that may be used when constructing a fuzzy model. Usually, membership functions can be subjectively determined in an ad-hoc style from experience or hunch. In order to reduce the search space for tuning the membership function, only one membership function shape is considered in this research. Although any appropriate fuzzy membership function representation is possible, triangular membership functions were used because they are easier to represent and also to work with. This selection was made on the basis that triangular membership functions were continuous, normal and convex (Ying, 2000). Triangular membership functions are among the most popular and widely used membership function nowadays. Furthermore, by using triangular membership functions, the membership function tuning (as described later) could be simplified. That is, in order to determine the fuzzy sets for the three linguistic term (*small*, *medium* and *high*), only one centre point (*cp*) was required. This reduced the computational time as compared to determining three different fuzzy sets for the three linguistic terms for each of the fuzzy variable.

In this implementation, the search was arbitrarily restricted based on the membership functions, as shown in Figure 4.7. Triangular shape membership functions were employed to represent *small*, *medium* and *high*. However, the fuzzy model was then altered by moving the point *cp* along the universe of discourse. This single point corresponded to the right edge for the term *small*, the centre point for the term *medium* and the left edge for the term *high*. Thus, there was one *cp* parameter for each fuzzy variable (two inputs and one output). The membership functions were refined by adjusting them until the best possible system performance was achieved. The three *cp* parameters were systematically altered while assessing the performance of the system.

A search was then carried out to find the best set of *cp* parameters. During this search, each point *cp* (for any of the fuzzy variables) can take a value between 0.0 and 1.0 inclusive. Increments of 0.1 were used (i.e. the values 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1.0) for data sets that have 300 and fewer exams, and 0.25 increments (i.e. the values 0.0, 0.25, 0.5, 0.75 and 1.0) for data sets that have more than 300 exams. The effect of varying the point *cp* from 0.0 to 1.0 is shown in Figure 4.8.

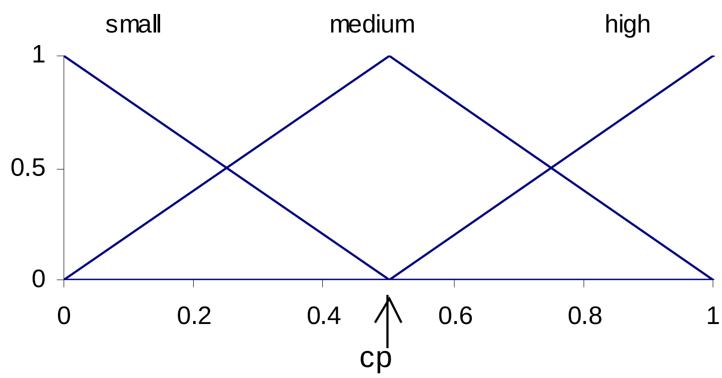


Figure 4.7: The membership function for tuned fuzzy model

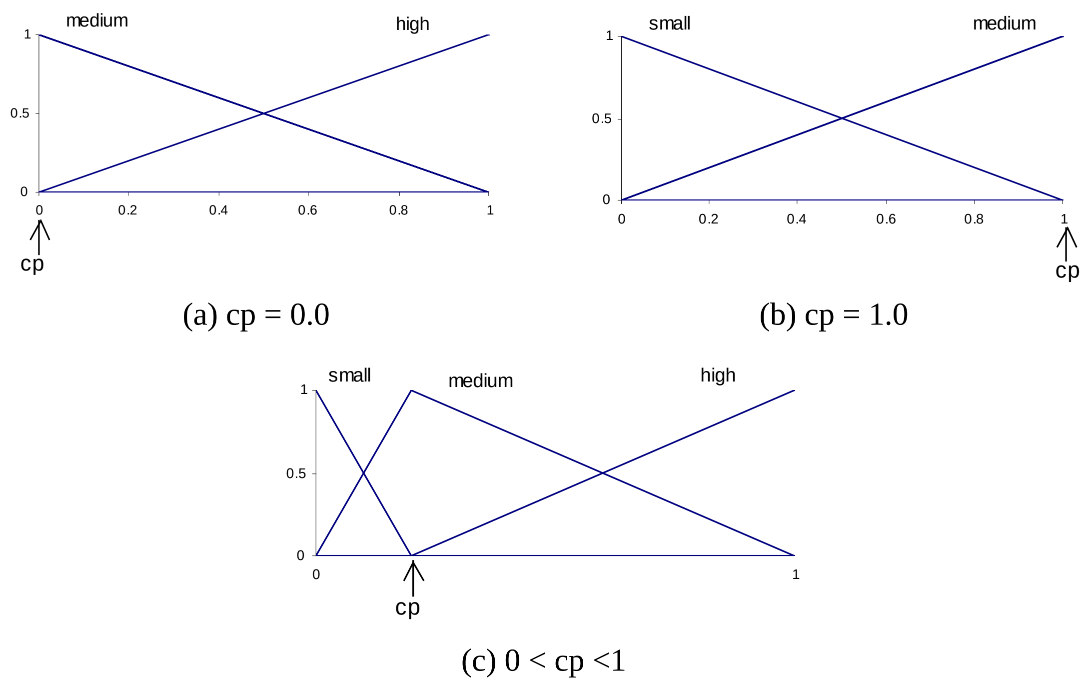


Figure 4.8: Range of possible membership functions

This tuning procedure is then applied to the three different combinations of multiple heuristic orderings, as follows:

Fuzzy LD+LE Model - the combination of *LD* and *LE* heuristic orderings were again used as the fuzzy input variables.

Fuzzy SD+LE Model - the combination of *SD* and *LE* heuristic orderings were used as the fuzzy input variables.

Fuzzy SD+LD Model - the combination of SD and LD heuristic orderings were used as the fuzzy input variables.

For each heuristic ordering combination, a fuzzy rule set connecting the input variables (any two of LD, LE or SD) to the output variable, *examweight* was constructed. All three fuzzy rule sets were motivated by the assumption that exams should be placed into a timetable in order of how difficult they are to schedule (most difficult first) and encapsulating the following heuristics:

1. If an exam has a large number of other exams in conflict, it is more difficult to schedule than one with fewer exams in conflict (LD).
2. If an exam has a large number of students enroled in it, it is more difficult to schedule than one with fewer students enroled (LE).
3. An exam with a small number of time slots available into which it can be placed is more difficult to schedule than one with more time slots available (SD).

These assumptions were used in order to get a symmetric, balanced set of fuzzy rules for each heuristic ordering combination, to ensure that all possible input values were covered. Note that the interpretation of the SD heuristic (smaller is more difficult) is linguistically opposite to that of LD and LE (larger is more difficult). Thus, care must be taken when considering SD as one of the heuristic orderings in a combination. The fuzzy rules sets for the *Fuzzy LD+LE Model*, *Fuzzy SD+LE Model* and *Fuzzy SD+LD Model* are shown in Tables 4.3 to 4.5, respectively.

4.4.2 Experiments and Results

4.4.2.1 Description of Experiments

A number of experiments were carried out in which progressively more sophisticated fuzzy mechanisms were created to order the exams. In each experiment this ordering is

Table 4.3: The fuzzy rule set for the *Fuzzy LD+LE Model*

		LE		
		S	M	H
LD	S	VS	S	M
	M	S	M	H
	H	M	H	VH

VS: very small
S: small
M: medium
H: high
VH: very high

Table 4.4: The fuzzy rule set for the *Fuzzy SD+LE Model*

		SD		
		S	M	H
LE	S	M	S	VS
	M	H	M	S
	H	VH	H	M

VS: very small
S: small
M: medium
H: high
VH: very high

Table 4.5: The fuzzy rule set for the *Fuzzy SD+LD Model*

		SD		
		S	M	H
LD	S	M	S	VS
	M	H	M	S
	H	VH	H	M

VS: very small
S: small
M: medium
H: high
VH: very high

simply inserted into the basic general algorithm presented in Figure 4.1.

Experiment 1 **: Single Heuristic Ordering** In order to provide a comparative test, the algorithm was initially run without implementing fuzzy ordering. That is, in this experiment, the exams in the problem instances were ordered based on a single heuristic ordering. All the exams were then selected to be scheduled based on this ordering.

Experiment 2 **: *Fixed Fuzzy LD+LE Model*** This experiment is designed to test the initial fuzzy model. Based on the results of this experiment, a better fuzzy model is defined.

Experiment 3 : Tuning the Fuzzy Model In this experiment, each of fuzzy models described in Section 4.4.1.3 is used to search for the 'best' fuzzy model for each heuristic ordering combinations.

4.4.2.2 Experimental Results

In this section the results obtained in each experiment are presented. In all experiments, the basic algorithm shown diagrammatically in Figure 4.1 was employed. The only difference was the heuristic ordering used. The experiments were carried out with twelve benchmark data sets made publicly available by Carter *et al.* (1996). Table 2.1 reproduces the problem characteristics. A proximity cost function described in Section 2.3.1 is used to measure the timetable quality.

The algorithm was developed using java based object oriented programming. The fuzzy inference engine developed by Sazonov *et al.* (2002) was implemented. The experiments were run on a PC with a 1.8 GHz Pentium 4 and 256MB of RAM. In the case of the *Single Heuristic Ordering* and the *Fixed Fuzzy LD+LE Model* each instance was run five times. In the other experiments (that involved tuning the fuzzy model), the aim was to search for the best fuzzy model to guide the constructive algorithm. In order to reduce the size of the search space, only the membership functions are tuned, whereas the fuzzy rule set is fixed. In this tuning process, for problem instances that have 300 and fewer exams, the algorithm was tested on 1331 (3 variables and 11 options - 11^3) membership function combinations. Problem instances that have more than 300 exams were tested on 125 (3 variables and 5 options - 5^3) membership function combinations. Because of this, each instance was only run twice. For all experiments, only the best results are selected and presented in Table 4.6.

For comparison, the best results obtained by Carter *et al.* (1996) when using various different heuristics to order the exams are shown in the second column of Table 4.6. The results obtained for our three varieties of *Single Heuristic Ordering* are presented

Table 4.6: Experimental results for single and fuzzy multiple heuristic orderings

Data Set	Carter *et al.* (1996)	Single Heuristic Ordering *LD*	*LE*	*SD*	*Fixed Fuzzy LD+LE Model*	*Fuzzy LD+LE Model*	*Fuzzy SD+LE Model*	*Fuzzy SD+LD Model*
CAR-F-92	6.2	5.56	5.03	5.50	5.65	4.62	**4.56**	4.62
CAR-S-91	7.1	6.38	5.90	5.91	6.31	5.60	**5.29**	5.77
EAR-F-83	36.4	40.58	45.88	49.10	48.14	38.41	**37.02**	39.27
HEC-S-92	10.8	14.98	14.94	14.27	16.93	12.53	**11.78**	12.55
KFU-S-93	14.0	18.63	16.46	18.60	18.29	16.53	15.81	**15.80**
LSE-F-91	10.5	15.08	14.52	13.46	16.84	12.35	**12.09**	12.95
RYE-F-92	7.3	12.95	11.12	11.60	12.98	11.75	**10.38**	12.71
STA-F-83	161.5	173.09	171.87	178.24	161.21	**160.42**	160.75	171.42
TRE-S-92	9.6	10.98	9.93	10.81	10.36	9.05	**8.67**	9.80
UTA-S-92	3.5	4.48	4.78	3.83	5.16	3.87	**3.57**	3.86
UTE-S-92	25.8	35.19	28.80	33.14	30.54	28.65	**28.07**	31.05
YOR-F-83	41.7	45.60	43.53	45.27	46.41	41.37	**39.80**	44.70

in the third to fifth columns. The results obtained for the *Fixed Fuzzy LD+LE Model* are shown in the sixth column. In general, these results are worse than for the best *Single Heuristic Ordering*, except for the *STA-F-83* data set, where the fixed fuzzy model obtained the best result. This observation suggested that there might be promise in the fuzzy approach and prompted us to undertake further investigations with tuned fuzzy models. The results for the *Fuzzy LD+LE Model*, *Fuzzy SD+LE Model* and *Fuzzy SD+LD Model* are shown in the seventh to ninth columns respectively.

The best results obtained in Table 4.6 are highlighted in bold font. The corresponding membership functions of the fuzzy model which obtained the best result for each data set are presented in Table 4.7. The graphical representation of the membership functions are shown in Figures 4.9 and 4.10. It can be seen that the membership functions differ in each case — i.e. there is no generic fuzzy model which suits all the data sets.

Table 4.8 shows a comparison of *cp* parameters combinations between the best fuzzy model and the second best fuzzy model for nine of the data sets. In the table, the

Table 4.7: Values for *cp* parameters

Data Set	Fuzzy LD+LE Model			Fuzzy SD+LE Model			Fuzzy SD+LD Model		
	LD	*LE*	examweight	*SD*	*LE*	examweight	*SD*	*LD*	examweight
CAR-F-92	0.00	0.50	0.50	0.50	0.25	0.25	0.50	1.00	1.00
CAR-S-91	0.50	0.00	0.25	0.25	0.00	0.50	0.75	0.75	0.75
EAR-F-83	0.40	1.00	0.30	0.50	1.00	0.50	0.80	0.40	0.20
HEC-S-92	0.40	0.20	0.40	0.40	1.00	1.00	0.20	0.40	0.00
KFU-S-93	0.75	0.00	0.00	0.50	1.00	0.50	0.50	1.00	0.50
LSE-F-91	0.75	0.50	0.25	0.25	1.00	0.25	0.25	0.75	0.50
RYE-F-92	0.75	0.25	0.00	1.00	0.00	0.00	0.75	1.00	0.50
STA-F-83	0.60	0.70	0.90	0.20	0.30	0.00	0.60	0.80	0.50
TRE-S-92	0.00	0.50	0.40	0.60	1.00	0.20	0.20	0.30	0.10
UTA-S-92	0.00	0.50	0.75	0.25	0.00	0.50	0.50	0.50	0.75
UTE-S-92	0.30	0.60	0.00	0.30	0.90	0.70	0.40	0.00	0.50
YOR-F-83	0.90	1.00	0.00	0.60	0.80	0.70	0.00	0.00	0.50

cp value of second best fuzzy model (Second Model) is highlighted in bold font if it is different to the *cp* values of the best fuzzy model (First Model). In terms of robustness of the best fuzzy model, it can be seen that for seven out of these nine data sets, the membership functions for the antecedents are the same; only the membership functions for the consequences are slightly different. For the other two data sets (*EAR-F-83* and *STA-F-83*), the membership functions for both antecedents and consequence are slightly different.

4.4.2.3 Discussion of Results

Amongst the three *Single Heuristic Ordering*, it would appear that *LE* is the 'best' in this context as it produced the best solution for eight out of the twelve data sets, compared to only one for *LD* (for EAR-F-83) and three for *SD* (for *HEC-S-92*, *LSE-F-91* and *UTA-S-92*). It also can be seen that, when compared to Carter *et al.*'s best results, our simplified version of their algorithm produced worse results in ten out of the twelve data sets, but a slightly better timetable was obtained for the *CAR-F-92* and *CAR-S-91*

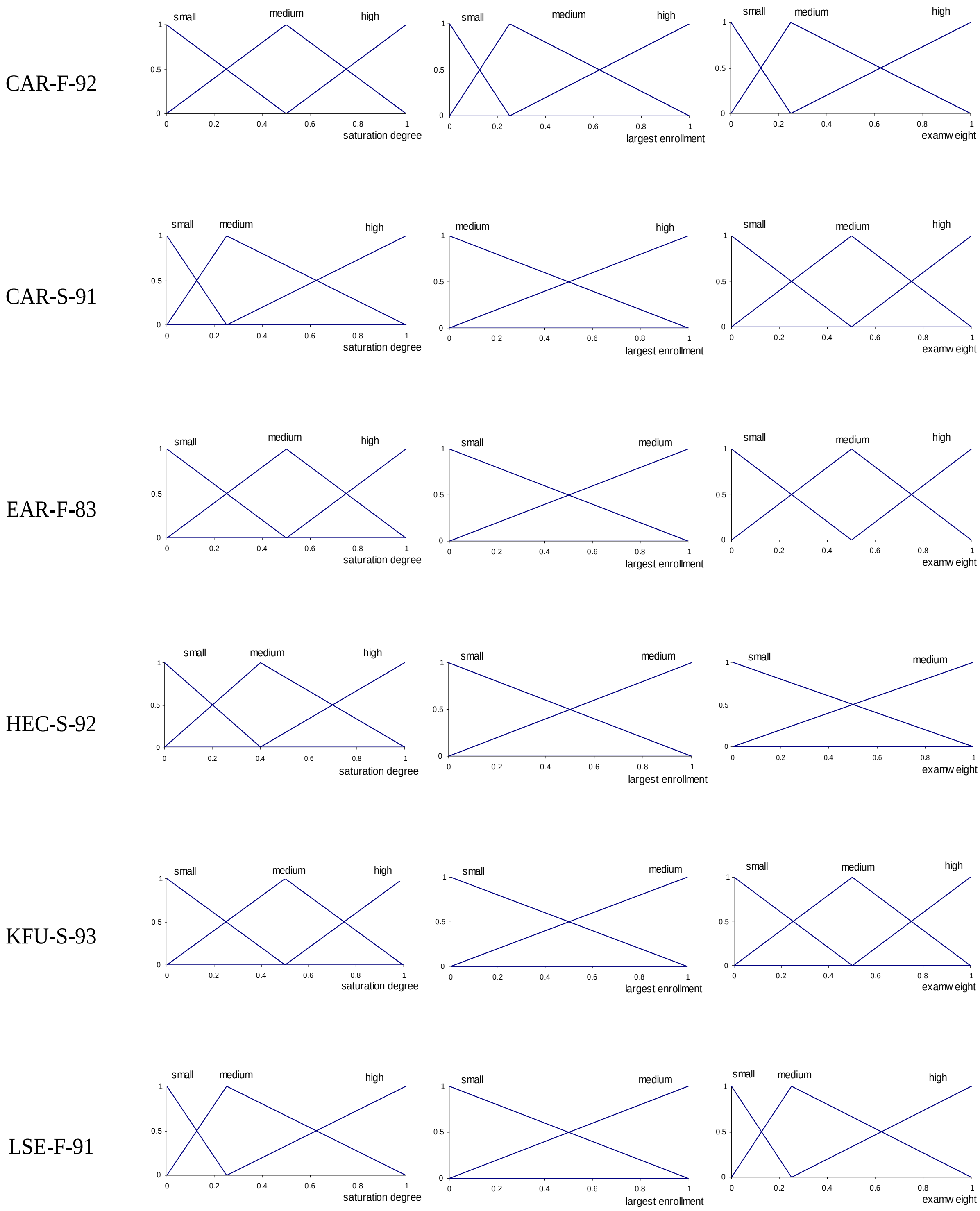


Figure 4.9: Best fuzzy model for data sets *CAR-F-92*, *CAR-S-91*, *EAR-F-83*, *HEC-S-92*, *KFU-S-93* and *LSE-F-91*

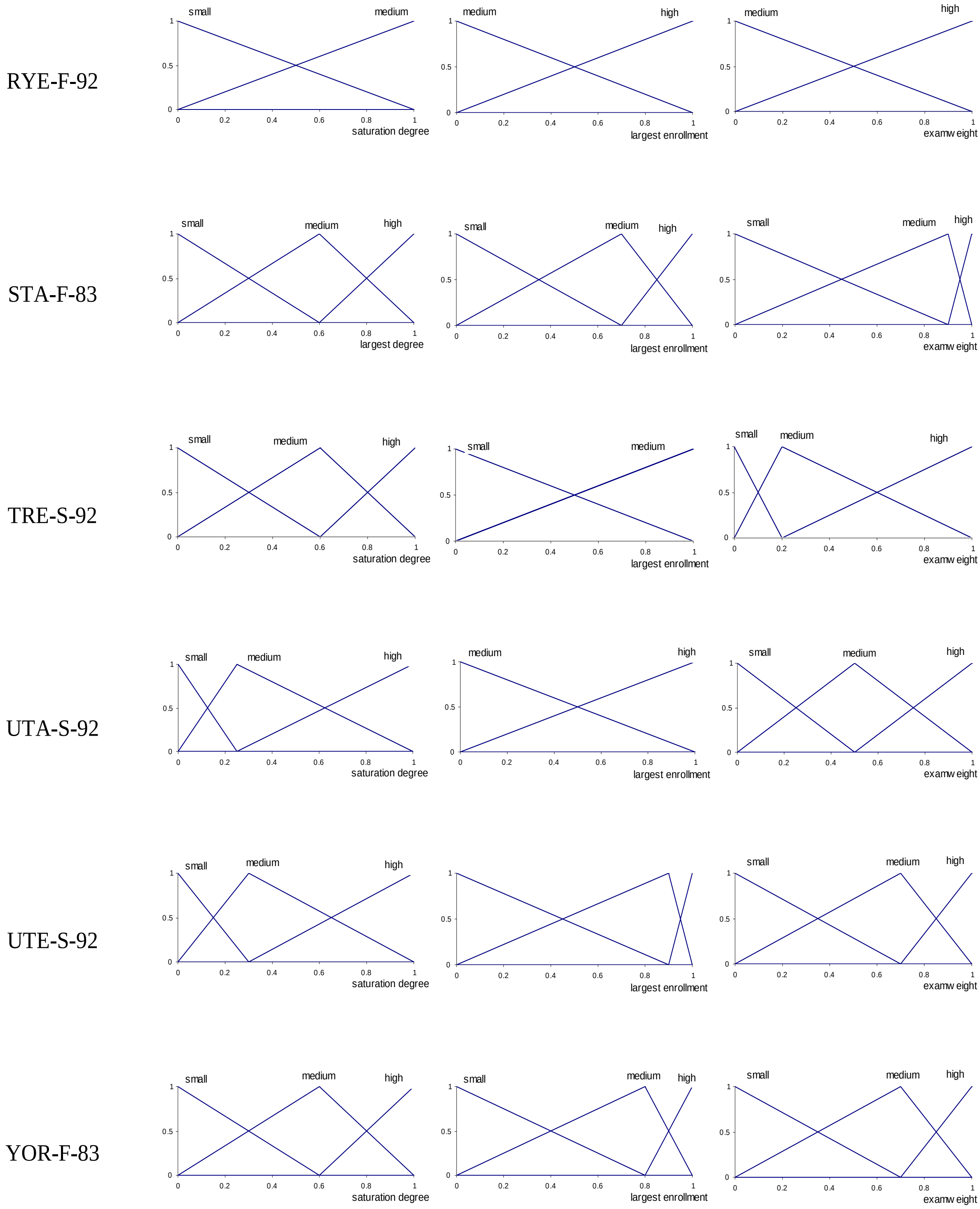


Figure 4.10: Best fuzzy model for data sets *RYE-F-92*, *STA-F-83*, *TRE-S-92*, *UTA-S-92*, *UTE-S-92* and *YOR-F-83*

Table 4.8: Comparison of *cp* parameters combinations for the best fuzzy model and the second best fuzzy model

Data Set		*Fuzzy LD+LE Model*		
		Heuristic1	*Heuristic2*	examweight
EAR-F-83	First Model	0.50	1.00	0.50
	Second Model	**0.40**	1.00	**0.70**
HEC-S-92	First Model	0.40	1.00	1.00
	Second Model	0.40	1.00	**0.60**
KFU-S-93	First Model	0.50	1.00	0.50
	Second Model	0.50	1.00	**0.75**
LSE-F-91	First Model	0.25	1.00	0.25
	Second Model	0.25	1.00	**0.50**
RYE-F-92	First Model	1.00	0.00	0.00
	Second Model	1.00	0.00	**0.25**
STA-F-83	First Model	0.60	0.70	0.90
	Second Model	**0.90**	**0.90**	**0.50**
TRE-S-92	First Model	0.60	1.00	0.20
	Second Model	0.60	1.00	**0.40**
UTE-S-92	First Model	0.30	0.90	0.70
	Second Model	0.30	0.90	**1.00**
YOR-F-83	First Model	0.60	0.80	0.70
	Second Model	0.60	0.80	**0.60**

Note:
For *STA-F-83*, *Heursitic1=LD* and *Heuristic2=LE*;
For other data sets, *Heursitic1=SD* and *Heuristic2=LE*;

cases. The *Fixed Fuzzy LD+LE Model* only achieves a better result than the best *Single Heuristic Ordering* in one out of the twelve data sets (*STA-F-83*). However, the rules and membership functions for this initial fixed fuzzy model were completely arbitrary, so it could be considered surprising that it achieved a best result even once.

It is evident that the *Fuzzy LD+LE Model* produced better results than the *Fixed Fuzzy LD+LE Model* in all cases. Although entirely expected, this observation was taken as confirmation that the fuzzy system was capturing meaningful information and that the tuning procedure, although not finding the truly optimal fuzzy model (in the sense of the globally best set of membership functions for the given set of rules and other fixed aspects of the fuzzy system), was operating successfully. In comparison with best *Single Heuristic Ordering*, the *Fuzzy LD+LE Model* obtained better results in all cases except for the *KFU-S-93*, *RYE-F-92* and *UTA-S-92* data sets.

The *Fuzzy SD+LE Model* went on to produce better results than the *Fuzzy LD+LE Model* for all cases except the *STA-F-83* data set. When compared to Carter *et al.*'s original results, the tuned fuzzy models operating on two heuristics simultaneously (taking the best tuned fuzzy model for each data set) obtained better results for five out of the twelve data sets. These were the *CAR-F-92*, *CAR-S-91*, *STA-F-83*, *TRE-S-92* and *YOR-F-83* data sets. Although these results have since been bettered by some authors (see the discussion of Table 4.9 below), these have been based on iterative improvement techniques rather than the constructive approach employed by Carter *et al.* (1996) and the proposed approach.

Initially, the choice to use a combination of the *LD* and *LE* heuristics was based on the fact that these heuristics are static in the sense that they only have to be calculated once at the beginning of the ordering process. In contrast, the *SD* heuristic must be recalculated after each exam is assigned to a slot. Thus, it was felt that tuning the fuzzy model based on the *LD+LE* combination would be quicker. The choice to use the *SD+LE* combination in the subsequent model was based on the observation that the *LE* heuristic ordering, when used alone, obtained the minimum penalty cost for eight out of the twelve data sets while the *SD* heuristic ordering obtained the minimum cost for three out of twelve. Thus it was felt that these offered the most promising combination of two heuristics.

The design of the fuzzy rule sets was based on three assumptions:

- if *LD* is *High* then examweight is *High*
- if *LE* is *High* then examweight is *High*
- if *SD* is *Small* then examweight is *High*

However, it must be emphasized that the rule sets specified in Tables 4.2 to 4.5 are only one possible instance (in the case of each experiment) out of a very large number of alternatives. Due to the very large number of degrees-of-freedom in any fuzzy model, it is very rare that the first fuzzy system constructed will perform at an acceptable level. Usually some form of optimisation or performance tuning of the system will need to be undertaken. The most significant influences on performance of a fuzzy system are likely to be the number and location of the membership functions and the number and form of the rules. In this implementation, the number and form of the rules are kept fixed in all cases. Although the fuzzy membership functions were, to a certain extent, tuned to obtain good performances, there was no attempt in the current work to tune the rule sets. It is highly likely that, given sufficient time to perform the tuning, a set of fuzzy rules leading to better performance of the fuzzy models could be obtained.

Table 4.6 demonstrates that, in all cases, tuning the fuzzy model produces better results, as might be expected. Comparison of the best fuzzy model and the second best fuzzy model as presented in Table 4.8 show the robustness of the results of the best parameters for each data sets as the membership functions are just slightly different. This confirms the hypothesis that simultaneous ranking of multiple heuristic orderings *can* produce better results. The fact that the best fuzzy results are all obtained using different fuzzy membership functions, as shown in Figures 4.9 and 4.10, means that no *generic* fuzzy model has been obtained at this stage. Such a generic model would be necessary if the approach is to be applied quickly and efficiently to novel data sets. The lack of such a generic fuzzy model may cast doubt regarding the usability and flexibility of this approach. This indicates that care must be taken when applying fuzzy techniques: it is certainly not the case that just because it is fuzzy it is necessarily better. Despite the fact that, across different data sets, a somewhat consistent pattern can be seen, especially for heuristic *Largest Enrolment* where in five data sets (*EAR-F-83*, *HEC-S-92*, *KFU-S-93*, *LSE-F-91* and *TRE-S-92*) the *cp* values are set to 1.0, in three data sets (*STA-F-83*, *UTE-S-92* and *YOR-F-83*) the *cp* values are set between 0.7 and 0.9 (toward 1.0), while in three data sets (*CAR-S-91*, *RYE-F-92* and *UTA-S-92*) the *cp* values are set to 0.0. This scenario suggests that there may indeed be a possibility of finding a generic model. Further work is clearly possible on this issue.

Table 4.9 shows the performance of this algorithm in comparison with selected recently published results on Carter *et al.*'s benchmarks. The best result amongst the compared techniques for each data set is highlighted in bold font. Collectively, these results have been selected to show the best known results for each data set. Although the fuzzy algorithm has not beaten the best known result for any data set, its performance is broadly competitive with the others in the sense that it is *not* the worst in six out of the twelve data sets. It is also worth pointing out that the fuzzy algorithm produces solutions for all the twelve data sets, and that in two of the cases where it produces the worst result, at least one of the other papers did not quote any result. However, it has to be kept in mind that the fuzzy method is a simple constructive initial solution, compared to the other methods which are iterative improvement approaches. Although these results are worse than more recent results, especially those of Caramia *et al.* (2001), interestingly the fuzzy constructive algorithm can beat Caramia *et al.*'s results for data sets *CAR-F-92*, *CAR-S-91* and *TRE-S-92*.

Finally, some remarks should be made concerning the time required for the algorithm. In doing so, it is vital that a distinction must be made between the time taken to perform the tuning of the fuzzy models and the time taken to construct a solution once each fuzzy model is fixed. Once the fuzzy model is fixed, the time taken to construct a solution is no longer (in a practical sense) than the time taken when using a single heuristic ordering — that is, the additional time taken for the fuzzy system to perform its ordering is negligible. Indeed, there is some evidence (as discussed further in the following Section) that, once the fuzzy model is fixed, solutions are constructed more quickly using the fuzzy ordering. It seems that this may be due to the lack of required backtracking when the fuzzy ordering is used. However, the time taken in tuning each fuzzy model is very significant. Of course, if a generic fuzzy model could be found — that is a single fuzzy model that produces good quality initial solutions for all data sets (including the twelve benchmark data sets used here and novel data sets) — then the approach could be widely adopted, with significant impact.

4.5 Consistency of the Different Heuristic Ordering

Due to the randomness in the '*rescheduling procedure*', a different timetable may be constructed each time the algorithm is run. Therefore, in order to determine and compare the performance of the various fuzzy heuristic orderings, repeated runs were performed to generate 30 solutions with each fuzzy multiple heuristic ordering model and each of the single heuristic orderings (*LD*, *LE* and *SD*), for each of the twelve data sets. For tuned fuzzy multiple heuristic orderings, the 'best' fuzzy models that had been identified during the membership functions tuning phase were utilised — i.e shown in Table 4.7.

4.5.1 Experimental Results

Table 4.10 shows a comparison of the cost penalties obtained based on 30 runs of each data set. The best results among the different heuristic orderings used are highlighted in bold font. It is evident that, overall, the fuzzy multiple heuristic ordering have outperformed any of the single heuristic orderings in that, for each data set, a fuzzy ordering obtained the best constructed timetable quality. Specifically, the *Fuzzy SD+LE Model* obtained ten best results and the *Fuzzy LD+LE Model* and *Fuzzy SD+LD Model* each obtained one best result. Amongst the single heuristic orderings, it appears that *LE* is the best because it obtained eight best results, followed by *SD* with three best results (*HEC-S-92*, *LSE-F-91* and *UTA-S-92*) and lastly *LD* with only one best result (*EAR-F-83*).

Table 4.9: Results comparison

Data Set	Fuzzy Best Results	Abdullah *et al.* (2006a)	Abdullah and Burke (2006)	Burke and Newall (2003)	Burke *et al.* (2004a)	Burke *et al.* (2006a)	Caramia *et al.* (2001)	Casey and Thompson (2003)	Di Gaspero and Schaerf (2001)	Kendall and Mohd Hussin (2005b)	Merlot *et al.* (2003)	White *et al.* (2004)	Yang and Petrovic (2005)
CAR-F-92	4.56	4.4	4.1	4.10	4.2	5.36	6.0	4.4	5.2	4.67	4.3	4.63	**3.93**
CAR-S-91	5.29	5.2	4.8	4.65	4.8	4.53	6.6	5.4	6.2	5.67	5.1	5.73	**4.50**
EAR-F-83	37.02	34.9	36.0	37.05	35.4	37.92	**29.3**	34.8	45.7	40.18	35.1	45.8	33.70
HEC-S-92	11.78	10.3	10.8	11.54	10.8	12.25	**9.2**	10.8	12.4	11.86	10.6	12.9	10.83
KFU-S-93	15.81	**13.5**	15.2	13.90	13.7	15.20	13.8	14.1	18.0	15.84	**13.5**	17.1	13.82
LSE-F-91	12.09	10.2	11.9	10.82	10.4	11.33	**9.6**	14.7	15.5	-	11.0	14.7	10.35
RYE-F-92	10.38	8.7	-	-	8.9	-	**6.8**	-	-	-	8.4	11.6	8.53
STA-F-83	160.42	159.2	159.0	168.73	159.1	158.19	158.2	**134.9**	160.8	157.38	157.3	158.0	151.50
TRE-S-92	8.67	8.4	8.5	8.35	8.3	8.92	9.4	8.7	10.0	8.39	8.4	8.94	**7.92**
UTA-S-92	3.57	3.6	3.6	3.20	3.4	3.88	3.5	-	4.2	-	3.5	4.44	**3.14**
UTE-S-92	28.07	26.0	26.0	25.83	25.7	28.01	**24.4**	25.4	29.0	27.60	25.1	29.0	25.39
YOR-F-83	40.66	**36.2**	**36.2**	37.28	36.7	41.37	**36.2**	37.5	41.0	-	37.4	42.3	36.35

Table 4.10: The penalty costs obtained by the different heuristic orderings on each of the twelve benchmark data sets. In each case the best result, the worst result, the average result and the standard deviation obtained over 30 repeated runs are given.

Data Set		Single Heuristic Ordering *LD*	*LE*	*SD*	*Fuzzy LD+LE Model*	*Fuzzy SD+LE Model*	*Fuzzy SD+LD Model*
CAR-F-92	*Best*	5.51	4.86	5.50	4.62	**4.54**	4.62
	Average	6.10	5.42	5.74	4.63	4.54	4.62
	Worst	6.81	6.40	7.25	4.64	4.54	4.62
	Std. Dev.	0.41	0.38	0.43	0.01	0.00	0.00
CAR-S-91	*Best*	6.13	5.89	5.91	5.57	**5.29**	5.77
	Average	6.66	6.36	5.91	5.67	5.29	5.77
	Worst	7.40	6.89	5.91	5.88	5.29	5.77
	Std. Dev.	0.31	0.26	0.00	0.08	0.00	0.00
EAR-F-83	*Best*	40.58	44.86	48.99	42.61	**37.02**	40.85
	Average	42.05	51.06	51.49	45.16	37.02	42.16
	Worst	45.09	59.14	54.79	49.90	37.02	44.46
	Std. Dev.	1.03	2.99	1.67	1.52	0.00	1.27
HEC-S-92	*Best*	14.73	14.41	14.23	12.43	**11.78**	12.55
	Average	16.25	16.98	16.36	14.25	11.78	12.55
	Worst	18.70	21.40	20.80	18.18	11.78	12.55
	Std. Dev.	1.31	1.76	1.86	1.74	0.00	0.00
KFU-S-93	*Best*	18.38	16.46	18.62	16.45	15.81	**15.80**
	Average	19.53	16.47	18.62	17.84	15.81	15.80
	Worst	21.81	16.50	18.62	21.75	15.81	15.80
	Std. Dev.	0.94	0.01	0.00	1.64	0.00	0.00
LSE-F-91	*Best*	14.79	14.41	13.46	12.35	**12.09**	12.95
	Average	17.12	16.45	13.46	12.35	12.09	12.95
	Worst	19.70	18.79	13.46	12.35	12.09	12.95
	Std. Dev.	1.37	1.20	0.00	0.00	0.00	0.00
RYE-F-92	*Best*	13.02	11.22	11.60	11.75	**10.38**	12.71
	Average	14.54	12.86	11.60	12.47	10.38	13.92
	Worst	17.38	14.60	11.60	13.70	10.38	15.42
	Std. Dev.	1.10	0.84	0.00	0.52	0.00	0.69
STA-F-83	*Best*	173.09	171.80	178.24	**160.42**	160.75	171.42
	Average	173.09	172.22	178.24	160.42	160.75	171.42
	Worst	173.09	172.57	178.24	160.42	160.75	171.42
	Std. Dev.	0.00	0.23	0.00	0.00	0.00	0.00
TRE-S-92	*Best*	10.65	9.92	10.81	9.05	**8.67**	9.80
	Average	11.42	10.73	10.81	9.05	8.67	9.80
	Worst	12.32	12.02	10.81	9.05	8.67	9.80
	Std. Dev.	0.43	0.49	0.00	0.00	0.00	0.00
UTA-S-92	*Best*	4.26	4.63	3.83	3.86	**3.57**	3.86
	Average	5.14	5.31	3.83	4.03	3.57	3.86
	Worst	6.28	6.32	3.83	4.30	3.57	3.86
	Std. Dev.	0.49	0.33	0.00	0.13	0.00	0.00
UTE-S-92	*Best*	35.19	28.79	33.26	28.65	**28.07**	31.05
	Average	35.51	28.93	33.61	28.68	28.07	31.05
	Worst	36.10	29.63	34.43	28.74	28.07	31.05
	Std. Dev.	0.26	0.20	0.28	0.03	0.00	0.00
YOR-F-83	*Best*	45.32	43.33	45.26	41.02	**39.80**	44.70
	Average	46.27	45.75	46.57	43.05	39.80	44.70
	Worst	47.91	49.12	48.53	47.95	39.80	44.70
	Std. Dev.	0.79	1.81	1.01	1.40	0.00	0.00

Table 4.11 shows the number of skipped exams obtained before the '*rescheduling procedure*' was called. The total number of exams that need to be scheduled for each data instance are shown in the second column. As described, the number of skipped exams is the number of exams that could not be scheduled after the completion of the initial phase of the constructions process (i.e. after *Process 2* to *Process 5* had been completed). It is simply the number of exams added to the 'skipped list' due to the fact that no valid time slot was available. It can be seen that *SD* most often (seven out of twelve data sets) produced the solutions without any skipped exams. This behaviour (most data sets resulting in no skipped exams) is also seen in the fuzzy multiple heuristic orderings that used *SD* as one of its heuristic orderings. However, this was not true for two data sets (*RYE-F-92* and *STA-F-83*) when the *Fuzzy SD+LD Model* was implemented — i.e. for these two data sets the *SD* heuristic alone resulted in no skipped exams, but when combined with the *LD* heuristic in the fuzzy approach some exams were skipped. The number of skipped exams determines whether it is necessary to invoke the '*rescheduling procedure*' or not. Obviously, it is not necessary to invoke the '*rescheduling procedure*' if there are no skipped exams.

Table 4.12 shows a comparison of the number of iteration of the '*rescheduling procedure*' required. This table shows the number of iterations of the loop in the '*rescheduling procedure*' that were required by each heuristic ordering in order to produce the solutions. As mentioned earlier, the number of skipped exams has an effect on the number of iterations of the '*rescheduling procedure*' are required. If there are no skipped exam, then no '*rescheduling procedure*' is required. On the other hand, if there are some skipped exam, then it is necessary to invoke the '*rescheduling procedure*', and there will always be at least that number of iterations of the '*rescheduling procedure*' required. For example, when *LD* ordering was applied to the *YOR-F-83* data set, it caused 5 skipped exams (see second column of Table 4.11). However, on average, 27 iterations of the '*rescheduling procedure*' were required (see second column of Table 4.12) in order to

Table 4.11: The number of skipped exams obtained due to the fact that there was no valid time slot available in the first attempt to assign the exam into the time slots — i.e. the number of exams in the skipped list after *Process 5*

Data Set	Total number of exams (N)	Single Heuristic Ordering *LD*	*LE*	*SD*	*Fuzzy LD+LE Model*	*Fuzzy SD+LE Model*	*Fuzzy SD+LD Model*
CAR-F-92	543	12	11	1	1	**0**	**0**
CAR-S-91	682	10	15	**0**	3	**0**	**0**
EAR-F-83	190	3	8	1	7	**0**	1
HEC-S-92	81	2	6	2	5	**1**	**0**
KFU-S-93	461	4	4	**0**	8	**0**	**0**
LSE-F-91	381	3	5	**0**	**0**	**0**	**0**
RYE-F-92	486	2	5	**0**	1	**0**	2
STA-F-83	139	24	2	**0**	7	**0**	24
TRE-S-92	261	6	7	**0**	1	**0**	**0**
UTA-S-92	622	7	13	**0**	2	**0**	**0**
UTE-S-92	184	2	3	**1**	**1**	**1**	**1**
YOR-F-83	181	5	10	3	13	**0**	**0**

produce the solutions.

Finally, Table 4.13 shows a comparison of the computational time required to construct the solutions for each heuristic ordering methods for each data set. As might be expected, when dynamic heuristic ordering was used, much longer times were required in order to produce the solutions, as each time around the loop the heuristic needed to be recalculated and the exams reordered. This happened either when single or multiple heuristic ordering was implemented.

4.5.2 Performance Analysis and Discussions

When constructing solutions for timetabling problems, one of the most important aspects that will affect the solution quality is the sequence in which the events should be selected to be scheduled (Boizumault *et al.*, 1996). Many ordering strategies have been implemented by other researchers. One of the strategies that is widely used is to base various heuristics on graph theory (Burke and Newall, 2004). However, to the best of our knowledge, although there are many such criteria derived from graph theory that

Table 4.12: The number of iterations of the '*rescheduling procedure*' required for each data set

Data Set		Single Heuristic Ordering *LD*	*LE*	*SD*	*Fuzzy LD+LE Model*	*Fuzzy SD+LE Model*	*Fuzzy SD+LD Model*
CAR-F-92	*Smallest*	58	31	5	1	0	0
	Average	204	81	58	1	0	0
	Worst	459	223	261	1	0	0
CAR-S-91	*Smallest*	39	34	0	4	0	0
	Average	99	70	0	10	0	0
	Worst	287	152	0	33	0	0
EAR-F-83	*Smallest*	4	17	7	11	0	2
	Average	7	95	49	24	0	12
	Worst	12	265	167	57	0	53
HEC-S-92	*Smallest*	8	19	9	6	1	0
	Average	29	41	39	22	1	0
	Worst	101	80	121	115	1	0
KFU-S-93	*Smallest*	6	4	0	10	0	0
	Average	13	4	0	29	0	0
	Worst	80	4	0	117	0	0
LSE-F-91	*Smallest*	13	24	0	0	0	0
	Average	59	71	0	0	0	0
	Worst	182	181	0	0	0	0
RYE-F-92	*Smallest*	9	9		6	0	6
	Average	88	28	0	22	0	59
	Worst	365	86	0	73	0	217
STA-F-83	*Smallest*	24	2	0	7	0	24
	Average	24	2	0	7	0	24
	Worst	24	2	0	7	0	24
TRE-S-92	*Smallest*	12	13	0	1	0	0
	Average	38	31	0	1	0	0
	Worst	121	67	0	1	0	0
UTA-S-92	*Smallest*	37	65	0	4	0	0
	Average	186	239	0	34	0	0
	Worst	413	543	0	82	0	0
UTE-S-92	*Smallest*	3	3	3	1	1	1
	Average	9	3	9	1	1	1
	Worst	66	11	32	1	1	1
YOR-F-83	*Smallest*	8	18	11	14	0	0
	Average	27	60	50	33	0	0
	Worst	65	181	142	107	0	0

Table 4.13: A comparison of the computational time (in seconds) required to construct the solutions for each heuristic ordering methods for each data set

Data Set		Single Heuristic Ordering			*Fuzzy*	*Fuzzy*	*Fuzzy*
		LD	*LE*	*SD*	*LD+LE Model*	*SD+LE Model*	*SD+LD Model*
CAR-F-92	*Shortest*	45.09	20.50	396.30	2.13	442.98	725.08
	Average	185.27	70.50	446.86	2.18	446.77	733.13
	Worst	440.08	216.67	666.81	2.67	458.31	763.75
CAR-S-91	*Shortest*	47.16	36.08	922.58	6.06	1006.70	1620.36
	Average	135.72	87.14	965.61	14.16	1023.50	1653.55
	Worst	403.24	197.70	1161.44	49.66	1055.53	1767.08
EAR-F-83	*Shortest*	0.41	1.13	12.61	0.83	19.34	33.34
	Average	0.63	8.26	16.62	1.88	19.38	34.82
	Worst	1.13	23.74	27.70	4.88	19.47	40.77
HEC-S-92	*Shortest*	0.11	0.22	0.95	0.11	2.28	2.27
	Average	0.37	0.52	1.29	0.32	2.36	2.36
	Worst	1.33	1.03	2.36	1.45	3.49	3.49
KFU-S-93	*Shortest*	1.17	0.89	64.28	2.05	112.44	179.50
	Average	2.74	0.91	64.54	7.77	113.92	182.91
	Worst	17.19	0.97	67.03	31.64	115.30	187.50
LSE-F-91	*Shortest*	1.77	3.24	37.92	0.52	70.27	114.55
	Average	8.25	9.77	38.00	0.53	70.57	118.33
	Worst	27.50	24.33	38.61	0.58	70.88	136.47
RYE-F-92	*Shortest*	2.94	2.84	149.94	2.11	215.24	333.50
	Average	22.68	7.54	150.44	6.01	221.01	359.11
	Worst	96.94	22.64	151.75	19.55	246.77	417.64
STA-F-83	*Shortest*	0.19	0.05	3.33	0.16	6.58	7.66
	Average	0.21	0.06	3.34	0.16	6.59	9.72
	Worst	0.27	0.14	3.39	0.22	6.64	11.05
TRE-S-92	*Shortest*	1.08	1.31	30.02	0.47	43.55	75.39
	Average	4.12	3.57	30.08	0.49	43.70	76.94
	Worst	12.77	8.34	30.23	0.55	44.86	85.88
UTA-S-92	*Shortest*	39.38	71.22	597.94	4.95	675.06	1101.94
	Average	229.01	296.84	639.26	40.40	695.52	1111.75
	Worst	501.64	697.88	809.13	93.91	818.70	1160.22
UTE-S-92	*Shortest*	0.06	0.08	4.23	0.14	12.67	18.41
	Average	0.11	0.09	4.32	0.17	13.02	19.51
	Worst	0.41	0.23	4.95	0.39	13.33	24.52
YOR-F-83	*Shortest*	0.42	0.88	15.99	0.78	22.47	37.22
	Average	1.34	3.06	18.03	1.74	22.51	38.78
	Worst	3.17	9.39	23.53	5.16	22.59	46.23

could be used as an heuristic ordering, only one criterion has been used on its own at any one time, except the works of Johnson (1990) where *LE* and *LD* heuristic are employed simultaneously. The other closest approach is recently published by (Burke *et al.*, 2007) where a different graph colouring heuristics are applied in sequence to construct solutions for the examination and course timetabling problem.

This Chapter presents a new heuristic ordering method in which two heuristic orderings are considered simultaneously using a fuzzy methodology to combine them. The experimental results, shown in Table 4.10, indicates that this new approach is promising. Concentrating on the quality of the solutions, it can be seen in Table 4.10 that all best results were obtained when fuzzy multiple heuristic orderings were implemented. This indicates that, in these timetabling problems, determining the difficulty of scheduling exams into time slots by taking into account multiple heuristic orderings simultaneously has resulted in initial solutions with better quality.

Nevertheless, there are a few cases in which fuzzy multiple heuristic orderings produced worst solutions compared with specific single heuristic orderings. For example, for the *RYE-F-92*, *UTE-S-92* and *YOR-F-83* data sets the *LE* heuristic ordering beat the *Fuzzy SD+LD Model* (see Table 4.10), and there are other similar such occurrences. These observations suggest that care must be taken when choosing which heuristic orderings are to be uses simultaneously for any given problem instance.

When looking at 'effectiveness' in terms of both solution quality and variation in solution quality, the results indicate that the *Fuzzy SD+LE Model* is the most effective heuristic ordering. For all twelve data sets, the 30 multiple runs of this heuristic ordering obtain the same solution quality. Although the *Fuzzy SD+LD Model* also managed to obtain the same solution quality for ten data sets, this fuzzy model only produced one best result out of the twelve data sets. Meanwhile, *SD* ordering and the *Fuzzy LD+LE Model* only managed to produce the same solution for a few of the data sets, while *LD* ordering only managed to obtain the same solution quality for the *STA-F-83* data set.

Since the only stochastic element in our algorithm is when selecting a time slot in the '*rescheduling procedure*', any heuristic ordering that produces an exam ordering which causes no skipped exams will always obtain the same solution in multiple runs. On the other hand, in situations where there are skipped exams (which depends on the problem instance and the heuristic ordering used) these can only be scheduled by reshuffling the already scheduled exams into another time slot, or 'bumping' the scheduled exams back to the unscheduled exam list. It seems obvious that the higher the number of iterations of the '*rescheduling procedure*' required, the higher the possibility of obtaining a solution with a *different* cost penalty.

This scenario may explain why the fuzzy membership function tuning process took a long time to finish, particularly for the problem instances that have more than 400 exams. It is assumed that during the fuzzy model tuning process, when a bad fuzzy model is evaluated, it will generate an ordering of the exams which for some reason cannot guide the constructive algorithm towards a good solution. In the case of a bad ordering of exams such as this, many of the exams cannot be scheduled without reshuffling exams that have already been scheduled earlier.

In Table 4.11, it can be observed that the *SD* heuristic ordering, the *Fuzzy SD+LE Model* and the *Fuzzy SD+LD Model* often produced solutions without invoking the '*rescheduling procedure*'. An interesting point here is that, although the *SD* heuristic ordering is capable of generating an ordering of exams that required no '*rescheduling procedure*', when compared against the other single heuristic orderings it only produced three best results out of twelve data sets (see Table 4.10). In contrast, the exam ordering generated using the *Fuzzy SD+LE Model* not only can guide the constructive algorithm without requiring the '*rescheduling procedure*', but it also can find solutions in which it outperformed other heuristics in ten out of twelve data sets.

In addition, although the *Fuzzy SD+LE Model* needed to reschedule one exam in the case of *HEC-S-92* and *UTE-S-92*, the solutions were produced by performing only one iteration of the '*rescheduling procedure*'. For the same *HEC-S-92* data set, the *SD* heuristic ordering also produced only one skipped exam but it required 39 iterations, *on average*, of the '*rescheduling procedure*' to produce the solution. When the *UTE-S-92* data set is considered, although having only one unscheduled exam, an average of nine iterations of the '*rescheduling procedure*' were required to produce the solution.

Taking these facts into consideration, let us now speculate as to what might be the factors that cause the *Fuzzy SD+LE Model* to perform uniformly well across the twelve data sets with different complexity. This fuzzy model combines two heuristic orderings, each of which may feature a strength that contributes to the effectiveness of this fuzzy model. Amongst the single heuristic orderings, *LE* performed well in eight out of twelve data sets (see Table 4.10), while *SD* often managed to find solutions during which no exam was skipped (see the fourth column of Table 4.11). By combining these two heuristic orderings simultaneously, it might be the case that the combination is benefitting from these two strengths to improve the overall performance of the search algorithm.

In can be seen that twenty-four exams are skipped when the single heuristic ordering *LD* and the *Fuzzy SD+LD Model* were applied to the *STA-F-83* data set (the second and seventh columns of Table 4.11). Interestingly, all these skipped exams are then scheduled by performing the '*rescheduling procedure*' with the same number of iterations, i.e. 24 (see the third and eighth columns of Table 4.12). That means that none of the already scheduled exams needed to be bumped back to the unscheduled list in order to create spaces for the skipped exams. Further investigation has shown that the 24 skipped exams are the same in each case. This was examined closely in order to understand what might have caused this curious effect.

In essence, the initial part of the construction process is a greedy algorithm that minimises the penalty of placing each exam, one by one, into the timetable (in the order given by the heuristic determination of difficulty). With the tendency to assign each unscheduled exam into the time slot with least penalty cost, the available time slots are usually occupied at an early stage of the scheduling process. In the case of the *STA-F-83* data set with the *Fuzzy SD+LD Model*, the first 13 exams were assigned to the 13 time slots available, although some of these exams could have been scheduled together in the same time slot — i.e. these 13 exams did *not* necessarily clash with each other. In effect, this situation had caused a 'bottleneck', after which no more valid time slots were available. In the next step of the construction process, the '*rescheduling procedure*' attempts to schedule each of the skipped exams by considering multiple simultaneous moves of already placed exams in order to obtain feasible solutions. For the *STA-F-83* data set, each of the skipped exams could be placed without need to 'unschedule' ('bump-back') any exams already placed.

Turning now to the computational time, it seems that the *Fuzzy LD+LE Model* can be considered the best amongst the multiple heuristic orderings experimented with since this heuristic always found good quality solutions in relatively low computational time. As seen in Table 4.10, in terms of solution quality, the *Fuzzy LD+LE Model* and *Fuzzy SD+LE Model* were approximately the same. Furthermore, when compared to the various single heuristic orderings, it is apparent that the *Fuzzy LD+LE Model* heuristic ordering obtained the minimum penalty cost for nine out of twelve data sets. However, in terms of computational time (see Table 4.13), the *Fuzzy SD+LE Model* and the *Fuzzy SD+LD Model* consistently perform worse than the other heuristic orderings.

Considering that the *Fuzzy LD+LE Model* combines two single heuristic ordering which are both categorised as *static* heuristics, it might be expected that this fuzzy model will take more computational time to produce the solution than the two heuristics on which it depends. However, the results presented in Table 4.13 indicate that with at least six out of the twelve cases the *Fuzzy LD+LE Model* is actually quicker than the single heuristics; specifically for the *CAR-F-92*, *CAR-S-91*, *LSE-F-91*, *RYE-F-92*, *TRE-S-92*, and *UTA-S-92* data sets. (It is arguable that is it also quicker for the 7^{th} data set, *HEC-S-92*, as the *Fuzzy LD+LE Model* has a lower average than the other heuristics.) It can be seen that this fuzzy heuristic ordering always obtains the solutions in shorter execution time for the data sets that consist of more than 300 exams, except for *KFU-S-93*. For the rest of the data sets, the time taken to construct the solution is very reasonable compared to the other single static heuristics.

If the *longest* time required to produced the solutions is now compared among the static heuristic orderings (i.e not including *SD*, *Fuzzy SD+LE Model* and *Fuzzy SD+LD Model*), it is evident that the *Fuzzy LD+LE Model* always produced the solutions in relatively short time (except for *KFU-S-93*). This is obvious for the data sets that contains more than 300 exams particularly for *CAR-F-92*, *CAR-S-91* and *UTA-S-92* (see Table 4.13). For example, in the case of the *CAR-F-92* data set (looking at the *Worst* row), the *Fuzzy LD+LE Model* only took approximately 3 seconds, whereas the other heuristics took at least 217 seconds. Although it takes a long time to search for the 'best' fuzzy model, it is important to notice how quick the 'best' fuzzy model finds the solution compared to the other heuristic orderings.

However, the capability to produce solutions quickly is not achievable when the dynamic heuristic is implemented. As seen in Table 4.13, the *Fuzzy SD+LD Model* required the longest time in all problem instances as compared to the other heuristics, followed by the *Fuzzy SD+LE Model*. It is believed that most of the time is used to recalculate the number of valid time slots available for the remainder of the unscheduled exams, and not to calculate the fuzzy exam weight. This assumption is based on the observation mentioned earlier, that the *Fuzzy LD+LE Model* always obtained the solutions in quick time, meaning that the time taken to calculate exam fuzzy weight must be relatively very small. Moreover, in ten out of the twelve problem instances, the *Fuzzy SD+LE Model* found the solutions without invoking the '*rescheduling procedure*' (and the other two data sets with only one iteration of the '*rescheduling procedure*'), which means no time was spent reshuffling the scheduled exams.

4.6 Chapter Summary

As far as the author is aware, no other published work has described the exploration of fuzzy methodologies for simultaneously ordering exams in the construction of examination timetables. In this study, fuzzy methodology to use multiple heuristic ordering simultaneously has been investigated.

The performance of three fuzzy multiple heuristic ordering and three single heuristic orderings were measured on the basis of the standard examination timetabling problem instances. It was found that better solutions were generated when two heuristic orderings were used simultaneously (provided that the 'best' tuned fuzzy model is applied). The results have been analyses in terms of criteria deemed important to the construction process. The potential of the fuzzy multiple heuristic ordering approach has been demonstrated as an important construction ordering technique using the simplest sequential constructive algorithm. The objective was to understand how all relevant criteria can be used simultaneously to enhance the provision of the initial feasible solution, as opposed to obtaining solutions simply to beat previously published results. It is the author believed that this research marks the beginning of a process which has the capability to incorporate all important user and technical data at all stages of the construction and improvement phases and hence will have the capability of producing much enhanced solutions. The main focus of the work presented here is to investigate an alternative fuzzy-based approach to assess the difficulty of scheduling exams to time slots. A multiple heuristic ordering has been introduced in which the conflicts between heuristic orderings is resolved by means of fuzzy reasoning, and the results obtained have been extensively analysed in order to further the understanding of how heuristics can be combined in various circumstances. This has been achieved by reference to a number of essential criteria.

It can be seen that these experiments have confirmed that the fuzzy multiple heuristic ordering approach can reliably perform better than the single heuristic orderings considered on repeated runs. However, the success of this fuzzy approach is highly dependent on the individual 'best' fuzzy membership functions tuned for each data set. Finding a generic fuzzy multiple heuristic ordering model that is applicable to all potential problem instances which can provide consistently good results is an interesting and open research problem. Based on the work presented, it is believed that further investigation is warranted into fuzzy techniques in all areas of the provision and evaluation of solutions to the examination timetabling problem.

The work presented in the first part of the Chapter is published in the Selected Volume of the 5th International Conference for the Practice and Theory of Automated Timetabling (PATAT'2004) (Asmuni *et al.*, 2005). The second part of the Chapter has be accepted to be published in the *Journal of Computers & Operations Research* (Asmuni *et al.*, 2008).

Chapter 5

Comparison of Fuzzy and Non-Fuzzy Multiple Heuristic Ordering

5.1 Introduction

This Chapter further investigates the efficiency and effectiveness of fuzzy multiple heuristic orderings. Due to the large amount of time required in tuning the fuzzy models used, the algorithm identified and used in the previous Chapter is modified in order to shorten the computational time. This allows for more detailed analysis of various combinations of heuristics. In addition, the concept of utilising more than one heuristic ordering simultaneously, proposed in the previous Chapter, is extended by considering up to three heuristic ordering simultaneously. The modified sequential constructive algorithm was implemented with a single heuristic ordering and multiple heuristic ordering, both by fuzzy reasoning and linear combinations. As in the previous Chapter, the performance of various heuristic orderings was compared on a set of standard benchmark problems.

5.2 Extension to Three Heuristic Ordering

In this Chapter, the multiple heuristic ordering technique described in the previous Chapter is extended incorporating three heuristic orderings which are considered simultaneously. This effectively means that, the fuzzy system is extended to be able to deal with 3 input variables and 1 output variable. It was decided that the same tuning method (see Section 4.4.1.3) would be used. Therefore, with four fuzzy variables, there are $11^4 = 14641$ *cp* combinations that need to be tested for data sets with 300 exams or less. Taking the *EAR-F-83* data set as an example, the shortest time to produced the solution shown in Table 4.13 is equal to 19.34 seconds when the *Fuzzy SD+LE Model* was employed. If this time is used to estimate the total time to test all the 14641 combinations, the total time to tune the fuzzy model would be 3 days and 6 hours approximately. It is pointed out that this is the total tuning time if the shortest time is considered. However, as discussed in the previous Chapter, the number of '*rescheduling procedure*' required have significant influence over the time to produce the solution.

Furthermore, by definition a 'bad' fuzzy model will produce a 'bad' ordering of exams, which will tend to cause the constructive algorithm to take a longer time to reach solutions. Therefore, it was clear that the original algorithm needed to be improved with the goal of reducing the computational time. This was necessary to ensure sufficient experimentation took place in establishing the effectiveness of the proposed approach.

5.2.1 Algorithmic Changes to Reduce Computational Time

With the aim of reducing the computational time, the following changes to the algorithm were implemented:

ALG1.1 The first changes is, when no clash free time slot is available in which to insert an exam, the '*rescheduling procedure*' is performed straight away. Whereas in previous algorithm (see Figure 4.1), the exam is skipped to be processed after all the 'conflict free' exams are scheduled. In doing so it is assumed that it is better to allocate a time slot to the 'stuck' exam in the earlier stage of the assignment process, rather than do it in the later stage.

ALG1.2 The second changes is, the exam difficulty reordering is performed after five exams are inserted into the timetable, instead of after only one. It is expected that, by reducing the number of times the exams ordering is recalculated, less time is require to produce the solution. This is motivated by the fact that, the static heuristic ordering will always produced the solution in the shortest time (as shown by the *Fuzzy LD+LE Model*, see Table 4.13). However, it is questionable whether this change will maintain the solution quality, especially when the dynamic heuristic ordering is implemented.

ALG2.0 The third change is to implement both *ALG1.1* and *ALG1.2* in parallel. The motivation for this is to look into the impacts of applying both changes at the same time.

In order to illustrate the impact of making these changes, a number of simple experiments were set up. The following two fuzzy models are used: the *Fuzzy LD+LE Model* and *Fuzzy SD+LE Model*. These two models was chosen to represents the *static-static* combination and *static-dynamic* combination. In order to understand the effect of the impact, the investigation focused on the impact of the number of '*rescheduling procedure*' required, the computational time and the proximity cost. These are considered key attributes of the construction process. Table 5.1 shows the comparison of the 4 algorithms. Only three data sets results were presented in this table. These three data sets (*CAR-S-91*, *KFU-S-93* and *UTA-S-92*) are selected because the effects of the changes made to the original algorithm described in Section 4.2 (from this point onwards, it is termed '*ALG1.0*') are more clearly observed in these three data sets. For the full results, please refer to Appendix A.

Table 5.1: A comparison of the results obtained by the different algorithms on the *CAR-S-91*, *KFU-S-93* and *UTA-S-92* data sets

Data Set		*Fuzzy LD+LE Model*				*Fuzzy SD+LE Model*			
		ALG1.0	*ALG1.1*	*ALG1.2*	*ALG2.0*	*ALG1.0*	*ALG1.1*	*ALG1.2*	*ALG2.0*
CAR-S-91									
Proximity Cost	Best	5.58	5.56	5.57	5.60	5.29	5.29	6.20	5.59
	Worst	5.81	5.63	5.82	5.65	5.29	5.29	7.06	5.59
	Average	5.65	5.59	5.67	5.62	5.29	5.29	6.54	5.59
Comp. Time (s)	Shortest	6.34	3.05	5.05	3.09	902.27	885.14	25.94	183.50
	Worst	30.44	3.63	21.95	3.64	908.56	900.41	199.91	184.38
	Average	13.48	3.26	11.65	3.33	905.15	889.51	102.30	184.08
Backtracking	Min	5	3	4	3	0	0	28	0
	Max	27	6	18	6	0	0	152	0
	Average	12.2	3.8	9.6	4.4	0	0	84.4	0
KFU-S-93									
Proximity Cost	Best	16.54	15.99	16.59	15.84	15.81	15.81	22.20	17.48
	Worst	19.17	16.72	18.72	16.24	15.81	15.81	25.48	17.48
	Average	17.60	16.35	17.29	16.13	15.81	15.81	23.79	17.48
Comp. Time (s)	Shortest	2.27	0.77	1.78	0.81	106.31	104.88	11.55	22.30
	Worst	5.52	0.92	5.06	1.17	109.34	107.56	28.56	22.48
	Average	3.64	0.83	3.42	0.90	107.86	106.49	18.16	22.36
Backtracking	Min	12	4	10	6	0	0	56	0
	Max	25	9	24	8	0	0	125	0
	Average	18	7	16.8	7.2	0	0	79	0
UTA-S-92									
Proximity Cost	Best	3.87	3.85	3.88	3.86	3.57	3.57	4.19	3.82
	Worst	4.64	3.86	4.13	3.90	3.57	3.57	4.82	3.88
	Average	4.23	3.85	3.98	3.88	3.57	3.57	4.52	3.86
Comp. Time (s)	Shortest	11.55	2.25	15.95	2.25	600.81	590.97	47.30	124.19
	Worst	56.08	2.31	40.34	2.55	603.81	593.52	279.67	125.11
	Average	31.11	2.28	28.36	2.36	602.47	591.97	119.23	124.61
Backtracking	Min	10	2	18	2	0	0	51	1
	Max	49	2	42	4	0	0	274	1
	Average	29.8	2	29	2.8	0	0	121.2	1

When the *Fuzzy LD+LE Model* is used, no change is expected by employing *ALG1.1* and *ALG1.2*, because no exams reordering is needed for the static heuristic ordering type. Overall, although little improvement can be seen in terms of proximity cost, the implementation of *ALG1.0* and *ALG2.0* cause some decrease in the number of '*rescheduling procedure*' required. As a result, the computational time is also reduced (this can be seen clearly in *UTA-S-92* data set).

The decrease in computational time can also be seen when the *Fuzzy SD+LE Model* is implemented. In the seventh and eighth columns of Table 5.1, the number of '*rescheduling procedure*' required are equal to zero. That means that all the unscheduled exams can be assigned to time slots without the need to reshuffling the already scheduled exams. Thus, applying *ALG1.0* and *ALG1.1* will always produce the same solution quality. Clearly, for the *Fuzzy SD+LE Model*, in many cases, the results were worst in terms of 'proximity cost' when the *ALG1.2* and *ALG2.0* are compared to the *ALG1.0*. This is due to the fact that the membership functions implemented are tuned for the *ALG1.0*, not for *ALG1.2* or *ALG2.0*.

Based on these observations, it was decided that the new improved algorithm should be used for the rest of the experiments relating to measuring the difficulty of scheduling exams to time slots by considering multiple heuristic ordering simultaneously. The modified sequential constructive algorithm is shown in Figure 5.1. In the implementation shown, the k value is set to 5. The '*rescheduling procedure*' is reproduced with very minor changes as shown in Figure 5.2. In the previous Chapter, the number of '*rescheduling procedure*' required is referred to the number of iterations of the procedure because it was dealing with 'skipped exams' and 'bumped back exams'. The '*rescheduling procedure*' is only activated if the 'skipped exams' list is consisting at least one element after all exams with valid time slot are scheduled in the timetable. Meanwhile, in this Chapter, the number of '*rescheduling procedure*' required refers to the number of time this procedure is called. The outer *WHILE* loop statement (see Figure 4.2) is removed because this new procedure is only activated when the reshuffling of the conflicting scheduled exams is required.

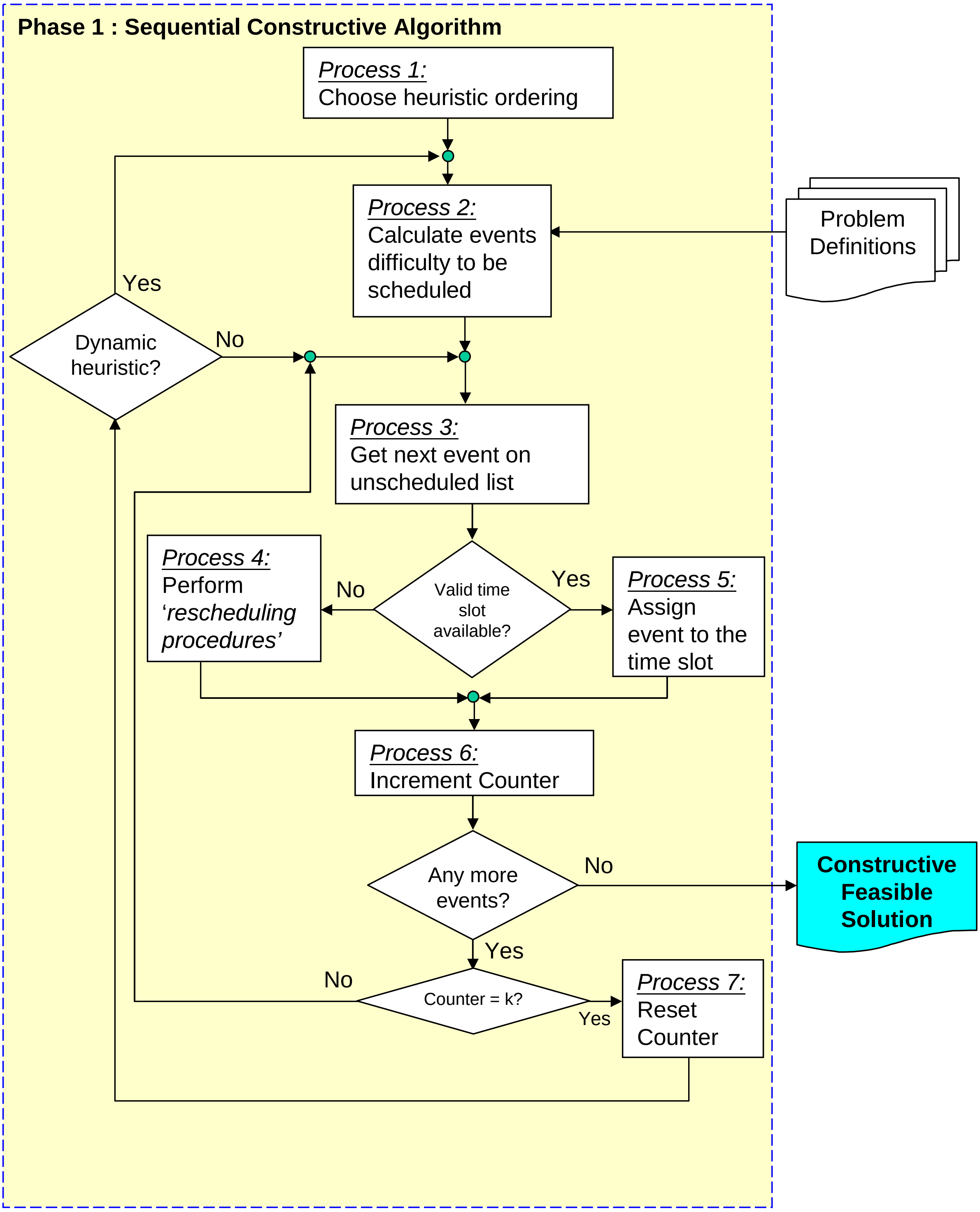


Figure 5.1: The modified algorithm

```
E* = current unscheduled event that need to be scheduled;
Find time slots where event E* can be inserted with minimum number of scheduled events need to
be removed from the time slot;
If found more than one time slot with the same number of scheduled events need to be removed
      Select a time slot t randomly from the candidate list of time slots;
End if
While there exist events that conflict with event E* in time slot t
    Et = first event in time slot t ;
    If found another time slot with minimum penalty cost to move event Et
        Move event Et to the time slot;
    else
        Bump back event Et to unscheduled events list;
    End if
End While
Insert event E* to timeslot t;
Remove event E* from unscheduled event list;
```

Figure 5.2: Pseudo code for the new '*rescheduling procedure*'

5.2.2 Experiments with Revised Algorithm

A series of experiments were carried out to test the new algorithm. Ultimately, the objective of these experiments was to compare the solution quality when the different kinds of heuristic ordering were employed to measure the difficulty of scheduling exams to time slots. The heuristic orderings considered in the experiments are described below.

5.2.2.1 Linear Multiple Heuristic Ordering

One way to simultaneously consider several heuristic orderings in measuring the exam difficulty weight is to multiply the value of the particular attribute of that exam with a weighting factor. In this approach, the exams in the problem instances were ordered based on a linear multiple heuristic ordering. All the exams were then selected to be scheduled based on this ordering. When this method is used, the linear weighted function becomes, for example:

$$W(e_j) = w_d LD_j + w_e LE_j + w_s SD_j$$

where $j = 1, 2, ...N$; $w_d = w_e = w_s = 0.0, 0.1, , 1.0$ if $N <= 300$; or $w_d = w_e = w_s = 0.0, 0.25, 0.5, 0.75, 1.0$ if $N > 300$; and w_d, w_e, w_s are weighting factors for LD, LE and SD respectively.

In the implementation, if one of the weighted factors is equal to zero, and the other two weighted factors are assigned with non-zero value, this situation represents the implementation of two heuristic ordering simultaneously. On the other hand, if two of the weighted factors are equal to zero, and the other one is equal to 1.0, this situation represents *Single Heuristic Ordering*. These non-fuzzy multiple heuristic orderings were developed for the purposes of comparison to the fuzzy multiple heuristic ordering detailed below.

5.2.2.2 Fuzzy Multiple Heuristic Ordering

As discussed in Section 4.4.1.3, the fuzzy models must be tuned using the new algorithm in order to search for the 'best' fuzzy model for each heuristic ordering combinations descried in Section 4.4.1. Similar procedures for tuning membership functions as described in Section 4.4.1.3 were implemented. From this point onwards, it is no longer interesting to compare with the *Fixed Fuzzy LD+LE Model* explained in Section 4.4.1.2. Therefore, this model will not be included in the current and the future experiments or, indeed, discussions. In this Chapter, the membership functions tuning process is performed for the three fuzzy model: *Fuzzy LD+LE Model*, *Fuzzy SD+LE Model* and *Fuzzy SD+LD Model*.

In addition to these three fuzzy models, a new fuzzy model that takes into account three heuristic orderings simultaneously was proposed. This is identified as *Fuzzy LD+SD+LE Model*. Therefore, a fuzzy system with three input variables and one output variable was developed. Again, the triangular shape membership functions depicted in Figure 4.7 were used as an initial membership function for all of the variables. Also, the same procedures explained in Section 4.4.1.3 were employed to tune this fuzzy model.

A set of fixed fuzzy rules applicable to the *Fuzzy LD+LE+SD* model are presented in Table 5.2.

Table 5.2: Fuzzy rule set for *Fuzzy LD+LE+SD Model*

	LD								
	S			*M*			*H*		
LE	**SD**			**SD**			**SD**		
	S	*M*	*H*	*S*	*M*	*H*	*S*	*M*	*H*
S	S	VS	VS	S	S	VS	M	S	S
M	S	S	VS	H	M	M	H	M	M
H	H	S	S	H	M	M	VH	H	M

VS=very small
S=small
M=medium
H=high
VH=very high

5.2.3 Experimental Results

The experiments were undertaken in two stages. The first stage focused on finding the appropriate weighted factor values for the linear multiple heuristic orderings and the *cp* values for the fuzzy multiple heuristic orderings. The results for the tuning process of the linear multiple heuristic orderings are presented in Table 5.3; while for the fuzzy multiple heuristic orderings, the results are presented in Table 5.4.

These values were then used in the second stage of the experiments in which repeated runs were performed to generate 30 solutions with each heuristic ordering, for each of the twelve data sets. In total, eleven heuristic orderings were tested in this experiment. The following list shows the full list of the heuristic orderings that were compared:

- Single Heuristic Ordering, *LD* (when $w_d = 1.0$, $w_e = 0.0$ and $w_s = 0.0$)
- Single Heuristic Ordering, *LE* (when $w_d = 0.0$, $w_e = 1.0$ and $w_s = 0.0$)
- Single Heuristic Ordering, *SD* (when $w_d = 0.0$, $w_e = 0.0$ and $w_s = 1.0$)
- Linear Two Heuristic Ordering, *Linear LD+LE* (when $w_s = 0.0$; w_d and w_e are assigned with the values in the second and third columns of Table 5.3 for respective data set)

Table 5.3: Values for weighted factors identified in the tuning process

Data Set	*Linear LD+LE*		*Linear SD+LE*		*Linear SD+LD*		*Linear LD+SD+LE*		
	w_d	w_e	w_s	w_e	w_s	w_d	w_d	w_s	w_e
CAR-F-92	0.50	0.75	0.00	1.00	0.75	0.00	0.25	0.75	0.75
CAR-S-91	0.75	1.00	0.25	0.25	1.00	0.25	0.75	0.75	1.00
EAR-F-83	0.90	0.70	0.00	0.00	0.50	0.40	1.00	0.10	0.80
HEC-S-92	0.10	0.70	1.00	0.40	0.90	0.00	0.10	0.00	0.70
KFU-S-93	0.00	0.25	0.25	1.00	0.75	0.50	0.25	0.75	0.50
LSE-F-91	0.75	0.75	0.00	1.00	0.50	0.50	0.75	0.75	0.50
RYE-F-92	0.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
STA-F-83	0.10	0.00	0.00	0.00	0.10	0.00	0.10	0.00	0.00
TRE-S-92	0.00	0.50	0.00	0.50	0.60	0.40	0.70	0.60	0.40
UTA-S-92	0.25	1.00	0.25	1.00	0.75	0.00	0.25	0.50	0.75
UTE-S-92	0.70	0.40	0.10	0.30	0.10	0.70	0.30	0.90	0.80
YOR-F-83	0.00	0.40	0.60	0.20	0.70	0.40	0.10	0.10	1.00

Table 5.4: Values for *cp* parameters obtained from the fuzzy tuning process

Data Set	*Fuzzy LD+LE Model*			*Fuzzy SD+LE Model*			*Fuzzy SD+LD Model*			*Fuzzy LD+SD+LE Model*			
	LD	*LE*	*exam weight*	*SD*	*LE*	*exam weight*	*SD*	*LD*	*exam weight*	*LD*	*SD*	*LE*	*exam weight*
CAR-F-92	0.75	1.00	0.00	1.00	0.25	0.25	0.25	0.50	1.00	0.00	1.00	0.50	0.50
CAR-S-91	1.00	0.75	0.00	0.50	0.25	0.75	0.75	0.00	0.25	0.00	0.75	0.25	0.50
EAR-F-83	0.40	0.00	0.80	0.20	0.80	0.80	1.00	0.20	0.80	0.50	0.90	0.70	0.10
HEC-S-92	0.30	0.90	0.70	0.40	1.00	1.00	0.90	0.00	0.20	0.30	0.20	0.70	0.70
KFU-S-93	0.75	0.00	0.00	0.00	0.25	0.00	0.75	1.00	0.50	0.00	0.50	0.50	0.00
LSE-F-91	1.00	0.25	1.00	0.25	0.00	0.00	0.50	1.00	0.25	0.00	0.50	0.50	0.25
RYE-F-92	1.00	0.50	0.50	0.50	1.00	1.00	0.25	0.25	0.00	0.00	0.25	0.25	0.50
STA-F-83	0.60	0.70	0.90	0.90	0.90	0.00	0.60	0.00	1.00	0.60	0.90	0.80	0.00
TRE-S-92	0.80	0.20	0.00	0.80	0.90	0.10	0.70	0.90	0.80	0.00	0.30	0.70	0.50
UTA-S-92	0.75	0.25	0.50	0.00	0.00	0.75	0.25	0.25	0.75	0.25	0.75	0.25	0.00
UTE-S-92	0.30	0.60	0.00	0.60	0.70	0.30	0.00	0.90	0.40	0.00	0.50	0.20	0.30
YOR-F-83	0.80	0.80	0.70	0.50	0.70	0.50	0.30	0.80	0.70	0.10	0.30	0.50	0.90

- Linear Two Heuristic Ordering, *Linear SD+LE* (when $w_d = 0.0$; w_s and w_e are assigned with the values in the fourth and fifth columns of Table 5.3 for respective data set)
- Linear Two Heuristic Ordering, *Linear SD+LD* (when $w_e = 0.0$; w_d and w_s are assigned with the values in the sixth and seventh columns of Table 5.3 for respective data set)
- Linear Three Heuristic Ordering, *Linear LD+SD+LE* (when w_e, w_d and w_s are assigned with the values in the eighth to tenth columns of Table 5.3 for respective data set)
- Fuzzy Two Heuristic Ordering, *Fuzzy LD+LE Model* (using cp values in the second to fourth columns of Table 5.4 for respective data set)
- Fuzzy Two Heuristic Ordering, *Fuzzy SD+LE Model* (using cp values in the fifth to seventh columns of Table 5.4 for respective data set)
- Fuzzy Two Heuristic Ordering, *Fuzzy SD+LD Model* (using cp values in the eighth to tenth columns of Table 5.4 for respective data set)
- Fuzzy Three Heuristic Ordering, *Fuzzy LD+SD+LE Model* (using cp values in eleventh to fourteenth columns of Table 5.4 for respective data set)

Table 5.5 shows a comparison of the cost penalties obtained based on 30 runs of each data set when implementing non-fuzzy heuristic orderings. The best results among the different non-fuzzy heuristic orderings used are highlighted in bold font. It can be seen that, in eleven out of twelve data sets, best results are produced when multiple heuristic orderings are implemented. In the case of the *STA-F-83* data set, the single heuristic ordering *LD* produced the best result and has the same solution quality compared to the solutions produced by the *Linear LD+LE* and the *Linear LD+SD+LE*. In comparison with the best result amongst the single heuristic orderings, *Linear SD+LD* combination produced worst results in all the 12 data sets.

Table 5.5: The penalty costs obtained by the different non-fuzzy heuristic orderings on each of the twelve benchmark data sets

Data Set		Single Heuristic			Linear Multiple Heuristic Ordering			
		LD	*LE*	*SD*	LD + LE	SD+LE	SD+LD	SD+LD+LE
CAR-F-92	Best	4.89	4.74	5.12	**4.66**	4.72	4.90	4.67
Exams 543	Average	5.14	4.86	5.28	4.84	4.84	5.04	4.96
Sessions 32	Worst	6.61	5.05	5.45	5.18	5.14	5.29	5.67
	Std. Dev	0.32	0.07	0.11	0.12	0.08	0.09	0.21
CAR-S-91	Best	5.86	5.64	5.97	5.47	5.78	5.83	**5.38**
Exams 682	Average	6.15	6.02	5.97	5.61	6.05	5.99	5.42
Sessions 35	Worst	7.36	6.79	5.97	5.83	6.44	6.33	5.46
	Std. Dev	0.33	0.29	0.00	0.07	0.16	0.11	0.02
EAR-F-83	Best	39.90	45.57	45.42	38.68	50.95	40.99	**38.17**
Exams 190	Average	42.31	49.63	45.42	41.99	55.81	42.18	41.12
Sessions 24	Worst	46.40	53.20	45.42	50.02	62.98	47.40	48.61
	Std. Dev	1.57	2.46	0.00	3.23	2.83	1.86	2.34
HEC-S-92	Best	14.56	13.36	13.70	12.76	13.05	14.56	**12.68**
Exams 81	Average	16.29	14.73	15.06	13.56	15.26	16.87	13.71
Sessions 18	Worst	19.45	19.30	19.11	15.45	19.09	21.80	18.36
	Std. Dev	1.05	1.43	1.53	0.67	1.49	1.73	1.16
KFU-S-93	Best	17.64	16.23	18.33	16.45	16.20	17.77	**16.02**
Exams 461	Average	18.69	16.59	18.83	16.73	16.74	19.00	16.63
Sessions 20	Worst	19.80	17.01	21.87	17.10	18.06	22.29	18.81
	Std. Dev	0.55	0.23	0.70	0.19	0.50	1.10	0.80
LSE-F-91	Best	13.98	13.25	12.76	13.03	13.10	14.24	**12.47**
Exams 381	Average	16.13	14.19	12.76	14.06	14.34	16.16	12.73
Sessions 18	Worst	18.62	18.35	12.76	20.01	17.30	19.25	13.03
	Std. Dev	1.19	0.94	0.00	1.23	1.18	1.23	0.18
RYE-F-92	Best	12.34	10.80	11.51	12.42	**10.73**	12.79	10.96
Exams 486	Average	13.65	12.35	11.51	13.55	11.95	14.08	11.87
Sessions 23	Worst	16.14	14.89	11.51	15.73	13.47	16.42	13.13
	Std. Dev	1.03	0.98	0.00	0.74	0.77	1.08	0.57
STA-F-83	Best	**167.05**	172.01	177.93	**167.05**	172.76	171.51	**167.05**
Exams 139	Average	167.84	172.26	178.83	167.62	172.76	172.07	167.70
Sessions 13	Worst	168.48	172.49	179.73	168.48	172.76	172.95	168.48
	Std. Dev	0.56	0.19	0.92	0.53	0.00	0.55	0.59
TRE-S-92	Best	10.45	9.25	10.50	9.26	9.56	9.97	**9.21**
Exams 261	Average	11.22	9.70	10.50	9.73	10.01	10.34	9.26
Sessions 23	Worst	12.29	10.73	10.50	10.21	10.86	11.17	9.29
	Std. Dev	0.43	0.31	0.00	0.29	0.29	0.32	0.04
UTA-S-92	Best	3.97	3.71	4.11	3.65	3.82	3.83	**3.61**
Exams 622	Average	4.84	4.12	4.11	4.09	4.05	4.32	3.97
Sessions 35	Worst	6.04	5.39	4.11	4.96	4.83	4.94	4.62
	Std. Dev	0.54	0.39	0.00	0.44	0.24	0.27	0.31
UTE-S-92	Best	35.19	28.93	33.72	28.68	28.93	33.27	**28.41**
Exams 184	Average	35.22	29.32	35.07	28.68	29.16	33.40	28.97
Sessions 10	Worst	35.27	30.89	36.56	28.68	31.31	33.59	29.92
	Std. Dev	0.03	0.43	0.86	0.00	0.48	0.16	0.63
YOR-F-83	Best	45.72	42.65	46.74	42.03	44.47	44.02	**41.52**
Exams 181	Average	47.49	44.58	48.32	44.80	47.00	46.25	45.37
Sessions 21	Worst	50.24	48.78	49.70	48.78	51.80	49.00	48.82
	Std. Dev	1.12	1.51	0.73	1.71	1.63	1.31	1.83

Table 5.6 shows a comparison of the cost penalties obtained based on 30 runs of each data set when the fuzzy multiple heuristic ordering are implemented. The best results among the different fuzzy multiple heuristic orderings used are highlighted in bold font. It appears that *Fuzzy LD+SD+LE Model* is the best amongst the fuzzy multiple heuristic ordering, because it obtained nine best results, followed by *Fuzzy SD+LE Model* with two best results (*CAR-F-92* and *EAR-F-83*). Both *Fuzzy LD+SD+LE Model* and *Fuzzy SD+LE Model* produced best solutions with the same solution quality for the *UTA-S-92* data set. Comparing Tables 5.5 and 5.6, it is evident that the fuzzy multiple heuristic orderings have outperformed all of the non-fuzzy heuristic orderings in terms of cost penalty. Furthermore, if the best results in Table 5.6 are compared with the best results obtained in previous the Chapter (see Table 4.10), it can be seen that the solutions produced in the previous Chapter are beaten by the results obtained in this experiments in all data sets except *YOR-F-83*. However, looking at the *Fuzzy SD+LE Model* specifically (this combination is the best in the previous Chapter), solutions obtained in these experiments for six data sets (*CAR-S-91*, *HEC-S-92*, *KFU-S-93*, *LSE-F-91*, *TRE-S-92* and *YOR-F-83*) are outperformed by the solutions produced using the old algorithm (*ALG1.0*).

A parametric statistical test (the *t-Test*) was performed to measure statistical significance for the differences in the means when comparing fuzzy and linear combinations for each heuristic combination ($LD + LE$, $SD + LE$, $SD + LD$ and $SD + LD + LE$). The *t-Test* is designed to detect differences in two population means, and is suitable for large sample sizes (less than 8 is considered as a small sample size) (Ross, 2005b). The Microsoft Excel Data Analysis Tool was employed for all the statistical tests. As there are 30 samples per test, the distributions of the population can be assumed to be approximately normal. The null hypothesis H_0 is that the means of the two populations are equal. H_0 will be rejected if the probability that H_0 is true, $p-value$, is smaller than the predetermined level of significance, α (i.e. $p - value < \alpha$). As this methodology requires much repeated testing, a lower value of α is used. In the experiment, α is set to 0.001.

The resulting p-values for the *t-Test* are shown in Table 5.7. In the table, $p-value$s that are far lower than α are marked with '< 0.0001'. It can be seen that H_0 cannot be rejected in five cases for $LD + LE$ heuristic combination, two cases for $SD + LD$ heuristic combination, and one case for $SD + LD + LE$ heuristic combination. That means that H_0 *can be* rejected in 83.33% of the total cases. Although it is obvious that not all the differences are statistically significant, overall, it can be seen that the fuzzy approach does indeed show promising performance. It should be remembered that, in this experiment, only the membership functions of the fuzzy models were tuned. In the next Chapter, the experimental results show that the performance of the fuzzy system can be further improved by also tuning the fuzzy rules.

Tables 5.8 and 5.9 show comparisons of the number of '*rescheduling procedure*' required for the non-fuzzy heuristic orderings and the fuzzy multiple heuristic orderings respectively. In each case the smallest, the worst and the average number of '*rescheduling procedure*' required is given. Considering the single heuristic orderings (see the third to fifth columns of Table 5.8) and fuzzy two heuristic orderings (see the third to fifth columns of Table 5.9), it can be seen that the number of cases that required no '*rescheduling procedure*' is reduced, as compared to the results presented in Table 4.12 of the previous Chapter. Overall, the fuzzy two heuristic orderings show some increases in the number of '*rescheduling procedure*' required, but the increments are not so obvious (considering the average number of '*rescheduling procedure*' required). Specifically, the *Fuzzy SD+LE Model* now needs to invoke the '*rescheduling procedure*' to construct the timetable solutions in eight out of twelve data sets. In contrast, it only required one iteration of the '*rescheduling procedure*' for two data sets in the previous Chapter. On the other hand, it can be observed that the number of '*rescheduling procedure*' required is reduced in most cases when the different single heuristic ordering were applied with

Table 5.6: The penalty costs obtained by the different fuzzy multiple heuristic orderings on each of the twelve benchmark data sets

Data Set		*Fuzzy LD+LE Model*	*Fuzzy SD+LE Model*	*Fuzzy SD+LD Model*	*Fuzzy LD+SD+LE Model*
CAR-F-92	Best	4.57	**4.47**	4.62	4.53
Exams 543	Average	4.66	4.55	4.91	4.60
Sessions 32	Worst	4.75	4.75	5.15	4.76
	Std. Dev	0.06	0.07	0.12	0.06
CAR-S-91	Best	5.45	5.31	5.45	**5.21**
Exams 682	Average	5.68	5.31	5.45	5.30
Sessions 35	Worst	6.20	5.31	5.45	5.52
	Std. Dev	0.14	0.00	0.00	0.08
EAR-F-83	Best	38.80	**36.99**	39.34	37.11
Exams 190	Average	41.90	36.99	39.34	39.28
Sessions 24	Worst	49.72	36.99	39.34	41.77
	Std. Dev	2.42	0.00	0.00	1.42
HEC-S-92	Best	12.09	12.03	12.69	**11.70**
Exams 81	Average	13.56	12.51	14.54	12.28
Sessions 18	Worst	16.83	16.20	15.88	14.30
	Std. Dev	1.27	0.83	0.95	0.77
KFU-S-93	Best	15.73	15.90	16.09	**15.41**
Exams 461	Average	16.24	16.02	17.25	15.86
Sessions 20	Worst	17.46	16.34	20.23	17.57
	Std. Dev	0.37	0.11	0.85	0.42
LSE-F-91	Best	11.97	12.16	14.22	**11.43**
Exams 381	Average	12.30	12.32	15.19	11.43
Sessions 18	Worst	12.44	12.47	18.08	11.43
	Std. Dev	0.09	0.16	0.92	0.00
RYE-F-92	Best	13.02	10.25	13.40	**10.21**
Exams 486	Average	14.16	10.49	15.08	10.93
Sessions 23	Worst	17.02	10.63	16.93	14.03
	Std. Dev	0.89	0.19	1.15	1.16
STA-F-83	Best	159.82	159.59	165.25	**159.34**
Exams 139	Average	160.14	161.17	168.12	160.26
Sessions 13	Worst	160.42	163.62	172.53	161.29
	Std. Dev	0.30	1.20	2.86	0.47
TRE-S-92	Best	8.99	8.92	9.26	**8.64**
Exams 261	Average	9.18	9.12	9.26	8.64
Sessions 23	Worst	9.67	9.67	9.26	8.64
	Std. Dev	0.17	0.24	0.00	0.00
UTA-S-92	Best	3.77	**3.55**	3.73	**3.55**
Exams 622	Average	4.17	3.55	3.73	3.55
Sessions 35	Worst	5.81	3.55	3.73	3.55
	Std. Dev	0.45	0.00	0.00	0.00
UTE-S-92	Best	28.59	27.99	30.37	**27.64**
Exams 184	Average	28.66	28.25	30.76	28.80
Sessions 10	Worst	28.70	29.57	31.97	30.90
	Std. Dev	0.04	0.29	0.59	0.86
YOR-F-83	Best	41.10	40.71	43.00	**40.46**
Exams 181	Average	42.33	40.71	45.47	42.11
Sessions 21	Worst	43.60	40.71	46.34	46.60
	Std. Dev	0.70	0.00	0.76	1.41

Table 5.7: The Best Fuzzy, Best Linear and t-Test (Two-Sample Assuming Unequal Variances) Result of the twelve benchmark data sets

Data Set		Linear Multiple Heuristic Ordering			
		LD + LE	SD+LE	SD+LD	SD+LD+LE
CAR-F-92	Best Linear	4.66	4.72	4.90	4.67
	Best Fuzzy	4.57	**4.47**	4.62	4.53
	p-value	< 0.0001	< 0.0001	< 0.0001	< 0.0001
CAR-S-91	Best Linear	5.47	5.78	5.83	5.38
	Best Fuzzy	5.45	5.31	5.45	**5.21**
	p-value	0.0226	< 0.0001	< 0.0001	< 0.0001
EAR-F-83	Best Linear	38.68	50.95	40.99	38.17
	Best Fuzzy	38.80	**36.99**	39.34	37.11
	p-value	0.9063	< 0.0001	< 0.0001	0.0006
HEC-S-92	Best Linear	12.76	13.05	14.56	12.68
	Best Fuzzy	12.09	12.03	12.69	**11.70**
	p-value	0.9912	< 0.0001	< 0.0001	< 0.0001
KFU-S-93	Best Linear	16.45	16.20	17.77	16.02
	Best Fuzzy	15.73	15.90	16.09	**15.41**
	p-value	< 0.0001	< 0.0001	< 0.0001	< 0.0001
LSE-F-91	Best Linear	13.03	13.10	14.24	12.47
	Best Fuzzy	11.97	12.16	14.22	**11.43**
	p-value	< 0.0001	< 0.0001	0.0011	< 0.0001
RYE-F-92	Best Linear	12.42	10.73	12.79	10.96
	Best Fuzzy	13.02	10.25	13.40	**10.21**
	p-value	0.0052	< 0.0001	0.0009	0.0003
STA-F-83	Best Linear	167.05	172.76	171.51	167.05
	Best Fuzzy	159.82	159.59	165.25	**159.34**
	p-value	< 0.0001	< 0.0001	< 0.0001	< 0.0001
TRE-S-92	Best Linear	9.26	9.56	9.97	9.21
	Best Fuzzy	8.99	8.92	9.26	**8.64**
	p-value	< 0.0001	< 0.0001	< 0.0001	< 0.0001
UTA-S-92	Best Linear	3.65	3.82	3.83	3.61
	Best Fuzzy	3.77	**3.55**	3.73	**3.55**
	p-value	0.4965	< 0.0001	< 0.0001	< 0.0001
UTE-S-92	Best Linear	28.68	28.93	33.27	28.41
	Best Fuzzy	28.59	27.99	30.37	**27.64**
	p-value	0.0002	< 0.0001	< 0.0001	0.3907
YOR-F-83	Best Linear	42.03	44.47	44.02	41.52
	Best Fuzzy	41.10	40.71	43.00	**40.46**
	p-value	< 0.0001	< 0.0001	0.0071	< 0.0001

the new algorithm compared to the figures shown in Table 4.12. Comparing the linear multiple heuristic ordering with the fuzzy multiple heuristic ordering, it is clear that the fuzzy multiple heuristic ordering requires fewer '*rescheduling procedure*'.

Finally, Tables 5.10 and 5.11 show a comparison of the computational time required to construct the solutions for each non-fuzzy heuristic ordering and each fuzzy multiple heuristic ordering for each data set, respectively. Due to the fact that different computer

Table 5.8: The number of '*rescheduling procedure*' required for non-fuzzy heuristic orderings for each data set

Data Set		Single Heuristic			Linear Multiple Heuristic Ordering			
		LD	*SD*	*LE*	*LD + LE*	*SD +LE*	*SD +LD*	*SD +LD +LE*
CAR-F-92	Smallest	6	8	1	7	6	6	10
Exams 543	Average	25	13	1	20	14	10	19
Sessions 32	Worst	221	28	3	57	75	17	50
CAR-S-91	Smallest	9	13	0	5	18	6	4
Exams 682	Average	24	35	0	10	29	11	7
Sessions 35	Worst	145	85	0	18	57	18	12
EAR-F-83	Smallest	3	12	0	5	27	1	5
Exams 190	Average	7	72	0	14	85	4	13
Sessions 24	Worst	26	252	0	72	318	17	57
HEC-S-92	Smallest	3	4	5	4	8	4	4
Exams 81	Average	10	20	23	11	30	24	12
Sessions 18	Worst	43	74	158	35	118	84	64
KFU-S-93	Smallest	3	3	2	3	2	4	3
Exams 461	Average	7	4	6	5	5	13	7
Sessions 20	Worst	29	5	67	8	13	98	22
LSE-F-91	Smallest	4	3	0	2	4	3	2
Exams 381	Average	52	10	0	10	21	24	3
Sessions 18	Worst	249	67	0	66	160	150	4
RYE-F-92	Smallest	7	4	1	4	3	5	4
Exams 486	Average	58	35	1	63	20	50	20
Sessions 23	Worst	284	116	1	231	134	211	79
STA-F-83	Smallest	3	1	1	3	0	3	3
Exams 139	Average	3	1	1	3	0	3	3
Sessions 13	Worst	3	1	1	3	0	3	3
TRE-S-92	Smallest	6	2	0	2	5	3	1
Exams 261	Average	22	5	0	4	10	8	1
Sessions 23	Worst	74	11	0	8	24	20	2
UTA-S-92	Smallest	6	7	0	5	13	5	5
Exams 622	Average	60	39	0	32	20	28	12
Sessions 35	Worst	163	199	0	321	44	221	34
UTE-S-92	Smallest	2	2	2	0	2	3	2
Exams 184	Average	2	8	3	0	6	3	6
Sessions 10	Worst	2	59	7	0	55	3	31
YOR-F-83	Smallest	19	11	2	12	29	10	13
Exams 181	Average	109	24	11	27	66	45	39
Sessions 21	Worst	294	61	148	95	124	194	157

Table 5.9: The number of '*rescheduling procedure*' required for fuzzy multiple heuristic orderings for each data set

Data Set		*Fuzzy LD+LE Model*	*Fuzzy SD+LE Model*	*Fuzzy SD+LD Model*	*Fuzzy LD+SD+LE Model*
CAR-F-92	Smallest	6	1	2	5
Exams 543	Average	8	2	5	6
Sessions 32	Worst	11	6	9	8
CAR-S-91	Smallest	9	0	0	4
Exams 682	Average	14	0	0	7
Sessions 35	Worst	23	0	0	11
EAR-F-83	Smallest	4	0	0	3
Exams 190	Average	15	0	0	6
Sessions 24	Worst	95	0	0	52
HEC-S-92	Smallest	2	2	1	2
Exams 81	Average	9	3	8	10
Sessions 18	Worst	48	13	43	94
KFU-S-93	Smallest	4	2	1	3
Exams 461	Average	7	3	4	6
Sessions 20	Worst	8	7	12	12
LSE-F-91	Smallest	3	1	2	0
Exams 381	Average	3	1	11	0
Sessions 18	Worst	3	1	117	0
RYE-F-92	Smallest	12	1	3	2
Exams 486	Average	85	2	24	7
Sessions 23	Worst	367	2	105	46
STA-F-83	Smallest	1	2	6	2
Exams 139	Average	1	2	11	2
Sessions 13	Worst	1	3	26	2
TRE-S-92	Smallest	1	2	0	0
Exams 261	Average	1	3	0	0
Sessions 23	Worst	4	7	0	0
UTA-S-92	Smallest	9	0	0	0
Exams 622	Average	52	0	0	0
Sessions 35	Worst	296	0	0	0
UTE-S-92	Smallest	1	3	1	2
Exams 184	Average	1	5	1	6
Sessions 10	Worst	2	17	2	16
YOR-F-83	Smallest	4	0	1	4
Exams 181	Average	26	0	3	29
Sessions 21	Worst	146	0	6	147

specifications were utilised to construct the solutions, it is not possible to compare directly the time taken to construct the solutions in this Chapter to the time taken to construct the solutions in the previous Chapter (see Table 4.13). In this Chapter, the experiments were undertaken on a *Pentium(R) 4 CPU 2.20GHz* with *512MB RAM*, meanwhile a *Pentium(R) 4 CPU 1.80GHz* with *256MB RAM* were used in the previous Chapter. However, presumably the computational time reduction can be justified by analysing the percentage of improvement (reduction in computational time) as shown in Tables 5.12 and 5.13. In Table 5.12, it can be seen that the old algorithm (*ALG1.0*) required between 0.5 to 16 percent less computational time when the old algorithm (*ALG1.0*) was implemented on the *P4 2.20GHz* with *512MB RAM* computer. On the other hand, in Table 5.13 it can be observed that the computational time was reduced by at least 77.10 percent when the new algorithm (*ALG2.0*) was implemented (on a *P4 2.20GHz* with *512MB RAM* computer). This indicates that the changes made to the old algorithm results in less computational time being required to construct the timetable solution.

5.2.4 Discussion of Results

In *Chapter 4*, it was demonstrated that multiple heuristic orderings, utilising fuzzy techniques to consider two heuristic orderings simultaneously, could outperform any single heuristic ordering in the benchmark data sets used. In this Chapter, these experiments have been extended by utilising up to three heuristic orderings simultaneously. In Table 5.6, it can be seen that, for ten out of the twelve benchmark data sets used, better results were obtained when the three heuristic orderings simultaneously were applied compared to the two heuristics orderings simultaneously applied. It is not the case, however, that three heuristic orderings always performed better than two heuristic orderings. For two data sets (*CAR-F-92* and *EAR-F-83*), the best overall results are produced when the *Fuzzy SD+LE Model* is employed. This is probably due to the fact

Table 5.10: A comparison of the computational time (in seconds) required to construct the solutions for non-fuzzy heuristic ordering for each data set

Data Set		Single Heuristic			Linear Multiple Heuristic Ordering			
		LD	*SD*	*LE*	*LD + LE*	*SD +LE*	*SD +LD*	*SD +LD +LE*
CAR-F-92	Shortest	1.80	2.20	48.88	1.92	43.48	42.67	44.00
Exams 543	Average	12.84	3.80	49.46	7.42	45.68	43.20	46.45
Sessions 32	Worst	169.23	11.81	50.42	26.02	76.05	45.00	61.50
CAR-S-91	Shortest	3.78	5.77	119.36	3.22	111.78	100.28	100.05
Exams 682	Average	17.12	21.34	119.69	4.81	119.46	103.61	101.18
Sessions 35	Worst	161.74	74.34	120.44	12.72	162.14	111.72	102.61
EAR-F-83	Shortest	0.19	0.44	1.81	0.23	3.27	1.44	1.50
Exams 190	Average	0.33	4.62	2.57	0.80	7.22	1.57	1.91
Sessions 24	Worst	1.64	17.89	4.72	5.20	23.66	2.33	4.41
HEC-S-92	Shortest	0.03	0.05	0.19	0.05	0.20	0.14	0.19
Exams 81	Average	0.10	0.20	0.38	0.12	0.42	0.30	0.26
Sessions 18	Worst	0.41	0.73	1.69	0.36	1.27	0.89	0.72
KFU-S-93	Shortest	0.48	0.52	11.28	0.53	11.14	8.44	11.78
Exams 461	Average	0.80	0.56	11.82	0.57	11.40	9.41	12.21
Sessions 20	Worst	4.27	0.80	19.88	0.64	12.13	20.47	13.63
LSE-F-91	Shortest	0.33	0.33	6.42	0.30	5.69	5.22	5.44
Exams 381	Average	4.89	0.74	6.89	0.72	7.04	7.04	5.51
Sessions 18	Worst	24.09	6.09	8.27	5.91	16.94	19.58	5.58
RYE-F-92	Shortest	0.89	0.88	21.75	0.80	21.47	18.20	21.44
Exams 486	Average	10.59	6.26	21.83	11.46	24.17	25.24	24.04
Sessions 23	Worst	52.44	22.69	21.94	35.69	47.81	57.61	33.31
STA-F-83	Shortest	0.05	0.03	0.64	0.05	0.44	0.31	0.31
Exams 139	Average	0.05	0.05	0.69	0.06	0.44	0.32	0.33
Sessions 13	Worst	0.09	0.09	0.99	0.14	0.45	0.36	0.38
TRE-S-92	Shortest	0.42	0.27	5.06	0.28	4.08	3.70	3.89
Exams 261	Average	1.74	0.34	5.29	0.41	4.25	4.04	3.93
Sessions 23	Worst	6.11	0.47	5.63	0.92	5.05	4.97	4.00
UTA-S-92	Shortest	2.27	2.69	72.48	2.30	68.34	66.42	67.27
Exams 622	Average	49.81	31.97	72.61	23.52	71.25	77.55	71.06
Sessions 35	Worst	148.22	181.20	72.78	299.05	85.81	217.02	80.92
UTE-S-92	Shortest	0.05	0.05	0.73	0.05	0.67	0.55	0.69
Exams 184	Average	0.06	0.09	0.92	0.06	0.71	0.57	0.71
Sessions 10	Worst	0.11	0.36	1.59	0.08	0.97	0.63	0.81
YOR-F-83	Shortest	0.66	0.47	2.48	0.45	2.64	1.86	2.02
Exams 181	Average	3.99	1.08	3.12	1.27	4.16	3.06	3.13
Sessions 21	Worst	10.91	2.84	8.27	3.72	6.28	8.95	8.30

Table 5.11: A comparison of the computational time (in seconds) required to construct the solutions for fuzzy multiple heuristic orderings for each data set

Data Set		*Fuzzy LD+LE Model*	*Fuzzy SD+LE Model*	*Fuzzy SD+LD Model*	*Fuzzy LD+SD+LE Model*
CAR-F-92	Shortest	2.27	54.56	55.67	93.89
Exams 543	Average	2.55	54.83	56.32	97.23
Sessions 32	Worst	2.84	55.23	58.13	100.06
CAR-S-91	Shortest	4.33	123.47	125.17	185.25
Exams 682	Average	5.73	123.75	125.27	188.36
Sessions 35	Worst	10.98	124.25	125.42	200.81
EAR-F-83	Shortest	0.28	3.02	3.03	6.59
Exams 190	Average	0.88	3.02	3.04	7.12
Sessions 24	Worst	6.22	3.05	3.06	10.86
HEC-S-92	Shortest	0.06	0.45	0.41	1.13
Exams 81	Average	0.15	0.47	0.46	1.22
Sessions 18	Worst	0.52	0.58	0.75	1.95
KFU-S-93	Shortest	0.70	18.70	17.66	45.17
Exams 461	Average	0.77	18.81	17.89	47.20
Sessions 20	Worst	0.86	18.97	18.73	49.77
LSE-F-91	Shortest	0.47	11.49	11.49	31.81
Exams 381	Average	0.49	11.83	12.45	32.18
Sessions 18	Worst	0.63	15.00	22.44	32.34
RYE-F-92	Shortest	2.30	29.47	28.24	60.31
Exams 486	Average	16.38	29.63	31.92	61.64
Sessions 23	Worst	67.94	30.03	45.74	71.02
STA-F-83	Shortest	0.09	1.50	1.45	3.34
Exams 139	Average	0.10	1.51	1.52	3.60
Sessions 13	Worst	0.16	1.56	1.69	5.61
TRE-S-92	Shortest	0.38	6.95	6.89	14.30
Exams 261	Average	0.39	7.01	6.91	14.83
Sessions 23	Worst	0.45	7.20	6.94	16.03
UTA-S-92	Shortest	2.94	86.08	82.75	118.14
Exams 622	Average	39.22	86.71	83.38	128.30
Sessions 35	Worst	279.99	87.78	84.75	146.80
UTE-S-92	Shortest	0.13	2.09	2.00	5.39
Exams 184	Average	0.13	2.14	2.01	5.94
Sessions 10	Worst	0.19	2.28	2.06	7.36
YOR-F-83	Shortest	0.31	3.80	3.53	6.55
Exams 181	Average	1.07	3.81	3.57	7.57
Sessions 21	Worst	4.86	3.81	3.72	12.23

Table 5.12: A comparison of the average computational time required for *Fuzzy SD+LE Model* when old algorithm (*ALG1.0*) were run in two different computers. Values in the second column are extracted from Table 4.13, and values in the third column are extracted from Table A.2. For each data set, the percentage improvement is shown in the fourth column.

Data Set	P4 1.80 GHz PC	P4 2.20 GHz PC	Improvement
CAR-F-92	446.77	399.25	10.64%
CAR-S-91	1023.50	905.15	11.56%
EAR-F-83	19.38	19.14	1.19%
HEC-S-92	2.36	2.22	5.71%
KFU-S-93	113.92	107.86	5.32%
LSE-F-91	70.57	68.57	2.84%
RYE-F-92	221.01	185.25	16.18%
STA-F-83	6.59	6.36	3.51%
TRE-S-92	43.70	43.00	1.60%
UTA-S-92	695.52	602.47	13.38%
UTE-S-92	11.56	11.31	2.15%
YOR-F-83	22.51	22.39	0.50%

that a fixed fuzzy rule set was implemented in each case — no tuning of fuzzy rules was implemented. Here, the terms 'fixed fuzzy rule set' is referring to the 'best' fuzzy model obtained in Section 5.2.3, where only the membership functions are tuned but the fuzzy rules remain the same. This fixed fuzzy model is not related to the *Fixed Fuzzy LD+LE Model* described in Section 4.4.1.2. If the rule set were tuned, then it should be possible to find a model based on three heuristic orderings to outperform that based on two (assuming that it is possible to search a reasonable proportion of the overall model search space). This indicates that the selection of which heuristic orderings need to be combined and the number of heuristic orderings that need to be considered simultaneously are important in order to get good quality solutions. In addition, this study also confirms that, as might be expected, fuzzy reasoning does result in better solutions compared to linear combinations. Although fuzzy techniques required longer processing time (for tuning the membership functions), this is acceptable because once the best

Table 5.13: A comparison of the average computational time required for *Fuzzy SD+LE Model* when the new algorithm (*ALG2.0*) and the old algorithm (*ALG1.0*) were run in two different computers. Values in the second column are extracted from Table 4.13, and values in the third column are extracted from Table 5.11. For each data set, the percentage improvement is shown in the fourth column.

Data Set	*ALG1.0* (on P4 1.80 GHz PC)	*ALG2.0* (on P4 2.20 GHz PC)	Improvement
CAR-F-92	446.77	54.83	87.73%
CAR-S-91	1023.50	123.75	87.91%
EAR-F-83	19.38	3.02	84.40%
HEC-S-92	2.36	0.47	79.91%
KFU-S-93	113.92	18.81	83.49%
LSE-F-91	70.57	11.83	83.24%
RYE-F-92	221.01	29.63	86.59%
STA-F-83	6.59	1.51	77.10%
TRE-S-92	43.70	7.01	83.97%
UTA-S-92	695.52	86.71	87.53%
UTE-S-92	11.56	2.14	81.47%
YOR-F-83	22.51	3.81	83.09%

fuzzy model is known for the problem instances, the constructive algorithm can produce the solution in a reasonable time.

Tables 5.14 and 5.15 compare the results obtained in Chapter 4 (i.e. using the old algorithm (*ALG1.0*)) to the results produced using the new algorithm (*ALG2.0*). Focusing on the single heuristic ordering (see Table 5.14), it can be seen that, overall, better results were produced when different single heuristic orderings were implemented with the new algorithm (*ALG2.0*). Similarly, in Table 5.15, when two heuristic orderings simultaneously were applied with the new algorithm, improvements in the results produced can be observed in many cases. Also note that, in the previous Chapter, it was shown that in most cases the best results were obtained when the *Fuzzy SD+LE Model* is implemented, in which the solutions were constructed without needing to reshuffling the exams that had been scheduled earlier (i.e. the number of iteration in the '*rescheduling procedure*' was zero). In Table 5.9, it can be seen that, when the new algorithm (*ALG2.0*) were used, only solutions for four data sets (*CAR-S-91*, *EAR-F-83*, *UTA-S-92* and *YOR-F-83*) are constructed without '*rescheduling procedure*'. For the other eight data sets, the average number of '*rescheduling procedure*' required was between 2 to 5. Most probably, the number of '*rescheduling procedure*' required is slightly affected if exams reordering is performed only after k exams are successfully assigned to valid time slots (in this experiment, $k = 5$). While dynamic heuristic ordering depends on the current structure of the problem, it is likely that fewer '*rescheduling procedure*' are required (or might be not required at all) if the exams reordering is performed each time an exam is successfully assigned to a valid time slot. Yet, when the *Fuzzy SD+LE Model* is utilised, the new algorithm (*ALG2.0*) managed to produced better solutions compared to the old algorithm (*ALG1.0*) for six data sets.

Arguably, for any iterative improvement methods, better final solutions may be obtained when better initial solutions are used. However, for the cases of timetable solutions generated in this research, there is no conclusive proof that better final solutions will be produced. In this thesis, no attempt was made to iteratively improve the constructed timetable solution. This is due to the fact that the main objective of this research was specifically to investigate the applicability of fuzzy techniques in timetable construction. Despite this, many researchers have reported work on improving initial solutions that have not been constructed in random form (generally, it is expected that randomly constructed solutions are inferior to solutions constructed using heuristic or optimisation methods). Perttunen (1994) has described that the performance of iterative improvement methods are dependent on the heuristic to be utilised, the properties of the problem itself and the processing time available. Considering the *Travelling Salesmen Problem*, Perttunen (1994) also reported that improvements made to initial solutions generated using a construction heuristic generally produced better final solutions than those produced by improvement made on randomly generated initial solutions.

Furthermore, there has been a recent tendency to utilise hybrid approaches across

Table 5.14: Comparison of results for single heuristic orderings using two different algorithms

Data Set	Results from *Chapter 4* (ALG1.0)			Results with new algorithm (*ALG2.0*)		
	LD	*LE*	*SD*	*LD*	*LE*	*SD*
CAR-F-92	5.51	4.86	5.5	4.89	**4.74**	5.12
CAR-S-91	6.13	5.89	5.91	5.86	**5.64**	5.97
EAR-F-83	40.58	44.86	48.99	**39.90**	45.57	45.42
HEC-S-92	14.73	14.41	14.23	14.56	**13.36**	13.70
KFU-S-93	18.38	16.46	18.62	17.64	**16.23**	18.33
LSE-F-91	14.79	14.41	13.46	13.98	13.25	**12.76**
RYE-F-92	13.02	11.22	11.6	12.34	**10.80**	11.51
STA-F-83	173.09	171.8	178.24	**167.05**	172.01	177.93
TRE-S-92	10.65	9.92	10.81	10.45	**9.25**	10.50
UTA-S-92	4.26	4.63	3.83	3.97	**3.71**	4.11
UTE-S-92	35.19	**28.79**	33.26	35.19	28.93	33.72
YOR-F-83	45.32	43.33	45.26	45.72	**42.65**	46.74

different optimisation problems, including the timetabling problem (Burke and Newall, 2003; Merlot *et al.*, 2003), the bin-packing problem (Fleszar and Hindi, 2002) and dynamic cellular manufacturing systems (N. Safaei and Jabal-Ameli, 2008). Basically, in any hybrid approach, the idea is to generate a feasible solution by using a certain method or heuristic, and then apply another different method in order to improve the solution generated earlier. Therefore, to some extent, such improvement phase (or phases) can be considered as operating on solutions that somehow have better quality than an initial solution that has been randomly generated. Having said that, it is the author belief that better solution can be obtained by considering good initial solutions.

Overall, the *Fuzzy SD+LE Model* can be considered as the best model for two heuristic ordering combinations. As can be seen in Table 5.15, for six out of the twelve data sets, 'best' results were produced when this model was applied using the new algorithm, and four other 'best' results were obtained when this model was applied using the old algorithm. The *Fuzzy LD+LE Model* is the second best fuzzy model with two 'best'

Table 5.15: Comparison of results for two heuristic orderings used simultaneously when two different algorithms were applied

Data Set	Results from *Chapter 4* (ALG1.0)			Results with new algorithm (*ALG2.0*)		
	LD +LE	*SD +LE*	*SD +LD*	*LD +LE*	*SD +LE*	*SD +LD*
CAR-F-92	4.62	4.54	4.62	4.57	**4.47**	4.62
CAR-S-91	5.57	**5.29**	5.77	5.45	5.31	5.45
EAR-F-83	42.61	37.02	39.27	38.80	**36.99**	39.34
HEC-S-92	12.43	**11.78**	12.55	12.09	12.03	12.69
KFU-S-93	16.45	15.81	15.80	**15.73**	15.90	16.09
LSE-F-91	12.35	12.09	12.95	**11.97**	12.16	14.22
RYE-F-92	11.75	10.38	12.71	13.02	**10.25**	13.40
STA-F-83	160.42	160.75	171.42	159.82	**159.59**	165.25
TRE-S-92	9.05	**8.67**	9.80	8.99	8.92	9.26
UTA-S-92	3.86	3.57	3.86	3.77	**3.55**	3.73
UTE-S-92	28.65	28.07	31.05	28.59	**27.99**	30.37
YOR-F-83	41.02	**39.80**	44.70	41.10	40.71	43.00

results when the new algorithm is employed. Therefore, these observation indicate that, if an exam is found with no available slot, rather that skip that exam and deal with it later on, it is better to resolve the conflict as soon as the problem is identified. This is especially applicable to the heuristic orderings where in the previous Chapter they were required to reshuffled the already scheduled exams in order to create feasible solutions. Presumably, by immediately performing the '*rescheduling procedure*', more valid time slots are available to move the conflicting scheduled exams from the selected time slot. With more valid time slots available, the chance of finding a time slot with lower penalty cost is higher (but this penalty cost is not lower than the current penalty cost in the current time slot). If the 'stuck' exam were to be skipped and dealt with later on, most probably the timetable is already compact. In that situation, it is more likely that the chance of finding a time slot with minimum penalty cost to move the conflicting scheduled exams is lower; there may exist some valid time slots, but the penalty cost might be far too high compared to the current penalty in the current time slot. This modification (immediately performing the '*rescheduling procedure*') also adhered to the main idea of the employed sequential constructive algorithm, in which the most difficult exam (in terms of scheduling the exam) is given the higher priority to be scheduled first. In addition to the improvements in quality of solutions, the modifications made results in less computational time being taken to construct the solutions (see Tables 5.10 and 5.11).

Table 5.16 shows a comparison of the best results obtained here with results previously published by other researchers. Although the best results did not beat any of the best benchmark results, the fuzzy based ordering produced better results for *CAR-F-92*, *CAR-S-91*, *STA-F-83*, *TRE-S-92* and *YOR-F-83* than Carter *et al.*'s constructive approach. It also produced the best overall result for *YOR-F-83* compared to any other purely constructive approach. Note also that the result obtained here for *STA-F-83* beats any other already published results, but this has recently been bettered by Burke *et al.* (2007).

Table 5.16: A comparing of results obtained herein with results published by other researchers

Data Set	Fuzzy Multiple Heuristic Orderings	Carter *et al.* (1996)	Abdullah *et al.* (2006a)	Abdullah and Burke (2006)	Burke and Newall (2003)	Burke *et al.* (2004a)	Burke *et al.* (2006a)	Caramia *et al.* (2001)	Casey and Thompson (2003)	Di Gaspero and Schaerf (2001)	Kendall and Mohd Hussin (2005b)	Merlot *et al.* (2003)	White *et al.* (2004)	Yang and Petrovic (2005)
CAR-F-92	4.47	6.2	4.4	4.1	4.10	4.2	5.36	6.0	4.4	5.2	4.67	4.3	4.63	**3.93**
CAR-S-91	5.21	7.1	5.2	4.8	4.65	4.8	4.53	6.6	5.4	6.2	5.67	5.1	5.73	**4.50**
EAR-F-83	36.99	36.4	34.9	36.0	37.05	35.4	37.92	**29.3**	34.8	45.7	40.18	35.1	45.8	33.70
HEC-S-92	11.70	10.8	10.3	10.8	11.54	10.8	12.25	**9.2**	10.8	12.4	11.86	10.6	12.9	10.83
KFU-S-93	15.41	14.0	**13.5**	15.2	13.90	13.7	15.20	13.8	14.1	18.0	15.84	**13.5**	17.1	13.82
LSE-F-91	11.43	10.5	10.2	11.9	10.82	10.4	11.33	**9.6**	14.7	15.5	-	11.0	14.7	10.35
RYE-F-92	10.21	7.3	8.7	-	-	8.9	-	**6.8**	-	-	-	8.4	11.6	8.53
STA-F-83	159.34	161.5	159.2	159.0	168.73	159.1	158.19	158.2	**134.9**	160.8	157.38	157.3	158.0	151.50
TRE-S-92	8.64	9.6	8.4	8.5	8.35	**8.3**	8.92	9.4	8.7	10.0	8.39	8.4	8.94	7.92
UTA-S-92	3.55	3.5	3.6	3.6	3.20	3.4	3.88	3.5	-	4.2	-	3.5	4.44	**3.14**
UTE-S-92	27.64	25.8	26.0	26.0	25.83	25.7	28.01	**24.4**	25.4	29.0	27.60	25.1	29.0	25.39
YOR-F-83	40.46	41.7	**36.2**	**36.2**	37.28	36.7	41.37	**36.2**	37.5	41.0	-	37.4	42.3	36.35

Table 5.17: A comparison of results obtained using different *constructive* approaches

Data Set	Carter *et al.* (1996)	Burke and Newall (2004)	Burke *et al.* (2007)	Fuzzy Multiple Heuristic
CAR-F-92	6.2	**4.32**	4.53	4.47
CAR-S-91	7.1	**4.97**	5.36	5.21
EAR-F-83	36.4	**36.16**	37.92	36.99
HEC-S-92	**10.8**	11.61	12.25	11.70
KFU-S-93	**14**	15.02	15.2	15.41
LSE-F-91	**10.5**	10.96	11.33	11.43
RYE-F-92	**7.3**	**-**	-	10.21
STA-F-83	161.5	170.35	**158.19**	159.34
TRE-S-92	9.6	**8.38**	8.92	8.64
UTA-S-92	3.5	**3.36**	3.88	3.55
UTE-S-92	**25.8**	27.42	28.01	27.64
YOR-F-83	41.7	40.77	41.37	**40.46**

5.3 Chapter Summary

This Chapter has presented the extended version of the proposed fuzzy multiple heuristic ordering approach. Two mechanisms of the algorithm developed in Chapter 4 have been modified with the intention of reducing the computational time required for the timetable constructions.

The main objective of this work was to investigate the effect of implementing two different approaches to simultaneously considering multiple heuristic orderings when finding solutions for examination timetabling problems, namely linear combination and fuzzy reasoning. It can be seen that these investigations have confirmed that fuzzy reasoning produces better ordering of exams compared to linear combinations. These investigations have also shown that better timetable solutions can be obtained when three heuristic orderings are simultaneously considered in measuring the difficulty of exams to be scheduled.

Chapter 6

Generalisation of the Fuzzy Multiple Heuristic Ordering

6.1 Introduction

This Chapter examines the issue of generalisation of the fuzzy approach in relation to the University timetabling problem. This Chapter is divided into three parts. In the first part, the potential of applying fuzzy multiple heuristic orderings to course timetabling is presented. The purpose of this work is to investigate the applicability of the developed approach to a different kind but related type of timetabling problem.

The second part of the Chapter describes extensions to the fuzzy multiple heuristic orderings that was proposed and implemented in Chapter 4 and Chapter 5. In the previous two Chapters, the fuzzy multiple heuristic orderings are constructed based upon three single heuristic ordering namely *Largest Degree*, *Saturation Degree* and *Largest Enrolment*. So far, four variations of fuzzy multiple heuristic orderings that combining two or three of the above mentioned single heuristic ordering have been investigated. In this Chapter, an extensive series of experiments are presented in which another two single heuristic orderings (*Largest Coloured Degree* and *Weighted Largest Degree*) were considered. All together, when taking into account five single heuristic orderings, there are 20 possible combinations of two and three heuristic orderings.

Finally, the third part of this Chapter describes alternative ways for tuning the fuzzy models. Instead of only tuning the membership functions, the effects of tuning the fuzzy rules was investigated. Four alternative tuning approaches are described in detail and their results are compared.

6.2 Application to Course Timetabling

In Chapters 4 and 5, fuzzy methodology was used to rank exams based on an assessment of how difficult they were to schedule, taking into account multiple heuristics. It was shown that when two heuristic orderings were simultaneously considered to rank the exams, better results were obtained as compared to single heuristic ordering (Chapter 4). Orderings using three heuristics simultaneously were also considered, and a comparison was made between fuzzy ordering with single and linear combination of heuristic orderings (Chapter 5). All this previous work has been concerned with the problem of creating timetables for *examinations*.

In this Section, the same underlying methodologies (i.e. fuzzy multiple heuristic orderings) is applied to a novel context; that of *course timetabling*. We apply the same algorithms to create fuzzy inferencing systems as in Chapters 4 and 5, with a different penalty function to capture the different domain characteristics. In order to provide a comparative test, the algorithm was initially run without implementing fuzzy ordering. That is, in this approach, the events in the problem instances were ordered based on a single heuristic ordering. All the events were then selected to be scheduled based on this ordering. All the five single heuristic ordering described in Section 2.2.2 are utilised in the experiments.

Based on observations of implementing fuzzy multiple heuristic orderings on examination timetabling problems (see Chapter 5), it was found that in many cases considering three heuristic orderings simultaneously produced better solutions compared to single heuristic ordering or two heuristic orderings. Inspired by that finding, in this present work the focus is on creating a fuzzy inferencing system based on three of the five single heuristic orderings, *Largest Degree* (*LD*), *Largest Enrolment* (*LE*) and *Saturation Degree* (*SD*). These three heuristics were selected as these were the ones that featured in the previous work on examination timetabling, and based on the fact that the design of a fuzzy system utilising all five heuristics would have been intractable (if the same tuning methodology had been utilised).

Indeed, the same restricted form of exhaustive search described in Section 4.4.1.3 was used to find the most appropriate shape for the fuzzy membership functions in the system. As explained, each variable has 11 options of the membership function's shape. For fuzzy systems with 3 fuzzy variables, the search in tuning process needs to explore $11^3(1331)$ combinations of membership functions. If we consider a fuzzy system with 5 input variables, the tuning process would need to explore 11^5 $(161,051)$ combinations. As we have 11 data sets on which the system is run, experiments with 5 heuristic orderings would take months to finish. The fuzzy rule set shown in Table 5.2 illustrates the 27 fuzzy rules that being used in the experiments. The solution quality calculation described in Section 6.2.1 is used as the criteria to determine the fitness of the membership functions for the combinations of three heuristic ordering, namely *Fuzzy LD+SD+LE Model*.

6.2.1 Problem Definition

Table 6.1 reproduces the characteristics of the data sets that were used for these experiments (Socha *et al.*, 2002). These problems deal with the assignment of courses into time slots such that rooms do not violate any of the following hard constraints:

1. No student is required to attend more than one course at the same time
2. A course can only be scheduled to a room which satisfies the features required

by the course

3. A course can only be scheduled to a room which has enough room to accommodate all students registered for it
4. Only one course can be scheduled in one room at any time slot

Table 6.1: Course timetabling problem characteristics

Data sets	No. of events	No. of rooms	No. of students	No. of features
Small1	100	5	80	5
Small2	100	5	80	5
Small3	100	5	80	5
Small4	100	5	80	5
Small5	100	5	80	5
Medium1	400	10	200	5
Medium2	400	10	200	5
Medium3	400	10	200	5
Medium4	400	10	200	5
Medium5	400	10	200	5
Large	400	10	400	10

Any solutions which do not violate any of the above hard constraints are defined as feasible solutions. Only feasible solutions are accepted. Besides these hard constraints, the solutions should also try to satisfy the following soft constraints:

1. No student should be scheduled to attend only one course on a day
2. No course should be scheduled at the last time slot of the day for any student
3. No student should be scheduled to attend more than two courses consecutively in any one day

An attempt is made to best satisfy these soft constraints, but they are not compulsory. The quality of any feasible solution is measured by simply summing the number of students that fail to satisfy the soft constraints. Hence, the less the number of students that violate the soft constraints, the better the solution quality.

The timetable is developed for one week, from Monday to Friday. For each day, there are 9 time slots available. Hence, the number of time slots available is 45 x *number of rooms*.

6.2.2 Experimental Results

In order to reduce the computational time, the number of '*rescheduling procedure*' allowed was limited to 500 for small and medium data sets, whereas for large data set it was limited to 1000 times. This meant that during the search for a solution, if too many events needed to be reshuffled, the fuzzy model that was currently under consideration was skipped and the solution for that fuzzy model was treated as an infeasible solution. A new fuzzy model was then tested. This was because, from observation, it was found that in many cases good quality solutions were usually produced when only a small number of '*rescheduling procedure*' were required. The same setting for the maximum number of required '*rescheduling procedure*' was implemented for *Single Heuristic Ordering*.

The experimental results are shown in Table 6.2. The best results amongst the heuristic orderings implemented are highlighted in bold font. The '-' in Table 6.2 indicates that no feasible solution was generated within the specified maximum number of '*rescheduling procedure*'. It can be seen that the fuzzy multiple heuristic orderings has outperformed all single heuristic orderings in all tested problem instances.

In term of feasibility, the *Fuzzy Multiple Heuristic Ordering* managed to produce feasible solutions for all data sets, whereas the best *Single Heuristic Ordering*, *SD*, only managed to produce ten feasible solutions out of eleven data sets. Other *Single Heuristic Orderings* were worse. Moreover, no *Single Heuristic Ordering* was able to produce a feasible solution for the *Large* problem instance.

Table 6.2: Comparison of solution quality between *Single Heuristic Ordering* and *Fuzzy Multiple Heuristic Ordering*

Data Sets	Best	Single Heuristic				
	Fuzzy	*LD*	*SD*	*LCD*	*LE*	*WLD*
Small1	**10**	78	31	48	79	80
Small2	**9**	45	44	55	34	52
Small3	**7**	28	30	42	41	27
Small4	**17**	42	50	48	51	48
Small5	**7**	41	29	74	43	47
Medium1	**243**	423	345	433	465	445
Medium2	**325**	-	398	-	-	-
Medium3	**249**	-	298	-	-	-
Medium4	**285**	-	403	-	-	-
Medium5	**132**	296	252	307	399	445
Large	**1138**	-	-	-	-	-

Table 6.3 summarises the performance of each heuristic ordering in terms of the number of iterations of '*rescheduling procedure*' required. It can be seen that, all heuristics obtained the solutions for the small size problem instances without any iterations of '*rescheduling procedure*'. On the other hand, only the *Fuzzy Multiple Heuristic Ordering* and *Single Heuristic Ordering SD* managed to find solutions for all of the medium size problem instances without any '*rescheduling procedure*' (except for *Medium4* problem instance in which the *Fuzzy Multiple Heuristic Ordering* needed to perform the '*rescheduling procedure*' for one event). However, for the *Large* problem instance the *Fuzzy Multiple Heuristic Ordering* needed to perform the '*rescheduling procedure*' for 307 iterations before it found the solution, whereas *Single Heuristic Ordering SD* was unable to find a feasible solution (refer to Table 6.3).

Table 6.3: Comparison of number of iterations of '*rescheduling procedure*' required to produce the solutions shown in Table 6.2

Data Sets	Best	Single Heuristic				
	Fuzzy	*LD*	*SD*	*LCD*	*LE*	*WLD*
Small1	0	0	0	0	0	0
Small2	0	0	0	0	0	0
Small3	0	0	0	0	0	0
Small4	0	0	0	0	0	0
Small5	0	0	0	0	0	0
Medium1	0	40	0	122	60	59
Medium2	0	-	0	-	-	-
Medium3	0	-	0	-	-	-
Medium4	1	-	0	-	-	-
Medium5	0	2	0	51	41	40
Large	307	-	-	-	-	-

6.2.3 Discussion of Results

Looking at the quality of the produced solutions summarised in Table 6.2, for all test instances of small and medium size, the *Fuzzy Multiple Heuristic Ordering* resulted in better solutions compared to any of the single heuristic orderings. For the *Large* data set, a feasible result was obtained only when the *Fuzzy Multiple Heuristic Ordering* was implemented. This is consistent with the implementation of *Fuzzy Multiple Heuristic Ordering* on examination timetabling problems previously described. Hence, these observations seem to indicate that this *Fuzzy Multiple Heuristic Ordering* approach may be applicable to a wider range of timetabling and scheduling problems.

Table 6.4 shows the best results obtained here in comparison with the approaches of other researchers applied to the same data sets. However, it has to be kept in mind that the fuzzy method is constructive, as opposed to the other methods which are iterative improvement approaches (except (Burke *et al.*, 2007)). Burke *et al.* (2003c) and Socha

Table 6.4: Comparison of solution quality with other results in literature

Data Set	FMHO	MA	GHH	THH	RRLS	AMM
Small1	10	**0**	6	1	8	1
Small2	9	**0**	7	2	11	3
Small3	7	**0**	3	**0**	8	1
Small4	17	**0**	3	1	7	1
Small5	7	**0**	4	**0**	5	**0**
Medium1	243	221	372	**146**	199	195
Medium2	325	**147**	419	173	202.5	184
Medium3	249	**246**	359	267	-	248
Medium4	285	165	348	169	177.5	**164.5**
Medium5	132	**130**	171	303	-	219.5
Large	1138	**529**	1068	1166	-	851.5

FMHO - *Fuzzy Multiple Heuristic Ordering*
MA - Memetic Approach (Abdullah, 2006, Chap. 9)
GHH - Graph-Based Hyperheuristic (Burke *et al.*, 2007)
THH - Tabu-Search Hyperheuristic (Burke *et al.*, 2003c)
RRLS - Random Restart Local Search (Socha *et al.*, 2002)
AMM - Ant MAX-MIN Algorithm (Socha *et al.*, 2002)

et al. (2002) start finding the solution by constructing an infeasible initial solution and then iteratively improving the timetable within a limited number of evaluations. Abdullah (2006, Chap. 9) started with feasible solutions and used a *memetic approach* with randomised iterative improvement techniques to improve the solutions. Burke *et al.* (2007) used a sequence of heuristic orderings to construct the initial solution and applied steepest descent local search to improve the solution. Although the approach here did not perform particularly well for small size problem instances, it is evident that our results are comparable to the other approaches for the medium and large size problem instances.

In terms of constructive approaches, it is more interesting to compare with Burke *et al.* (2007)'s approach because they used a sequence of heuristic orderings to construct the solution whereas here several heuristic orderings are used simultaneously to construct the timetable. When comparing these two constructive approaches, the approach here produced better results for all of the medium size problem instances, but slightly worse solutions were obtained for small and large size problem instances. It is believed that these initial solutions can be easily improved by applying a simple optimisation algorithm.

6.3 Alternative Combinations of Heuristic Orderings

In Chapter 4 it can be seen that, all of the 'best' results produced by fuzzy multiple heuristic ordering approach are constructed without the need to bump back the exams that were already scheduled earlier. Specifically, for nine data sets no '*rescheduling procedure*' is performed. Although few exams are skipped in three cases (*HEC-S-92*, *STA-F-83* and *UTE-S-92*), the number of iterations for the '*rescheduling procedure*' are equal to the number of skipped exam(s). Hence, the same timetable solutions for each data set are produced every time the 'best' fuzzy multiple heuristic ordering model is applied in the sequential constructive algorithm. This means that, no stochastic element is involved in the timetable constructions (recalled that in the '*rescheduling procedure*' (see Figure 4.2), time slot is choose randomly if more than one time slot with minimum number of exams need to be removed from the time slot are available). Therefore, it can be assumed that fuzzy multiple heuristic orderings (with the 'best' tuned fuzzy model) are capable of producing exams orderings that can guide the search algorithm towards better solutions.

However, the experimental results presented in Chapter 5 has shown that the modified version of the original sequential algorithm (see Figure 4.1) cause slightly increased numbers of iterations of the '*rescheduling procedure*' in order to construct the timetable solutions. Moreover, it is obvious that the number of timetable solutions that are constructed without the need to call the '*rescheduling procedure*' has been reduced. Looking at the 'best' solutions produced using fuzzy multiple heuristic ordering, timetable solutions for four data sets (*EAR-F-83*, *LSE-F-91*, *TRE-S-92* and *UTA-S-92*) have been constructed without need to call '*rescheduling procedure*' i.e no non-deterministic factor involved. For the remaining eight data sets, when run for 30 times, variations of timetable solutions with different timetable quality have been constructed for each data set. The non-deterministic features in the '*rescheduling procedure*' has provided larger search space to be explored by the search algorithm. Accordingly, better solutions might exist within the larger search space. Although it was expected that the use of three heuristic ordering simultaneously would significantly contribute to the better timetable solutions as compared to two heuristic orderings, it is believed that non-deterministic feature embedded in the '*rescheduling procedure*' also play important role in this achievement. The changes made to the sequential algorithm have somehow effected the sequence of exams being scheduled determined earlier.

Considering the fact that a larger search space provides more chances to explore for better solutions, a deliberately non-deterministic feature was added in the sequential constructive algorithm. The motivation behind the idea of introducing a more random element in the algorithm was based on the approach proposed by Burke *et al.* (1998a). Burke *et al.* applied two different types of random selection to select which exam to schedule next. Basically, the idea was to not select the most difficult exam to be scheduled every time they wanted to choose an exam. In order to achieve this, an exam was selected from smaller group of unscheduled exams. The member of the smaller group was selected using either *Tournament Selection* or *Bias Selection*. Their experimental results showed that the random selection approach produced better solutions compared to the approach without randomization. Other related work that has used random selection approach for examination timetabling was published by Broder (1964).

The work presented in this Section did not use random selection of exams to be scheduled. Instead, the selection of the next exam to be scheduled was based on the ordering of exams generated by the specified heuristic ordering. The modified version of the sequential constructive algorithm explained in Chapter 5 was used with changes in the way that time slots were selected. Referring to Figure 5.1, in *Process 5*, rather than assigning exam to the last time slot with least penalty cost, the time slot was randomly selected if more than one valid time slot with the same penalty cost was available. In addition to this change, an additional two single heuristic orderings were considered - *Largest Coloured Degree* (*LCD*) and *Weighted Largest Degree* (*WLD*). All possible combinations of two and three heuristic orderings are shown in Figure 6.1 and Figure 6.2, respectively.

For each fuzzy multiple heuristic ordering, the fuzzy rules used were based on the set of rules defined in Tables 6.5 to 6.8. When combining two heuristic orderings, the fuzzy rules shown in Table 6.5 were applied if one of the heuristic orderings was *SD*, otherwise

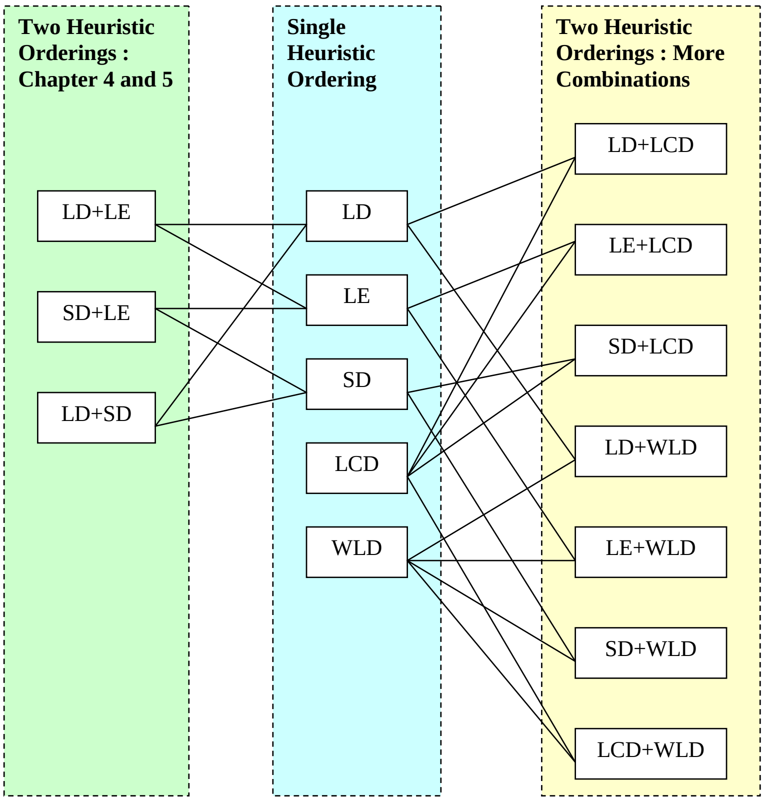


Figure 6.1: Possible combinations of two heuristic orderings

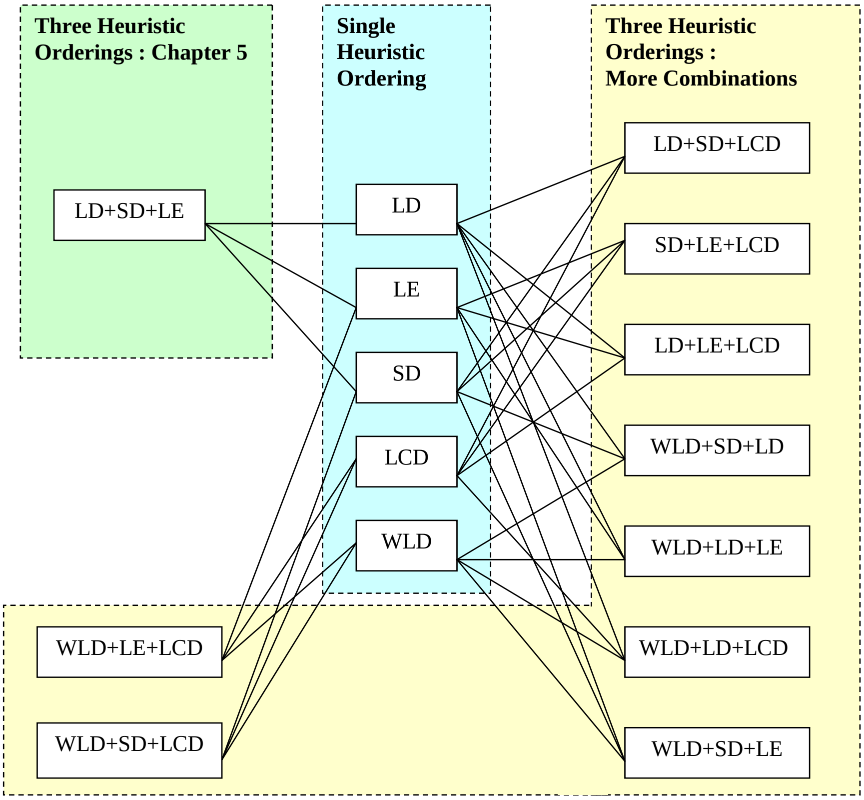


Figure 6.2: Possible combinations of three heuristic orderings

Table 6.5: Fuzzy rule set when combining two heuristic orderings (with *SD* as one of the variable)

		SD		
		S	M	H
HEUR-1	S	M	S	VS
	M	H	M	S
	H	VH	H	M

VS: very small
S: small
M: medium
H: high
VH: very high

fuzzy rules shown in Table 6.6 were used. When considering three heuristic orderings simultaneously, the fuzzy rules shown in Table 6.7 were applied if one of the heuristic ordering is *SD*, otherwise fuzzy rules shown in Table 6.8 were used. During the implementation, the variables *HEUR-1*, *HEUR-2* and *HEUR-3* were replaced with the specific heuristic orderings that constitute the considered fuzzy multiple heuristic ordering.

Table 6.6: Fuzzy rule set when combining two heuristic orderings (without *SD*)

		HEUR-2		
		S	M	H
HEUR-1	S	VS	S	M
	M	S	M	H
	H	M	H	VH

VS: very small
S: small
M: medium
H: high
VH: very high

Table 6.7: Fuzzy rule set when combining three heuristic orderings (with *SD* as one of the variable)

HEUR-1	***HEUR-2***								
	S			*M*			*H*		
	SD			***SD***			***SD***		
	S	*M*	*H*	*S*	*M*	*H*	*S*	*M*	*H*
S	S	VS	VS	S	S	VS	M	S	S
M	S	S	VS	H	M	M	H	M	M
H	H	S	S	H	M	M	VH	H	M

VS=very small
S=small
M=medium
H=high
VH=very high

Table 6.8: Fuzzy rule set when combining three heuristic orderings (without *SD*)

HEUR-1	***HEUR-2***								
	S			*M*			*H*		
	HEUR-3			***HEUR-3***			***HEUR-3***		
	S	*M*	*H*	*S*	*M*	*H*	*S*	*M*	*H*
S	VS	VS	S	VS	S	M	S	M	H
M	VS	S	M	S	M	H	M	H	VH
H	S	M	H	M	H	VH	H	VH	VH

VS=very small
S=small
M=medium
H=high
VH=very high

6.3.1 Experimental Results

Carter *et al.*'s benchmark data sets were used in this experiment. Similar tuning membership functions procedures described in Section 4.4.1.3 were implemented. As shown in Figures 6.1 and 6.2, all together twenty variations of multiple heuristic orderings are possible when considering simultaneously combinations of two and three of the five single heuristic orderings. The four fuzzy multiple heuristic ordering (i.e *Fuzzy LD+LE Model*, *Fuzzy SD+LE Model*, *Fuzzy SD+LD Model* and *Fuzzy LD+SD+LE Model*) that have been used in the last two Chapters were not used in this experiment. Taking into account sixteen different fuzzy multiple heuristic orderings and five single heuristic ordering that needed to be considered in the experiments, only proximity cost was used for the purpose of comparison.

In the preliminary investigations, all of the five single heuristic orderings were applied with both types of assignment of time slots — last time slot and random time slot selection (both were applied within the context of time slots with the same least penalty cost). In the cases of *LD*, *SD* and *LE*, results presented in Chapter 5 are reproduced here. All experiments for single heuristic orderings were run 30 times. The purpose of these experiments was to analyse the performance of the five different single heuristic orderings when they are used individually within the sequential constructive algorithm with different way of time slot selection. Results from these experiments were used for comparison with results produced by variations of fuzzy multiple heuristic orderings.

Results for the five single heuristic ordering when applied with last time slot selection are shown in Table 6.9, where the third to fifth columns were reproduced from Table 5.5 in Chapter 5. It can be seen that, for eleven out of the twelve data sets, the 'best' results are produced when *WLD* and *LE* were applied. In particular, five 'best' results were obtained by *WLD*, three by *LE* and only one by *LD*. *WLD* and *LE* produced solutions that were almost the same quality (decimal point rounding) for *CAR-F-92* and

TRE-S-92 data sets. *WLD* also produced the best result that has the same solution quality compared to the best solution produced by *LCD* for *CAR-S-91* data set. It seems that, amongst the single heuristic ordering, *WLD* appears to be the best (bear in mind that *LE* is the best single heuristic in the previous experiments presented in the last two Chapters).

Turning our attention to the random time slot selection approach, in Table 6.10, it can be observed that the best results for ten out of the twelve data sets were once again obtained when *WLD* and *LE* were implemented. Specifically, six 'best' results were obtained by *WLD*, four by using *LE*, while *LD* and *SD* produced one best solution each.

Considering the computational time required for tuning membership functions, only the 'best' results obtained by each of the sixteen new fuzzy multiple heuristic ordering are reported. Tables 6.11 and 6.12 show comparisons of 'best' results obtained by fuzzy multiple heuristic ordering when two and three heuristic orderings were considered simultaneously. In the case of two heuristic orderings, better results were produced in nine out of the twelve data sets compared to the results produced by the two heuristics ordering applied in Chapter 5. Concerning the three heuristic ordering, the *Fuzzy LD+SD+LE Model* has been outperformed by the new fuzzy multiple heuristic ordering in all of the data sets.

Table 6.9: A comparison of results for five *Single Heuristic Ordering* with last time slot selection

Data Set		Heuristic Ordering (last time slot)				
		LD	***LE***	***SD***	***LCD***	***WLD***
CAR-F-92	Best	4.89	**4.74**	5.12	5.11	**4.74**
Exams 543	Average	5.14	4.86	5.28	5.34	4.99
Sessions 32	Worst	6.61	5.05	5.45	5.73	6.34
	Std. Dev	0.32	0.07	0.11	0.17	0.31
CAR-S-91	Best	5.86	5.64	5.97	**5.56**	**5.56**
Exams 682	Average	6.15	6.02	5.97	5.66	5.79
Sessions 35	Worst	7.36	6.79	5.97	5.79	6.57
	Std. Dev	0.33	0.29	0.00	0.06	0.22
EAR-F-83	Best	39.90	45.57	45.42	41.45	**38.85**
Exams 190	Average	42.31	49.63	45.42	42.21	41.35
Sessions 24	Worst	46.40	53.20	45.42	44.73	44.78
	Std. Dev	1.57	2.46	0.00	0.55	1.58
HEC-S-92	Best	14.56	13.36	13.70	14.13	**12.77**
Exams 81	Average	16.29	14.73	15.06	15.68	14.67
Sessions 18	Worst	19.45	19.30	19.11	16.96	22.09
	Std. Dev	1.05	1.43	1.53	1.08	1.93
KFU-S-93	Best	17.64	**16.23**	18.33	17.61	17.65
Exams 461	Average	18.69	16.59	18.83	18.57	18.55
Sessions 20	Worst	19.80	17.01	21.87	19.04	21.43
	Std. Dev	0.55	0.23	0.70	0.41	1.02
LSE-F-91	Best	13.98	13.25	12.76	13.55	**12.55**
Exams 381	Average	16.13	14.19	12.76	14.78	13.13
Sessions 18	Worst	18.62	18.35	12.76	19.21	14.32
	Std. Dev	1.19	0.94	0.00	1.18	0.42
RYE-F-92	Best	12.34	10.80	11.51	11.56	**9.85**
Exams 486	Average	13.65	12.35	11.51	11.96	10.30
Sessions 23	Worst	16.14	14.89	11.51	13.30	12.47
	Std. Dev	1.03	0.98	0.00	0.57	0.66
STA-F-83	Best	**167.05**	172.01	177.93	169.58	172.01
Exams 139	Average	167.84	172.26	178.83	170.75	172.21
Sessions 13	Worst	168.48	172.49	179.73	171.54	172.49
	Std. Dev	0.56	0.19	0.92	0.80	0.18
TRE-S-92	Best	10.45	**9.25**	10.50	10.02	**9.25**
Exams 261	Average	11.22	9.70	10.50	10.44	9.66
Sessions 23	Worst	12.29	10.73	10.50	10.84	10.67
	Std. Dev	0.43	0.31	0.00	0.21	0.31
UTA-S-92	Best	3.97	3.71	4.11	3.88	**3.63**
Exams 622	Average	4.84	4.12	4.11	3.92	3.76
Sessions 35	Worst	6.04	5.39	4.11	3.97	3.98
	Std. Dev	0.54	0.39	0.00	0.03	0.09
UTE-S-92	Best	35.19	**28.93**	33.72	31.28	29.59
Exams 184	Average	35.22	29.32	35.07	31.28	30.42
Sessions 10	Worst	35.27	30.89	36.56	31.28	31.50
	Std. Dev	0.03	0.43	0.86	0.00	0.55
YOR-F-83	Best	45.72	**42.65**	46.74	46.31	44.19
Exams 181	Average	47.49	44.58	48.32	49.37	46.46
Sessions 21	Worst	50.24	48.78	49.70	55.67	48.67
	Std. Dev	1.12	1.51	0.73	1.70	1.50

Table 6.10: A comparison of results for five *Single Heuristic Ordering* with random time slots selection

Data Set		Heuristic Ordering (random time slot)				
		LD	***LE***	***SD***	***LCD***	***WLD***
CAR-F-92	Best	4.91	**4.65**	4.89	4.74	4.67
Exams 543	Average	5.26	5.12	5.12	5.10	4.97
Sessions 32	Worst	6.26	6.17	5.44	5.35	5.69
	Std. Dev	0.27	0.32	0.14	0.17	0.22
CAR-S-91	Best	5.64	5.45	**5.39**	5.45	5.42
Exams 682	Average	5.99	5.84	5.74	5.61	5.79
Sessions 35	Worst	7.11	6.24	6.12	5.94	6.31
	Std. Dev	0.28	0.19	0.15	0.11	0.22
EAR-F-83	Best	**38.83**	42.43	42.49	40.94	42.56
Exams 190	Average	44.61	47.03	46.68	44.66	47.45
Sessions 24	Worst	49.69	51.79	50.21	49.15	55.82
	Std. Dev	2.89	2.69	1.84	2.08	3.25
HEC-S-92	Best	13.47	12.72	12.50	13.55	**12.45**
Exams 81	Average	15.38	15.60	14.18	15.53	14.84
Sessions 18	Worst	18.41	21.21	19.57	19.72	20.07
	Std. Dev	1.19	2.18	1.31	1.37	1.74
KFU-S-93	Best	17.04	15.60	16.89	16.31	**15.32**
Exams 461	Average	18.88	16.77	18.54	18.04	17.32
Sessions 20	Worst	22.56	19.45	21.38	20.26	20.60
	Std. Dev	1.41	0.93	0.97	0.79	1.34
LSE-F-91	Best	13.27	12.53	11.91	12.65	**11.68**
Exams 381	Average	15.06	14.08	13.16	13.71	13.44
Sessions 18	Worst	17.85	19.78	14.18	17.87	17.10
	Std. Dev	1.25	1.59	0.51	1.12	1.37
RYE-F-92	Best	12.18	10.78	11.01	11.20	**10.44**
Exams 486	Average	13.67	12.38	11.98	12.66	11.54
Sessions 23	Worst	15.65	14.88	13.50	15.43	13.23
	Std. Dev	0.86	1.15	0.69	0.87	0.80
STA-F-83	Best	166.43	163.85	163.66	166.28	**162.62**
Exams 139	Average	182.64	170.27	173.34	179.00	170.06
Sessions 13	Worst	192.73	174.99	186.71	194.74	176.77
	Std. Dev	6.92	2.69	5.28	8.34	3.24
TRE-S-92	Best	9.57	**9.33**	9.62	9.57	9.43
Exams 261	Average	10.63	10.06	10.51	10.06	9.95
Sessions 23	Worst	11.92	12.09	11.53	10.67	10.89
	Std. Dev	0.58	0.55	0.44	0.30	0.35
UTA-S-92	Best	3.95	3.67	3.71	3.71	**3.60**
Exams 622	Average	4.53	4.16	3.94	3.98	4.07
Sessions 35	Worst	5.52	5.65	4.35	4.35	5.38
	Std. Dev	0.41	0.49	0.16	0.15	0.42
UTE-S-92	Best	30.87	**28.63**	29.96	29.73	28.66
Exams 184	Average	33.54	30.90	32.73	32.64	31.05
Sessions 10	Worst	38.69	37.29	36.14	35.33	36.52
	Std. Dev	1.98	1.95	1.59	1.31	1.58
YOR-F-83	Best	43.87	**43.21**	44.09	44.96	44.15
Exams 181	Average	47.28	46.77	47.77	47.07	46.91
Sessions 21	Worst	52.75	50.66	50.29	50.37	50.16
	Std. Dev	1.99	1.71	1.39	1.21	1.54

Table 6.11: Experimental results for two heuristic orderings applied simultaneously

Data Set	**Results from Chapter 5**									
	Fuzzy LD+LE Model	*Fuzzy SD+LE Model*	*Fuzzy SD+LD Model*	*Fuzzy LD+LCD Model*	*Fuzzy SD+LCD Model*	*Fuzzy LE+LCD Model*	*Fuzzy LE+WLD Model*	*Fuzzy LCD+WLD Model*	*Fuzzy SD+WLD Model*	*Fuzzy LD+WLD Model*
CAR-F-92	4.57	4.47	4.62	4.57	4.72	4.51	4.85	4.48	**4.46**	4.62
CAR-S-91	5.45	5.31	5.45	5.29	5.37	5.28	5.78	5.26	**5.20**	5.29
EAR-F-83	38.80	**36.99**	39.34	38.36	40.53	38.55	37.44	39.09	38.06	38.26
HEC-S-92	12.09	12.03	12.69	12.28	12.18	11.89	12.02	11.59	11.59	**11.46**
KFU-S-93	15.73	15.90	16.09	16.19	16.55	15.04	17.43	15.03	**14.77**	16.33
LSE-F-91	11.97	12.16	14.22	12.26	12.12	11.87	13.55	11.92	11.49	**11.29**
RYE-F-92	13.02	10.25	13.40	10.80	10.63	10.26	11.46	9.85	**9.84**	10.42
STA-F-83	159.82	**159.59**	165.25	161.97	160.87	159.72	165.94	159.72	160.80	159.69
TRE-S-92	8.99	8.92	9.26	8.95	9.20	8.90	9.05	8.89	**8.76**	8.87
UTA-S-92	3.77	3.55	3.73	3.63	3.72	3.56	3.92	3.52	**3.51**	3.59
UTE-S-92	28.59	27.99	30.37	28.43	29.66	**27.27**	28.45	27.43	28.08	28.26
YOR-F-83	41.10	**40.71**	43.00	41.25	42.87	40.84	40.91	41.62	41.41	41.98

Table 6.12: Experimental results for three heuristic orderings applied simultaneously

	Results from Chapter 5									
Data Set	ThreeHO-1	ThreeHO-2	ThreeHO-3	ThreeHO-4	ThreeHO-5	ThreeHO-6	ThreeHO-7	ThreeHO-8	ThreeHO-9	ThreeHO-10
CAR-F-92	4.52	4.64	4.71	4.52	4.42	4.51	4.42	4.52	**4.38**	4.42
CAR-S-91	5.24	5.49	5.42	5.48	5.27	5.26	5.22	5.26	**5.19**	5.20
EAR-F-83	37.11	38.85	38.93	37.62	37.53	37.74	37.30	37.64	**36.57**	37.29
HEC-S-92	11.71	12.16	12.57	11.92	11.64	11.72	11.74	11.61	11.72	**11.52**
KFU-S-93	15.34	14.79	15.99	15.28	15.49	15.54	15.37	14.92	**14.58**	14.61
LSE-F-91	11.43	12.05	12.82	12.27	11.55	**11.30**	11.43	11.62	11.63	11.43
RYE-F-92	10.30	10.35	11.18	10.81	9.86	10.32	9.81	9.89	9.82	**9.71**
STA-F-83	159.15	159.65	159.51	158.87	159.17	159.38	158.47	159.16	158.72	**158.31**
TRE-S-92	8.64	8.76	8.91	8.92	**8.59**	8.71	8.76	8.62	8.62	8.78
UTA-S-92	3.55	3.61	3.74	3.66	3.55	3.62	3.54	**3.49**	3.51	3.52
UTE-S-92	27.64	27.81	28.78	28.65	27.45	27.37	27.13	27.24	27.13	**27.03**
YOR-F-83	40.68	41.41	42.72	41.34	41.16	41.72	**40.15**	40.89	40.45	41.17

ThreeHO-1 = *Fuzzy LD+SD+LE Model*
ThreeHO-2 = *Fuzzy SD+LE+LCD Model*
ThreeHO-3 = *Fuzzy LD+SD+LCD Model*
ThreeHO-4 = *Fuzzy LD+LE+LCD Model*
ThreeHO-5 = *Fuzzy WLD+SD+LD Model*
ThreeHO-6 = *Fuzzy WLD+LD+LE Model*
ThreeHO-7 = *Fuzzy WLD+LD+LCD Model*
ThreeHO-8 = *Fuzzy WLD+SD+LE Model*
ThreeHO-9 = *Fuzzy WLD+SD+LCD Model*
ThreeHO-10 = *Fuzzy WLD+LE+LCD Model*

6.3.2 Discussion of Results

The experimental results of this research serves to confirm earlier research by Burke *et al.* (1998a) with regard to the non-deterministic factor that can aid with finding better timetable solutions. Generally, in terms of proximity cost, it can be observed that better solutions were obtained in this Chapter compared to the solutions produced in Chapter 4 and Chapter 5. Either these heuristic orderings are used on their own or they are combined by means of fuzzy reasoning. Across all experiments, the most notable single heuristic ordering was *WLD*. Using *WLD* either on its own or combining it with other heuristic ordering produced considerably better results than the other heuristic orderings. When *WLD* was applied on its own, as shown in the preliminary investigation results (see Tables 6.9 and 6.10), it outperformed other single heuristic orderings in both types of time slot selection.

Concerning the multiple heuristic ordering, it can be seen that eight of the best results in Table 6.11 and all the best results in Table 6.12 were produced when *WLD* is included as one of the heuristic orderings that constitute the performed multiple heuristic ordering combinations. *LE* appears to be the second 'best' single heuristic ordering. An interesting point is that both *WLD* and *LE* are static heuristic orderings that rely on the number of enrolments in each exam. Taking into account that *LE* was the 'best' in Chapter 4 and Chapter 5, these observations would seem to suggest that the number of students enroled in each exam is a good feature to use as a heuristic ordering in the context of the problem instances and penalty function that have been used here. Although utilising *LCD* on its own only produced comparable results, it is worth highlighting that better timetable solutions were produced when *LCD* was used simultaneously with other heuristic orderings. In Table 6.12, implementing *Fuzzy WLD+SD+LCD Model* and *Fuzzy WLD+LE+LCD Model* obtained four 'best' results each, and *Fuzzy WLD+LD+LCD Model* produced one 'best' result.

Tables 6.13 and 6.14 are referred to in order to analyse the effects of using two different types of time slot selection (i.e. last time slot or random time slot) when utilising single heuristic orderings. For both tables, the lowest value amongst the results for each data set is considered as the 'best'. In Table 6.13 it can be seen that 'best' results for nine out of the twelve data sets were obtained when the random time slot selection was implemented. However, in terms of average penalty (see Table 6.14), utilising the last time slot selection lead to lower average penalty cost for ten data sets compared to only two by the random time slot selection. One possible reason for this is that more variations of timetable solutions might be found in the larger search space that needs to be explored when time slots are selected in random. This is demonstrated by the higher standard deviations of the results of the thirty runs, that can be observed if we compare the *Std. Dev.* values shown in Tables 6.9 and 6.10, in which most the largest value for *Std. Dev.* for each data set are found in Table 6.10. Because of the non-deterministic factor, there is no guarantee that the timetable solution will always be constructed with good quality. Careful investigations of Table 6.13 also shows that it is not always the case that utilising the single heuristic ordering with random time slot selection will produce better solutions compared to the use of last time slot selection. This happens in the following cases:

- utilising *WLD* for *EAR-F-83*, *RYE-F-92* and *TRE-S-92*
- utilising *LE* for *TRE-S-92* and *YOR-F-83*
- utilising *LD* for *CAR-F-92*

The fact that most of the 'best' results shown in Table 6.13 were produced when the single heuristic ordering *WLD* or *LE* is utilised, would seem to suggest that the non-deterministic approach is not capable of finding a good quality solution without applying an appropriate heuristic ordering for the particular problem. This means that the success of the non-deterministic approach is dependent on the heuristic ordering applied. Con-

Table 6.13: A comparison of 'best' penalty cost for the five single heuristic orderings. The lowest value are highlighted with bold font.

Data Set	Heuristic Ordering(last time slot)					Heuristic Ordering(random time slot)				
	LD	*LE*	*SD*	*LCD*	*WLD*	*LD*	*LE*	*SD*	*LCD*	*WLD*
CAR-F-92	4.89	4.74	5.12	5.11	4.74	4.91	**4.65**	4.89	4.74	4.67
CAR-S-91	5.86	5.64	5.97	5.56	5.56	5.64	5.45	**5.39**	5.45	5.42
EAR-F-83	39.90	45.57	45.42	41.45	38.85	**38.83**	42.43	42.49	40.94	42.56
HEC-S-92	14.56	13.36	13.70	14.13	12.77	13.47	12.72	12.50	13.55	**12.45**
KFU-S-93	17.64	16.23	18.33	17.61	17.65	17.04	15.60	16.89	16.31	**15.32**
LSE-F-91	13.98	13.25	12.76	13.55	12.55	13.27	12.53	11.91	12.65	**11.68**
RYE-F-92	12.34	10.80	11.51	11.56	**9.85**	12.18	10.78	11.01	11.20	10.44
STA-F-83	167.05	172.01	177.93	169.58	172.01	166.43	163.85	163.66	166.28	**162.62**
TRE-S-92	10.45	**9.25**	10.50	10.02	**9.25**	9.57	9.33	9.62	9.57	9.43
UTA-S-92	3.97	3.71	4.11	3.88	3.63	3.95	3.67	3.71	3.71	**3.60**
UTE-S-92	35.19	28.93	33.72	31.28	29.59	30.87	**28.63**	29.96	29.73	28.66
YOR-F-83	45.72	**42.65**	46.74	46.31	44.19	43.87	43.21	44.09	44.96	44.15

Table 6.14: A comparison of average penalty cost for the five single heuristic orderings. The lowest value are highlighted with bold font.

Data Set	Heuristic Ordering(last time slot)					Heuristic Ordering(random time slot)				
	LD	*LE*	*SD*	*LCD*	*WLD*	*LD*	*LE*	*SD*	*LCD*	*WLD*
CAR-F-92	5.14	**4.86**	5.28	5.34	4.99	5.26	5.12	5.12	5.10	4.97
CAR-S-91	6.15	6.02	5.97	5.66	5.79	5.99	5.84	5.74	**5.61**	5.79
EAR-F-83	42.31	49.63	45.42	42.21	**41.35**	44.61	47.03	46.68	44.66	47.45
HEC-S-92	16.29	14.73	15.06	15.68	14.67	15.38	15.60	**14.18**	15.53	14.84
KFU-S-93	18.69	**16.59**	18.83	18.57	18.55	18.88	16.77	18.54	18.04	17.32
LSE-F-91	16.13	14.19	**12.76**	14.78	13.13	15.06	14.08	13.16	13.71	13.44
RYE-F-92	13.65	12.35	11.51	11.96	**10.30**	13.67	12.38	11.98	12.66	11.54
STA-F-83	**167.84**	172.26	178.83	170.75	172.21	182.64	170.27	173.34	179.00	170.06
TRE-S-92	11.22	9.70	10.50	10.44	**9.66**	10.63	10.06	10.51	10.06	9.95
UTA-S-92	4.84	4.12	4.11	3.92	**3.76**	4.53	4.16	3.94	3.98	4.07
UTE-S-92	35.22	**29.32**	35.07	31.28	30.42	33.54	30.90	32.73	32.64	31.05
YOR-F-83	47.49	**44.58**	48.32	49.37	46.46	47.28	46.77	47.77	47.07	46.91

sidering multiple heuristic orderings, it is obvious that better results were produced as compared to the single heuristic ordering. While implementing *Fuzzy WLD+SD+LCD Model* and *Fuzzy WLD+LE+LCD Model* with non-deterministic time slot selection appears promising, it is difficult to determine which fuzzy multiple heuristic ordering is the most prominent. As mentioned above, the observation that the success of the random approach is down to the heuristic ordering chosen, might be applied in multiple heuristic

Table 6.15: A comparison of 'best' results obtained in Chapter 4, Chapter 5 and this Chapter

Data Set	Chapter 4	Chapter 5	This Chapter
CAR-F-92	4.54	4.47	**4.38**
CAR-S-91	5.29	5.21	**5.19**
EAR-F-83	37.02	36.99	**36.57**
HEC-S-92	11.78	11.70	**11.46**
KFU-S-93	15.80	15.41	**14.58**
LSE-F-91	12.09	11.43	**11.29**
RYE-F-92	10.38	10.21	**9.71**
STA-F-83	160.42	159.34	**158.31**
TRE-S-92	8.67	8.64	**8.59**
UTA-S-92	3.57	3.55	**3.49**
UTE-S-92	28.07	27.64	**27.03**
YOR-F-83	**39.80**	40.46	40.15

ordering as well. On that basis, it is expected that using *Fuzzy WLD+SD+LCD Model* and *Fuzzy WLD+LE+LCD Model* within the sequential constructive algorithm with last time slot selection (without randomisation) will produce better quality solutions.

Table 6.15 compares the 'best' results obtained by three different algorithms (with a variation of single and multiple heuristic ordering combinations). Overall, the 'best' results produced in this Chapter have outperformed all 'best' results that were produced earlier in Chapter 4 and Chapter 5. Note that, the 'best' results for *HEC-S-92* and *LSE-F-91* data sets were produced using two heuristic orderings (i.e. *Fuzzy LD+WLD Model*). Again, this shows that the number of heuristic orderings *and* which heuristic orderings are considered simultaneously in measuring the difficulty of scheduling exams will affect the performance of the construction algorithm.

6.4 Alternative Approaches to Tuning the Fuzzy System

Up to this point, all the fuzzy systems have featured tuning of the membership functions, while the fuzzy rules have been fixed. In this section, a series of experiments are presented to explore the influences of tuning the fuzzy rules. Basically, two approaches were implemented. Firstly, a simple enumerative rule tuning process was implemented and, secondly, a stochastic approach to generating a fuzzy system to measure the difficulty of scheduling exams to time slots was developed (see Section 6.4.2 for more details).

6.4.1 Tuning Fuzzy Rules with Fixed Membership Functions

The objective of these experiments was to investigate whether tuning the fuzzy rules would offer any improvement in performance over the predefined fixed set of fuzzy rules. For this purpose, the membership functions identified in experiments reported in Chapter 5, specifically Table 5.4 were implemented as fixed membership functions for the respective data sets. The sequential constructive algorithm (*ALG2.0*) explained in Section 5.2.1 was applied. As the fuzzy multiple heuristic ordering that considered three heuristic ordering simultaneously (i.e *Fuzzy LD+SD+LE Model*) was studied, the fuzzy rules set shown in Table 5.2 was used as the benchmark fuzzy rules set.

In order to help with understanding the fuzzy rules tuning process, Table 5.2 has been reproduced with more details as shown in Table 6.16. The number in the cell represents the rule number. In the tuning process, the only modification was in the consequence part of each rule, one at a time in sequence from *Rule 1* to *Rule 27* (as numbered in Table 6.16). The antecedent part of each rule remained the same. As described in Section 4.4.1.2, there were five possible values for each rule consequence: *very small*, *small*, *medium*, *high* and *very high*. Beside these five values, one additional value, *not_inuse* was added to represent the non existence of the rule. If the *not_inuse*

Table 6.16: Fuzzy rule set for *Fuzzy LD+LE+SD Model*

LE	***LD***								
	S			*M*			*H*		
	SD			***SD***			***SD***		
	S	*M*	*H*	*S*	*M*	*H*	*S*	*M*	*H*
S	S^{1}	VS^{4}	VS^{7}	S^{10}	S^{13}	VS^{16}	M^{19}	S^{22}	S^{25}
M	S^{2}	S^{5}	VS^{8}	H^{11}	M^{14}	M^{17}	H^{20}	M^{23}	M^{26}
H	H^{3}	S^{6}	S^{9}	H^{12}	M^{15}	M^{18}	VH^{21}	H^{24}	M^{27}

VS=very small
S=small
M=medium
H=high
VH=very high

value was assigned to the consequence part of a rule, that meant the rule was not applicable. For each rule, its consequence part was changed by assigning one of the six possible values in the sequence of *not_inuse*, *very small*, *small*, *medium*, *high* and *very high*; one at a time. Considering 27 fuzzy rules and six possible values that can be assigned to the consequence part of each rule, there are 162 possible sets of fuzzy rules. For each set, the tuned fuzzy rules were tested over three runs. Initially, the total number of the fuzzy rules was 27. However, the number of rules might be reduced if, by removing any of the rules, solution quality improved.

Note that, by changing the values of the consequence part of a rule, the consistency or completeness of the set of rules might be affected. In evaluating any set of rules, the output of the fuzzy system is set to zero if none of the rules is fired for a certain input value. That means that, for any exam with *LD*, *LE* or *SD* values that cannot be handled by the proposed set of fuzzy rules, the weight of difficulty of scheduling the exam is set to zero. Such an exam will be considered as not difficult to be scheduled and therefore it will be given a lower priority in terms of the sequence of processing the exams. Recall that the main purpose of applying the fuzzy technique in *Process 1* is to measure the difficulty of scheduling the exams, where the sequence of scheduling which exams might affect the overall scheduling process. As a result, any set of fuzzy rules that produces an 'inappropriate' exams ordering (where some exams result in zero difficulty due to an incompleteness in the set of rules) will simply guide the scheduler towards a lower quality timetable. As the main objective of this experiment is to improve the initial fuzzy model for *Fuzzy LD+SD+LE Model* (implementing the fuzzy rules shown in Table 5.2 and membership functions shown in Table 5.4), any new fuzzy model (with new tuned fuzzy rules) that produces a worse performance compared to the initial fuzzy model will be rejected.

At the end of the experiment, for each data set, there were 162 sets of fuzzy rules with corresponding timetable solutions. The fuzzy rules set with the lowest penalty cost were selected as the 'best' sets of fuzzy rules for the specific data sets. Two experiments were conducted:

- *Tuned Fuzzy Rules 1*— The 'best' set of tuned fuzzy rules that improved the current solution quality was kept and used as the initial set of fuzzy rules for the next set of tuned fuzzy rules. A simple deepest descent enumerative search algorithm was employed in this experiment. The pseudo-code is shown in Figure 6.3.

- *Tuned Fuzzy Rules 2*— Each of the rules was changed in isolation; no changes made in the earlier iterations was taken into account. In this experiment, whenever the consequence part of any rule was changed, the fuzzy rules were reinitialised to the initial set of fuzzy rules as shown in Table 6.16, before moving to the next iteration. The pseudo-code is shown in Figure 6.4.

6.4.2 Randomly Generated Fuzzy Models

The aim of this experiment was to examine alternative approaches for implementing *Process 1* (described in Chapter 4). Instead of using fixed fuzzy models (either fixed membership functions or fixed fuzzy rules) for the particular combination of heuristic ordering, a non-deterministic approach to define the fuzzy model was utilised. Each step in *Process 1* is now performed randomly. In order to make the experiment more man-

```
DECLARE INTEGER rulesCode[50][4]  // Holds rules code for 50 rules of 4 variables
DECLARE DOUBLE penalty  // Penalty cost for the adjacent exams for a new timetable
DECLARE DOUBLE proximityCost
DECLARE INTEGER consequence[6]  ← {0,1,2,3,4,5} //Consequence code of a rule
//where 0=`not-inuse'; 1=`very small'; 2=`small';3=`medium';4=`high';5=`very high'

proximityCost ← 9999.0
ruleCode ← initiliase the fuzzy rules using the fixed fuzzy rules (as shown in Table 6.16)

For i = 1 to 27  // For fuzzy rule number start from 1 to 27
        For j =  0 to 5 // For consequence value represent by 0, 1, 2, 3, 4 and 5, in turn
                // Copy the current consequence part value into a temporary variable
                tempCode ← ruleCode[i][4]
                // Change the consequence part of rule i
                ruleCode[i][4] ←  consequence[j]

                Construct a timetable using the fuzzy model with the new set of fuzzy rules
                penalty ← calculate penalty cost of the constructed timetable
                If (penalty < proximityCost)
                        proximityCost ← penalty
                        Write fuzzy model into a file
                        Write timetable into a file
                Else
                        Reset  rulesCode ruleCode[i][4] ←  tempCode
                End If
        End For
End For
```

Figure 6.3: Pseudo-code for *Tuned Fuzzy Rules 1*

ageable, only fuzzy multiple heuristic orderings that combine three heuristic orderings from the five available single heuristic orderings - LD, SD, LE, LCD and WLD, were considered. This was based upon the previous observations, in which most of the 'best' results were produced when three heuristic ordering were considered simultaneously.

In the implementation, the first step was to randomly select three heuristic orderings to be considered simultaneously. The next step was to create a set of fuzzy rules for the chosen heuristic orderings, also selected in random fashion. Any rule should contain at least one antecedent, and the maximum is three antecedents. The last step was to choose cp points for membership functions for all of the fuzzy variables. As a fuzzy system with

```
DECLARE INTEGER rulesCode[50][4]  // Holds rules code for 50 rules of 4 variables
DECLARE DOUBLE penalty  // Penalty cost for the adjacent exams for a new timetable
DECLARE DOUBLE proximityCost
DECLARE INTEGER consequence[6]  ← {0,1,2,3,4,5} //Consequence code of a rule
//where 0=`not-inuse'; 1=`very small'; 2=`small';3=`medium';4=`high';5=`very high'

proximityCost ← 9999.0
ruleCode ← initiliase the fuzzy rules using the fixed fuzzy rules (as shown in Table 6.16)

For i = 1 to 27  // For fuzzy rule number start from 1 to 27
        For j =  0 to 5 // For consequence value represent by 0, 1, 2, 3, 4 and 5, in turn
                // Change the consequence part of rule i
                ruleCode[i][4] ←  consequence[j]

                Construct a timetable using the fuzzy model with the new set of fuzzy rules
                penalty ← calculate penalty cost of the constructed timetable
                If (penalty < proximityCost)
                        proximityCost ← penalty
                        Write fuzzy model into a file
                        Write timetable into a file
                End If
                ruleCode ← reinitiliase the fuzzy rules using the fixed fuzzy rules (as shown in Table 6.16)
        End For
End For
```

Figure 6.4: Pseudo-code for *Tuned Fuzzy Rules 2*

three inputs and one output was implemented, four *cp* points were randomly chosen. The integer values used to represent the heuristic orderings and fuzzy rules are shown in Table 6.17.

An example is represented graphically in Figure 6.5 to show how the random fuzzy model was developed. In *STEP 1*, the three heuristic orderings chosen are identified as *LE*, *SD* and *WLD*. Based on these heuristic orderings, the randomly generated rules were translate into '*IF ... THEN ...*' form. The rules were represented in a two dimensional array. Each row of the array represented one rule. In each row, the first column corresponded to the antecedent for the first heuristic ordering, the second column corresponded to the antecedent value for the second heuristic and the third column to the value for the third heuristic; the last column corresponded to the consequence part (i.e *examweight*). In the example, three rules were randomly generated and their translated form are given. Note that *Rule 2* only consisted of two antecedents as *SD* was set to

Table 6.17: Integer codes assigned to fuzzy model parameters

Heuristic	*LD*	*LE*	*SD*	*LCD*	*WLD*
Heuristic Code	1	2	3	4	5

Antecedent linguistic variable	not_inuse	small	medium	high
Antecedent Code	0	1	2	3

Consequence linguistic variable	not_inuse	very small	small	medium	high	very high
Consequence Code	0	1	2	3	4	5

not_inuse (antecedent code = 0). These fuzzy rules generations were performed without concerning the logical order of the rule — any rule could be accepted even if it contrasted with the basic knowledge of heuristic ordering. Considering the membership functions, *STEP 3* shows the four *cp* points that are randomly picked and the related membership function graphical representations is given. Again, the first three elements of the array correspond to the membership functions for the three chosen heuristic ordering in the sequence order; while the last element represents the *cp* point for *examweight*.

To evaluate this non-deterministic approach using fuzzy model tuning, two experiments were performed as follows:

- *Random Model 1* — Experiments were performed for 100 iterations for each data set. In each iteration, a new fuzzy model was created by randomly choosing the heuristic orderings, 27 fuzzy rules and the four *cp* points for the membership functions. Each fuzzy model was tested three times within the sequential constructive algorithm. The pseudo-code is shown in Figure 6.6.
- *Random Model 2* — Experiments were conducted for 1000 iterations for nine data sets (*EAR-F-83*, *HEC-S-92*, *KFU-S-93*, *LSE-F-91*, *RYE-F-92*, *STA-F-83*, *TRE-S-92*, *UTE-S-92* and *YOR-F-83*), while for *CAR-F-92*, *CAR-S-91* and *UTA-S-92*, the experiments were run for 100 iterations. For this experiment, the

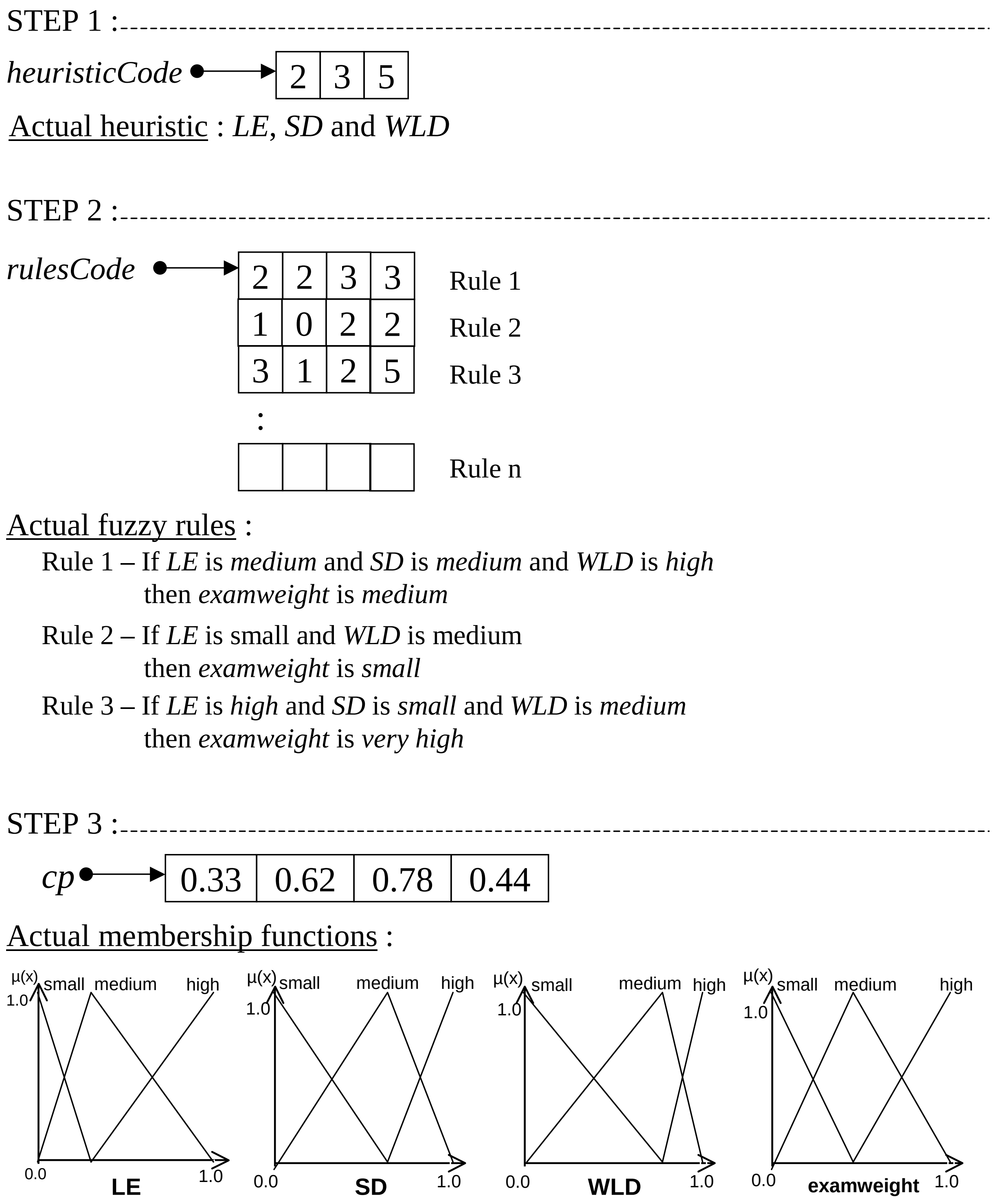


Figure 6.5: An example of defining a random fuzzy model

heuristic orderings and *cp* points were randomly chosen only once for each data set. Initially the fuzzy rules set was empty. In the first iteration, a fuzzy rule was randomly created and set as the first rule. Having created the fuzzy model, the sequential constructive algorithm was run three times. The best timetable constructed was set as a benchmark. For each of the remaining iterations, a fuzzy rule was randomly created and appended to the end of the list of rules. The sequential constructive algorithm was then run with the new fuzzy model (i.e. only the rules were changed). The rules were kept if a better solution was produced with the new fuzzy model, and the new best solution was then set as the new benchmark. Otherwise, the newly added rule was removed. This process continued until the number of iteration exceeded the maximum number of iterations allowed for the particular data set. The pseudo-code is shown in Figure 6.7.

In both experiments, non-applicable rules (rule with consequence part assigned to *not_inuse* or rule with all the antecedents part were assigned to *not_inuse*) were removed. Because the fuzzy rules were randomly selected, in the case of experiment with *Random Model 1*, it was possible to have a fuzzy model that contains less than 27 rules. Meanwhile, in the case of experiments with *Random Model 2*, the issue of completeness of the fuzzy rule is relevant because the experiment starts with only one fuzzy rule. Therefore, care must be taken when evaluating the fuzzy rules. During the experiment, any randomly generated set of rules will be tested — including a set of rules which is inconsistent and incomplete. As described in Section 6.4, in such a situation, the fuzzy system will simply set the exam difficulty to zero if none the rules is fired for the certain input values.

6.4.3 Testings and Results

Table 6.18 shows a comparison of the results obtained using fixed and tuned fuzzy rules. The first column indicates the penalty cost for the timetable solution of each data set that has been constructed with a set of fixed fuzzy rules (extracted from the sixth column of

```
DECLARE INTEGER heuristicCode[3]  // Holds heuristics code
DECLARE INTEGER cp[4]  // Holds cp points for the membership functions
DECLARE INTEGER rulesCode[27][4]  // Holds rules code for 27 rules of 4 variables
DECLARE DOUBLE penalty  // Penalty cost for the adjacent exams for a new timetable
DECLARE INTEGER maxLoop // Number of iteration
DECLARE DOUBLE proximityCost

proximityCost ← 9999.0
maxLoop ← 100

For i = 1 to maxLoop
        heuristicCode ← randomly choose 3 heuristis
        rulesCode ← randomly choose 27 rules
        cp ← randomly choose 4 cp points
        For j = 1 to 3
                Construct a timetable using the randomly generated fuzzy model
                penalty ← calculate penalty cost of the constructed timetable
                If (penalty < proximityCost)
                        proximityCost ← penalty
                        Write fuzzy model into a file
                        Write timetable into a file
                End If
        End For
End For
```

Figure 6.6: Pseudo-code for *Random Model 1*

Table 5.6). In the next two columns, the qualities of the timetable solutions produced by using the sequential constructive algorithm with *Tuned Fuzzy Rules 1* and *Tuned Fuzzy Rules 2* tuning approaches are given. It can be seen that in all data sets, better solutions were produced by tuning the fuzzy rules (either by *Tuned Fuzzy Rules 1* or *Tuned Fuzzy Rules 2*), compared to the approach that only used fixed fuzzy rules. The results show that tuning the fuzzy rules has produced considerably better timetable solutions.

In the previous Chapter, it was demonstrated that combining three heuristic orderings produced better solutions compared to combining two heuristic orderings. However, in two cases (*CAR-F-92* and *EAR-F-83*), two heuristic orderings outperformed three heuristic orderings. It was argued that, this can be rectified if the fuzzy rules were tuned. Indeed, as can be observed, the *EAR-F-83* data set now has a penalty cost equal

```
DECLARE INTEGER heuristicCode[3]  // Holds heuristics code
DECLARE INTEGER cp[4]  // Holds cp points for the membership functions
DECLARE INTEGER rulesCode[50][4]  // Holds rules code for 50 rules of 4 variables
DECLARE INTEGER newRule[4]  // A new rule code
DECLARE DOUBLE penalty  // Penalty cost for the adjacent exams for a new timetable
DECLARE INTEGER maxLoop // Number of iteration
DECLARE DOUBLE proximityCost, ruleCounter, examSize

proximityCost ← 9999.0
ruleCounter ← 0
If (examSize > 500)
        maxLoop ← 1000
Else
        maxLoop ← 100
End If

heuristicCode ← randomly choose 3 heuristis
cp ← randomly choose 4 cp points
For i = 1 to maxLoop
        newRule ← randomly create a new rule
        ruleCounter ←  ruleCounter + 1
        rulesCode[ruleCounter][] ← append newRule at the end of the set of rules
        Construct a timetable using the randomly generated fuzzy model
        penalty ← calculate penalty cost of the constructed timetable
        If (penalty < proximityCost)
                proximityCost ← penalty
                Write fuzzy model into a file
                Write timetable into a file
        Else
                Reset  rulesCode[ruleCounter][] ← {0,0,0,0}
                ruleCounter ←  ruleCounter - 1
        End If
End For
```

Figure 6.7: Pseudo-code for *Random Model 2*

to 36.16. This penalty cost value is smaller than the penalty cost incurred when the *Fuzzy SD+LE Model* was used — i.e. 36.99. Although the result produced by the *Fuzzy SD+LE Model* model for the *CAR-F-92* (in the fourth column of Table 5.6) still outperformed the result obtained in this experiment, overall the results indicate the potential of expanding the tuning of the fuzzy model to include tuning the fuzzy rules.

Table 6.18: A comparison of results for *Fuzzy LD+SD+LE Model* when utilising fixed and tuned fuzzy rules

Data Set	Fixed Fuzzy Rules	*Tuned Fuzzy Rules 1*	*Tuned Fuzzy Rules 2*
CAR-F-92	4.53	**4.51**	**4.51**
CAR-S-91	5.21	**5.19**	**5.19**
EAR-F-83	37.11	**36.16**	36.64
HEC-S-92	11.70	11.61	**11.60**
KFU-S-93	15.41	**15.34**	**15.34**
LSE-F-91	11.43	**11.35**	**11.35**
RYE-F-92	10.21	**10.02**	10.05
STA-F-83	159.34	**159.09**	160.79
TRE-S-92	8.64	8.62	**8.47**
UTA-S-92	3.55	**3.52**	**3.52**
UTE-S-92	27.64	27.64	**27.55**
YOR-F-83	40.46	**39.25**	39.79
Total	335.23	332.30	334.8

Table 6.19 compares the results obtained by the experiments outlined in Section 6.4.2 to the results produced by the experiments explained in Section 6.4.1 and the 'best' results of using three heuristic orderings (from Table 6.12). This comparison is on the basis that all these results were produced when three heuristic orderings were simultaneously implemented. However, note that the algorithms used in the experiments are slightly different. While the results in the second column were produced with random time slot selection (details described in Section 6.3), results for the remaining three columns were obtained with last time slot selection (i.e. *ALG2.0* — as implemented in Chapter 5). In Table 6.19 the best results across all experiments for each data set is highlighted in bold font. It can be seen that the best results for seven data sets were produced by fuzzy models that featured fixed fuzzy rules and tuned membership functions (see the first column); while the best result for three data sets were with tuned fuzzy rules (see the second column). Although experiments which applied *Random Model 1* did not pro-

Table 6.19: A comparison of results for tuning fuzzy model randomly

Data Set	Best results of combining three heuristic ordering (from Table 6.12)	Best results of tuning fuzzy rules only (from Table 6.18)	*Random Model 1*	*Random Model 2*
CAR-F-92	4.38	4.51	4.59	**4.32**
CAR-S-91	**5.19**	**5.19**	5.58	5.54
EAR-F-83	36.57	**36.16**	40.93	37.05
HEC-S-92	**11.52**	11.60	12.55	12.31
KFU-S-93	**14.58**	15.34	15.74	15.03
LSE-F-91	**11.30**	11.35	12.58	12.65
RYE-F-92	**9.71**	10.02	10.58	9.75
STA-F-83	**158.31**	159.09	159.22	158.64
TRE-S-92	8.59	**8.47**	9.24	8.79
UTA-S-92	**3.49**	3.52	3.69	4.31
UTE-S-92	**27.03**	27.55	29.77	29.10
YOR-F-83	40.15	**39.25**	43.88	42.30
Total	330.82	332.05	348.35	339.79

duce any best results, the experiments that used *Random Model 2* produced one best result. The best result for *CAR-F-92* shown in the fourth column was obtained using the following fuzzy model (which was randomly created):

- heuristic orderings : *LCD*, *LE* and *SD*
- *cp* points for membership functions : 0.550, 0.110, 0.296, and 0.132
- number of fuzzy rules : 16

Taking into account that the fuzzy model is defined in random fashion, this best result was found in an arbitrary fashion. One possible reason why only one best result was found in the experiments that applied the random fuzzy model is due to the fact that the number of iterations in the experiments (100 for *Random Model 1* and 1000 for *Random Model 2*) was quite small when compared to the huge search space that needs to be explored in order to find the 'optimal' fuzzy model.

Taking a different view, it could be stated that tuning membership functions and fuzzy rules at different stages is better than tuning both membership functions and fuzzy rules at the same time with the non-deterministic approach. Note that three of the 'best' results in Table 6.19 (i.e. for *EAR-F-83*, *TRE-S-92* and *YOR-F-83*) were obtained when the *Fuzzy LD+SD+LE Model* was applied with fixed membership functions and tuned fuzzy rules. Therefore, it can be expected that the solutions presented in the second column (the results produced with fixed fuzzy rules) may be improved by tuning the fuzzy rules with the membership functions that have been identified in the initial set of experiments performed earlier (explained in Section 6.3). It also worthy of mention that only 16 rules are required to produce the solution. This indicates that not all possible rules are required to be embedded in the system in order to get a better solution. With fewer rules, the fuzzy model is more understandable for the developer and user. Therefore, a more sophisticated optimisation approach should be devised to tackle the tuning process more systematically.

6.5 Chapter Summary

Several issues regarding the generalisation of the proposed multiple heuristic orderings have been explored in this Chapter. Firstly, the issue of the applicability of the proposed approach to a different timetabling problem. The experimental results obtained when the fuzzy multiple heuristic ordering was implemented on course timetabling problems suggests that the proposed approach may be suitable for generalisation to other domains.

Secondly, work was presented on exploring all possible combinations of two and three heuristic orderings based upon the five single heuristics. The experimental results on the range of benchmark examination timetabling data sets showed that the use of *WLD* as one of the variables in the fuzzy heuristic ordering combinations leads to better solutions in most of the problem instances. Due to the non-deterministic factor, it is difficult to determine which heuristic ordering combination is superior amongst all the possible heuristic ordering combinations. Furthermore, earlier work (i.e. when each of the single heuristic orderings was implemented on its own) on implementing random time slot selection indicates that the method of time slot selection and best heuristic ordering method is inter-dependent.

Finally, alternative approaches for tuning the fuzzy models were developed and evaluated. The results obtained demonstrated that tuning the fuzzy rules has improved the chances of constructing better solutions. Given that only a simple enumerative search was implemented to modify the rules on the fixed membership functions and randomly create fuzzy models, it is possible that more sophisticated optimisation techniques will improve the search for the 'optimal' fuzzy model.

Part III

Fuzzy Evaluation

Chapter 7

A Novel Fuzzy Approach to Evaluate the Examination Timetabling

This chapter introduces a new fuzzy evaluation function for examination timetabling. Fuzzy reasoning is employed to evaluate the quality of a constructed timetable by considering two criteria, namely the average penalty per student and the highest penalty imposed on any of the students. A fuzzy system was created based on a series of easy to understand rules featuring the combination of these two criteria. A significant problem encountered was how to determine the lower and upper bounds of the decision criteria for any given problem instance, in order to allow the fuzzy system to be fixed and, hence, applicable to new problems without alteration. In this work, two different methods for determining boundary settings are proposed. Experimental results are presented and the implications analysed. These results demonstrate that fuzzy reasoning can be successfully applied to evaluate the quality of timetable solutions by simultaneously taking into consideration multiple decision criteria.

7.1 Introduction

Previous studies such as Asmuni *et al.* (2005) and Petrovic *et al.* (2005), demonstrated that fuzzy reasoning is a promising technique that can be used both for modeling timetabling problems and for constructing solutions. These studies indicated that the utilisation of fuzzy methodologies in university timetabling is an encouraging research topic. In this Chapter, a new evaluation function is introduced that is based on fuzzy methodologies. The research focuses on evaluating the constructed timetable solutions by considering two decision criteria. Although the constructed timetable solutions were developed based on the specific objectives mentioned above, the method is general in the sense that a user could, in principle, define additional criteria he or she wished to be taken into account in evaluating any constructed timetables. This research is motivated by the fact that, in practice, the quality of the timetable solution is usually assessed by the timetabling officer considering several criteria/objectives.

A brief description of the existing evaluation methods is discussed in Chapter 2. In the next section, the drawbacks of existing evaluation methods is presented, followed by a detailed explanation of the proposed novel approach. Section 7.3 presents descriptions of the experiments carried out and the results obtained, followed by discussions in Section 7.3.3. Finally, some concluding comments and future research directions are given in Section 7.4.

7.2 Assessing Timetable Quality

7.2.1 Disadvantages/Drawbacks of Current Evaluation Functions

In the evaluation function shown in Equation 2.4, it can be seen that the final value of the proximity cost penalty function is a measure only of the average penalty per student. Although this penalty function has been widely used by many researchers in the context of uncapacitated problem of the benchmark data set (Carter *et al.*, 1996), in practice, considering only the average penalty per student is not sufficient to evaluate the quality of the constructed timetable. The final value does not, for example, represent the relative fairness of spreading out each student's schedule. For example, when examining the resultant timetable, it may be the case that a few students have an examination timetable in which many of their exams are scheduled in adjacent time slots. These students are likely to not be happy with their timetable, as they will not have enough time to prepare adequately. On the other hand, the remaining students enjoy a 'good' examination timetable.

As a specific example, consider the following two cases. *Case 1*: there are 100 students with each student having a penalty cost of one; *Case 2*: there are 100 students, but now ten students have a penalty cost of ten, the rest zero. In both cases the average penalty per student is equal to one, but obviously the solution in *Case 2* is 'worse' than the solution in *Case 1*.

One of the co-supervisors of this thesis (McCollum) has extensive experience of real-world timetabling, having spend 12 years as a timetabling officer and with continuing links with the timetabling industry. It is his experience that 'proximity cost' is not the only factor considered by timetabling officers when evaluating the quality of an actual timetable in practice. Usually, a timetable evaluation is based on several factors, several of which are subjective and/or based on ambiguous information. Furthermore, to the best of the author's knowledge, all the evaluation functions mentioned in Section 2.3.2 are integrated into the timetabling construction process. These objective functions are used to measure the satisfaction of specific soft constraints. This means that the timetable solution is optimised against these soft constraints. In practice, the user may consider other criteria in evaluating the final timetable solution *after* the solution has been arrived at. A review of other objective functions that have been proposed and which have been used in timetable optimisation were given in Section 2.3.2.

One way to handle multiple criteria decision making is to use simple linear combinations of the various criteria. This works by multiplying the value of each criterion by a constant weighting factor and summing to form an overall result. Each weight represents the relative important of each criterion compared to the other criteria. As with the case in Section 4.3, there is usually no simple way to determine the precise values for these weights, especially weights that can be used across several problem instances with different complexity.

In this Chapter, a new evaluation function utilising fuzzy methodologies is introduced. Basically, the idea is to develop an independent evaluation function that can be used to measure the quality of any examination timetable solution. This evaluation function may be based on more than one criterion. The timetable can have been generated using any (single or multi-objective) approach, featuring any construction and/or iterative improvement. Subsequently, the timetable solution with the problem description and the list of criteria that need to be evaluated are submitted to the evaluation function. Hence, the methods presented in Chapters 7 and 8 represent a form of *multi-criteria evaluation* of timetables carried out on constructed timetables, and are *not* a form of multi-objective optimisation.

7.2.2 The Proposed Fuzzy Evaluation Function

As an initial investigation, this proposed approach was implemented on solutions which were generated based on the proximity cost requirements (*average penalty*). Once generated, one additional criterion other than the average penalty per student, namely the highest penalty that occurred amongst the students (*highest penalty*) was also taken into account in the evaluation. There is no specific or external reason why this criterion was chosen, other than the fact that it was felt that this was likely to be a factor which is taken into account (particularly by the students themselves) in the real-world. It would also appear to be quite general and fairly uncontentious, in the sense that minimising the maximum penalty for any one student (however that penalty is derived) would seem to be a reasonable thing to do, in addition to minimising the average of the same penalty function over all students. Once again, it is emphasised that the latter factor was *only* considered *after* the timetable solution was constructed. That is to say, there was no attempt to include this criterion in the process of constructing the timetable. Such a process (which would involve, in the terminology adopted in this thesis, turning the fuzzy evaluation function into a fuzzy objective function) might be an interesting avenue for future research.

A fuzzy system with these two input variables (*average penalty* and *highest penalty*) and one output variable (*quality*) was constructed. Each of the input variables were associated with three linguistic terms; fuzzy sets corresponding to a meaning of *low*, *medium* and *high*. In addition to these three linguistic terms, the output variable (*quality*) has two extra terms that correspond to meanings of *very low* and *very high*. These terms were selected as they were deemed the simplest possible to adequately represent the problem. Gaussian functions of the form $e^{-(x-c)^2/2\sigma^2}$, where c and σ are constants representing the centre and width of the fuzzy set respectively (see Figure 7.1), were used to define the fuzzy set for each linguistic term. As shown in Figure 7.1, σ_k should be the width between the central point c_k and a value on the x-axis for which the membership function has value 0.5 (so-called cross-over value). The standard Gaussian membership function always has its peak value at one.

As this experiment aimed to move towards mimicking human decision making, smooth function were required. Thus, Gaussians were selected on the basis that they are the simplest and most common choice, given that smooth, continuously varying functions were desired, particularly in the context of modelling human reasoning.

The membership functions defined for the two inputs, *average penalty* and *highest penalty*, and the output *quality* are depicted in Figure 7.2 (a) – (c), respectively. For such

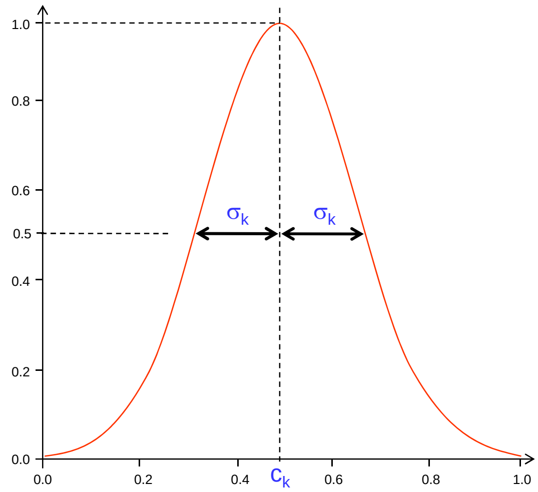


Figure 7.1: Gaussian membership function for $\mu(x_k, \sigma_k)$

a system with two inputs with three linguistic terms, there are nine possible fuzzy rules that can be defined in which each input variable is associated with one linguistic term. As already known, from the definition of proximity cost, the objective is to minimise the penalty cost — i.e. the lower the penalty cost, the better the timetable quality. Also, based on everyday experience, the highest penalty for any one student should be as low as possible, as this will create a fairer timetable for all students. Based upon this knowledge, a fuzzy rule set was defined consisting of all nine possible rule combinations. Each rule connects the input variables to the single output variable, *quality*. The fuzzy rule set is presented in Figure 7.3. As stated previously, standard Mamdani style fuzzy inference was used to obtain the fuzzy output for a given set of inputs. The Centre of Gravity defuzzification method described in Section 3.1.5.1 was then used to obtain a single crisp (real) value for the output variable. This single crisp output was then taken as the *quality* of the timetable.

7.2.3 Input Normalisation

With this proposed fuzzy evaluation function, a set of experiments was carried out to determine whether the fuzzy evaluation system was able to distinguish a range of

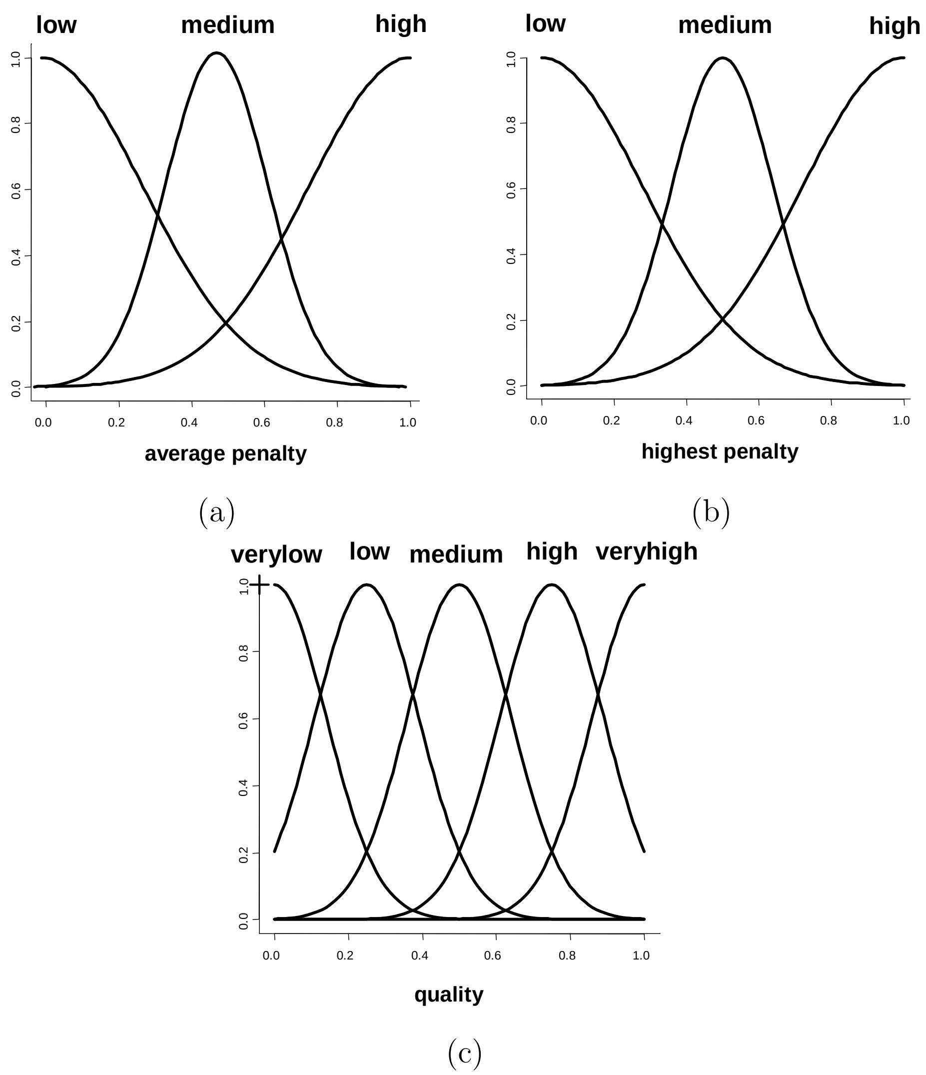


Figure 7.2: Membership functions for input and output variables

timetable solutions based on the average penalty per student and the highest penalty imposed on any of the students. All the constructed timetables for the given problem instance were evaluated using the same fuzzy system, and their quality determined based on the output of the fuzzy system. The constructed timetable with the biggest output value was selected to be the 'best' timetable.

Based on previous experience outlined in Chapters 4 to 6, the average penalty values for different data sets result in widely different scales due to the different complexity of the problem instances. For example, in the *STA-F-83* data set an average penalty of

Rule 1: IF (*average penalty* is *low*) AND (*highest penalty* is *low*)
THEN (*quality* is *very high*)
Rule 2: IF (*average penalty* is *low*) AND (*highest penalty* is *medium*)
THEN (*quality* is *high*)
Rule 3: IF (*average penalty* is *low*) AND (*highest penalty* is *high*)
THEN (*quality* is *medium*)
Rule 4: IF (*average penalty* is *medium*) AND (*highest penalty* is *low*)
THEN (*quality* is *high*)
Rule 5: IF (*average penalty* is *medium*) AND (*highest penalty* is *medium*)
THEN (*quality* is *medium*)
Rule 6: IF (*average penalty* is *medium*) AND (*highest penalty* is *high*)
THEN (*quality* is *low*)
Rule 7: IF (*average penalty* is *high*) AND (*highest penalty* is *low*)
THEN (*quality* is *medium*)
Rule 8: IF (*average penalty* is *high*) AND (*highest penalty* is *medium*)
THEN (*quality* is *low*)
Rule 9: IF (*average penalty* is *high*) AND (*highest penalty* is *high*)
THEN (*quality* is *very low*)

Figure 7.3: Fuzzy rules for *Fuzzy Evaluation Function*

160.42 was obtained, whereas for *UTA-S-92*, the average penalty was 3.57 (these values are extracted from the second column of Table 4.9).

As can be seen in Figure 7.2(a) and Figure 7.2(b), the input variables have their universe of discourse defined between 0.0 and 1.0. Therefore, in order to use this fuzzy model, both of the original input variables must be normalised within the range [0.0, 1.0]. The initial transformation used was as follows:

$$v' = \frac{(v - lowerLimit)}{(upperLimit - lowerLimit)} \tag{7.1}$$

where v is the actual value in the initial range $[lowerLimit, upperLimit]$. In effect, the range $[lowerLimit, upperLimit]$ represents the actual lower and upper boundaries for the fuzzy linguistic terms.

By applying this normalisation technique, the same fuzzy model can be used for any problem instance, either for the benchmark data sets as used here, or for a new real-world problem. This would provide flexibility when problems of various complexity are presented to the fuzzy system. In such a scheme, the membership functions do not need to be changed from their initial shapes and positions. In addition, rather than recalculate the parameters for each input variable's membership functions, it is much easier to transform the crisp input values into normalised values in the range of $[0.0, 1.0]$. The problem thus becomes one of finding suitable lower and upper limits for each problem instance.

7.3 Preliminary Investigations

7.3.1 Experiments Setup

In order to test the fuzzy evaluation system, the Carter *et al.*'s (1996) benchmark data sets were used again (see Table 2.1). For each instance of the twelve data sets, 40 timetable solutions were constructed using a simple sequential constructive algorithm with backtracking, as previously described in Chapter 4. Eight different heuristics were used to construct the timetable solutions; for each of which the algorithm was run five times to obtain a range of solutions. However, due to the nature of the heuristics used, in some cases, a few of the constructed timetable solutions had the same proximity cost value. Therefore, for the purpose of standardization, 35 different timetable solutions were selected out of the 40 constructed timetable solutions, by firstly removing any repeated solution instances and then just removing at random any excess. The objective was to obtain a set of timetable solutions with variations of timetable solution quality, in which none of the solutions had the same quality in terms of proximity cost (i.e average penalty per student). The timetable solutions were constructed by implementing the following heuristics:

- Three different single heuristic orderings:
  - Least Saturation Degree First (*SD*),
  - Largest Degree First (*LD*),
  - Largest Enrolment First (*LE*),
- Three different fuzzy multiple heuristic orderings:
  - a *Fixed Fuzzy LD+LE Model*,
  - a *Tuned Fuzzy LD+LE Model*, and
  - a *Tuned Fuzzy SD+LE Model* (see Chapter 4 for details of these),
- random ordering, and
- a deliberately 'poor' ordering (see below).

A specific 'poor' heuristic was utilised in an attempt to purposely construct bad solutions. The idea was to attempt to determine the upper limit of solution quality (in effect, though not formally, the 'worst' timetable for the given problem instance). Basically the method was to deliberately assign student exams in adjacent time slots. In order to construct bad solutions, *LD* was initially employed to order the exams. Next, the exams were sequentially selected from this ordered exams list and assigned to the time slot that caused the highest proximity cost; this process continued until all the exams were scheduled.

The 35 timetable solutions were analysed in order to determine the minimum and the maximum values for both the input variables, *average penalty* and *highest penalty*. These values were then used for the normalisation process (see Section 7.2.3). However, because the twelve data sets have various complexity (see Table 2.1), the determination of the initial range for each data set is not a straight-forward process. Thus, two alternative boundary settings were implemented in order to identify the appropriate set of *lowerLimit* and *upperLimit* for each data set.

The first boundary setting used $lowerLimit = 0.0$ and the $upperLimit = maxValue$, where *maxValue* was the largest value obtained from the set of 35 solutions. However, from the literature, the lowest value yet obtained for the *STA-F-83* data set is around 130 (Casey and Thompson, 2003). Thus, it did not seem sensible to use zero as the lower limit in this case. In order to attempt to address this, the use of a non-zero lower limit was investigated. Of course, a formal method for determining the lower limit for any given timetabling instance is not currently known. Hence, the second boundary setting used *lowerLimit* = *minValue* and *upperLimit* = *maxValue*, where *minValue* was the smallest value obtained from the set of 35 constructed solutions for the respective data set.

In this implementation, both input variables, *average penalty* and *highest penalty*, were independently normalised based on their respective *minValue* and *maxValue*. The fuzzy evaluation system described earlier (see Section 7.2.2) was then employed to evaluate the timetable solutions. The same processes were applied to all of the data sets listed in Table 2.1. The fuzzy evaluation system was implemented using the 'R' language (*The R Foundation for Statistical Computing Version 2.2.0*) (R Development Core Team, 2005).

7.3.2 Experimental Results

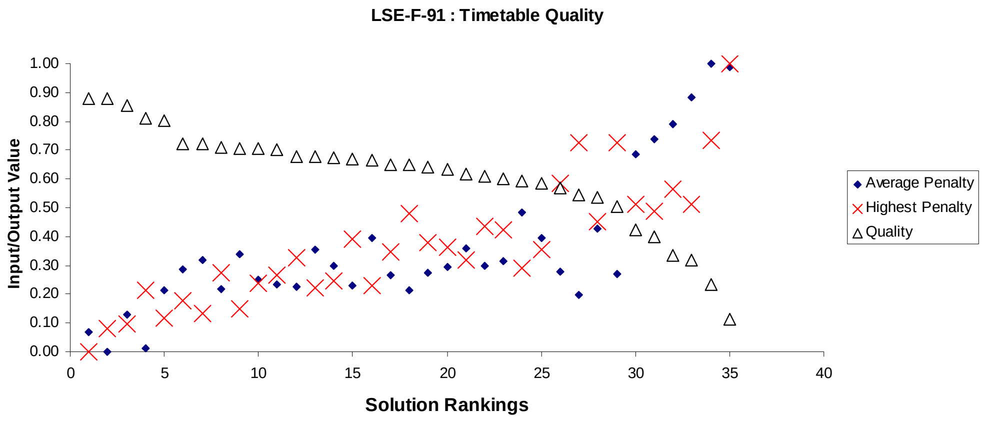
In this Section, the experiment results are presented. Table 7.1 shows the minimum and maximum values obtained for both evaluation criteria (the input variables). Figures 7.4(a) and 7.4(b) show the evaluation results obtained by the fuzzy evaluation system for the *LSE-F-91* and *TRE-S-92* data sets. These two data sets are shown as representative examples chosen at random. Both graphs show the results obtained when the boundary setting $[minValue, maxValue]$ was implemented. In the graph, the x-axis (Solution Rankings) represents the ranking of the timetable solution quality evaluated by using the fuzzy evaluation function; in order from the best solution to the worst solution. The y-axis represents the normalised input values (*average penalty* and *highest penalty*) and the output values (*quality*) obtained for the particular timetable solution.

These two graphs show that the fuzzy evaluation function has performed as desired, in that the overall (fuzzy) quality of the solutions varies from close to zero to close to one.

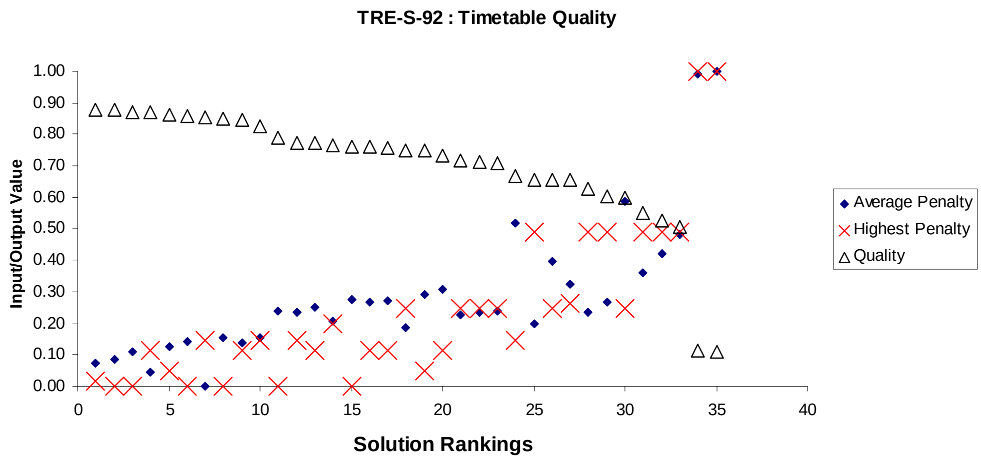
Tables 7.2 – 7.4 show a comparison of the results obtained using the two alternative forms of the normalisation process. The *Solution Number* is used to identify a particular solution within the 35 timetable solutions used in the experiments for each data set, where *Solution Number* is assigned based on the ranking by *average penalty* (i.e. the solution with lowest *average penalty* is labelled *Solution Number* 1, the next lowest 2, etc.). In both tables, the fifth and sixth columns (labeled as 'Range $[minValue, maxValue]$') indicates the fuzzy evaluation value and the rank of the solution relative to the other solutions, when the boundary range $[minValue, maxValue]$ was used. The last two columns in the tables show the evaluation values and solution ranking obtained when the boundary range $[0, maxValue]$ was used. Only the first ten 'best' timetable solutions for each of the data sets are presented, based on the ranking produced when the boundary range $[minValue, maxValue]$ was used.

Table 7.1: Minimum and maximum values for *Average Penalty* and *Highest Penalty* obtained from the 35 timetable solutions for each data set

Data Set	*Average Penalty*		*Highest Penalty*	
	Minimum Value	Maximum Value	Minimum Value	Maximum Value
CAR-F-92	4.54	11.42	65.0	132.0
CAR-S-91	5.29	13.33	68.0	164.0
EAR-F-83	37.02	71.28	105.0	198.0
HEC-S-92	11.78	31.88	75.0	136.0
KFU-S-93	15.81	43.40	98.0	191.0
LSE-F-91	12.09	32.38	78.0	191.0
RYE-F-92	10.38	36.71	87.0	191.0
STA-F-83	160.75	194.53	227.0	284.0
TRE-S-92	8.67	17.25	68.0	129.0
UTA-S-92	3.57	8.79	63.0	129.0
UTE-S-92	28.07	56.34	83.0	129.0
YOR-F-83	39.80	64.48	228.0	331.0



(a) *LSE-F-91*



(b) *TRE-S-92*

Figure 7.4: Indicative illustrations of the range of normalised inputs and associated output obtained for the *LSE-F-91* and *TRE-S-92* data sets

Table 7.2: A comparison of the results obtained using the two alternative forms of the normalisation process for data sets *CAR-F-92*, *CAR-S-91*, *EAR-F-83* and *HEC-S-92*

Data Set	Timetable Criteria			Range[minValue, maxValue]		Range[0, maxValue]	
	Solution Number	Average Penalty	Highest Penalty	Evaluation Value	Solution Ranking	Evaluation Value	Solution Ranking
CAR-F-92	1	4.544	65	0.888503	1	0.534427	1
	2	4.624	71	0.876804	2	0.517946	2
	3	4.639	71	0.876791	3	0.517485	3
	4	4.643	71	0.876788	4	0.517366	4
	5	5.148	68	0.876583	5	0.510084	5
	6	5.192	69	0.873279	6	0.506692	6
	8	5.508	68	0.858276	7	0.500729	7
	9	5.532	68	0.856617	8	0.500120	8
	11	5.595	68	0.851966	9	0.498538	9
	12	5.609	68	0.850863	10	0.498184	10
CAR-S-91	1	5.292	68	0.888524	1	0.557585	1
	2*	5.573	75	0.880205	2	0.537593	3
	7*	5.911	68	0.879621	3	0.542750	2
	3	5.654	75	0.879244	4	0.535472	4
	6	5.842	75	0.875877	5	0.530812	5
	10*	6.079	76	0.868161	6	0.523516	8
	11*	6.393	71	0.860211	7	0.526116	6
	13*	6.509	71	0.853145	8	0.523572	7
	4	5.688	83	0.850233	9	0.520297	9
	5	5.690	83	0.850227	10	0.520255	10
EAR-F-83	1	37.018	116	0.868135	1	0.467867	1
	4*	41.860	118	0.834883	2	0.444700	3
	6*	43.637	105	0.827016	3	0.454672	2
	7	44.147	118	0.798099	4	0.432416	4
	3	41.324	131	0.748303	5	0.415267	5
	5*	43.628	129	0.733864	6	0.411292	7
	9*	44.968	127	0.718542	7	0.411481	6
	18	49.662	114	0.710776	8	0.392966	8
	2*	41.178	144	0.699109	9	0.370814	11
	10*	44.980	135	0.674252	10	0.385906	9
HEC-S-92	1	11.785	83	0.863057	1	0.506506	1
	10	14.774	75	0.854699	2	0.495547	2
	2	13.236	84	0.853706	3	0.489407	3
	5*	14.162	83	0.847966	4	0.482514	5
	7*	14.635	83	0.838633	5	0.477754	7
	6*	14.217	85	0.832653	6	0.476641	8
	13*	15.594	78	0.828916	7	0.481021	6
	17*	15.911	75	0.817611	8	0.485117	4
	15	15.763	84	0.801080	9	0.463727	9
	4*	14.124	94	0.727535	10	0.446459	11

Table 7.3: A comparison of the results obtained using the two alternative forms of the normalisation process for data sets *KFU-S-93*, *LSE-F-91*, *RYE-F-92* and *STA-F-83*

Data Set	Timetable Criteria			Range[minValue, maxValue]		Range[0, maxValue]	
	Solution Number	Average Penalty	Highest Penalty	Evaluation Value	Solution Ranking	Evaluation Value	Solution Ranking
KFU-S-93	1	15.813	98	0.888529	1	0.541211	1
	7	16.904	101	0.884358	2	0.526210	2
	10	17.336	100	0.883340	3	0.524294	3
	11	17.920	104	0.876034	4	0.513226	4
	22*	20.022	102	0.852341	5	0.501383	11
	2*	16.463	113	0.847871	6	0.509402	5
	3*	16.471	113	0.847868	7	0.509339	6
	4*	16.500	113	0.847858	8	0.509119	7
	5*	16.500	113	0.847858	9	0.509119	8
	6*	16.500	113	0.847858	10	0.509119	9
LSE-F-91	3*	13.458	78	0.881499	1	0.552817	2
	1*	12.094	87	0.879126	2	0.555747	1
	4*	14.720	89	0.855424	3	0.523229	4
	2*	12.349	102	0.812127	4	0.527563	3
	6	16.408	91	0.804048	5	0.504874	5
	17*	17.942	98	0.722929	6	0.480142	7
	22*	18.564	93	0.720053	7	0.481747	6
	8*	16.486	109	0.707889	8	0.476028	9
	23*	18.979	95	0.707212	9	0.474395	11
	12*	17.174	105	0.704871	10	0.476479	8
RYE-F-92	1	10.384	87	0.888528	1	0.610225	1
	7	12.180	97	0.871582	2	0.558378	2
	10	12.337	97	0.870489	3	0.556102	3
	8	12.264	98	0.868672	4	0.555205	4
	12	12.976	97	0.864830	5	0.547756	5
	11	12.417	102	0.854386	6	0.545595	6
	6	12.094	105	0.839576	7	0.544225	7
	16*	13.678	104	0.831331	8	0.527428	12
	23*	14.441	104	0.817334	9	0.519821	14
	24*	14.581	104	0.814229	10	0.518513	15
STA-F-83	1	160.746	227	0.888536	1	0.215426	1
	2	161.151	227	0.887829	2	0.214107	2
	3	164.375	228	0.871792	3	0.202156	3
	4	167.394	227	0.824391	4	0.196779	4
	5	168.195	227	0.805614	5	0.194967	5
	7	168.863	227	0.788882	6	0.193535	6
	6*	168.781	232	0.788385	7	0.182500	17
	8*	169.100	227	0.782864	8	0.193043	7
	10*	171.249	227	0.733062	9	0.188900	8
	11*	171.391	227	0.730410	10	0.188645	9

Table 7.4: A comparison of the results obtained using the two alternative forms of the normalisation process for data sets *TRE-S-92*, *UTA-S-92*, *UTE-S-92* and *YOR-F-83*

Data Set	Timetable Criteria			Range[minValue, maxValue]		Range[0, maxValue]	
	Solution Number	Average Penalty	Highest Penalty	Evaluation Value	Solution Ranking	Evaluation Value	Solution Ranking
TRE-S-92	3*	9.311	69	0.880078	1	0.478231	2
	4*	9.389	68	0.878204	2	0.479078	1
	5	9.598	68	0.871588	3	0.475325	3
	2*	9.039	75	0.868946	4	0.468005	6
	6*	9.757	71	0.864316	5	0.465758	8
	8*	9.885	68	0.858365	6	0.469941	4
	1*	8.671	77	0.855435	7	0.469016	5
	10*	10.003	68	0.851293	8	0.467596	7
	7	9.856	75	0.846708	9	0.454514	9
	9*	9.981	77	0.826007	10	0.446743	11
UTA-S-92	1	3.567	63	0.888536	1	0.532771	1
	2	3.833	68	0.878185	2	0.511100	2
	3	3.911	68	0.876019	3	0.508369	3
	4	3.927	68	0.875482	4	0.507798	4
	5	3.977	68	0.873738	5	0.506065	5
	6	4.143	68	0.866816	6	0.500466	6
	8	4.531	73	0.807693	7	0.475697	7
	9	4.573	73	0.802872	8	0.474319	8
	10	4.581	73	0.801938	9	0.474053	9
	13	4.976	68	0.762605	10	0.472232	10
UTE-S-92	6	30.323	83	0.879116	1	0.438284	1
	4	29.718	86	0.878651	2	0.429775	2
	1	28.069	90	0.853031	3	0.420748	3
	17	32.804	88	0.835146	4	0.400981	4
	11	31.522	91	0.826953	5	0.392480	5
	20	33.935	91	0.780095	6	0.378000	6
	23	34.928	90	0.767341	7	0.377994	7
	18*	32.996	94	0.758297	8	0.367082	9
	3*	29.695	98	0.723270	9	0.369027	8
	8	30.555	98	0.721926	10	0.362837	10
YOR-F-83	1	39.801	234	0.883004	1	0.372139	1
	2*	44.158	233	0.837983	2	0.363036	3
	3*	44.412	231	0.831362	3	0.365581	2
	4	45.645	228	0.791749	4	0.359602	4
	6	45.736	238	0.785008	5	0.345675	5
	10	46.810	234	0.751639	6	0.341781	6
	12	46.862	235	0.749650	7	0.340088	7
	15	47.142	240	0.736830	8	0.330597	8
	14*	46.947	244	0.731929	9	0.324728	10
	19*	47.396	242	0.726141	10	0.324908	9

7.3.3 Discussion

The fuzzy system presented here provides a mechanism to allow an overall decision in evaluating the quality of a timetable solution to be made based on common sense rules that encapsulate the notion that the timetable solution quality increases as both the *average penalty* and the *highest penalty* decrease. The rules are in a form that is easily understandable by any incumbent timetabling officer.

Looking at Figures 7.4(a) and 7.4(b) it can be seen that, in many cases, it is not guaranteed that timetable solutions with low *average penalty* will also have low *highest penalty*. This observation confirmed the assumption that considering only the proximity cost to measure timetable solution quality is not sufficient. As an example, if the detailed results obtained for the $[0, maxValue]$ boundary range for *LSE-F-91* in Table 7.3 are analysed, it can be seen that solution 1 (with the lowest *average penalty*) is not ranked as the 'best' solution by the fuzzy evaluation. The same effect can be observed for the *TRE-S-92* data set (see Table 7.4) and for the *UTE-S-92* data set in Table 7.4.

In these three data sets (*LSE-F-91*, *TRE-S-92* and *UTE-S-92*), the timetable solutions with the lowest *average penalty* were not evaluated as the 'best' timetable solution, because the decision made by the fuzzy evaluation system also takes into account another criterion, the *highest penalty*. This finding can also be seen in the other data sets, but it is not so obvious especially if only the first three 'best' solutions are focussed on. Regardless of this, in terms of functionality, these results indicate that the fuzzy evaluation system has performed as intended in measuring the timetable's quality by considering two criteria simultaneously.

Analysing Tables 7.2 – 7.4 further, it can also be observed that the decision made by the fuzzy evaluation function is affected slightly when the different boundary settings are used to normalise the input values. The consequence of this is that the same timetable solution might be ranked in a different order, dependent on the boundary conditions. In Tables 7.2 – 7.4, solutions in which the different boundary settings have resulted in different ranking position are marked with *. For the *CAR-F-92* (in Table 7.2) and *UTA-S-92* data sets (in Table 7.4), the solution rankings are unchanged by altering the boundary settings. In several cases, the solution rankings are only changed slightly. It is also interesting to note that, in a few cases, for example solution 22 for *KFU-S-93* (in Table 7.3) and solution 6 for *STA-F-83* (in Table 7.3), the ranking change is quite marked.

Overall, the performance of the fuzzy evaluation system utilizing the boundary range $[0.0, maxValue]$ did not seem as satisfactory as when the boundary range $[minValue, maxValue]$ was used. This observation is highlighted by Table 7.5, which presents the fuzzy quality measure obtained for the 'worst' and 'best' solutions as evaluated under the two different boundary settings.

When the boundary range $[0.0, maxValue]$ was used, it can be seen that the fuzzy evaluation system evaluated the quality of the timetable solutions for the twelve data sets in the overall range of 0.111464 to 0.610225. In the case of *STA-F-83*, the 'best' solution was only rated as 0.215426 in quality. The quality of timetable solutions falls only in the regions of linguistic terms that correspond to meanings of *very low*, *low* and *medium* in the *quality* linguistic variable (see Figure 7.2(c)). This is because the lower limit value used here (i.e. $lowerLimit = 0.0$) is far smaller than the smallest values observed in practice. Consequently, the input values for even the lowest values (i.e. the 'best' solution qualities) are transformed to normalised values that always fall within the regions of the *medium* and *high* linguistic terms in the input variables. As a result, the normalised input values will not cause any rule to be fired or, the firing level for any rule is relatively very low. This is illustrated in Figure 7.5(a), in which the activation level of the consequent part for **Rule 1** is equal to 0.13. Although the possibility exists for any input to fall into more than one fuzzy set, so that more than one rule can be fired, the aggregation of fuzzy output for all rules will obtain a final shape that will only

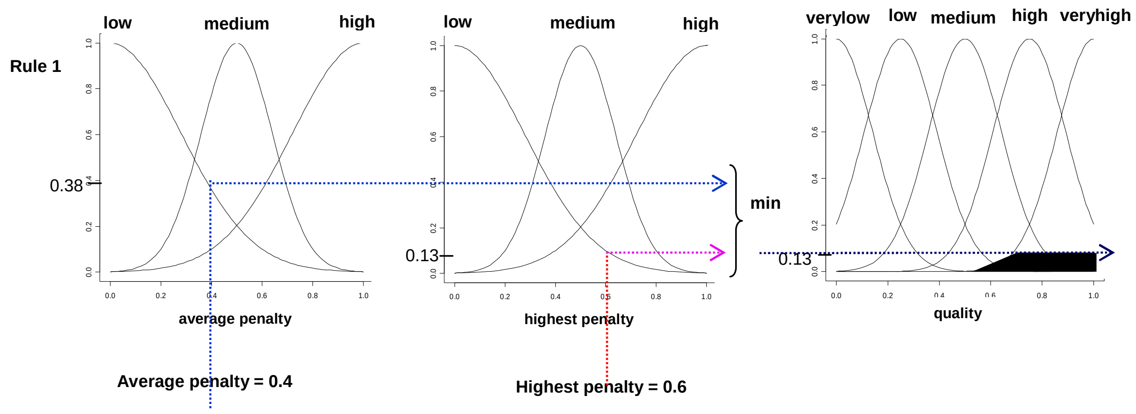
Table 7.5: Range of timetable quality

Data Set	Range $[0, maxValue]$ Worst Solution	Range $[0, maxValue]$ Best Solution	Range $[minValue, maxValue]$ Worst Solution	Range $[minValue, maxValue]$ Best Solution
CAR-F-92	0.111464	0.534427	0.111464	0.888503
CAR-S-91	0.111464	0.557585	0.111464	0.888524
EAR-F-83	0.111465	0.467867	0.111465	0.868135
HEC-S-92	0.127502	0.506506	0.155374	0.863057
KFU-S-93	0.111466	0.541211	0.111466	0.888529
LSE-F-91	0.111895	0.555747	0.112182	0.881499
RYE-F-92	0.115999	0.610225	0.119240	0.888528
STA-F-83	0.111464	0.215426	0.111464	0.888536
TRE-S-92	0.111476	0.479078	0.111488	0.880078
UTA-S-92	0.111464	0.532771	0.111464	0.888536
UTE-S-92	0.111464	0.438284	0.111464	0.879116
YOR-F-83	0.120046	0.372139	0.213388	0.883004

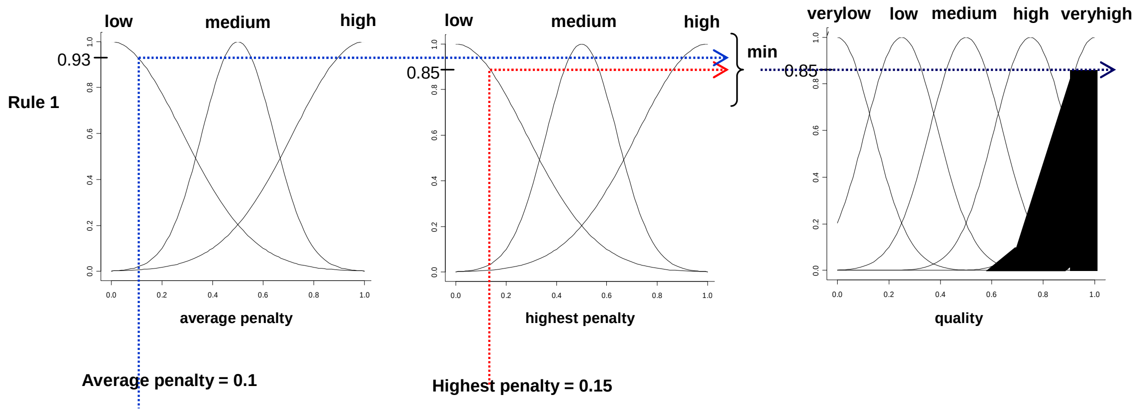
produce a low defuzzification value.

In contrast, Figure 7.5(b) illustrates the situation when the normalised input values fall in the regions of linguistic term that correspond to the meaning of *low*. In this situation, a high defuzzification value will be obtained due to the fact that most of the rules will have a high firing level. Thus, all of the solutions being ranked first had quality values more than 0.8, when the initial range $[minValue, maxValue]$ was used. In this case, the quality of timetable solutions falls in the regions of the linguistic terms that correspond to meanings of *high* and *very high* for the timetable *quality* fuzzy set (see Figure 7.2(c)). As might be expected, from the fact that the actual minimum and maximum values from the 35 constructed timetable solutions were used, the fuzzy evaluation results were nicely distributed along the universe of discourse of the timetable *quality* fuzzy set.

For a clearer comparison of the effect of the two boundary settings, the distribution of input and output values for the *UTA-S-92* data set are presented in Figure 7.6. As can be seen, the input values (Figure 7.6(b) and Figure 7.6(c)) are concentrated in the middle regions $(0.4 - 0.7)$ of the graphs when the boundary range $[0.0, maxValue]$ was



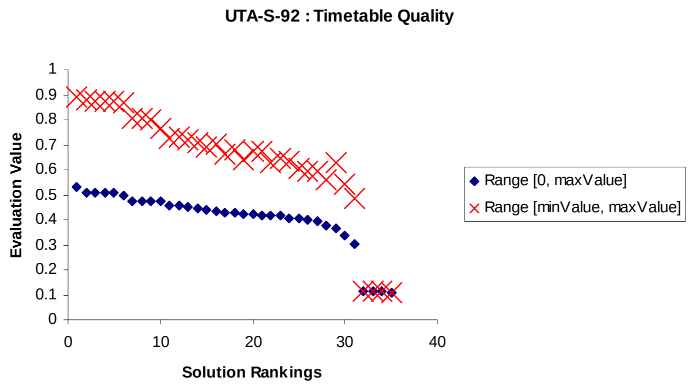
(a) Normalised value falls in the middle regions of the universe of discourse



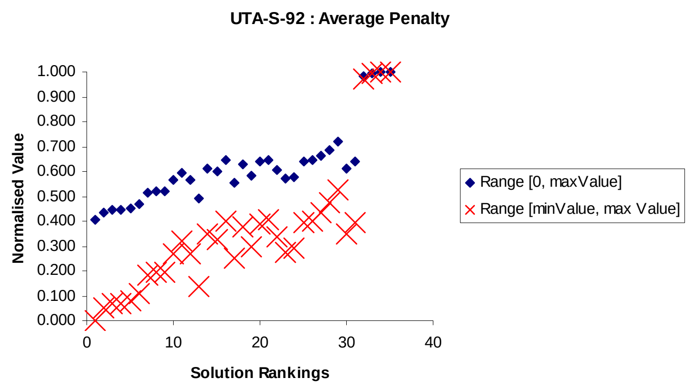
(b) Normalised value falls in the left regions of the universe of discourse

Figure 7.5: Firing level for **Rule 1** with different normalised input values

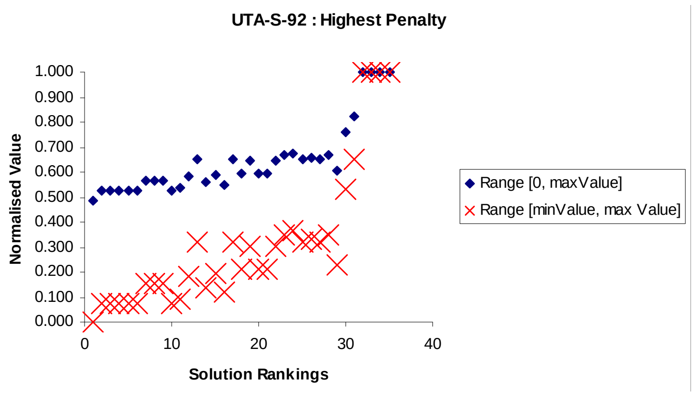
used. In contrast, when the boundary range [$minValue$, $maxValue$] was used, the input values were concentrated in the bottom regions of the graphs. Based upon the defined fuzzy rules, we know that the timetable quality increases with a decrease in both input values. Indeed, this behavior of the output can be observed for both boundary setting (see Figure 7.6(a)). Using either of the boundary settings, the fuzzy evaluation system is capable of ranking the timetable solutions. It is purely a matter of choosing the appropriate boundary settings of the fuzzy sets for the input variables.



(a) Timetable quality



(b) Average penalty



(c) Highest penalty

Figure 7.6: A graphical comparison of the effect of the two boundary settings for *UTA-S-92*

One of the deficiencies of this fuzzy evaluation, at present, appears to be that there is no simple way of selecting the boundary settings of the input variables. The drawback is that both boundary settings implemented so far can only be applied after a number of timetable solutions are generated. Therefore significant amounts of time are required to construct and analyse the solutions. Furthermore, if boundary setting are based on the actual minimum and maximum values from the existing timetable solutions, the fuzzy evaluation system might not be able to evaluate a newly constructed timetable solution if the input values for the decision criteria for the new solution lie outside the range of the fuzzy sets. Actually, output values *can* always be calculated — the real problem is that the resultant solution quality will always be the same once both criteria reach the left-hand boundary of their variables.

7.4 Chapter Summary

In conclusion, the experimental results presented here demonstrate the capability of a fuzzy approach of combining multiple decision criteria in evaluating the overall quality of a given timetable solution. This novel approach, in which fuzzy evaluation is applied to evaluate constructed timetables (as opposed to the objective functions used in solution generation), represents a significant addition to how the majority of current research work decides which is the best solution. It is suggested that this approach may have significant potential for more sophisticated evaluation of a range solutions compared to previous approaches. This could be of significant benefit in the real-world in which timetabling officers subjectively evaluate a range of alternative timetable solutions in order to select the 'best' to be used. The fuzzy evaluation function presented here could be used to support such decision making.

However, in the fuzzy system implementation the selection of the *lowerLimit* and *upperLimit* for the normalisation process is extremely important because it has a significant effect on the overall quality obtained. Thus it would be highly beneficial if approximate boundary settings could be determined, particularly some form of estimate of the lower limit of the assessment criteria, based upon the problem structure itself. In next Chapter, two novel approximation approaches are introduced to determine the boundary setting for *average penalty* and *highest penalty*.

Chapter 8

Determination of Boundary Settings

8.1 Introduction

This Chapter presents novel research into designing and utilising a variety of approaches for determining the boundary settings to be used in the normalisation process (see Section 7.2.3). These boundaries are termed *approximate boundaries* as they are informal lower and upper limits to be used within the normalisation process as opposed to lower and upper *bounds* which have been formally proven. The main feature of these approaches is that the approximate boundaries for *average penalty* and *highest penalty* are calculated merely by analysing the underlying structure of the given problem instance, without the need to construct an actual timetable.

One of the benefit of this approach is that the lower limit and upper limit for the boundary setting can be determined without the need to construct a range of timetable solutions. The other benefit is that the lower limit for proximity cost determined using one of the proposed approaches outlined in this Chapter might indeed represent a lower bound for the proximity cost as used by many researchers in Carter *et al.*'s benchmark data set. As such, it provides an interesting new perspective into how close the best published results on these widely researched benchmark data sets are to the optimum.

In this Chapter, the approach proposed for fuzzy evaluation in the previous Chapter is expanded in order to make the fuzzy system applicable to a wider range of problem solutions (i.e. beyond the solutions generated for system training purposes). In order to achieve this, two new approaches are introduced in which the underlying structure of the problem instances is exploited in order to determine the boundary settings for *average penalty* and *highest penalty* for each data set. With this approach, it will be possible to measure the approximate boundary settings without the need to actually build any actual timetables. The goal is to define boundary settings that have lower limit and upper limit that cover all possible feasible solutions generated by any algorithm or optimisation technique for any particular problem instance. This concept is illustrated graphically in Figure 8.1.

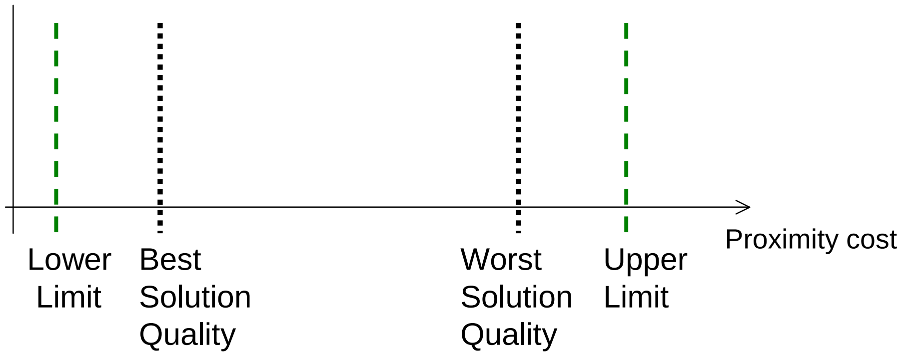


Figure 8.1: Illustration of boundary coverage concept

The benefit of this approach is that it will reduce the need to generate many timetable solutions in order to test the system (even if many solutions are generated, there is still a problem in terms of testing coverage in that the solutions that have been generated may still not be sufficiently representative to determine the lower limit). This gives a distinct advantage if the system is applied to new real-world timetabling problems in which no best and worst solutions are previously available. In the following Sections, two alternative methods for determining approximate boundary settings are explained.

8.2 Approximate Boundaries using Weighting Factors

8.2.1 Approximate Boundaries for *Average Penalty*

The idea is illustrated in Figure 8.2. The first step is to determine the approximate 'average' (or medium) proximity cost (P_{approx}), which will then be multiplied by one constant factor to give an approximate lower limit and multiplied by another constant factor to give an approximate upper limit. In order to do so, it is necessary to calculate the maximum proximity cost (P_{max}) obtainable if the worst timetable was to be constructed. It is assumed that the worst timetable (in terms of proximity cost) is constructed in the situation where every student has all of their enroled exams scheduled in adjacent time slots. In reality, it is not possible to assign all exams enroled by each student in adjacent time slots. This is because it is necessary to consider constraints amongst the exams across students.

However, in the approach presented here, constraints amongst the exams across students will *not* be considered. Only the fact that exams enroled by a student should be scheduled in different time slots will be taken into account. To give an example,

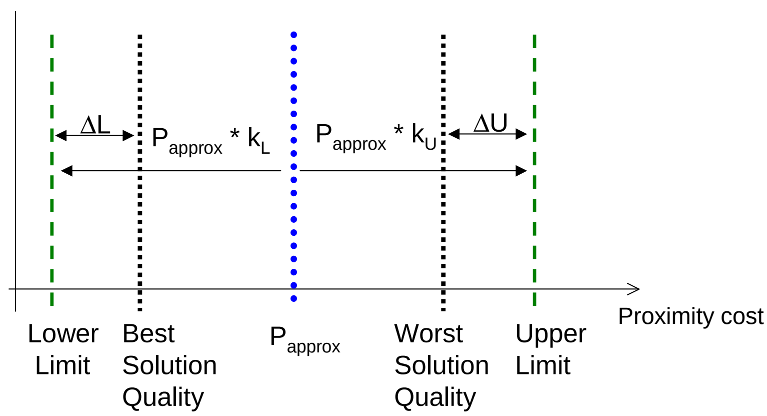


Figure 8.2: Boundary coverage using weighted factors

suppose that *Student1* is enroled in the set of seven exams ($e1$, $e2$, $e3$, $e4$, $e5$, $e6$, $e7$) and that *Student2* is enroled in the set of seven exams ($e10$, $e22$, $e3$, $e34$, $e15$, $e19$, $e70$). Despite the fact that, in reality, both students are enroled in exam $e3$ and hence exam $e3$ will be timetabled at the same time for both students, these two sets of exams are treated as being entirely independent in the calculation of P_{max} here. The pseudo code for calculating P_{max} is shown in Figure 8.3.

In the search for the P_{max}, the number of exams enroled by each student needs to be analysed. By doing so, it can be determined how many students are enroled for any particular number of exams. Then, the penalty imposed on all students having their enroled exams scheduled in adjacent time slots are calculated. As defined in the proximity cost formulation, when a particular student has to sit two exams scheduled t time slots apart, he or she is given a penalty weight of $w_t = 2^{5-t}$ proximity cost, in which the applicable weight values are $w_1 = 16$, $w_2 = 8$, $w_3 = 4$, $w_4 = 2$ and $w_5 = 1$. Note that only exams on the right-hand side (at most five time slots apart — represented by variable *maxTimeSlot*) of the exam that is currently under consideration are involved in the penalty calculation. The penalty that is imposed on the students who enroled in the corresponding number of exams is then calculated (represented by *total1* in Figure 8.3). Finally, the maximum average penalty is obtained by summing all the *total2* values. The maximum proximity cost, P_{max}, obtained when the worst possible timetable is generated, can be obtained using the following formula:

$$maximum\ proximity\ cost,\ P_{max} = \frac{\sum_{i=2}^{R_{max}}(total2_i)}{\sum_{i=1}^{R_{max}}(examCount_i)}, \tag{8.1}$$

where R_{max} is the largest number of exams enroled on by any student, $total2_i$ is the total proximity cost for all students who enroled for i exams, and $examCount_i$ is the number of students who enroled for i exams.

```
DECLARE INTEGER examEnroled  // Number of exams enroled by a student
DECLARE DOUBLE penalty  // Penalty cost for the adjacent exams
DECLARE INTEGER maxStudent  // Total number of student
DECLARE INTEGER maxTimeSlot  // Maximum number of adjacent time slots that penalty will incurred
DECLARE INTEGER Rmax  // Maximum number of exams enroled by any one student
DECLARE INTEGER examCount [Rmax] // Number of students enroled for x exams
DECLARE INTEGER total1[Rmax] // Penalty impose on a student enroled for x exams
DECLARE INTEGER total2[Rmax] // Penalty impose on all the students enroled for x exams
DECLARE INTEGER totalStudent
DECLARE DOUBLE Pmax, totalPenalty

// Traverse the student array in order to read each student record
For s = 1 to maxStudent
    // Get the number of exams enroled by student s
    examEnroled  ← studentArray[s].examEnroled
    // Increase the counter by 1
    examCount[examEnroled] ← examCount[examEnroled] + 1
End For
For i = 1 to Rmax
    If examCount[i] > 0  // If there is at least one student enroled for i exams
        For e = 1 to (i – 1)  // Calculate penalty cost for i adjacent exams
            maxTimeSlot  ←  e + 5
            If (maxTimeSlot  > i)
                maxTimeSlot  ← i
            End If
            For j = (e + 1) to maxTimeSlot
                penalty ← 2  ^  (5 - (j - e))
                total1[i] ← total1[i]  + penalty
            End For
        End For
    End If
    // Accumulate the number of student
    totalStudent ← totalStudent + examCount[i]
    // Multiply the penalty cost for i adjacent exams with number of student enroled for i exams
    total2[i] ←  total1[i] *  examCount[i]
    // Accumulate the penalty cost
    totalPenalty  ←  totalPenalty + total2[i]
End For
// Calculate the approximate value of maximum total penalty
Pmax ← totalPenalty / totalStudent
```

Figure 8.3: Pseudo code for approximation of maximum total penalty, P_{max}

An illustrative example of applying this algorithm to the *LSE-F-91* data set is given in Figure 8.4. Using the enrolment information, the average number of exams enroled on per student can be determined. The formula is as follows:

$$average\ exams\ per\ student,\ E_{avg} = \frac{\sum_{i=1}^{R_{max}} (examCount_i * i)}{\sum_{i=1}^{R_{max}} examCount_i}, \tag{8.2}$$

No of exam	1	2	3	4	5	6	7	8				Min	Max	Avg	Std Dev
No of student	99	80	302	1638	474	103	27	3	2726			1	8	3.97	0.99

Number of Exam	E1	E2	E3	E4	E5	E6	E7	E8	*total1*	Number of Student	*total2*
8		16	8	4	2	1					
			16	8	4	2	1				
				16	8	4	2	1			
					16	8	4	2			
						16	8	4			
							16	8			
								16	191	3	573
7		16	8	4	2	1					
			16	8	4	2	1				
				16	8	4	2				
					16	8	4				
						16	8				
							16		160	27	4320
6		16	8	4	2	1					
			16	8	4	2					
				16	8	4					
					16	8					
						16			129	103	13287
5		16	8	4	2						
			16	8	4						
				16	8						
					16				98	474	46452
4		16	8	4							
			16	8							
				16					68	1638	111384
3		16	8								
			16						40	302	12080
2		16							16	80	1280
1		0							0	99	0
									Total	**2726**	**189376**

Figure 8.4: A graphical illustrations of P_{max} calculations for *LSE-F-91*

Logically, it can be expected that the proximity cost will increase with an increase in the number of exams enroled on by students. By having many exams, it is more difficult to spread out each student's schedule. Therefore, it can be stated that

$$E_{avg} \uparrow \;\Rightarrow\; P_{max} \uparrow$$

Moreover, it is obvious that the fewer the number of time slots (T) available, the higher the proximity cost will be. That is, it is more difficult to spread out each student's schedule when there are only a limited number of time slots available. Thus, it follows that

$$T \uparrow \;\Rightarrow\; P \downarrow$$

Based on these observations, the following formulation can be used for an approximation of proximity cost:

$$approximate\ proximity\ cost,\ P_{approx} = \frac{(P_{max})(E_{avg})}{(T)} \tag{8.3}$$

Having calculated the approximate value of proximity cost (P_{approx}), the final step is to multiply P_{approx} with weighting factors k_L and k_U, to determine the *lowerLimit* and *upperLimit* of the proximity cost (*average penalty*) for each data set.

8.2.1.1 Calculation of Weighting Factors

To choose appropriate values for k_L and k_U is not an easy task. Therefore a set of experiments were performed on the benchmark data sets in order to determine both of the weighting factors. In these experiments, the 'best' results available in literature and the purposely generated 'worst' solutions were used as guidelines to indicate the range of coverage required. Table 8.1 shows the minimum and maximum values for each data set that the *lowerLimit* and *upperLimit* should cover.

Table 8.1: The 'best' and 'worst' timetable solutions known

Data set	Best in literature	Worst Solution	
		Max	Average
CAR-F-92	3.93	13.34	13.28
CAR-S-91	4.00	11.42	11.35
EAR-F-83	29.30	71.28	67.89
HEC-S-92	9.20	32.00	27.21
KFU-S-93	13.00	43.40	43.40
LSE-F-91	9.60	32.38	30.32
RYE-F-92	6.80	36.71	32.17
STA-F-83	157.03	194.53	194.53
TRE-S-92	7.90	17.25	17.22
UTA-S-92	3.14	8.79	8.76
UTE-S-92	24.40	56.34	56.34
YOR-F-83	36.20	64.82	63.96

Results obtained for running the algorithm depicted in Figure 8.3 on the benchmark data sets are shown in Table 8.2. After careful examination of these results, it was determined that setting $k_L = 0.55$ and $k_U = 3.10$ produced boundary settings that covered the penalty costs of all timetable solutions quality within the 'best' and 'worst' results defined in Table 8.1. Two further important values were then examined. These were ΔU, the difference between the highest observed penalty cost (for the 'worst' solution) and the approximate upper limit, and ΔL, the difference between the lowest observed penalty cost (for the 'best' solution) and the approximate lower limit (see Figure 8.2). It can be seen from Table 8.2 that ΔU for *STA-F-83* is very high, and that ΔU for *EAR-F-83* and *YOR-F-83* are also quite high. This means that the *upperLimit* for these data sets is set far too high above the worst available solutions. As the timetabling problem is a minimisation problem, it might be naturally expected that most timetables are generated towards 'best' solutions, not towards 'worst' solution. Therefore ΔU should be minimised in order to get satisfactory fuzzy evaluation results.

Further investigations indicated that these three data sets (*STA-F-83*, *EAR-F-83*

Table 8.2: Approximate boundaries derived by considering all students

Data sets	**No of Students**	**Enrolments**	E_{avg}	P_{max}	P_{approx}	**Average Penalty**		**Differences**	
						lowerLimit ($k_L = 0.55$)	*upperLimit* ($k_U = 3.10$)	ΔL	ΔU
CAR-F-92	18419	55522	3.01	45.01	4.24	2.33	13.14	1.60	0.02
CAR-S-91	16925	56877	3.36	54.55	5.24	2.88	16.24	1.12	5.07
EAR-F-83	1125	8109	7.21	166.50	50.01	27.50	155.02	1.80	86.24
HEC-S-92	2823	10632	3.77	64.67	13.53	7.44	41.94	1.76	10.62
KFU-S-93	5349	25113	4.69	90.71	21.29	11.71	66.01	1.29	23.68
LSE-F-91	2726	10918	4.12	69.47	15.46	8.50	47.92	1.10	16.31
RYE-F-92	11483	45051	3.92	71.28	12.16	6.69	37.69	0.11	1.59
STA-F-83	611	5751	9.41	234.79	169.99	93.50	526.98	58.02	340.95
TRE-S-92	4360	14901	3.42	55.20	8.20	4.51	25.43	3.39	8.59
UTA-S-92	21266	58979	2.77	39.42	3.12	1.72	9.68	1.42	1.05
UTE-S-92	2749	11793	4.29	77.85	33.38	18.36	103.49	6.04	48.81
YOR-F-83	941	6034	6.41	142.51	43.51	23.93	134.89	12.27	72.25

and *YOR-F-83*) have a very small number of students with only one exam, compared to the other nine data sets. The distributions of the number of students enroled for a particular number of the exams for each data set are presented in Table 8.3. There are two interesting observations that can be made from this Table. Firstly, it can be seen that in four data sets (*CAR-F-92*, *CAR-S-91*, *RYE-F-92* and *UTA-S-92*), the number of students with only one exam is between 2025 and 6180, while for the other eight data sets the number of students with only one exam is between only 0 and 667. As the proximity cost is calculated by dividing the total penalty by the total number of students, it is obvious that the proximity cost is highly affected by the number of students with only one exam for each data set. In one sense, students with only one exam should not given any penalty. Secondly, the fact that, for *STA-F-83*, 610 out of 611 of the students are enroled for 8, 9 or 11 exams appears to explain why this data set has a very high proximity cost compared to the other data sets. Data sets *EAR-F-83* and *YOR-F-83* also show the same pattern (most of the students enroled for many exams) but to a less extreme extent. The average number of exams enroled by each student (E_{avg}) for these three data sets is between 6 and 10 (see the fourth column of Table 8.2).

Table 8.3: Distribution of students enroled for a particular number of exams

Data sets	**Number of exams**													
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
CAR-F-92	3969	3330	3436	4168	3212	275	29							
CAR-S-91	3409	2145	2107	4098	4569	565	27	4	1					
EAR-F-83	1	1	1	20	72	201	302	409	109	9				
HEC-S-92	321	315	351	659	1071	105	1							
KFU-S-93	276	234	277	765	2515	1082	189	11						
LSE-F-91	99	80	302	1638	474	103	27	3						
RYE-F-92	2025	1463	1492	1714	1884	1487	939	459	19	1				
STA-F-83					1			162	239		209			
TRE-S-92	667	524	744	1191	1214	20								
UTA-S-92	6180	3866	3717	4026	3073	381	23							
UTE-S-92	78	118	276	754	1503	20								
YOR-F-83	1	20	93	64	44	174	172	372						1

One possible way to reduce the ΔU for *STA-F-83*, *EAR-F-83* and *YOR-F-83* is to eliminate the students with only one exam in the P_{approx} calculations. Therefore, when no students with only one exam are considered, the initial value of variable j in Equation (8.1) is set to 2; while for Equation (8.2), the initial values of both variables i and j are set to 2. Accordingly, the pseudo code depicted in Figure 8.3 also needs to be amended in several lines. Table 8.4 shows the results obtained when students with only one exam are excluded from the calculations. Appropriate values for the *lowerLimit* and *upperLimit* were then obtained by using $k_L = 0.38$ and $k_U = 2.15$, respectively.

It can be seen that a smaller weighting factor, k_U, was required to obtained the *upperLimit* that cover the range up to the worst solution for all data sets. Hence, ΔU was also reduced in most cases, compared to the previous setting (i.e. those obtained for $k_U = 3.10$ when considering all students). A comparison of the boundary ranges (i.e. $upperLimit - lowerLimit$) is shown in Table 8.5. Except for three data sets (*CAR-F-92*, *CAR-S-91* and *UTA-S-92*), it can be seen that the boundary ranges are reduced when students with only one exam are excluded. For the purpose of comparison, both boundaries settings (see the seventh and eighth columns of Tables 8.2 and 8.4) were used in the normalisation process of the fuzzy evaluation experiments.

Table 8.4: Approximate boundaries derived by excluding students with only one exam

Data sets	**No of Students**	**Enrolments**	E_{avg}	P_{max}	P_{approx}	**Average Penalty**		**Different**	
						lowerLimit ($k_L = 0.38$)	*upperLimit* ($k_U = 2.15$)	ΔL	ΔU
CAR-F-92	14450	51553	3.57	57.37	6.40	2.43	13.75	1.50	0.42
CAR-S-91	13516	53468	3.96	68.31	7.73	2.93	16.60	1.07	5.17
EAR-F-83	1124	8108	7.21	166.65	50.04	19.03	107.69	10.27	36.41
HEC-S-92	2502	10311	4.12	72.96	16.70	6.35	35.92	2.85	3.91
KFU-S-93	5073	24837	4.90	95.65	23.43	8.90	50.34	4.10	6.94
LSE-F-91	2627	10819	4.12	72.09	16.50	6.27	35.46	3.33	3.08
RYE-F-92	9458	43026	4.55	86.54	17.12	6.50	36.80	0.30	0.09
STA-F-83	611	5751	9.41	234.79	169.99	64.60	365.48	86.92	170.95
TRE-S-92	3693	14234	3.85	65.17	10.91	4.15	23.48	3.75	6.24
UTA-S-92	15086	52799	3.50	55.57	5.56	2.11	11.95	1.03	3.15
UTE-S-92	2671	11715	4.39	80.12	35.17	13.35	75.52	11.05	19.18
YOR-F-83	940	6033	6.42	142.66	43.61	16.57	93.74	19.63	28.92

Table 8.5: A comparison of the range of boundary settings for *average penalty*

Data sets	**All students**	**Excluding students with only one exam**
CAR-F-92	11.02	11.32
CAR-S-91	13.62	13.66
EAR-F-83	130.02	88.66
HEC-S-92	35.18	29.57
KFU-S-93	55.37	41.44
LSE-F-91	40.19	29.19
RYE-F-92	31.61	30.30
STA-F-83	441.98	300.89
TRE-S-92	21.33	19.33
UTA-S-92	8.12	9.84
UTE-S-92	86.80	62.17
YOR-F-83	113.14	77.17

8.2.2 Approximate Boundaries for *Highest Penalty*

In terms of *highest penalty*, the *upperLimit* value was determined by the following formula:

$$maximum\ highest\ penalty,\ HP_{max} = \sum_{i=1}^{R_{max}-1} \sum_{j=j+1}^{maxgap} 2^{5-(j-i)}, \qquad (8.4)$$

where

$$maxgap = \begin{cases} i+5, & \text{if } (i+5) \leqslant R_{max} \\ R_{max}, & \text{otherwise} \end{cases}$$

Basically HP_{max} represents the proximity cost penalty for the maximum number of exams enroled on by an individual student. In the pseudo-code shown in Figure 8.3, this value is represented by the array *subtotal1* for element number *maxExam*. Referring to the given example (see Figure 8.4), $R_{max} = maxExam = 8$; hence $HP_{max} = subtotal1[8] = 191$.

As this is the first attempt to consider the highest penalty imposed on any individual student in evaluating timetable quality, wide experience of appropriate ranges for this variable was not available. Hence, only the minimum and maximum values of *highest penalty* for each data set from timetable solutions that had been generated in this research (see Section 7.3 for the descriptions of how the solutions were generated) were used as guidelines to determine the lower limit and upper limit for the boundary setting. Table 8.6 shows the boundary settings for *highest penalty* employed in the experiments. The phrase *'In hands timetable'* is referred to the set of timetable solutions obtained in the experiments as explained in Section 7.3. The *lowerLimit* value was simply obtained by multiplying the *upperLimit* (HP_{max}) with a weighting factor $k_{L_{HP}} = 0.3$ (which was determined empirically).

Table 8.6: The boundary settings for *highest penalty*

Data sets	In hands timetable solutions quality			Approximate boundary	
	Min	Max	Avg	*lowerLimit*	*upperLimit*
CAR-F-92	65.0	84.0	71.8	48.0	160.0
CAR-S-91	68.0	101.0	77.1	66.6	222.0
EAR-F-83	105.0	194.0	140.3	75.9	253.0
HEC-S-92	75.0	129.0	95.3	48.0	160.0
KFU-S-93	98.0	131.0	114.4	57.3	191.0
LSE-F-91	78.0	160.0	105.5	57.3	191.0
RYE-F-92	87.0	139.0	110.9	75.9	253.0
STA-F-83	227.0	248.0	228.2	85.2	284.0
TRE-S-92	68.0	98.0	79.3	38.7	129.0
UTA-S-92	63.0	106.0	75.0	48.0	160.0
UTE-S-92	83.0	129.0	100.4	38.7	129.0
YOR-F-83	228.0	301.0	252.7	101.1	337.0

8.3 Algorithmic Determination of the Lower Boundary

In the derivation of approximate boundaries detailed above, the assumption was made that maximum penalty cost for a student could be obtained by placing all their exams in adjacent time slots (see Equation (8.1)). In order to calculate an approximate *minimum* proximity cost (P_{min}), utilising the underlying structure of the problem instances, a contrasting assumption was applied. Conceptually, in order to mimimise the proximity cost, the task is to spread out the exams enroled on by each student as much as possible. That is, the objective is to assign the enroled exams into the time slots that will cause the least penalty cost for the particular number of enroled exams. In a similar approach to that described in the P_{max} calculation detailed in Section 8.2.1, no constraints amongst exams across students were considered. By ignoring constraints amongst exams, it is to be expected that any feasible solution must have an *average penalty* that is *higher* than the *lowerLimit* determined using this approach. In effect, ignoring this hard constraint means that the *lowerLimit* is applicable to some solutions which are infeasible. However, crucially, it is applicable to all solutions that are feasible. Of course, if it were possible to derive the formal lower bound for the set of feasible solutions only, then this lower bound would represent the global optimum for this minimisation problem.

Two algorithms to determine this *lowerLimit* are presented below. The first features a brute-force method in which all possible combinations of placements of exams are considered. When run, it was found that this can take a large amount of time (obviously dependent on the problem size) and so a refinement of the algorithm was developed. This refinement features a type of 'greedy' placement algorithm, which omits many placement combinations but runs in much faster time. In order to further reduce the computational time (for both forms of the algorithm), only the number of enroled exams that can cause penalty are taken into account during the calculations. For a problem with T time slots available, the minimum number of enroled exams that will cause penalty is given by:

$$minExams\_causePenalty = ((int)(T + 5)/6) + 1, \tag{8.5}$$

8.3.1 *Brute Force Lower Limit Approximation Algorithm*

The first algorithm is termed the *Brute Force Lower Limit Approximation Algorithm* (*BFLLAA*). *BFLLAA* starts with all the enroled exams assigned in adjacent time slots. Later on, in the sequence of iterations, the exams are moved in a systematic manner in the search for the placement of exams that causes the least penalty. It is difficult to represent the algorithm in pseudo-code, as the number of nested loops is dependent on the number of exams that is currently under consideration. For example, if the penalty cost for seven exams is being calculated, then seven nested loops are required.

Hence, an illustrative example is given in order to explain this algorithm. Consider a problem with only eight time slots. From Equation (8.5), the minimum number of exams that cause penalty is three. Let us assume that there are ten students enroled for three exams and five students enroled for four exams. Figure 8.5 shows the pseudo-code for *BFLLAA* when calculating the penalty for three enroled exams. In Figure 8.6, an illustration of the process is given (for 3 enroled exams), showing some of the steps of the iterative process. Steps (i) and (vi) represent the first and the final step, respectively. Steps (ii) – (v) are not in the full sequence, but are only used to show an illustrative sample of the steps in the process. At the end of the process, the minimum penalty found is accumulated as the total penalty. The same process is performed for each number of enroled exams in sequence, until the stopping criteria is reached, at which point the number of enroled exams is equal to the maximum number of exams enroled by any of the students. For example, Figure 8.7 shows how the pseudo-code proceeds in order to calculate the penalty for four enroled exams. Finally, P_{min} is obtained by dividing the accumulated penalty by the number of students.

8.3.2 *Greedy Lower Limit Approximation Algorithm*

The second algorithm is termed the *Greedy Lower Limit Approximation Algorithm* (*GLLAA*). The pseudo-code for *GLLAA* is shown in Figure 8.8. For each number of enroled exams in turn (starting with $minExams\_causePenalty$) the following is carried out. An empty timetable is created with the specified number of time slots. For each exam in turn, the exam is assigned to the time slot that incurs the least penalty. After assigning each of the exams into a time slot, the penalty incurred is calculated. This value is then multiplied by the number of students enroled on this specified number of exams. The result of this calculation is then accumulated to the total penalty. The process continues for each of the number of exams enroled until the maximum number of enroled exams is reached. P_{min} is determined by dividing the total penalty by the total number of students (considering all students). Note that, at each iteration the timetable is re-initialised as an empty timetable. This means that the process of assigning the exams to time slots for the current iteration is not affected by the exam assignments made in the previous iteration. However, the penalty incurred at each iteration is accumulated.

```
DECLARE INTEGER examEnroled  // Number of exams enroled by a student
DECLARE INTEGER maxStudent  // Total number of student
DECLARE DOUBLE penalty  // Penalty cost for the adjacent exams
DECLARE INTEGER maxPeriod  // Number of time slots available
DECLARE INTEGER maxExam // Maximum number of exams enroled by any one student
DECLARE INTEGER schedule[maxPeriod] // Holds timetable
DECLARE INTEGER examCount [maxExam] // Number of students enroled for x exams
DECLARE INTEGER subtotal[maxExam] // Penalty impose on a student enroled for x exams
DECLARE DOUBLE minPenalty, totalPenalty

// STEP 1 : Traverse the student array in order to read each student record
For s = 1 to maxStudent
        // Get the number of exams enroled by student s
        examEnroled  ← studentArray[s].examEnroled
        // Increase the counter by 1
        examCount[examEnroled] ← examCount[examEnroled] + 1
End For

// STEP 2 : Calculate and find the exams arrangement that will cause minimum penalty cost for 3 exams
minPenalty ←  10000.0
maxPeriod ←  8
examToSchedule  ←  3
DECLARE INTEGER  exam[examToSchedule] ← {1,2,3}  //Initialise exams list with size examEnroled
DECLARE INTEGER  stopindex [examToSchedule]  //Controller to avoid exams schedule in the same time slot
DECLARE INTEGER  schedule [maxPeriod] // Holds timetable
stopindex[1] ←  maxperiod
stopindex[2] ←  maxperiod - 1
stopindex[3] ←  maxperiod - 2

For L1 = 1 to stopindex[3]
        For L2 = L1+1 to stopindex[2]
                For L3 = L2+1 to stopindex[1]
                        schedule[L1] ← exam[1]
                        schedule[L2] ← exam[2]
                        schedule[L3] ← exam[3]
                        penalty ← calculate penalty of assigning examToSchedule exams into schedule[]
                        if (penalty < minPenalty)
                                minPenalty ←  penalty
                End For
        End For
End For
subtotal[examToSchedule] ← minPenalty * examCount[examToSchedule]
totalPenalty ←  totalPenalty + subtotal[examToSchedule]
```

Figure 8.5: Pseudo code for *BFLLAA* for three enroled exams

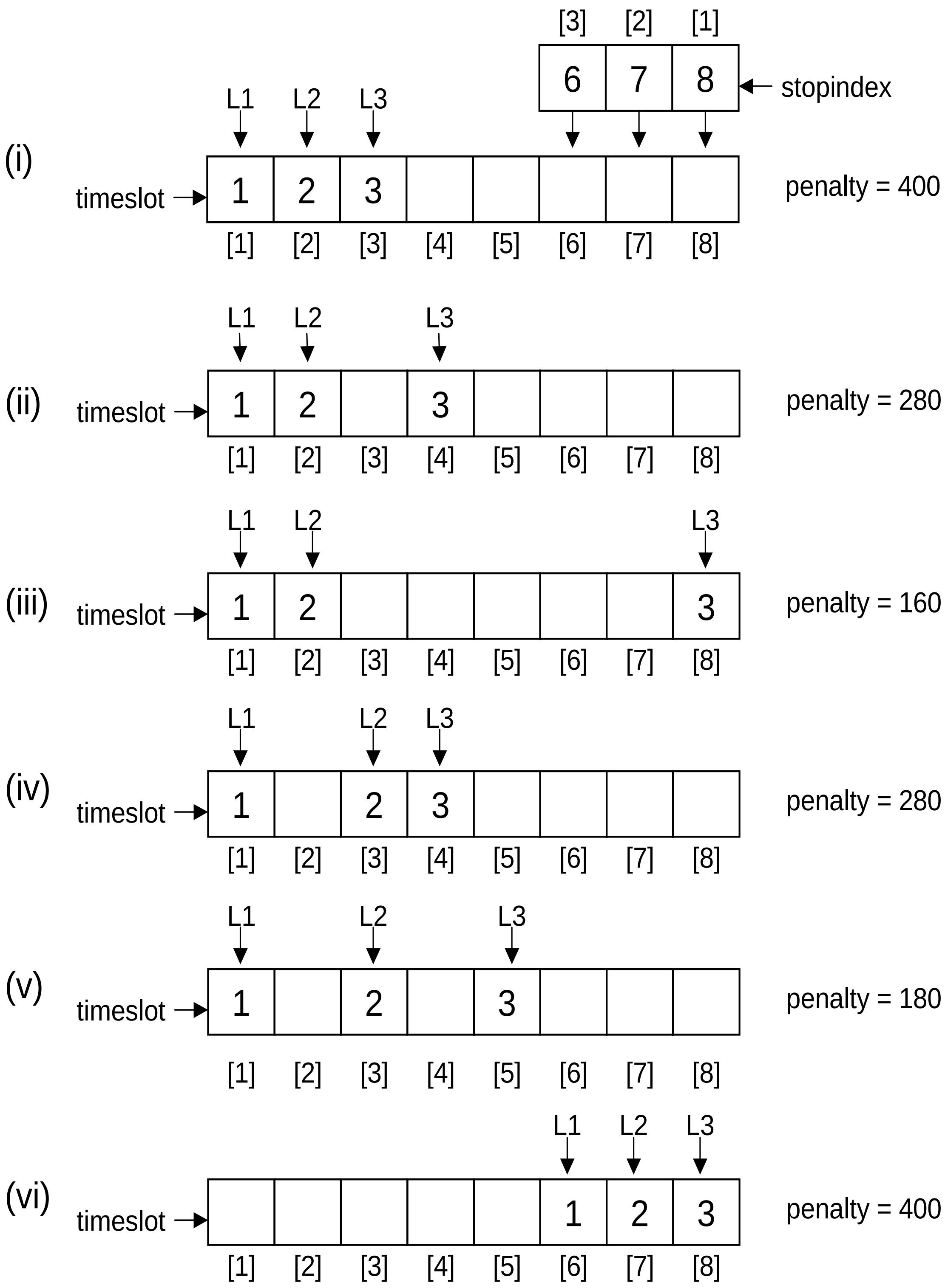


Figure 8.6: Illustrative example of BFLLAA for 3 enroled exams

```
// STEP 3 : Calculate and find the exams arrangement that will cause minimum penalty cost for 4 exams
minPenalty ← 10000
maxPeriod ← 8
examToSchedule ← 4
DECLARE INTEGER exam[examToSchedule] ← {1,2,3,4} //Initialise exams list with size examEnroled
DECLARE INTEGER stopindex [examToSchedule] //Controller to avoid exams schedule in the same time slot
DECLARE INTEGER schedule [maxPeriod] // Holds timetable
stopindex[1] ← maxperiod
stopindex[2] ← maxperiod - 1
stopindex[3] ← maxperiod - 2
stopindex[4] ← maxperiod - 3

For L1 = 1 to stopindex[4]
    For L2 = L1+1 to stopindex[3]
        For L3 = L2+1 to stopindex[2]
            For L4 = L3+1 to stopindex[1]
                schedule[L1] ← exam[1]
                schedule[L2] ← exam[2]
                schedule[L3] ← exam[3]
                schedule[L4] ← exam[4]
                penalty ← calculate penalty of assigning examToSchedule exams into schedule[]
                if (penalty < minPenalty)
                    minPenalty ← penalty
            End For
        End For
    End For
End For
subtotal[examToSchedule] ← minPenalty
totalPenalty ← totalPenalty + subtotal[examToSchedule]
```

Figure 8.7: Pseudo code for *BFLLAA* for four enroled exams (continue from Figure 8.5)

8.3.3 Comparison of Lower Limit Algorithms

An experiments was performed in order to evaluate and compare these two new methods for calculating the *lowerLimit* (*BFLLAA* and *GLLAA*). A comparison of P_{min} values obtained using the two alternative algorithms is presented in Table 8.7. The approximate time taken by the two algorithms is also shown for comparative purposes (run on the same hardware under the same experimental conditions, but not particularly carefully controlled). It can be seen that four of the data sets have values of P_{min} that are well above zero (*EAR-F-83*, *STA-F-83*, *UTE-S-92* and *YOR-F-83*), another four have values of P_{min} that are smaller but still definitely non-zero (*HEC-S-92*, *KFU-S-93*, *LSE-F-91* and *RYE-F-92*) and the other four have values quite close to zero (*CAR-F-92*,

```
DECLARE INTEGER examEnroled  // Number of exams enroled by a student
DECLARE DOUBLE penalty  // Penalty cost for the adjacent exams
DECLARE INTEGER maxStudent  // Total number of student
DECLARE INTEGER maxPeriod  // Number of time slots available
DECLARE INTEGER maxExam // Maximum number of exams enroled by any one student
DECLARE INTEGER schedule[maxPeriod] // Holds timetable
DECLARE INTEGER examCount [maxExam] // Number of students enroled for x exams
DECLARE INTEGER subtotal1[maxExam] // Penalty impose on a student enroled for x exams
DECLARE INTEGER subtotal2[maxExam] // Penalty impose on all students who enroled for x exams
DECLARE INTEGER minExams_causePenalty
DECLARE DOUBLE Pmi, totalPenalty

// Traverse the student array in order to read each student record
For s = 1 to maxStudent
        // Get the number of exams enroled by student s
        examEnroled  ← studentArray[s].examEnroled
        // Increase the counter by 1
        examCount[examEnroled] ← examCount[examEnroled] + 1
End For

//Set the minimum number of time slot that will cause penalty
minExams_causePenalty ← ((maxPeriod +5 ) / 6) + 1

For e = minExams_causePenalty to maxExam
        Reset schedule[]  as an empty timetable
        If examCount[e] > 0
                DECLARE INTEGER  unscheduledList[e]  // Declare dummy exams list with size e
                For i = 1 to e  //Assign exam into time slot
                        E# ←  unscheduledList[i]
                        Assign exam E# into a time slot in schedule[] with minimum penalty cost
                End For
                penalty ← calculate penalty of assigning e exams  into schedule[]
                subtotal1[e] ← penalty
        End If
        // Multiply the penalty cost for e exams with number of student enroled for e exams
        subtotal2[e] ← subtotal1[e] *  examCount[e]
        // Accumulate the penalty cost
        totalPenalty  ← totalPenalty + subtotal2[e];
End For
// Calculate the approximate value of minimum total penalty
Pmin = totalPenalty / maxStudent;
```

Figure 8.8: Pseudo code for *GLLAA*

CAR-S-91, *TRE-S-92* and *UTA-S-92*). Note that the time taken by *BFLLAA* is sometimes significant (many hours) for these data sets. Not also that the time taken by *GLLAA* is very much smaller This is the first time that an attempt has been made

Table 8.7: *lowerLimit* values for *average penalty* calculated using *BFLLAA* and *GLLAA*

Data sets	**BFLLAA**		**GLLAA**		**Percent Diff.**
	P_{min}	**Time (mins.)**	P_{min}	**Time (mins.)**	**in** P_{min}
CAR-F-92	0.0079	55	0.0126	< 1	37.30
CAR-S-91	0.0059	1458	0.0066	< 1	10.61
EAR-F-83	17.8471	52	19.4053	< 1	8.03
HEC-S-92	3.4945	1	4.2905	< 1	18.55
KFU-S-93	5.6338	2	7.9323	< 1	28.98
LSE-F-91	2.7649	1	3.1548	< 1	12.36
RYE-F-92	3.7868	32	4.2113	< 1	10.08
STA-F-83	152.0458	< 1	153.7136	< 1	1.09
TRE-S-92	0.5936	1	0.6028	< 1	1.53
UTA-S-92	0.00216	81	0.0022	< 1	1.82
UTE-S-92	21.5098	< 1	24.3647	< 1	11.72
YOR-F-83	18.9607	7	20.9915	< 1	9.67

to derive a lower limit for the proximity cost achievable on these data sets and it is interesting to note how high the lower limit for *STA-F-83* actually is. Indeed, initially it appeared that the lower limit derived here was *above* some results previously quoted in literature. Of course, if results lower than the lower limit derived here had been achieved, then it would imply that the assumptions made here (in order to derive the lower limit) were incorrect. Remember that, although not formally proven as a lower bound, the lower limit calculations derived here are *believed* to apply to *all* feasible solutions — i.e. it is believed that any feasible solution *must* lie above the lower limit derived by *BFLLAA*. As an aside, as *GLLAA* is known to be a greedy approximation of the brute-force limit, it is possible that a feasible solution lies below the *GLLAA* limit. However, it can be seen that the differences between the P_{min} values obtained by the two algorithms is quite small (within around ten percent or less of *BFLLAA*), while the time taken is very much quicker.

8.3.4 Algorithmic Derivation of Boundaries

Given that, for the Carter data sets, *BFLLAA* could be run in reasonable time, there was no reason not to use the values of P_{min} obtained using this method as the *lowerLimit*. and then to determine a method for deriving the *lowerLimit* and *upperLimit* based on these algorithms. Hence, the P_{min} derived by *BFLLAA* for each data set was assigned as the *lowerLimit* for *average penalty*. Having obtained an algorithmic value for the *lowerLimit*, the next step was to derive a method for calculating *upperLimit*. The algorithmic method for deriving P_{max} and P_{approx} calculated when considering all students, as presented in Section 8.2.1 (see Table 8.2), was reused in a slightly modified form. Firstly, a smaller multiplying factor of 2.0 for P_{approx} was utilised to give a smaller *upperLimit*. These values of *upperLimit* were determined empirically in order to bring the boundaries close to the lower and upper values observed in practice (so that the overall (fuzzy) quality for the 'best' solution and the 'worst' solution are easily differentiated). The *upperLimit* values should not be set too far from the *lowerLimit* as the intention is to construct timetable solutions with smaller proximity cost. It was also noted that for one data set (*STA-F-83*), even this value of *upperLimit* was *higher* than P_{max}, and so an additional condition was introduced limiting the *upperLimit* to P_{max}. Hence, the *upperLimit* was determined as follows:

$$upperLimit = \begin{cases} P_{max}, & \text{if } P_{approx} * 2.0 > P_{max} \\ P_{approx} * 2.0, & \text{otherwise} \end{cases}$$

where P_{max} and P_{approx} are calculated when considering all students (see Table 8.2). The resultant boundary settings that were obtained in this way are shown in Table 8.8.

Table 8.8: Boundary settings for *average penalty* using *BFLLAA lowerLimit*

Data sets	***lowerLimit***	***upperLimit***
CAR-F-92	0.008	8.48
CAR-S-91	0.006	10.47
EAR-F-83	17.847	100.01
HEC-S-92	3.495	27.06
KFU-S-93	5.634	42.59
LSE-F-91	2.765	30.92
RYE-F-92	3.787	24.32
STA-F-83	152.046	234.79
TRE-S-92	0.594	16.41
UTA-S-92	0.002	6.25
UTE-S-92	21.510	66.77
YOR-F-83	18.961	87.03

8.4 Evaluation of Boundary Settings

8.4.1 Methods

A similar experimental setup as described in Section 7.3.1 was implemented in order to examine the effect of the various methods introduced so far for determining the boundary of *average penalty*. Based on the methods explained in Sections 8.2.1 and 8.3, three new boundary settings were examined, and these were compared to the two methods introduction in Chapter 7. The five boundary methods compared were:

- the range[0.0, *maxValue*] described in previous Chapter (referred to as *Range1*);
- the range[*minValue*, *maxValue*] described in previous Chapter (*Range2*);
- the range[*lowerLimit*, *upperLimit*] using the approximate boundaries calculated by considering all students, as described in Section 8.2.1.1 and given in columns 7–8 of Table 8.2 (*Range3*);
- the range[*lowerLimit*, *upperLimit*] using the approximate boundaries calculated by excluding students sitting only one exam, as described in Section 8.2.1.1 and

given in columns 7–8 of Table 8.4 (*Range4*); and

- the range[*lowerLimit*, *upperLimit*] derived algorithmically based on *BFLLAA*, as described in Section 8.3.1 and given in Table 8.8 (*Range5*).

Note that, in terms of *highest penalty*, similar boundary settings to those implemented in the previous experiments (see Table 8.6) were employed unaltered.

8.4.2 Results

Table 8.9 shows the fuzzy quality measure obtained for the 'worst' and 'best' solutions as evaluated under the three new boundary settings introduced in the Chapter (termed Range 3 to Range 5, above). Figures 8.9 to 8.20 show graphs of the solutions ranked by the various fuzzy evaluation functions against the ranking obtained by the original proximity cost. In each graph, all the qualities obtained from the fuzzy evaluation using the five alternative boundary settings described above are plotted. If the ranking obtained by the new method is the same as that obtained by the original proximity cost solution, then the point will lie on the line $y = x$. For example, in Figure 8.9, the solution ranked 18^{th} lowest by proximity cost was also ranked 18^{th} lowest by the fuzzy evaluation measure based on *Range2*. Any point plotted either above or below the $y = x$ line represents that the rank of the solution obtained using the fuzzy evaluation is above or below the rank obtained when only considering proximity cost in measuring the solution quality. Overlapped markers for the boundary settings show that the respective boundary settings evaluated the solution to the same ranked position. For example, in Figure 8.9 again, the same solution was ranked best (rank 1) by all evaluation methods.

8.4.3 Discussion

It is immediately evident that the solution rankings have changed in comparison with the initial ranking (i.e. that based only upon proximity cost) when the fuzzy evaluation is utilised to rank the solutions. This is consistent with the results obtained in the

Table 8.9: A comparison of the results of fuzzy evaluation obtained by using the approximate boundary settings based on the three new methods introduced in this Chapter

Data sets	Range3		Range4		Range5	
	Worst	Best	Worst	Best	Worst	Best
CAR-F-92	0.250194	0.797254	0.269371	0.805984	0.219585	0.663828
CAR-S-91	0.330549	0.838201	0.333812	0.840984	0.287907	0.682057
EAR-F-83	0.481702	0.831354	0.393637	0.747719	0.374716	0.724998
HEC-S-92	0.327018	0.734226	0.275794	0.705584	0.189365	0.628090
KFU-S-93	0.298753	0.736131	0.155710	0.717846	0.111464	0.653105
LSE-F-91	0.295403	0.836315	0.130307	0.763360	0.111464	0.700172
RYE-F-92	0.286992	0.867045	0.284188	0.863133	0.283494	0.725924
STA-F-83	0.467568	0.566619	0.336497	0.476504	0.313114	0.571982
TRE-S-92	0.293612	0.676428	0.274126	0.653753	0.111464	0.544662
UTA-S-92	0.250256	0.786457	0.352895	0.834831	0.241961	0.658009
UTE-S-92	0.334524	0.678439	0.265393	0.611868	0.202785	0.659118
YOR-F-83	0.379577	0.659125	0.296110	0.555702	0.288389	0.552610

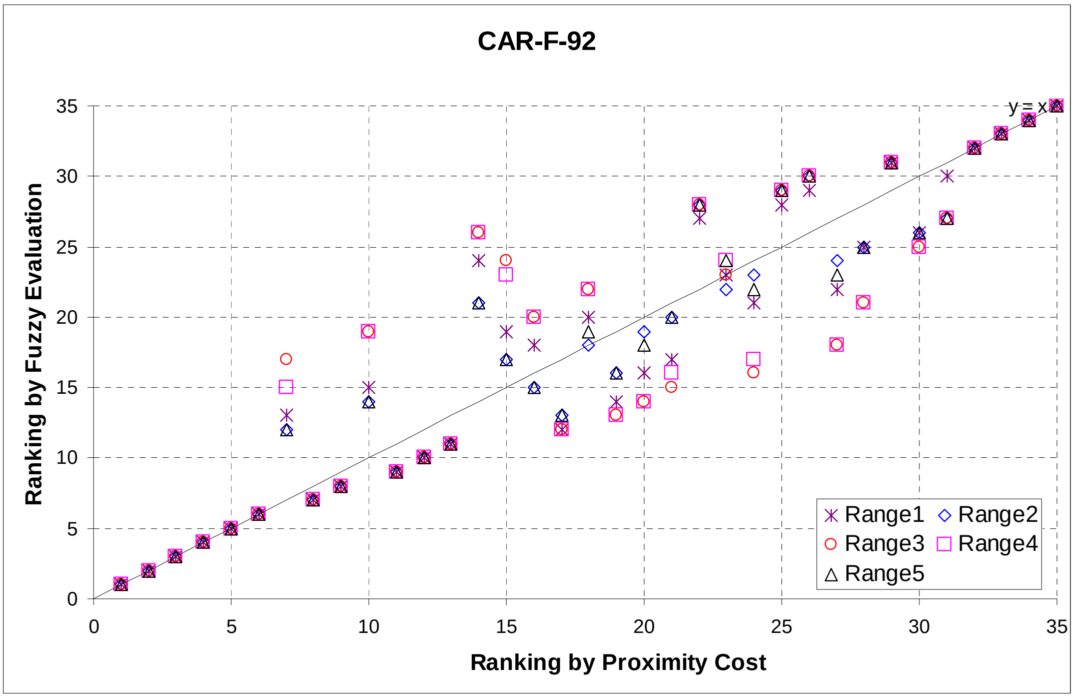


Figure 8.9: A comparison of rankings produced by the five boundary settings used for *CAR-F-92*

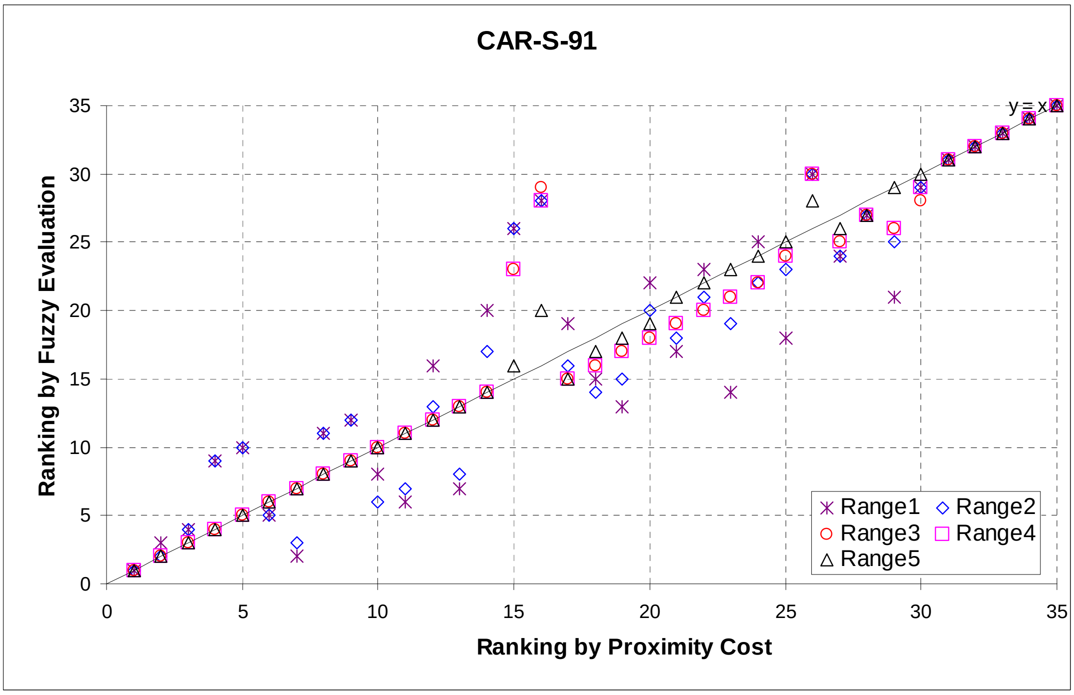


Figure 8.10: A comparison of rankings produced by the five boundary settings used for *CAR-S-91*

previous Chapter (when the boundary setting *Range1* and *Range2* were employed). In terms of functionality, these results indicate that the fuzzy evaluation system has performed as intended in measuring the timetable's quality by considering two criteria simultaneously. Although different boundary settings were utilised, the results show the same pattern of overall fuzzy quality in terms of evaluation performance. For example, in Figure 8.9, in the case of the solution that was ranked 7^{th} by proximity cost, the five different boundary settings ranked the solution between 12^{th} and 17^{th}. The reason why the rankings produced by different boundary conditions is slightly different has been discussed in the previous Chapter. Notice that in some cases the difference in rankings is quite marked. For example, this situation can be observed in the following cases (in this list the rank refers to *Ranking by Proximity Cost* (i.e the x-axis)):

- the solution ranked 15^{th} for *CAR-S-91* (Figure 8.10)
- the solution ranked 3^{rd} for *HEC-S-92* (Figure 8.12)

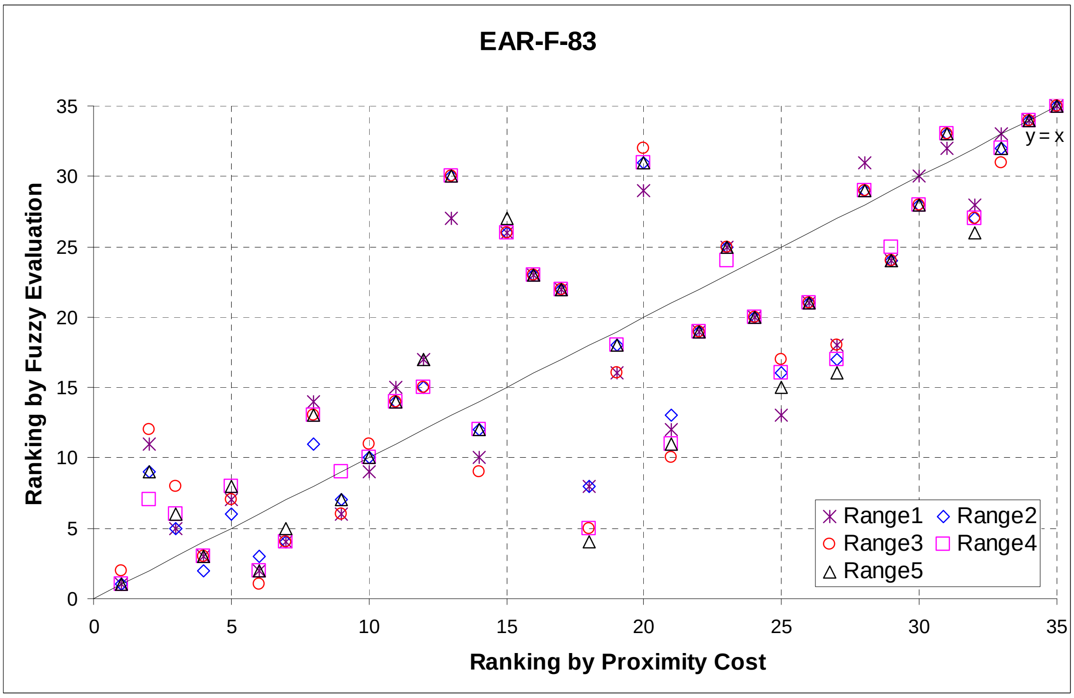


Figure 8.11: A comparison of rankings produced by the five boundary settings used for *EAR-F-83*

- the solution ranked 22^{nd} for *KFU-S-93* (Figure 8.13)
- the solutions ranked 6^{th} and 9^{th} for *STA-F-83* (Figure 8.16)
- the solution ranked 5^{th} for *UTE-S-92* (Figure 8.19)

One should notice that, even though the difference in ranking is quite marked, the overall fuzzy quality for the solutions calculated by the five boundary settings are in the same direction (i.e. they all lie either above or below the line $y = x$).

On the other hand, very close agreement can be observed for the solutions ranked 1^{st} to 5^{th} and 32^{nd} to 35^{th} for three data sets (*CAR-F-92*, *CAR-S-91* and *UTA-S-92*). Further investigation showed that the five 'best' solutions ranked 1^{st} to 5^{th} for each of these three data sets have *highest penalty* values that are almost identical (and hence only *average penalty* has a bearing on relative solution quality). Concerning the solutions ranked 32^{th} to 35^{th}, it can be observed that the last four worst solutions for *CAR-S-91* and *UTA-S-92* data sets have the same *highest penalty* value — 164 for *CAR-S-91* and

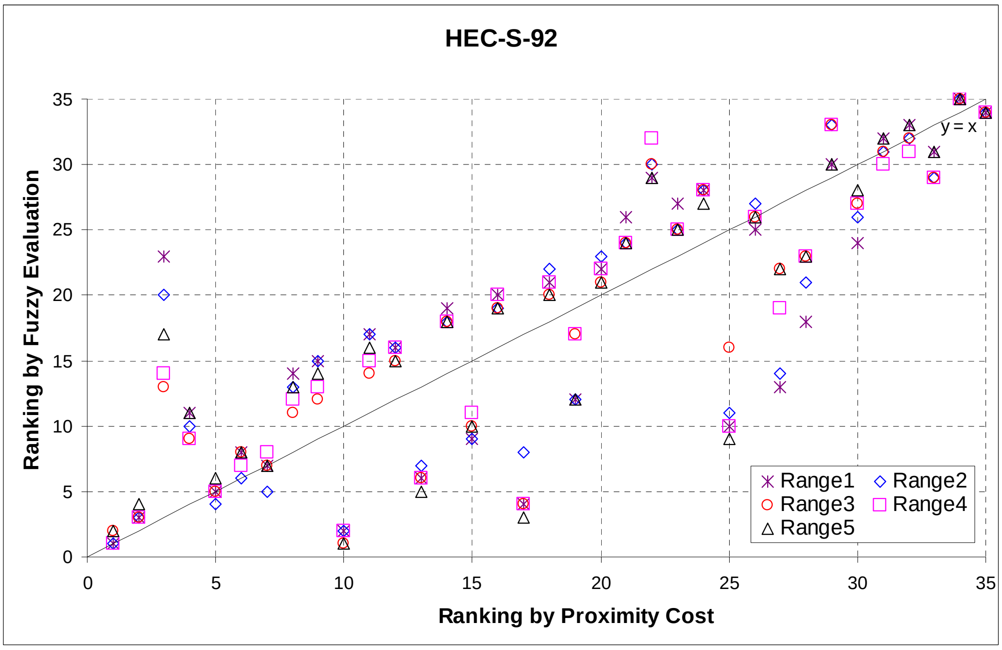


Figure 8.12: A comparison of rankings produced by the five boundary settings used for *HEC-S-92*

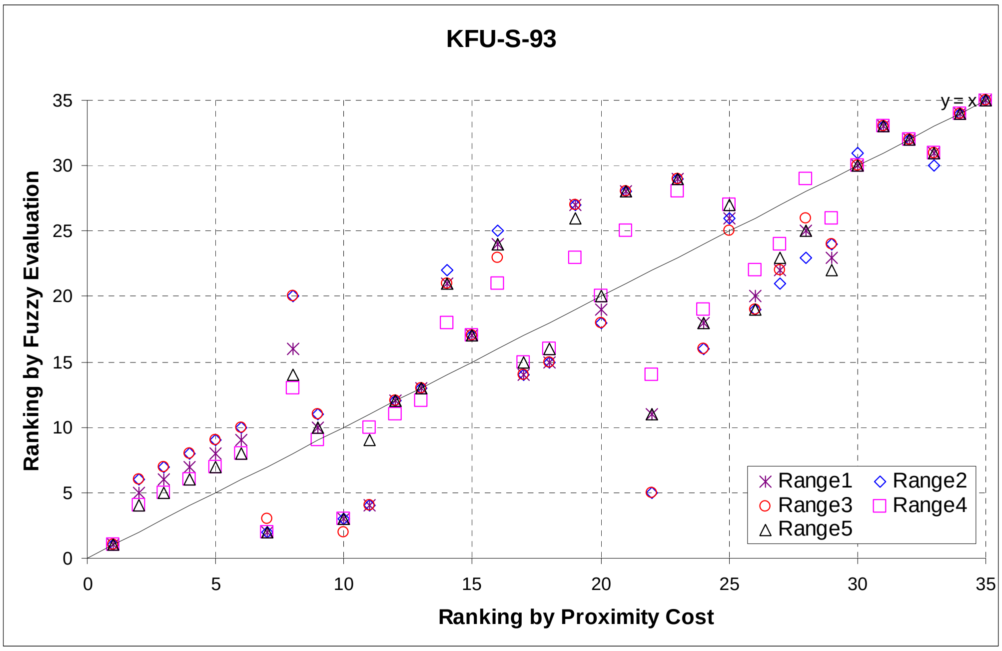


Figure 8.13: A comparison of rankings produced by the five boundary settings used for *KFU-S-93*

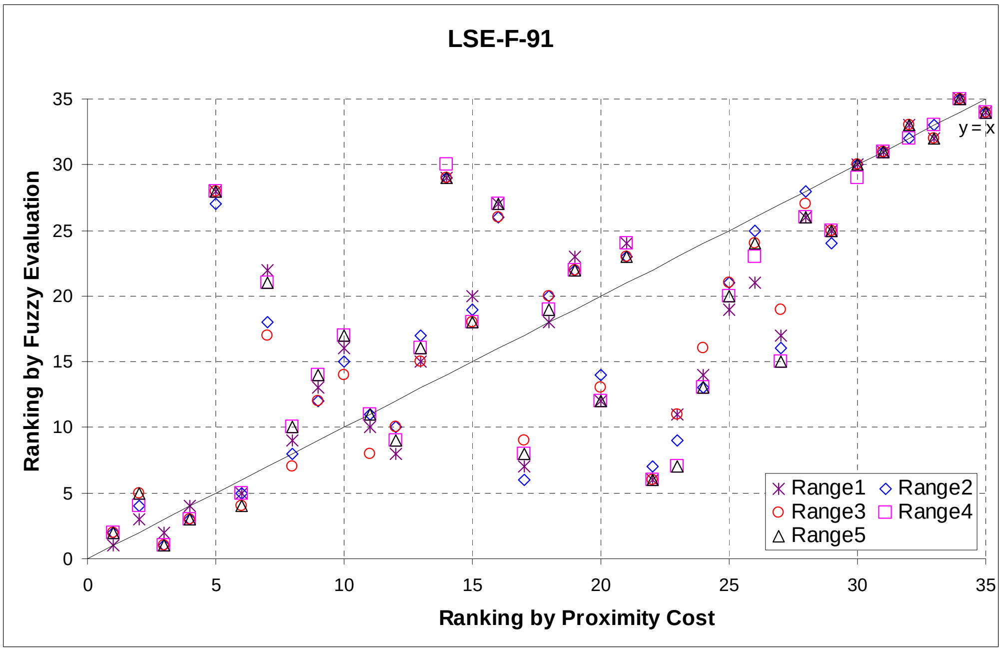


Figure 8.14: A comparison of rankings produced by the five boundary settings used for *LSE-F-91*

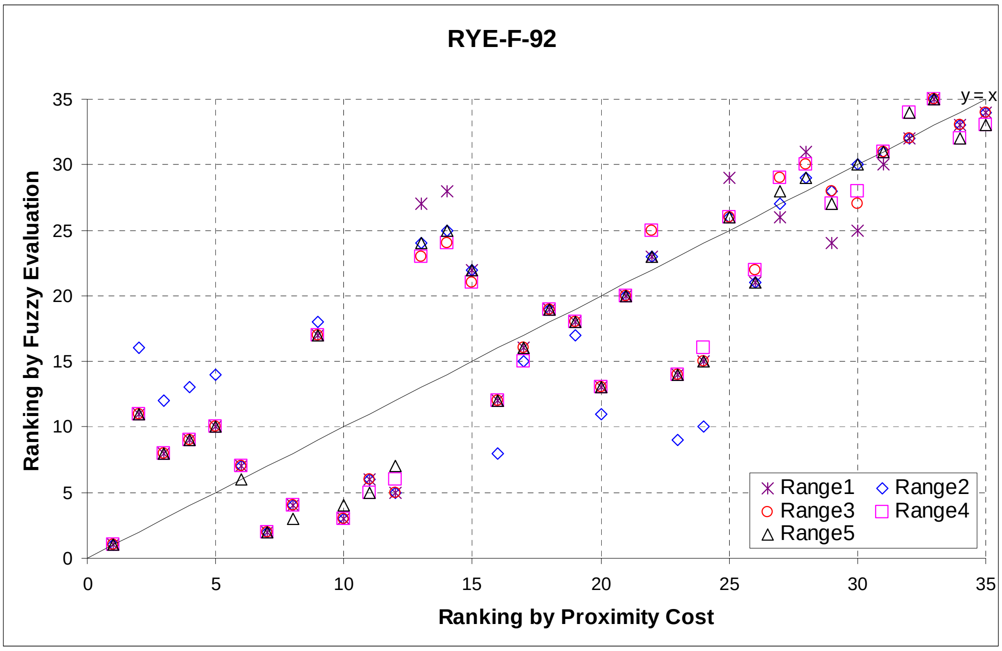


Figure 8.15: A comparison of rankings produced by the five boundary settings used for *RYE-F-92*

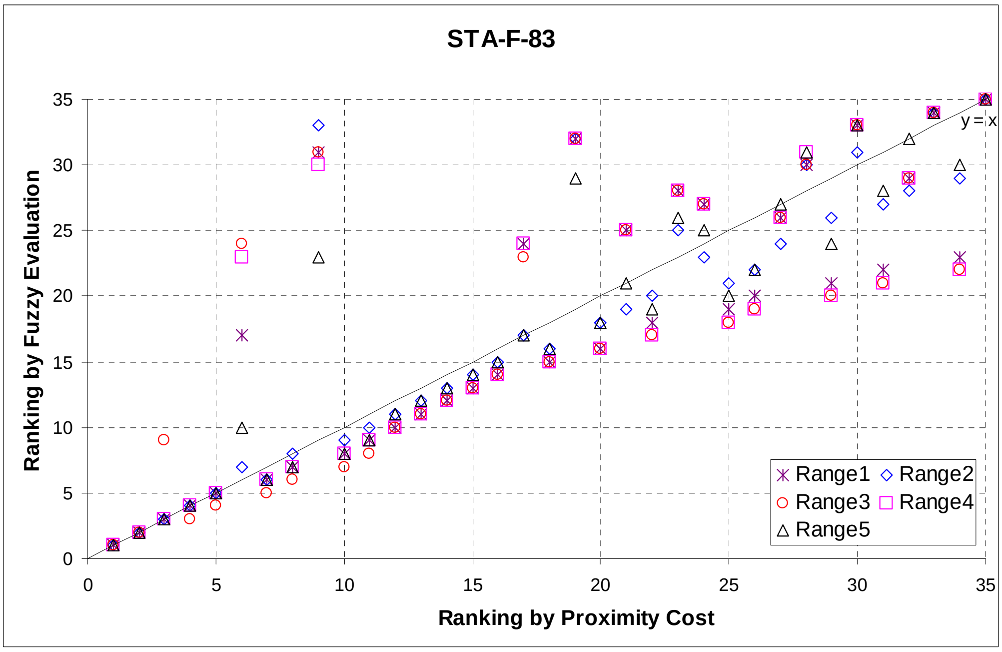


Figure 8.16: A comparison of rankings produced by the five boundary settings used for *STA-F-83*

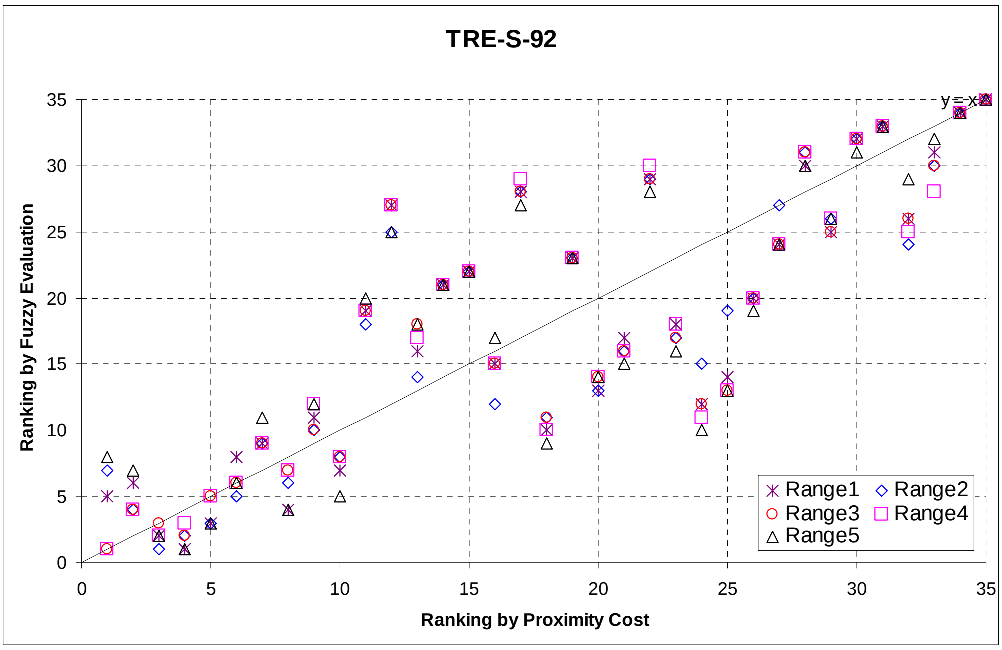


Figure 8.17: A comparison of rankings produced by the five boundary settings used for *TRE-S-92*

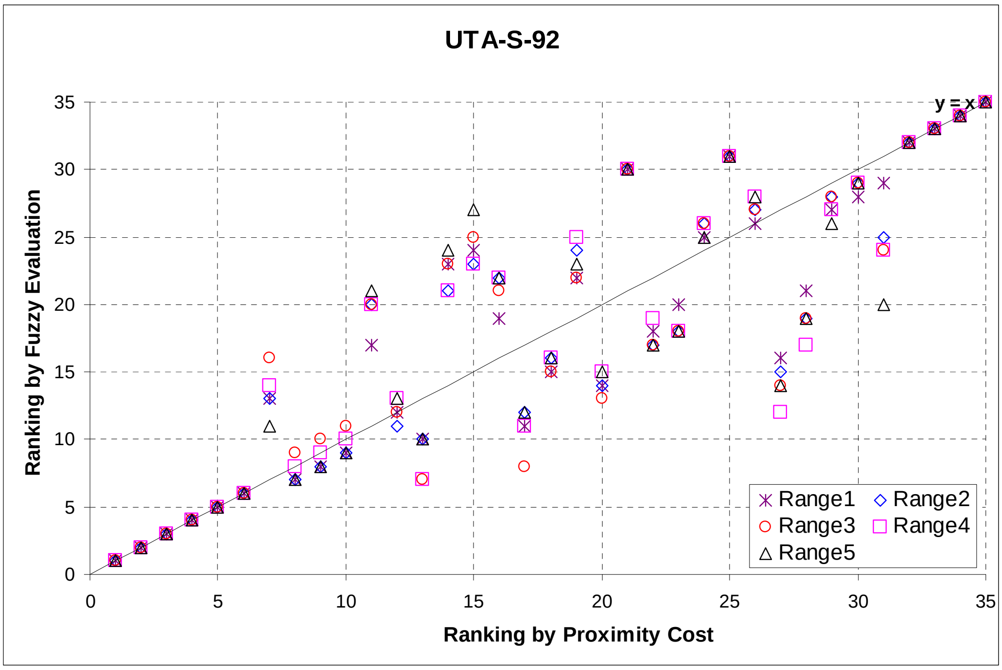


Figure 8.18: A comparison of rankings produced by the five boundary settings used for *UTA-S-92*

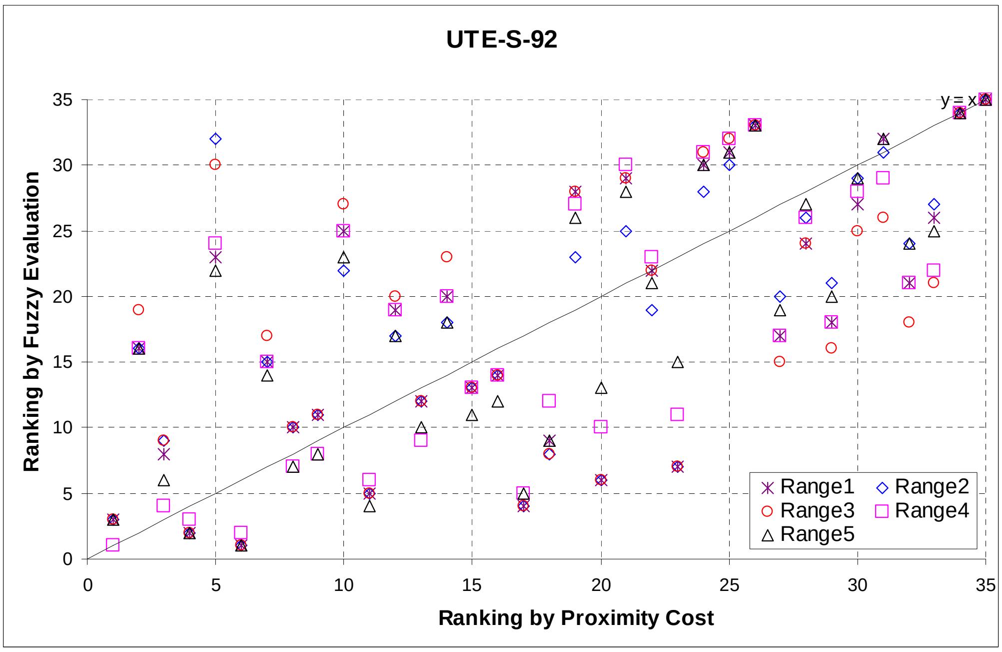


Figure 8.19: A comparison of rankings produced by the five boundary settings used for *UTE-S-92*

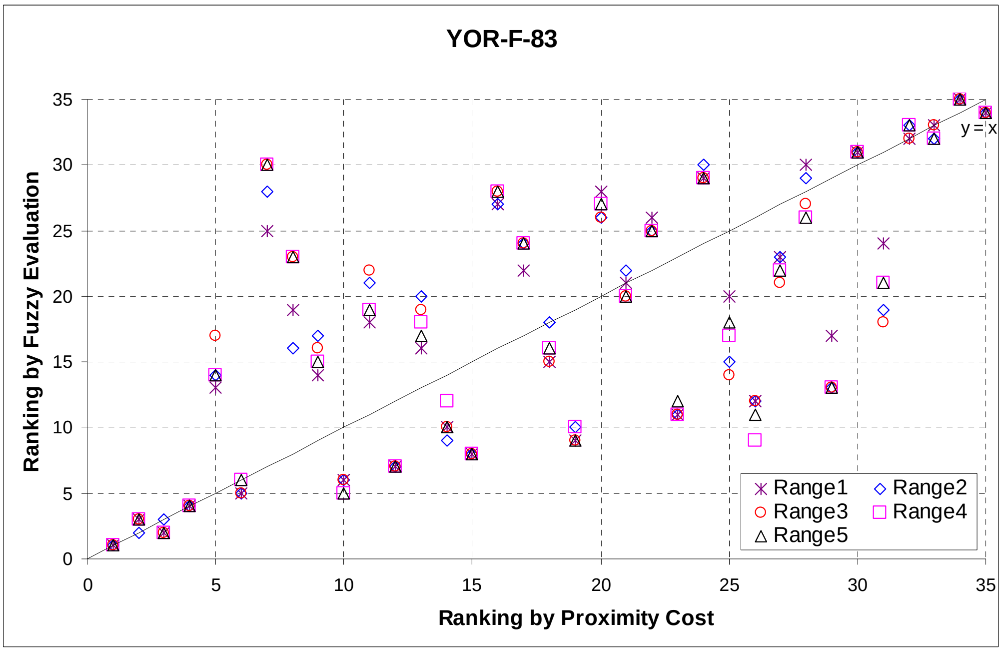


Figure 8.20: A comparison of rankings produced by the five boundary settings used for *YOR-F-83*

129 for *UTA-S-92*. In the case of *CAR-F-92*, the last three worst solutions have the same *highest penalty* value which is 132, while for solution in ranked 32^{th} has *highest penalty* is equal to 83. Details of the crisp values of *average penalty* and *highest penalty* for 35 solutions for each data set are presented in Appendix B. In these situations, the *average penalty* value has greater influence on the decision made by the fuzzy evaluation system, because the influence of *highest penalty* value on the decision will be the same for the different solutions (as the value are the same). Furthermore, none of the solutions ranked 6^{th} to 31^{st} (when ranked by proximity cost) is evaluated better than the five 'best' solutions when considering *average penalty* and *highest penalty* simultaneously.

There is no clear winner as to which boundary settings has given the most 'appropriate' ranking of solutions. In the end, it would be up to the practitioner to choose which boundary setting has produced solution ranking that is most satisfactory in reflecting his/her personal requirements. It might be better to employ the range that uses the actual minimum and maximum values of the criteria that are being considered in the evaluation if it is easy (in terms of resource availability and computational times) to construct a range of solution for testing purposes. On the other hand, the approximation approaches are more convenient if dealing with new timetabling problems (as no solutions need to be constructed).

Another interesting finding is in regard to the informal lower bound for proximity cost. This is the first time that an informal lower bound for proximity cost for each data set has been introduced. Figure 8.21 provides a graphical illustration of the ideal number of exams that can be placed in order to avoid any proximity cost penalty for the twelve data sets. In the Figure, the horizontal bar represents the time slots available for the particular data set; the vertical bar represents the location of exams that will impose zero penalty if enroled on by a particular student (that is, if a student has an exam at time slot 1, then the student's next exam must occur at, or after, time slot 7 if it is to incur zero proximity cost penalty). Simply counting the number of vertical bars crossed by the horizontal bar for a data set gives the maximum number of exams that a student may be enroled on before a proximity cost penalty must be incurred. Due to the limited number of time slots available, it is not always possible to assign all the enroled exams for a particular student in the ideal arrangement. By finding the minimum number of exams that cause penalty, it is obvious that a proximity cost will be imposed if the maximum number of exams enroled by any student is equal to or larger than the determined minimum number of exams that cause penalty. Comparing the second and third columns of Table 8.10 indicates that, for all of the twelve benchmark data sets, none of the values in the third column are less than the values in the second column. That is, the maximum number of exams enroled on by at least one student is *higher* than the maximum number of exams that can be timetabled without proximity cost penalty. That means that **it is not possible to obtain a solution with zero proximity cost for any of the twelve benchmark data sets**. Taking into account the fact that the lower bound value is calculated based on an approximation approach (i.e. without constructing the actual feasible timetable), it is believed that this lower bound value can be used as benchmark for the purpose of comparing proximity costs. For any feasible solution, it is *not possible* to have proximity cost that is lower than the lower bound value for the particular data set as determined here (as shown in the second column of Table 8.8).

Recall that these *lowerLimit* proximity costs were calculated by taking into account students who are enroled for the minimum number of exams that cause penalty, and above. When constructing an actual physical timetable, it not always the case that students who are enroled for less than the minimum number of exams that cause penalty are guaranteed to be able to have their exams scheduled in an ideally arrangement (i.e. with no penalty imposed). This is due to the constraints amongst the exams that limit the available time slots for the placement of exams into a conflict free time slot. Hence, it is expected that any feasible physical timetable constructed should have proximity cost that is *higher* than the *lowerLimit* proximity costs proposed here.

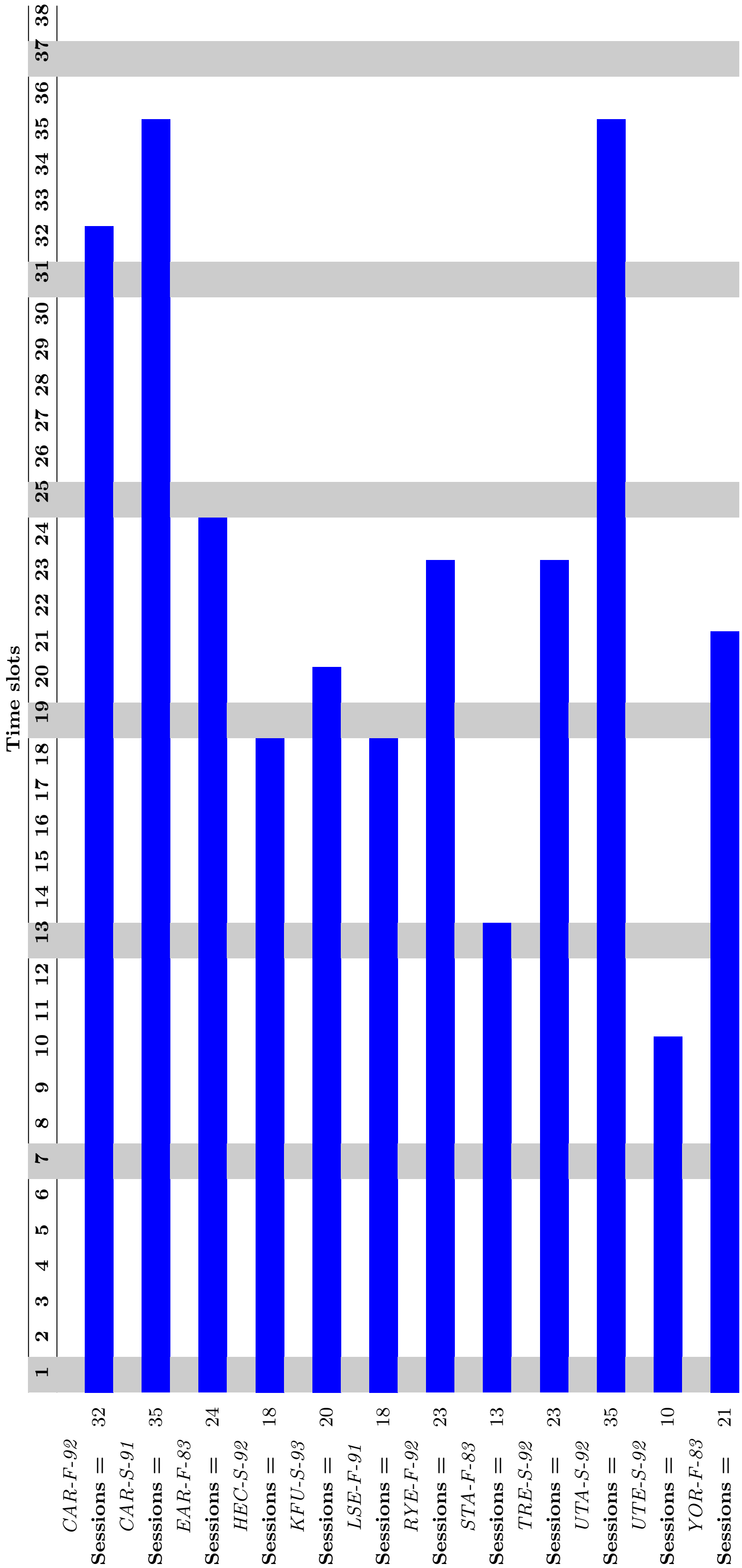


Figure 8.21: Illustrative of minimum exams that cause penalty

Table 8.10: Analysis of students involved in calculating the lower limit proximity cost

Data sets	**Minimum exams that cause penalty**	**Maximum exams enroled by any student**	**Total number of students**	**Students involved in penalty calculation**	**% of students involved**
CAR-F-92	7	7	18419	29	0.16
CAR-S-91	7	9	16925	32	0.19
EAR-F-83	5	10	1125	1102	97.96
HEC-S-92	4	7	2823	1836	65.04
KFU-S-93	5	8	5349	3797	70.99
LSE-F-91	4	8	2726	2245	82.36
RYE-F-92	5	10	11483	4789	41.71
STA-F-83	4	11	611	611	100.00
TRE-S-92	5	6	4360	1234	28.30
UTA-S-92	7	7	21266	23	0.11
UTE-S-92	3	6	2749	2553	92.87
YOR-F-83	6	14	941	763	81.08

8.5 Review of Previously Published Results

Lately, many researchers have published their work on finding better solutions for Carter *et al.*'s benchmark data set and there is much research which is still ongoing. Many of the latest publications have published results that have outperformed the results of earlier publications and the tendency to beat the current 'best' results still continues. As the authors of such published papers usually only provide the best solution results that they obtained (sometimes with average proximity cost and computational times), often there is no way of independently verifying their results. Currently (at the time of writing this thesis), one member of the Automated Scheduling, Optimisation and Planning (ASAP) Research Group (specifically Dr. Rong Qu) has initiated an effort to contact the authors with the 'best' published solutions in order to obtain their timetable solutions for verification. This is important in order to eliminate the confusing results that have (unfortunately) become prevalent between published papers due to the use of different data sets (under the same names) and/or different penalty functions.

Comparing the determined lower limit proximity cost with the published results on Carter *et al.*'s benchmarks presented in Table 4.9 has identified that, in the case of the *STA-F-83* data set, there are two papers in which the published results are lower than the lower limit value determined using the approach proposed here. The first paper was published by Casey and Thompson (2003) in which their 'best' result is 134.9. Dr. Qu has clarified that Casey and Thompson (2003) used a slightly different data set for *STA-F-83*. Apparently, due to an unfortunate error in which the data file was inadvertently altered, the data set that they used contained only 138 exams and 549 students. On the other hand, most of other published papers have used a data set that consists of 139 exams and 611 students. In the second paper, Yang and Petrovic (2005) published their result with a proximity cost for *STA-F-83* equal to 151.52. Via private communication, the corresponding author indicated that this solution has one exam unscheduled. This means that the solution was infeasible. The evaluated solution quality was calculated to be 151.52 on the basis that an extra penalty cost of 5000 was assigned to the solution to penalise the one unscheduled exam. The best feasible solution generated by the same author has a proximity cost of 158.35, which does not violate the lower limit proposed here.

Having mentioned the above example, it is believed that the determined lower limit proximity costs for Carter *et al.*'s benchmarks will be extremely beneficial to the timetabling research community. A comparison of the best (lowest proximity cost) results published in literature to date and the lower limit proposed in this thesis is shown in Table 8.11. Note that the determined lower limit for eleven of the data sets (excluding *STA-F-83*) are far lower than the 'best' published results. One possible reason for this is due to the number of students that are involved in the lower limit calculation. In the sixth column of Table 8.10, it can be seen that in three cases (*CAR-F-92*, *CAR-S-91* and *UTA-S-92*) less than 0.2% of the total students are involved in the calculations. For

Table 8.11: A comparison between lower limit with the 'best' results in literature

Data sets	Lower limit, L	'Best' results in literature, B	$((B-L)/B)*100$
CAR-F-92	0.008	3.93	99.80
CAR-S-91	0.006	4.00	99.85
EAR-F-83	17.847	29.30	39.09
HEC-S-92	3.495	9.20	62.02
KFU-S-93	5.634	13.00	56.66
LSE-F-91	2.765	9.60	71.20
RYE-F-92	3.787	6.80	44.31
STA-F-83	152.046	157.03	3.28
TRE-S-92	0.594	7.90	92.49
UTA-S-92	0.002	3.14	99.93
UTE-S-92	21.510	24.40	11.85
YOR-F-83	18.961	36.20	47.62

TRE-S-92 only 28.30% of the total students are involved. Whereas for *STA-F-83*, all of the students are involved in the lower limit calculation. Consequently, in Table 8.11 it can observed that for *CAR-F-92*, *CAR-S-91*, *UTA-S-92* and *TRE-S-92*, the determined *lowerLimit* proximity costs are at least 92.49% smaller than the 'best' published results. Nevertheless, it can be seen that the best result for *EAR-F-83* is within 40% of the lower limit, *RYE-F-92* is within 45%, *UTE-S-92* is within 12% and *YOR-F-83* is within 50%. Further, it can be seen that the best result for *STA-F-83* is within 4% of the lower limit (which, remember, may not be achievable by a feasible solution). These lower limits all provide researchers in the area a valuable new piece of information against which to compare their solutions.

8.6 Chapter Summary

The new algorithms, *BFLLAA* and *GLLAA*, provide (for the first time) an algorithmic method for deriving a lower limit to proximity cost for timetable solutions which can be calculated for any existing or novel data set. It is the first time that any lower limit of proximity cost has been published. Of course, a lower limit of zero has been implicitly assumed and, for some data sets (such as *CAR-F-92* and *CAR-S-91*) the new lower limit is not much above zero. On the other hand, for some data sets (most notably *STA-F-83*) the new lower limit is well above zero and is close to the best results observed. Indeed, for *STA-F-83* the new lower limit is less than 4% below the best published result in literature to date.

Of the two algorithms, *BFLLAA* gives more 'correct' results, in that the lower limit is a valid limit based on the underlying assumptions. However, it can take a long time to calculate and may, for novel real-world data sets, prove to take a prohibitively long time. The *GLLAA* variation provides a quick approximation to the lower limit proposed here. However, it is worth pointing out that the lower limit for proximity cost given by *GLLAA* is still lower than any of the published best results for any of the data sets; and this is for an algorithm that is very quick to run on any data set. Thus, in practice, it may be that obtaining a lower limit by *GLLAA* may be sufficient. It would appear from the analysis of the Carter data sets presented in this Chapter, that it is reasonable to state that any solution which is close to a limit given by *GLLAA* would represent a very high quality solution (in terms of proximity cost).

The lower limits presented here provide researchers in the area of timetabling a valuable new piece of information against which to compare their solutions for the Carter benchmark data sets. Further, they have provided, for the first time, guidance as to which of previously published results have been erroneous (or misleading) in that they have either utilised slightly different versions of the data sets (published under the same name) or have included infeasible solutions with arbitrary penalty costs for infeasibility 'hidden' within the measure of proximity cost penalty.

Taken as a whole, the methods outlined in the last two Chapters represent the first attempt to implement a more realistic evaluation of timetable solutions that is more appropriate to real-world contexts than an evaluation based on proximity cost alone. It does so by utilising fuzzy methods to combine two criteria, *average penalty* (proximity cost) and *highest penalty* (highest proximity cost for any one student). Although combining these two criteria using fuzzy methodologies is conceptually relatively simple, there are obstacles to the approach in practice. The problem of determining lower and upper limits of the criteria on which the assessment of quality is based is probably the most difficult challenge. This Chapter has presented methods for deriving appropriate lower and upper limits for proximity cost (currently the most common criterion for assessing timetable quality). Clearly more research will need to be undertaken on any new criteria used for evaluation of quality but it is hoped that the methods presented here will provide a starting point for all such research.

Chapter 9

Conclusions and Future Work

In this thesis fuzzy methodologies have been investigated in an attempt to construct solutions to university timetabling problems and to evaluate the timetable quality. This study focuses on exploring the basic but powerful features of fuzzy methodologies. In this context, the 'basic feature' is the concept of membership degree in fuzzy sets. The use of fuzzy boundaries instead of sharp boundaries as in classical sets has made possible the use of everyday linguistic terms in the development of computer systems. Another strength of fuzzy methodologies that is explored is the mechanism of fuzzy reasoning that naturally provides the platform for considering simultaneously more than one attribute (or factor) in decision making. This feature may be closer to human thinking and perception than other methods of combining multiple criteria. In this sense, fuzzy methodologies seem to provide mechanisms that more closely mimic the way human beings make decisions. In this Chapter, a list of contributions drawn from this research is provided, followed by a brief outline of some possibilities for future research.

9.1 Summary of Contributions

9.1.1 Fuzzy Construction of Timetables

The first theme of this thesis is the use of fuzzy techniques in the construction of timetable solutions. As far as the author is aware, this thesis is the first work to develop and analyse the simultaneous use of multiple heuristic to determine orderings. Different combinations of multiple heuristic orderings were examined, considering five graph-based heuristic orderings — *Largest Degree*, *Saturation Degree*, *Largest Enrolment*, *Largest Coloured Degree* and *Weighted Largest Degree*. This analysis has provided some key insights regarding the implementation of multiple heuristic orderings. Particularly, it has been demonstrated from the research findings that:

1. Generally, the fuzzy multiple heuristic orderings (with parameter tuning) have outperformed all of the single heuristic orderings.

2. Employing fuzzy techniques to measure the relative importance of each of the considered heuristic orderings produces better solutions compared to using non-fuzzy linear weighting factors.

3. Overall, considering three heuristic orderings produced better results compared to two heuristic orderings.

4. For any given heuristic ordering, incorporating a stochastic element in the time slot selection may permit better solutions to be found, as a bigger search space is explored.

5. The timetable solutions constructed by means of fuzzy constructive algorithms were comparable to the solutions produced with more sophisticated optimisation approaches developed by other researchers.

While considering multiple heuristic orderings for constructing feasible timetable solutions is, in itself, an original contribution, several other achievements are outlined, as follows:

1. Integrating fuzzy techniques in the basic sequential constructive algorithm. This approach provides a more realistic scheme for measuring the difficulty of assigning exams to time slots. Although the five graph-based heuristic orderings implemented in this research are well known within the timetabling community, each heuristic ordering is usually employed on its own. While each heuristic ordering can be used individually (usually with a 'backtracking' algorithm) to construct feasible solutions, it is interesting to see the effect of employing more than one heuristic ordering simultaneously. As expected, more accurate ordering of exams, in terms of their difficulty to schedule, were obtained when several heuristic orderings are combined.

2. Experimental results presented in Chapter 5 justified that it is worth exploring a more advanced approach such as the use of fuzzy techniques instead of using the simple linear weighting function when more than one factor needs to be considered in making decisions.

3. The developed approach produces reasonably good solutions when applied to benchmark exam and course timetabling problem instances. These promising results might suggest that this approach can be implemented in other combinatorial problems that can be represented as the graph colouring problem.

4. A comprehensive comparison of twenty combinations of two and three heuristic orderings that have been tested in Chapters 4 to 6 can be used as a guideline to choose which heuristic ordering combination is more suitable for particular problem instances.

9.1.2 Fuzzy Evaluation of Timetables

The second theme of this thesis is concerned with a new evaluation function for examination timetabling problems. In order to evaluate the fairness of the constructed timetables, two evaluation criteria, namely the proximity cost (average penalty per student) and the highest penalty among students, are considered. The evaluation function is modeled as a fuzzy system in order to take the advantage of the powerful features of fuzzy reasoning.

One common problem in developing a fuzzy system is the difficulty in defining the appropriate fuzzy model for the variables involved. In this thesis, a fixed fuzzy model was developed based on a common sense view of how one would define a 'fair' timetable when the above evaluation criteria are considered. The major problem that arose was related to determining the boundary settings for the universe of discourse of the membership functions. In the initial investigation, the boundary settings used for input normalisation were based on the minimum and maximum values of the constructed timetable solutions being evaluated. Then, new algorithms were developed to calculate the proximity cost based only on the underlying structure of the problem instances, without needing to build the actual physical timetable. Initially, this work was intended for the purpose of identifying the lower and upper bound of the universe of discourse for the fuzzy membership functions, particularly for the *average penalty* membership function. Subsequently, it was realised that the outcome of this work was, for the first time, a non-zero lower bound for timetable problems. This has provided valuable new information for the examination timetabling community, particularly in checking the validity of published results.

The work carried out has also made several original contributions to the state of the art. These are outlined as follows:

1. The development of a fuzzy based evaluation function for examination timetabling.

The presented approach provides a more realistic evaluation of timetables with regard to real-world timetabling problems in which the decision to choose the 'best' timetable is affected by more subjective factors than proximity cost alone.

2. The creation of a novel algorithm and an associated formula for measuring approximate proximity cost without having to build the physical timetable. This cost can provide researchers and practitioners with an idea of how good a solution to a previously unseen timetable instance is, without needing to construct alternative solutions for comparison.

3. The establishment of unofficial lower limit for the uncapacitated problem of Carter *et al.*'s benchmark data set. For the first time, an investigation of the lower limit for the proximity cost is presented. In addition to the requirement of a validation tool for timetable solutions discussed by Schaerf and Di Gaspero (2006), the proposed lower limit for proximity cost can be used instantly to check the validity of timetable solutions (any feasible solutions cannot have penalty value lower than the proposed lower limit).

9.2 Future Research

As mentioned earlier in Chapter 2, the amount of published literature on fuzzy methodologies for educational timetabling problems is very limited. It is obvious that the research work presented in this thesis has opened a new line of research in which a number of avenues of future work remain to be investigated.

Improvement of Initial Solutions. Having demonstrated that good quality initial solutions can be obtained using fuzzy multiple heuristic orderings within the simple sequential constructive algorithm, a particularly important future direction is to apply optimisation algorithms to iteratively improve the initial solutions. Such work would answer the question "Does an initial solution generated with fuzzy approach lead to better solutions compared with initial solutions generated by single or random heuristics?".

Improvement of the Fuzzy Modeling Technique. Observation of the attempts to identify fuzzy models based on simple exhaustive search and the stochastic approach presented in Section 6.4 suggest that the proposed fuzzy models for multiple heuristic ordering could be further improved. Of particular interest would be to employ more sophisticated optimisation algorithms in order to identify fuzzy model parameters (i.e. the shape of membership functions and the fuzzy rule set). The selection of heuristic ordering to be combined could also be incorporated into such fuzzy model optimisation. If a reliable model optimisation technique could be developed, it might then be possible to consider the combination of four and five heuristic orderings simultaneously, as determination of relevant fuzzy rules could then be performed automatically. This could overcome the fact that the number of fuzzy rules exponentially increases with the increase of input variables. In defining the behaviour of a fuzzy system, it is usually the case that the number of fuzzy rules required is much less than the actual possible number of rules.

Deriving a Generic Fuzzy Model. As noted in Chapters 4 and 5, in order to obtain the best initial solution, it is necessary to tune the fuzzy model for the particular data set. Therefore, another important avenue to explore is to search for generic fuzzy model(s), i.e. a model which able to guide the search algorithm to quite a good solution that is applicable across a range of problem instances. It is not expected that it will be possible to obtain the 'optimal' fuzzy model for any problem instance but more research is required on identifying a generic model that can produce quite satisfactory solution qualities that are better than any single heuristic ordering. The obvious benefit of such a fuzzy model would be that no tuning would be needed for each new problem instance.

Enrichment of the Fuzzy Evaluation Function. The fuzzy evaluation function proposed here is clearly extendable to include more criteria. The initial investigation presented in Chapters 7 and 8 only uses two decision criteria to evaluate the timetable quality. One possible direction for future research includes extending the application of the fuzzy evaluation system to real world educational timetabling problems in which more criteria are considered in the evaluation of timetables. Another aspect to be investigated further is in comparing the quality assessments produced by such fuzzy approaches with the subjective assessments of quality that timetabling officers make in real-world timetabling problems.

Furthermore, this fuzzy evaluation approach could be implemented in the context of choosing the next move during the exploration of the neighbourhood in any iterative improvement optimisation algorithm.

9.3 Dissemination

The research described in this thesis has been disseminated in conferences and publications in the field of timetabling and fuzzy applications. The following is the list of papers that have been produced.

9.3.1 Journal Paper

- Asmuni, H., Burke, E. K., Garibaldi, J. M., McCollum, B. & Parkes, A. J. An Investigation of Fuzzy Multiple Heuristic Orderings in the Construction of University Examination Timetables. Accepted to be published in the *Computers & Operations Research* journal.

9.3.2 Conference Papers

- Asmuni, H., Burke, E.K. & Garibaldi, J.M. (2004). A Comparison of Fuzzy and Non-Fuzzy Ordering Heuristics for Examination Timetabling. In A. Lotfi, ed., *Proceedings of 5th International Conference on Recent Advances in Soft Computing 2004*, 288–293, Nottingham, United Kingdom, 16th - 18th December 2004.

- Asmuni, H., Burke, E.K. & Garibaldi, J.M. (2005a). Fuzzy Multiple Heuristic Orderings for Course Timetabling Problem. In B. Mirkin & G. Magoulas, eds., *Proceedings of the 2005 UK Workshop on Computational Intelligence (UKCI 2005)*, 302–309, Birkbeck University of London, London, 5th - 7th September 2005.

- Asmuni, H., Burke, E.K., Garibaldi, J.M. & McCollum, B. (2005b). Fuzzy Multiple Heuristic Orderings for Examination Timetabling. In Burke and Trick (2005), 334–353. The earlier version of this paper appeared in Burke, E.K. & Trick, M.A., eds., *Proceedings of 5th International Conference on the Practice and Theory of Automated Timetabling 2004*, 51–66, Pittsburgh, USA, 18th - 20th August 2004.

- Asmuni, H., Burke, E.K., Garibaldi, J.M. & McCollum, B. (2006). A Novel Fuzzy Approach to Evaluate the Quality of Examination Timetabling. In E.K. Burke & H. Rudova, eds., *Proceedings of 6th International Conference on the Practice and Theory of Automated Timetabling VI (PATAT) 2006*, 82–102, Brno, Czech Republic, 30th August - 1st September 2006.

- Asmuni, H., Burke, E.K., Garibaldi, J.M. & McCollum, B. (2007). A Novel Fuzzy Approach to Evaluate the Quality of Examination Timetabling. In E.K. Burke and H. Rudova, eds., *6th International Conference, PATAT 2006 Brno, Czech Republic, August 30-September 1, 2006 Revised Selected Papers*, vol. 3867 of *LNCS*, 327–346.

- Asmuni, H., Burke, E.K., Garibaldi, J.M. & McCollum, B. (2007). Determining Rules in Fuzzy Multiple Heuristic Orderings for Constructing Examination Timetables. In P. Baptiste, G. Kendall, A. Munier-Kordon and F. Sourd, eds., *Proceedings of the 3rd Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA 2007), Paris, August 28-31, 2007*, 59–66.

9.3.3 Abstract

- CORS / Optimization Days 2006 Joint Conference,Montreal, 8th-10th May, 2006. Asmuni, H., Burke, E.K., Garibaldi, J.M. & McCollum, B. On Evaluating the Quality of Automatically Generated Examination Timetables.

Appendix A

Analysis of Modified Algorithms

Table A.1: Analysis of Changes in Algorithm for *Tuned Fuzzy LD+LE Model*

Data Set		Algo1.0	Algo1.1	Algo1.2	Algo2.0
CAR-F-92					
Proximity Cost	Best	4.62	4.62	4.62	4.62
	Average	4.64	4.63	4.63	4.64
	Worst	4.64	4.64	4.64	4.65
Comp. Time (s)	Shortest	1.80	1.64	1.80	1.63
	Average	1.86	1.67	1.83	1.65
	Worst	2.02	1.70	1.88	1.67
Backtracking	Min	1	1	1	1
	Average	1	1	1	1
	Max	1	1	1	1
CAR-S-91					
Proximity Cost	Best	5.58	5.56	5.57	5.60
	Average	5.65	5.59	5.67	5.62
	Worst	5.81	5.63	5.82	5.65
Comp. Time (s)	Shortest	6.34	3.05	5.05	3.09
	Average	13.48	3.26	11.65	3.33
	Worst	30.44	3.63	21.95	3.64

Continued on Next Page...

Data Set		Algo1.0	Algo1.1	Algo1.2	Algo2.0
Backtracking	Min	5	3	4	3
	Average	12.2	3.8	9.6	4.4
	Max	27	6	18	6
EAR-F-83					
Proximity Cost	Best	44.27	43.03	43.96	42.73
	Average	45.09	44.40	45.19	44.57
	Worst	46.41	46.46	47.11	47.56
Comp. Time (s)	Shortest	1.02	1.13	1.13	0.61
	Average	1.21	2.12	1.51	1.39
	Worst	1.48	5.05	2.13	3.34
Backtracking	Min	15	18	16	11
	Average	18.8	28.2	21.4	19.8
	Max	24	57	32	43
HEC-S-92					
Proximity Cost	Best	12.84	12.35	12.56	12.35
	Average	13.79	12.57	15.29	12.51
	Worst	15.91	12.80	19.31	12.72
Comp. Time (s)	Shortest	0.14	0.09	0.20	0.08
	Average	0.33	0.12	0.48	0.13
	Worst	0.55	0.25	0.81	0.25
Backtracking	Min	9	3	7	4
	Average	24	4	33	4.4
	Max	46	5	59	5
KFU-S-93					
Proximity Cost	Best	16.54	15.99	16.59	15.84
	Average	17.60	16.35	17.29	16.13
	Worst	19.17	16.72	18.72	16.24
Comp. Time (s)	Shortest	2.27	0.77	1.78	0.81

Continued on Next Page...

Data Set		Algo1.0	Algo1.1	Algo1.2	Algo2.0
	Average	3.64	0.83	3.42	0.90
	Worst	5.52	0.92	5.06	1.17
Backtracking	Min	12	4	10	6
	Average	18	7	16.8	7.2
	Max	25	9	24	8
LSE-F-91					
Proximity Cost	Best	12.35	12.35	12.35	12.35
	Average	12.35	12.35	12.35	12.35
	Worst	12.35	12.35	12.35	12.35
Comp. Time (s)	Shortest	0.44	0.45	0.45	0.45
	Average	0.45	0.48	0.45	0.46
	Worst	0.45	0.50	0.45	0.47
Backtracking	Min	0	0	0	0
	Average	0	0	0	0
	Max	0	0	0	0
RYE-F-92					
Proximity Cost	Best	12.17	12.14	11.68	11.51
	Average	12.70	12.68	12.11	11.93
	Worst	13.18	13.65	12.71	12.62
Comp. Time (s)	Shortest	2.03	1.00	1.42	1.02
	Average	8.17	1.60	3.34	1.60
	Worst	19.39	3.77	7.05	3.83
Backtracking	Min	7	3	4	3
	Average	40	7.2	12.8	7.2
	Max	96	18	29	20
STA-F-83					
Proximity Cost	Best	160.42	159.82	160.42	159.82
	Average	160.42	160.18	160.42	160.06

Continued on Next Page...

Data Set		Algo1.0	Algo1.1	Algo1.2	Algo2.0
	Worst	160.42	160.42	160.42	160.42
Comp. Time (s)	Shortest	0.14	0.11	0.14	0.11
	Average	0.14	0.13	0.14	0.13
	Worst	0.16	0.20	0.16	0.20
Backtracking	Min	7	1	7	1
	Average	7	1	7	1
	Max	7	1	7	1
TRE-S-92					
Proximity Cost	Best	9.05	9.06	9.05	9.06
	Average	9.05	9.12	9.05	9.09
	Worst	9.05	9.17	9.05	9.17
Comp. Time (s)	Shortest	0.41	0.38	0.41	0.38
	Average	0.42	0.39	0.42	0.40
	Worst	0.42	0.41	0.47	0.48
Backtracking	Min	1	1	1	1
	Average	1	1.2	1	1.2
	Max	1	2	1	2
UTA-S-92					
Proximity Cost	Best	3.87	3.85	3.88	3.86
	Average	4.23	3.85	3.98	3.88
	Worst	4.64	3.86	4.13	3.90
Comp. Time (s)	Shortest	11.55	2.25	15.95	2.25
	Average	31.11	2.28	28.36	2.36
	Worst	56.08	2.31	40.34	2.55
Backtracking	Min	10	2	18	2
	Average	29.8	2	29	2.8
	Max	49	2	42	4

Continued on Next Page...

Data Set		Algo1.0	Algo1.1	Algo1.2	Algo2.0
UTE-S-92					
Proximity Cost	Best	28.68	28.59	28.65	28.59
	Average	28.70	28.65	28.71	28.63
	Worst	28.74	28.67	28.74	28.69
Comp. Time (s)	Shortest	0.14	0.13	0.14	0.14
	Average	0.14	0.13	0.14	0.15
	Worst	0.16	0.16	0.14	0.17
Backtracking	Min	1	1	1	1
	Average	1	1.2	1	1.6
	Max	1	2	1	2
YOR-F-83					
Proximity Cost	Best	41.54	42.06	41.30	42.06
	Average	42.64	43.05	43.05	43.98
	Worst	43.53	43.98	44.13	46.37
Comp. Time (s)	Shortest	0.69	0.70	0.75	0.63
	Average	1.63	1.64	1.55	1.31
	Worst	3.19	4.61	2.19	1.92
Backtracking	Min	14	17	16	13
	Average	34.6	37	35.4	28
	Max	73	100	49	40

Table A.2: Analysis of Changes in Algorithm for *Tuned Fuzzy SD+LE Model*

Data Set		Algo1.0	Algo1.1	Algo1.2	Algo2.0
CAR-F-92					
Proximity Cost	Best	4.54	4.54	6.45	4.57
	Average	4.54	4.54	6.74	4.57
	Worst	4.54	4.54	7.09	4.57
Comp. Time (s)	Shortest	397.53	391.06	186.77	81.52
	Average	399.25	394.59	223.92	82.39
	Worst	402.17	397.27	307.50	83.94
Backtracking	Min	0	0	228	0
	Average	0	0	288	0
	Max	0	0	403	0
CAR-S-91					
Proximity Cost	Best	5.29	5.29	6.20	5.59
	Average	5.29	5.29	6.54	5.59
	Worst	5.29	5.29	7.06	5.59
Comp. Time (s)	Shortest	902.27	885.14	25.94	183.50
	Average	905.15	889.51	102.30	184.08
	Worst	908.56	900.41	199.91	184.38
Backtracking	Min	0	0	28	0
	Average	0	0	84.4	0
	Max	0	0	152	0
EAR-F-83					
Proximity Cost	Best	37.02	37.02	50.51	42.52
	Average	37.02	37.02	52.01	44.03
	Worst	37.02	37.02	54.39	45.31
Comp. Time (s)	Shortest	19.06	18.77	5.05	4.05

Continued on Next Page…

Data Set		Algo1.0	Algo1.1	Algo1.2	Algo2.0
	Average	19.14	18.83	7.19	4.12
	Worst	19.22	18.94	9.83	4.20
Backtracking	Min	0	0	63	4
	Average	0	0	90.2	5.6
	Max	0	0	128	7
HEC-S-92					
Proximity Cost	Best	11.78	11.78	17.01	12.00
	Average	11.78	11.78	18.16	13.18
	Worst	11.78	11.78	19.79	16.52
Comp. Time (s)	Shortest	2.16	2.16	0.45	0.47
	Average	2.22	2.22	0.59	0.63
	Worst	2.45	2.41	0.78	0.83
Backtracking	Min	1	1	32	2
	Average	1	1	38.8	8.8
	Max	1	1	53	26
KFU-S-93					
Proximity Cost	Best	15.81	15.81	22.20	17.48
	Average	15.81	15.81	23.79	17.48
	Worst	15.81	15.81	25.48	17.48
Comp. Time (s)	Shortest	106.31	104.88	11.55	22.30
	Average	107.86	106.49	18.16	22.36
	Worst	109.34	107.56	28.56	22.48
Backtracking	Min	0	0	56	0
	Average	0	0	79	0
	Max	0	0	125	0
LSE-F-91					
Proximity Cost	Best	12.09	12.09	17.89	12.87
	Average	12.09	12.09	18.09	12.87

Continued on Next Page. . .

Data Set		Algo1.0	Algo1.1	Algo1.2	Algo2.0
	Worst	12.09	12.09	18.25	12.87
Comp. Time (s)	Shortest	68.31	67.27	14.88	13.97
	Average	68.57	67.42	19.73	14.06
	Worst	68.77	67.58	30.83	14.11
Backtracking	Min	0	0	120	0
	Average	0	0	156.6	0
	Max	0	0	254	0
RYE-F-92					
Proximity Cost	Best	10.38	10.38	12.33	11.06
	Average	10.38	10.38	13.16	11.06
	Worst	10.38	10.38	13.97	11.06
Comp. Time (s)	Shortest	183.88	180.97	4.98	38.09
	Average	185.25	182.15	12.70	38.70
	Worst	187.16	183.97	26.47	39.59
Backtracking	Min	0	0	19	0
	Average	0	0	54.2	0
	Max	0	0	116	0
STA-F-83					
Proximity Cost	Best	160.75	160.75	172.42	168.86
	Average	160.75	160.75	172.42	168.86
	Worst	160.75	160.75	172.42	168.86
Comp. Time (s)	Shortest	6.23	6.13	0.20	1.31
	Average	6.36	6.29	0.21	1.33
	Worst	6.81	6.86	0.22	1.36
Backtracking	Min	0	0	16	0
	Average	0	0	16	0
	Max	0	0	16	0

Continued on Next Page. . .

Data Set		Algo1.0	Algo1.1	Algo1.2	Algo2.0
TRE-S-92					
Proximity Cost	Best	8.67	8.67	11.76	9.57
	Average	8.67	8.67	12.32	9.72
	Worst	8.67	8.67	13.24	10.02
Comp. Time (s)	Shortest	42.88	42.38	4.00	8.88
	Average	43.00	42.45	5.38	8.97
	Worst	43.28	42.56	7.31	9.22
Backtracking	Min	0	0	36	1
	Average	0	0	48	1.6
	Max	0	0	63	2
UTA-S-92					
Proximity Cost	Best	3.57	3.57	4.19	3.82
	Average	3.57	3.57	4.52	3.86
	Worst	3.57	3.57	4.82	3.88
Comp. Time (s)	Shortest	600.81	590.97	47.30	124.19
	Average	602.47	591.97	119.23	124.61
	Worst	603.81	593.52	279.67	125.11
Backtracking	Min	0	0	51	1
	Average	0	0	121.2	1
	Max	0	0	274	1
UTE-S-92					
Proximity Cost	Best	28.07	28.07	36.16	29.16
	Average	28.07	28.07	37.59	29.60
	Worst	28.07	28.07	39.69	30.62
Comp. Time (s)	Shortest	11.13	11.09	0.22	2.38
	Average	11.31	11.15	0.27	2.55
	Worst	11.41	11.25	0.36	2.75

Continued on Next Page…

Data Set		**Algo1.0**	**Algo1.1**	**Algo1.2**	**Algo2.0**
Backtracking	Min	1	1	8	5
	Average	1	1	10.8	22.2
	Max	1	1	15	58
YOR-F-83					
Proximity Cost	Best	39.80	39.80	51.73	44.62
	Average	39.80	39.80	52.09	45.35
	Worst	39.80	39.80	52.46	46.78
Comp. Time (s)	Shortest	22.36	22.02	3.08	4.63
	Average	22.39	22.09	4.02	4.89
	Worst	22.45	22.17	5.34	5.34
Backtracking	Min	0	0	53	5
	Average	0	0	74.2	7
	Max	0	0	102	10

Appendix B

Crisp Values for the 35 Solutions

Values for average penalty and highest penalty for the 35 solutions for the 12 data sets.

Table B.1: Crisp values of *average penalty* and *highest penalty* for *CAR-F-92*, *CAR-S-91* and *EAR-F-83*.

Ranking	*CAR-F-92*		*CAR-S-91*		*EAR-F-83*	
	average penalty	*highest penalty*	*average penalty*	*highest penalty*	*average penalty*	*highest penalty*
1	4.54422	65	5.29182	68	37.01778	116
2	4.62376	71	5.57294	75	41.17778	144
3	4.63923	71	5.65448	75	41.32444	131
4	4.64325	71	5.68804	83	41.85956	118
5	5.14805	68	5.68975	83	43.62756	129
6	5.19241	69	5.84201	75	43.63733	105
7	5.25045	76	5.91131	68	44.14667	118
8	5.50850	68	5.92502	83	44.96267	146
9	5.53228	68	5.94783	83	44.96800	127
10	5.58228	75	6.07876	76	44.98044	135
11	5.59466	68	6.39297	71	45.82578	146
12	5.60872	68	6.41448	83	46.81867	148
13	5.61670	68	6.50866	71	48.69511	188
14	5.67224	83	6.55433	85	49.26667	131
15	5.72262	77	6.55474	98	49.51822	167
16	5.76204	75	6.62381	101	49.52178	159
17	5.76513	68	6.67332	83	49.55467	158
18	5.96075	75	6.91628	75	49.66222	114
19	6.08958	68	6.94635	71	49.78311	144
20	6.27222	68	6.95403	84	49.84800	194
21	6.32857	68	7.10576	75	50.26578	130
22	6.34774	84	7.11728	83	50.54933	148
23	6.48960	75	7.20620	71	50.99378	159
24	6.68288	68	7.27131	83	51.55200	147
25	6.68891	83	7.60360	71	51.79911	137
26	6.78636	83	7.63391	98	52.28356	148
27	6.84049	68	7.76006	76	53.01156	136
28	6.98849	71	7.82151	84	53.14311	167
29	6.98958	84	8.01022	69	54.44889	149
30	6.99294	77	8.32804	83	54.50489	160
31	7.30794	77	8.87297	98	55.09511	198
32	7.99072	83	13.10665	164	57.03378	149
33	11.28563	132	13.25058	164	60.16000	176
34	11.30110	132	13.30192	164	67.60533	198
35	11.42386	132	13.33489	164	71.27911	198

Table B.2: Crisp values of *average penalty* and *highest penalty* for *HEC-S-92*, *KFU-S-93* and *LSE-F-91*.

Ranking	*HEC-S-92*		*KFU-S-93*		*LSE-F-91*	
	average penalty	*highest penalty*	*average penalty*	*highest penalty*	*average penalty*	*highest penalty*
1	11.78498	83	15.81342	98	12.09391	87
2	13.23592	84	16.46326	113	12.34886	102
3	13.77365	106	16.47149	113	13.45781	78
4	14.12363	94	16.49991	113	14.71974	89
5	14.16188	83	16.49991	113	16.11262	160
6	14.21714	85	16.49991	113	16.40829	91
7	14.63479	83	16.90447	101	16.44901	132
8	14.64081	98	16.91400	124	16.48606	109
9	14.77400	98	16.91999	113	16.65737	115
10	14.77435	75	17.33614	100	16.74248	122
11	14.87070	99	17.91961	104	16.84740	108
12	15.05066	98	18.26640	114	17.17425	105
13	15.59369	78	18.26827	114	17.48496	117
14	15.65781	98	18.62311	125	17.55686	160
15	15.76337	84	18.96654	118	17.64050	121
16	15.88771	98	19.02225	129	17.69919	144
17	15.91144	75	19.12600	113	17.94167	98
18	16.25717	98	19.37858	113	18.07520	119
19	16.49203	90	19.52384	131	18.12252	127
20	16.53737	98	19.98187	118	18.18305	106
21	16.70705	113	20.00916	131	18.48239	126
22	17.12611	129	20.02225	102	18.56420	93
23	17.23521	113	20.31595	131	18.97946	95
24	18.89586	113	20.42756	114	19.31805	103
25	18.92597	77	20.90428	126	19.36207	114
26	19.08254	106	22.80052	108	20.13720	118
27	19.70988	83	23.21817	111	20.13830	104
28	20.06801	87	24.07758	113	20.80227	129
29	20.33546	129	25.05721	105	21.87748	111
30	21.58519	98	25.22247	119	26.01761	136
31	23.17499	113	25.58478	129	27.02128	133
32	23.45484	113	26.31501	121	28.16288	142
33	23.85264	106	27.01383	115	30.01761	136
34	28.52426	136	28.59563	134	32.13610	191
35	31.88027	112	43.39877	191	32.37821	161

Table B.3: Crisp values of *average penalty* and *highest penalty* for *RYE-F-92*, *STA-F-83* and *TRE-S-92*.

Ranking	*RYE-F-92*		*STA-F-83*		*TRE-S-92*	
	average penalty	*highest penalty*	*average penalty*	*highest penalty*	*average penalty*	*highest penalty*
1	10.38378	87	160.74632	227	8.67064	77
2	11.60185	114	161.15057	227	9.03945	75
3	11.71001	111	164.37480	228	9.31101	69
4	11.71959	111	167.39444	227	9.38922	68
5	11.82783	111	168.19476	227	9.59794	68
6	12.09449	105	168.78069	232	9.75665	71
7	12.18035	97	168.86252	227	9.85596	75
8	12.26430	98	169.09984	227	9.88486	68
9	12.33406	122	170.35516	284	9.98119	77
10	12.33693	97	171.24877	227	10.00344	68
11	12.41723	102	171.39116	227	10.25344	83
12	12.97614	97	171.92471	227	10.36239	98
13	13.13716	138	172.11620	227	10.42546	80
14	13.24872	139	172.17021	227	10.62821	83
15	13.64269	129	172.60393	227	10.68073	83
16	13.67848	104	173.12602	227	10.68739	77
17	13.68266	110	173.50245	230	10.69679	98
18	13.74980	121	173.50409	227	10.70826	68
19	13.76295	120	173.56301	268	10.72523	83
20	13.87564	107	175.55483	227	10.82523	75
21	14.02639	121	175.77414	233	10.96651	75
22	14.41662	130	176.29951	227	10.97821	98
23	14.44135	104	176.65794	239	11.00757	75
24	14.58051	104	177.53191	236	11.01835	68
25	14.61691	138	177.86579	227	11.15757	71
26	14.80632	121	178.40098	227	11.29358	75
27	15.72847	135	178.87234	233	11.44817	84
28	16.31629	138	180.63011	248	11.74725	98
29	17.44953	125	181.09984	227	12.05757	83
30	18.88392	122	181.12275	260	12.26812	98
31	21.21197	122	182.29787	227	12.79633	98
32	32.28956	191	182.72668	242	13.10482	77
33	34.82827	191	184.73650	268	13.70229	83
34	35.50649	175	186.48445	227	17.18280	129
35	36.71062	175	194.53191	284	17.24610	129

Table B.4: Crisp values of *average penalty* and *highest penalty* for *UTA-S-92*, *UTE-S-92* and *YOR-F-83*.

Ranking	*UTA-S-92*		*UTE-S-92*		*YOR-F-83*	
	average penalty	*highest penalty*	*average penalty*	*highest penalty*	*average penalty*	*highest penalty*
1	3.56729	63	28.06945	90	39.80128	234
2	3.83321	68	29.22473	104	44.15834	233
3	3.91080	68	29.69491	98	44.41233	231
4	3.92711	68	29.71818	86	45.64506	228
5	3.97724	68	29.88400	129	45.66844	259
6	4.14253	68	30.32291	83	45.73645	238
7	4.29865	84	30.50836	101	45.78108	301
8	4.53122	73	30.55527	98	45.93305	267
9	4.57279	73	31.21964	98	46.56642	258
10	4.58069	73	31.48582	113	46.80978	234
11	4.88413	84	31.52182	91	46.82784	262
12	4.96718	75	31.65273	104	46.86185	235
13	4.97583	68	31.65455	98	46.87885	259
14	5.00912	86	32.14400	105	46.94687	244
15	5.09212	87	32.32691	98	47.14240	240
16	5.11304	83	32.49964	98	47.20616	286
17	5.24711	69	32.80400	88	47.36663	268
18	5.28026	76	32.99600	94	47.37938	256
19	5.32978	83	33.13855	113	47.39639	242
20	5.39095	72	33.93527	91	47.71945	281
21	5.39636	98	34.36545	113	47.87779	260
22	5.54434	77	34.59964	104	47.91923	275
23	5.60651	77	34.92764	90	48.89479	238
24	5.62358	84	35.17527	113	49.31775	284
25	5.63073	106	35.66982	113	49.37088	252
26	5.65485	85	35.98545	129	50.53879	232
27	5.66632	71	36.18691	98	50.68332	256
28	5.69482	77	36.63564	105	50.76302	277
29	5.83354	84	36.84909	98	50.92774	241
30	6.04223	86	37.45782	106	51.51753	289
31	6.32531	78	38.76545	106	52.63124	248
32	8.65259	129	39.64618	98	53.13390	331
33	8.76568	129	41.31382	98	56.90436	291
34	8.78125	129	43.41345	129	63.90223	306
35	8.79253	129	56.34291	129	64.48140	295

Bibliography

ABBOUD, N., INUIGUICHI, M., SAKAWA, M. and UEMURA, Y. (1998). Manpower Allocation Using Genetic Annealing. *European Journal of Operational Research*, **111**, 405–420.

ABDULLAH, S. (2006). *Heuristic Approaches for University Timetabling Problems.*. Ph.D. thesis, School of Computer Science and Information Technology, The University of Nottingham, United Kingdom.

ABDULLAH, S. and BURKE, E.K. (2006). A Multi-start Large Neighbourhood Search Approach with Local Search Methods for Examination Timetabling. In D. Long, S.F. Smith, D. Borrajo and L. McCluskey, eds., *The International Conference on Automated Planning and Scheduling (ICAPS 2006), Cumbria, UK, 6-10 June 2006.*, 334–337.

ABDULLAH, S., BURKE, E.K. and MCCOLLUM, B. (2005). An Investigation of Variable Neighbourhood Search for University Course Timetabling. In *Proceedings of The 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2005), New York, USA, July 18th-21st*, 413–427.

ABDULLAH, S., AHMADI, S., BURKE, E. and DROR, M. (2006a). Investigating Ahuja-Orlin's Large Neighbourhood Search Approach for Examination Timetabling. *OR Spectrum*, **29**, 351–372.

ABDULLAH, S., AHMADI, S., BURKE, E.K., DROR, M. and MCCOLLUM, B. (2006b). A Tabu-based Large Neighbourhood Search Methodology for the Capacitated Examination Timetabling Problem. *Journal of Operational Research Society*, advance online publication 13 September 2006; doi: 10.1057/palgrave.jors.2602258.

ABDULLAH, S., BURKE, E.K. and MCCOLLUM, B. (2006c). Using A Randomised Iterative Improvement Algorithm with Composite Neighbourhood Structures for the University Course Timetabling Problem. In K.F. Doerner, M. Gendreau, P. Greistorfer, W.J. Gutjahr, R.F. Hartl and M. Reimann, eds., *Computer Science Interfaces Book Series.*, Springer Operations Research, accepted.

AMINTOOSI, M. and HADDADNIA, J. (2005). Feature Selection in a Fuzzy Student Sectioning Algorithm. In Burke and Trick (2005), 147–160.

Appleby, J.S., Blake, D.V. and Newman, E.A. (1961). Techniques for Producing School Timetables on a Computer and their Application to other Scheduling Problems. *The Computer Journal*, **3**, 237–245.

Arani, T. and Lotfi, V. (1989). A Three Phased Approach to Final Exam Scheduling. *IIE Transactions*, **21**, 86–96.

Asmuni, H., Burke, E.K., Garibaldi, J.M. and McCollum, B. (2005). Fuzzy Multiple Heuristic Orderings for Examination Timetabling. In Burke and Trick (2005), 334–353.

Asmuni, H., Burke, E.K., Garibaldi, J.M., McCollum, B. and Parkes, A.J. (2008). An investigation of fuzzy multiple heuristic orderings in the construction of university examination timetables. *Computers & Operations Research*, (accepted to be published).

Aufm Hofe, H.M. (2001). Solving Rostering Tasks By Generic Methods For Constraint Optimization. *International Journal of Foundations of Computer Science*, **12**, 671–693.

Avella, P. and Vasil'Ev, I. (2005). A Computational Study of a Cutting Plane Algorithm for University Course Timetabling. *Journal of Scheduling*, **8**, 497–514.

Azimi, Z.N. (2005). Hybrid Heuristics for Examination Timetabling Problem. *Applied Mathematics And Computation*, **163**, 705–733.

Bardadym, V.A. (1996). Computer Aided School and University Timetabling : The New Wave. In Burke and Ross (1996), 22–45.

Bilgin, B., Özcan, E. and Korkmaz, E.E. (2006). An Experimental Study on Hyper-heuristics and Exam Scheduling. In Burke and Rudová (2006), 123–140.

Blum, C. and Roli, A. (2003). Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys*, **35**, 268–308.

Boizumault, P., Delon, Y. and Peridy, L. (1996). Constraint Logic Programming for Examination Timetabling. *The Journal of Logic Programming*, **26**, 217–233.

Brailsford, S.C., Potts, C.N. and Smith, B.M. (1999). Constraint Satisfaction Problems: Algorithms and Applications. *European Journal of Operational Research*, **119**, 557–581.

Broder, S. (1964). Final Examination Scheduling. *Communications of the ACM*, **7**, 494–498.

Brĕlaz, D. (1979). New Methods to Color the Vertices of A Graph. *Communications of the ACM*, **22**, 251–256.

BULLNHEIMER, B. (1997). An Examination Scheduling Model to Maximize Students' Study Time. In Burke and Carter (1998), 78–91.

BURKE, E. and LANDA SILVA, J. (2004). The Design of Memetic Algorithms for Scheduling and Timetabling Problems. In W. Krasnogor N.and Hart and J. Smith, eds., *Recent Advances in Memetic Algorithms, Studies in Fuzziness and Soft Computing*, vol. 166, 289–312, Springer.

BURKE, E., MACCARTHY, B., PETROVIC, S. and QU, R. (2000). Structured Cases in Case-Based Reasoning - Re-using and Adapting Cases for Time-tabling Problems. *Journal of Knowledge-Based Systems*, **13**, 159–165.

BURKE, E.K. and BYKOV, Y. (2006). Solving Exam Timetabling Problems with the Flex-Deluge Algorithm. In Burke and Rudová (2006), 370–372.

BURKE, E.K. and CARTER, M.W., eds. (1998). *Practice and Theory of Automated Timetabling II, Second International Conference, PATAT'97, Toronto, Canada, August 20-22, 1997, Selected Papers.*, vol. 1408 of *Lecture Notes in Computer Science*, Springer.

BURKE, E.K. and CAUSMAECKER, P.D., eds. (2003). *Practice and Theory of Automated Timetabling IV, 4th International Conference, PATAT 2002, Gent, Belgium, August 21-23, 2002, Selected Revised Papers.*, vol. 2740 of *Lecture Notes in Computer Science*, Springer.

BURKE, E.K. and ERBEN, W., eds. (2001). *Practice and Theory of Automated Timetabling III, Third International Conference, PATAT 2000, Konstanz, Germany, August 16-18, 2000, Selected Papers.*, vol. 2079 of *Lecture Notes in Computer Science*, Springer.

BURKE, E.K. and KENDALL, G., eds. (2005). *Search Methodologies - Introductory Tutorials in Optimization and Decision Support Techniques*. Springer.

BURKE, E.K. and NEWALL, J.P. (1999). A Multistage Evolutionary Algorithm for the Timetable Problem. *IEEE Transactions on Evolutionary Computation*, **3**, 63–74.

BURKE, E.K. and NEWALL, J.P. (2003). Enhancing Timetable Solutions with Local Search Methods. In Burke and Causmaecker (2003), 195–206.

BURKE, E.K. and NEWALL, J.P. (2004). Solving Examination Timetabling Problems through Adaption of Heuristic Orderings. *Annals of Operations Research*, **129**, 107–134.

BURKE, E.K. and PETROVIC, S. (2002). Recent Research Directions in Automated Timetabling. *European Journal of Operational Research*, **140**, 266–280.

BURKE, E.K. and ROSS, P., eds. (1996). *Practice and Theory of Automated Timetabling, First International Conference, Edinburgh, U.K., August 29 - September 1, 1995, Selected Papers.*, vol. 1153 of *Lecture Notes in Computer Science*, Springer.

Burke, E.K. and Rudová, H., eds. (2006). *Proceedings of The 6th International Conference on the Practice and Theory of Automated Timetabling, 30th August - 1st September 2006, Brno, Czech Republic.*, Faculty of Informatics, Masaryk University, Brno, The Czech Republic, Masaryk University.

Burke, E.K. and Trick, M.A., eds. (2005). *Practice and Theory of Automated Timetabling V, 5th International Conference, PATAT 2004, Pittsburgh, PA, USA, August 18-20, 2004, Revised Selected Papers.*, vol. 3616 of *Lecture Notes in Computer Science*, Springer.

Burke, E.K., Elliman, D.G. and Weare, R.F. (1994a). A Genetic Algorithm based University Timetabling System. In *Proceedings of the 2nd East-West International Conference on Computer Technologies in Education (Crimea, Ukraine, 19th-23rd Sept 1994)*, vol. 1, 35–40.

Burke, E.K., Elliman, D.G. and Weare, R.F. (1994b). A Genetic Algorithm for University Timetabling. In *Proceedings of the AISB Workshop on Evolutionary Computing (University of Leeds, UK, 11th-13th April 1994)*.

Burke, E.K., Elliman, D.G. and Weare, R.F. (1994c). A University Timetabling System Based on Graph Colouring and Constraint Manipulation. *Journal of Research on Computing in Education*, **27**, 1–18.

Burke, E.K., Elliman, D.G., Ford, P.H. and Weare, R.F. (1995a). Specialised Recombinative Operators for the Timetabling Problem. In *Proceedings of the AISB (Artificial Intelligence and Simulation of Behaviour) Workshop on Evolutionary Computing (University of Sheffield, UK, 3rd-7th April 1995)*, Lecture Notes in Computer Science, 75–85, Springer.

Burke, E.K., Elliman, D.G. and Weare, R.F. (1995b). A Hybrid Genetic Algorithm for Highly Constrained Timetabling Problems. In L. Eshelman, ed., *Proceedings of the 6th International Conference on Genetic Algorithms (ICGA'95, Pittsburgh, USA, 15th-19th July 1995)*, 605–610, Morgan Kaufmann, San Francisco, CA, USA.

Burke, E.K., Elliman, D.G., Ford, P.H. and Weare, R.F. (1996a). Examination Timetabling in British Universities - A Survey. In Burke and Ross (1996), 76–90.

Burke, E.K., Newall, J.P. and Weare, R.F. (1996b). A Memetic Algorithm for University Exam Timetabling. In Burke and Ross (1996), 241–250.

Burke, E.K., Jackson, K., Kingston, J.H. and Weare, R.F. (1997). Automated University Timetabling: The State of the Art. *The Computer Journal*, **40**, 565–571.

Burke, E.K., Newall, J.P. and Weare, R.F. (1998a). A Simple Heuristically Guided Search for the Timetable Problem. In E. Alpaydin and C. Fyte, eds., *Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems (EIS'98)*, 574–579, University of La Laguna, Spain, ICSC Academic Press.

Burke, E.K., Newall, J.P. and Weare, R.F. (1998b). Initialisation Strategies and Diversity in Evolutionary Timetabling. *Evolutionary Computation Journal (special issue on Scheduling)*, **6**, 81–103.

Burke, E.K., Bykov, Y. and Petrovic, S. (2001a). A Multicriteria Approach to Examination Timetabling. In Burke and Erben (2001), 118–131.

Burke, E.K., MacCarthy, B., Petrovic, S. and Qu, R. (2001b). Case-Based Reasoning in Course Timetabling: An Attribute Graph Approach. In D.W. Aha and I. Watson, eds., *Proceedings of the 4th International Conference on Case-Based Reasoning*, vol. 2080 of *Lecture Notes in Artificial Intelligence*, 90–104, Springer-Verlag, Berlin, Heidelberg.

Burke, E.K., MacCarthy, B.L., Petrovic, S. and Qu, R. (2002). Knowledge Discovery in a Hyper-heuristic for Course Timetabling Using Case-Based Reasoning. In Burke and Causmaecker (2003), 276–287.

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

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

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

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

Burke, E.K., Kingston, J. and de Werra, D. (2004b). Applications to Timetabling. In J. Yellen and J. Gross, eds., *Handbook of Graph Theory.*, chap. 5.6, 445–474, Chapman Hall,CRC Press.

Burke, E.K., MacCarthy, B.L., Petrovic, S. and Qu, R. (2006a). Multiple-Retrieval Case-Based Reasoning for Course Timetabling Problems. *Journal of the Operational Research Society*, **57**, 148–162.

Burke, E.K., Petrovic, S. and Qu, R. (2006b). Case Based Heuristic Selection for Timetabling Problems. *Journal of Scheduling*, **9**, 115–132.

Burke, E.K., McCollum, B., Meisels, A., Petrovic, S. and Qu, R. (2007). A Graph-Based Hyper Heuristic for Educational Timetabling Problems. *European Journal of Operational Research*, **176**, 177–192.

Caramia, M., DellOlmo, P. and Italiano, G.F. (2001). New Algorithms for Examination Timetabling. In S. Naher and D. Wagner, eds., *Algorithm Engineering 4th Int. Workshop, Proc. WAE 2000 (Saarbrucken, Germany, September)*, vol. 1982 of *Lecture Notes in Computer Science*, 230–241, Springer, Berlin.

Carter, M.W. (1986). A Survey of Practical Applications of Examination Timetabling Algorithms. *Operation Research*, **34**, 193–202.

Carter, M.W. (2001). A Comprehensive Course Timetabling and Student Scheduling System at the University of Waterloo. In Burke and Erben (2001), 64–82.

Carter, M.W. and Johnson, D.G. (1999). The Use of Cliques in Examination Timetabling. Research Series 1999:9, Loughborough University, Business School.

Carter, M.W. and Johnson, D.G. (2001). Extended Clique Initialisation in Examination Timetabling. *Journal of the Operational Research Society*, **52**, 538–544.

Carter, M.W. and Laporte, G. (1996). Recent Development in Practical Examination Timetabling. In Burke and Ross (1996), 3–21.

Carter, M.W. and Laporte, G. (1998). Recent Developments in Practical Course Timetabling. In Burke and Carter (1998), 3–19.

Carter, M.W., G. Laporte, G. and Lee, S.Y. (1996). Examination Timetabling: Algorithmic Strategies and Applications. *Journal of the Operational Research Society*, **47**, 373–383.

Casey, S. and Thompson, J. (2003). GRASPing the Examination Scheduling Problem. In Burke and Causmaecker (2003), 232–244.

Chiarandini, M., Birattari, M., Socha, K. and Rossi-Doria, O. (2006). An Effective Hybrid Algorithm for University Course Timetabling. *Journal of Scheduling*, **9**, 403–432.

Coello, C.A.C. (2006). Evolutionary Multi-Objective Optimization: A Historical View of the Field. *IEEE Computational Intelligence Magazine*, **1**, 28–36.

Cole, A.J. (1964). The Preparation of Examination Time-tables using a Small-store Computer. *The Computer Journal*, **7**, 117–121.

Costa, D. (1994). A Tabu Search Algorithm for Computing an Operational Timetable. *European Journal of Operational Research*, **76**, 98–110.

Costa, D. and Hertz, A. (1997). Ants Can Colour Graphs. *Journal of the Operational Research Society*, **48**, 295–305.

Cox, E. and O'Hagen, M. (1998). *The Fuzzy Systems Handbook : A Practitioner's Guide to Building, Using and Maintaining Fuzzy Systems.*. AP Professional, Cambridge, MA.

Dahal, K.P., Aldridge, C.J. and McDonald, J.R. (1999). Generator Maintenance Scheduling using a Genetic Algorithm with a Fuzzy Evaluation Function. *Fuzzy Sets and System*, **102**, 21–29.

Daskalaki, S., Birbas, T. and Housos, E. (2004). An Integer Programming Formulation for a Case Study in University Timetabling. *European Journal of Operational Research*, **153**, 117–135.

de Werra, D. (1985). An Introduction to Timetabling. *European Journal of Operational Research*, **19**, 151–162.

Desroches, S., Laporte, G. and Rousseau, J.M. (1978). HOREX: A Computer Program for the Construction of Examination Schedules. *INFOR*, **16**, 294–298.

Di Gaspero, L. and Schaerf, A. (2001). Tabu Search Techniques for Examination Timetabling. In Burke and Erben (2001), 104–117.

Di Gaspero, L. and Schaerf, A. (2003). Multi-Neighbourhood Local Search with Application to Course Timetabling. In Burke and Causmaecker (2003), 262–275.

Dimopoulou, M. and Miliotis, P. (2004). An Automated University Course Timetabling System Developed in a Distributed Environment: A Case Study. *European Journal of Operational Research*, **153**, 136–147.

Dorigo, M. and Gambardella, L. (1997). Ant Colonies for the Traveling Salesman Problem. *Biosystems*, **43**, 73–81.

Dorigo, M., Maniezzo, V. and Colorni, A. (1996). Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Trans. on Systems, Man, and Cybernetics–Part B*, **26**, 29–41.

Dowsland, K.A. and Thompson, J.M. (2005). Ant Colony Optimization for the Examination Scheduling Problem. *Journal of the Operational Research Society*, **56**, 426–438.

Dueck, G. (1993). New Optimization Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel. *Journal of Computational Physics*, **104**, 86–92.

Eley, M. (2006). Ant Algorithms for the Exam Timetabling Problem. In Burke and Rudová (2006), 167–180.

Erben, W. (2001). A Grouping Genetic Algorithm for Graph Colouring and Exam Timetabling. In Burke and Erben (2001), 132–158.

Fleszar, K. and Hindi, K.S. (2002). New heuristics for one-dimensional bin-packing. *Computers & Operations Research*, **29**, 821–839.

Foxley, E. and Lockyer, K. (1968). The Construction of Examination Timetables by Computer. *The Computer Journal*, **11**, 264–268.

Garibaldi, J. and Ifeachor, E. (1999). Application of Simulated Annealing Fuzzy Model Tuning to Umbilical Cord Acid-Base Interpretation. *IEEE Transactions on Fuzzy Systems*, **7**, 72–84.

Gendreau, A., Salvail, L. and Soriano, P. (1993). Solving the Maximum Clique Problem Using a Tabu Search Approach. *Annals of Operations Research*, **41**, 385–403.

Glover, F. (1986). Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research*, **13**, 533–549.

Glover, F. and Laguna, M. (1997). *Tabu Search*. Kluwer, Dordrecht.

Gómez-Skarmeta, A.F. and Jiménez, F. (1999). Fuzzy Modeling with Hybrid Systems. *Fuzzy Sets and Systems*, **104**, 199–208.

Hertz, A. (1991). Tabu Search for Large Scale Timetabling Problems. *European Journal of Operational Research*, **54**, 39–47.

Holland, J.H. (1992). *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, MA, USA.

Hwang, C.L. and Yoon, K., eds. (1981). *Multiple Attribute Decision Making - Methods and Applications: A State of the Art Survey*, vol. 186 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, Berlin.

Johnson, D. (1990). Timetabling University Examinations. *Journal of Operational Research Society*, **41**, 39–47.

Joslin, D.E. and Clements, D.P. (1999). Squeaky Wheel Optimization. *Journal of Artificial Intelligence Research*, **10**, 353–373.

Kasabov, N.K. (1996). *Foundation of Neural Networks, Fuzzy Systems, and Knowledge Engineering.*. A Bradford Book, The MIT Pres.

Kendall, G. and Mohd Hussin, N. (2005a). A Tabu Search Hyper-heuristic Approach to the Examination Timetabling Problem at the MARA University of Technology. In Burke and Trick (2005), 270–293.

Kendall, G. and Mohd Hussin, N. (2005b). An Investigation of A Tabu Search Based Hyper-heuristic for Examination Timetabling. In G. Kendall, E.K. Burke, S. Petrovic and M. Gendreau, eds., *Multidisciplinary Scheduling: Theory and Applications*, 309–328, Springer.

Kiaer, L. and Yellen, J. (1992). Weighted Graphs and University Course Timetabling. *Computers and Operations Research*, **19**, 59–67.

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983). Optimization by Simulated Annealing. *Science*, **220**, 671–680.

KOSTUCH, P. (2005). The University Course Timetabling Problem with a Three-Phase Approach. In Burke and Trick (2005), 109–125.

LANDA SILVA, J.D., BURKE, E.K. and PETROVIC, S. (2004). An Introduction to Multiobjective Metaheuristics for Scheduling and Timetabling. In X. Gandibleux, M. Sevaux, K. Sorensen and V. T'kindt, eds., *Metaheuristic for Multiobjective Optimisation*, vol. 535 of *Lecture Notes in Economics and Mathematical Systems*, 91–129, Springer.

LAPORTE, G. and DESROCHES, S. (1984). Examination Timetabling by Computer. *Computers and Operations Research*, **11**, 351–360.

LI, J. and KWAN, R.S.K. (2003). A Fuzzy Genetic Algorithm for Driver Scheduling. *European Journal of Operational Research*, **147**, 334–344.

LIM, M.H., RAHARDJA, S. and GWEE, B.H. (1996). A GA Paradigm for Learning Fuzzy Rules. *Fuzzy Sets and Systems*, **82**, 177–186.

LOO, E.H., GOH, T.N. and ONG, H.L. (1985). A Heuristic Approach to Scheduling University Timetables. *Computers & Education*, **10**, 379–388.

LOTFI, V. and CERVENY, R. (1991). A Final Exam-Scheduling Package. *Journal of the Operational Research Society*, **42**, 205–216.

MAMDANI, E.H. and ASSILIAN, S. (1975). An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. *International Journal of Man-Machine Studies*, **7**, 1–13.

MARÍN, H.T. (1998). *Combinations of GAs ans CSP Strategies for Solving the Examination Timetabling Problem.*. Ph.D. thesis, Instotuto Technólogy y de Estudios Superiores de Monterrey.

MCCOLLUM, B. (2006). University Timetabling: Bridging the Gap between Research and Practice. In Burke and Rudová (2006), 15–35.

MEHTA, N.K. (1981). The Application of a Graph Coloring Method to an Examination Scheduling Problem. *INTERFACES*, **11**, 57–65.

MERLOT, L.T.G., BOLAND, N., HUGHES, B.D. and STUCKEY, P.J. (2003). A Hybrid Algorithm for Examination Timetabling Problem. In Burke and Causmaecker (2003), 207–231.

MOSCATO, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Tech. Rep. Report 826, Caltech Concurrent Computation Program, California Institute of Technology.

MOSCATO, P. and NORMAN, M.G. (1992). A Memetic Approach for the Traveling Salesman Problem Implementation of a Computational Ecology for Combinatorial Optimization on Message-Passing Systems. In M. Valero, E. Onate, M. Jane, J.L. Larriba and B. Suarez, eds., *Parallel Computing and Transputer Applications*, 177–186, IOS Press, Amsterdam.

N. Safaei, M.S.M. and Jabal-Ameli, M. (2008). A hybrid Simulated Annealing for solving an extended model of dynamic cellular manufacturing system. *European Journal of Operational Research*, **185**, 563–592.

Negnevitsky, M. (2002). *Artificial Intelligence: Guide to Intelligent Systems.*. Addison Wesley, 1st edn.

Osman, I.H. and Kelly, J.P., eds. (1996). *Metaheuristics: Theory and Applications*. Kluwer, Dordrecht.

Pappis, C.P. and Siettos, C.I. (2005). *Fuzzy Reasoning*, chap. 15, 437–474. In Burke and Kendall (2005).

Perttunen, J. (1994). On the significance of the initial solution in travelling salesman heuristics. *The Journal of the Operational Research Society*, **45**, 1131–1140.

Petrovic, S. and Burke, E.K. (2004). University Timetabling. In J.Y.T. Leung, ed., *Handbook of Scheduling: Algorithms, Models, and Performance Analysis.*, chap. 45, Chapman & Hall/CRC.

Petrovic, S. and Bykov, Y. (2003). A Multiobjective Optimisation Technique for Exam Timetabling Based on Trajectories. In Burke and Causmaecker (2003), 179–192.

Petrovic, S. and Qu, R. (2002). Case-Based Reasoning as a Heuristic Selector in a Hyper-Heuristic for Course Timetabling Problems. In *Knowledge-Based Intelligent Information Engineering Systems and Allied Technologies, Conference Volume of KES'02,*, vol. 82, 336–340, IOS Press.

Petrovic, S., Patel, V. and Yang, Y. (2005). University Timetabling With Fuzzy Constraints. In Burke and Trick (2005), 313–333.

Qu, R. (2002). *Case-Based Reasoning for Course Timetabling Problems*. Ph.D. thesis, School of Computer Science and Information Technology, The University of Nottingham.

Qu, R., Burke, E.K., McCollum, B., Merlot, L.T.G. and Lee, S.Y. (2006). A Survey of Search Methodologies and Automated Approaches for Examination Timetabling. Tech. Rep. NOTT-CSTR-2006-4, University of Nottingham, School of Computer Science and Information Technology.

R Development Core Team (2005). *R: A Language and Environment for Statistical Computing.*. R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0.

Rahoual, M. and Saad, R. (2006). Solving Timetabling Problems by Hybridizing Genetic Algorithms and Tabu Search. In Burke and Rudová (2006), 467–472.

Rattadilok, P., Gaw, A. and Kwan, R.S.K. (2005). Distributed Choice Function Hyper-heuristics for Timetabling and Scheduling. In Burke and Trick (2005), 51–67.

Ross, P. (2005a). *Hyper-heuristics*, chap. 17, 529–556. In Burke and Kendall (2005).

Ross, P., Hart, E. and Corne, D. (1998). Some Observations about GA-Based Exam Timetabling. In Burke and Carter (1998), 115–129.

Ross, S.M. (2005b). *Introductory Statistics*. Elsevier Academic Press, 2nd edn., ISBN:0-12-597132-X.

Santiago-Mozos, R., Salcedo-Sanz, S., de Prado-Cumplido, M. and Bousoño-Calzón, C. (2005). A Two-phase Heuristic Evolutionary Algorithm for Personalizing Course Timetables: A Case Study in a Spanish University. *Computers and Operations Research*, **32**, 1761–1776.

Sazonov, E.S., Klinkhachorn, P., Gangarao, H.V.S. and Halabe, U.B. (2002). Fuzzy Logic Expert System for Automated Damage Detection from Changes in Strain Energy Mode Shapes. *Non-destructive Testing and Evaluation*, **18**, 1–20.

Schaerf, A. (1999). A Survey of Automated Timetabling. *Artificial Intelligent Review*, **13**, 87–127.

Schaerf, A. and Di Gaspero, L. (2001). Local Search Techniques for Educational Timetabling Problems. In L. Lenart, L. Stirn Zadnik and S. Drobne, eds., *Proceedings of the 6th International Symposium on Operational Research (SOR-01), Preddvor, Slovenia.*, 13–23.

Schaerf, A. and Di Gaspero, L. (2006). Measurability and Reproducibility in Timetabling Research: State-of-the-Art and Discussion. In Burke and Rudová (2006), 53–62.

Schmidt, G. and Strohlein, T. (1980). Timetable Construction - An Annotated Bibliography. *The Computer Journal*, **23**, 307–316.

Setnes, M. and Roubos, H. (2000). GA-Fuzzy Modeling and Classification: Complexity and Performance. *IEEE Transactions on Fuzzy Systems*, **8**, 509–522.

Shimojima, K., Fukuda, T. and Hasegawa, Y. (1995). Self-Tuning Fuzzy Modeling with Adaptive Membership Function, Rules, and Hierarchical Structure Based on Genetic Algorithms. *Fuzzy Sets and Systems*, **71**, 295–309.

Slany, W. (1996). Scheduling as a Fuzzy Multiple Criteria Optimization Problem. *Fuzzy Sets and Systems*, **78**, 197–222.

Slowinski, R. and Hapke, M., eds. (2000). *Scheduling Under Fuzziness*. Physica-Verlag.

Socha, K., Knowles, J. and Sampels, M. (2002). A MAX- MIN Ant System for the University Timetabling Problem. In M. Dorigo, G. Di Caro and M. Sampels, eds., *Proceedings of ANTS 2002 - Third International Workshop on Ant Algorithms*, vol. 2463 of *Lecture Notes in Computer Science*, 1–13, Springer, Berlin, Germany.

Teodorovic, D. and Lucic, P. (1998). A Fuzzy Set Theory Approach to the Aircrew Rostering Problem. *Fuzzy Sets and Systems*, **95**, 261–271.

Thompson, J.M. and Dowsland, K.A. (1996). General Cooling Schedules for a Simulated Annealing Based Timetabling System. In Burke and Ross (1996), 345–363.

Thompson, J.M. and Dowsland, K.A. (1998). A Robust Simulated Annealing Based Examination Timetabling System. *Computers & Operations Research*, **25**, 637–648.

Voβ, S., Martello, S., Osman, I.H. and Roucairol, C., eds. (1999). *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Kluwer, Boston.

Wang, C.H., Hong, T.P. and Tseng, S.S. (1998). Integrating Fuzzy Knowledge by Genetic Algorithms. *IEEE Transactions on Evolutionary Computation*, **2**, 138–149.

Welsh, D.J.A. and Powell, M.B. (1967). An Upper Bound for the Chromatic Number of a Graph and its Application to Timetabling Problems. *The Computer Journal*, **10**, 85–86.

White, G.M. and Xie, B.S. (2001). Examination Timetables and Tabu Search with Longer-Term Memory. In Burke and Erben (2001), 85–103.

White, G.M. and Zhang, J. (1998). Generating Complete University Timetables by Combining Tabu Search with Constraint Logic. In Burke and Carter (1998), 187–200.

White, G.M., Xie, B.S. and Zonjic, S. (2004). Using Tabu Search with Longer-Term Memory and Relaxation to Create Examination Timetables. *European Journal of Operational Research*, **153**, 80–91.

Wood, D.C. (1968). A System for Computing University Examination Timetables. *The Computer Journal*, **11**, 41–47.

Wood, D.C. (1969). A Technique for Colouring a Graph Applicable to Large Scale Timetabling Problems. *The Computer Journal*, **12**, 317–319.

Wren, A. (1996). Scheduling, Timetabling and Rostering - A Special Relationship?. In Burke and Ross (1996), 46–75.

Yang, Y. and Petrovic, S. (2005). A Novel Similarity Measure for Heuristic Selection in Examination Timetabling. In Burke and Trick (2005), 247–269.

Ying, H. (2000). *Fuzzy Control and Modeling - Analytical Foundations and Applications*. IEEE Press Series on Biomedical Engineering, IEEE Press, New York.

Zadeh, L.A. (1965). Fuzzy Sets. *Information and Control*, **8**, 338–353.

ZADEH, L.A. (1975). The Concept of a Linguistic Variable and its Application to Approximate Reasoning - I, II and III. *Information Sciences*, **8;8;9**, 199–249;301–357;43–80.

ZADEH, L.A. (1999). From Computing with Numbers to Computing with Words - From Manipulation of Measurements to Manipulation of Perceptions. *IEEE Transactions on Circuitss and Systems - I: Fundamental Theory and Applications*, **45**, 105–119.

ZIMMERMANN, H.J. (1996). *Fuzzy Set Theory and Its Applications.*. Kluwer Academic Publishers, 3rd edn.