Time-Predefined and Trajectory-Based Search: Single and Multiobjective Approaches to Exam Timetabling

by Yuri Bykov, BSc

Thesis submitted to the University of Nottingham
for the degree of Doctor of Philosophy
November 2003

Contents

List of Figures **5**
List of Tables **7**
Abstract **8**
Acknowledgements **9**
1. Introduction **10**
1.1 Timetabling Problems 10
1.2 University Exam Timetabling and its Automatisation 12
1.3 Exam Timetabling as Graph Colouring Problem 14
1.4 Motivation of Presented Research 18
1.5 Contribution 21
2. Overview of Algorithmic Approaches to Exam Timetabling **22**
2.1 Sequential Methods 22
2.2 Cluster Methods 25
2.3 Metaheuristics 26
2.3.1 Local Search Metaheuristics 27
2.3.2 Simulated Annealing 29
2.3.3 Tabu Search 32
2.3.4 Genetic Algorithms 36
2.3.5 Constraint Logic Programming 42
2.3.6 Ant Colony Optimisation 44
2.4 Recent Innovative Methodologies for Automated Exam Timetabling .. 45
2.5 Hybridisation of Different Methods 47
2.6 Summary 49
3. A Review of Multiobjective Techniques **51**
3.1 Basic Concepts of the Multiobjective Optimisation 51
3.2 A Priori Methods 55
3.3. A Posteriori Methods 58
3.3.1 Multiobjective Versions of Genetic Algorithms 59
3.3.2 Non-Evolutionary Pareto-Based Techniques 66
3.3.3 Hybridisation of Pareto-based Techniques 68
3.4 The Evaluation of Trade-off Surfaces 69
3.5 Multiobjective Exam Timetabling 72
3.6 Summary 73
4. Exam Timetabling Specification and Data **75**
4.1 A Formalisation of Exam Timetabling Problems 75
4.1.1 A Specification of the Basic Problem 75
4.1.2 A Specification of Additional Constraints 79
4.2 A Multiobjective Statement of Exam Timetabling Problem 90
4.3 Benchmark Exam Timetabling Datasets 92
5. A Time Predefined Approach to Examination Timetabling **95**
5.1 The Role of Computational Time in the Process of Solving Timetabling Problems 95
5.2 Time-Predefined Algorithms 97
5.2.1 The Time-Predefined Simulated Annealing 97
5.2.2 The Great Deluge Algorithm 100

5.3 Experiments with Time-Predefined Techniques.....103
5.3.1 An Initialisation Phase.....104
5.3.2 Neighbourhood Structure.....105
5.4 Investigating the Properties of the Algorithms.....105
5.5 Analysis of the Relationship between Time and Cost.....110
5.6 A Comparison of Time-Predefined Simulated Annealing and the Great Deluge Algorithm with the Current State-of-the-Art.....117
5.6.1 A Comparison on Carter's Benchmarks.....117
5.6.2 Experiments with More Advanced Problems.....121
5.7 On the Comparison of the Performance of the Time-Predefined Algorithms with other Approaches.....127
5.7.1 A Comparison with a Threshold Acceptance Method.....128
5.7.2 A Comparison with Hill-Climbing.....130
5.8 Conclusions.....131
6. Multiobjective Methods for Exam Timetabling Problems.....133
6.1 An Aggregation Multiobjective Technique based on Compromise Programming.....133
6.1.1 Criteria and Preference Spaces.....134
6.1.2 An Algorithm for Heuristic Search of the Preference Space.....137
6.1.3 A Real Timetabling Problem: Results and Discussion.....140
6.2 A Case Study of the Application of Pareto-Based Approach to Exam Timetabling Problems.....144
6.2.1 Non-Dominated Sorting Genetic Algorithm for Exam Timetabling.....144
6.2.2 Pareto Archived Evolutionary Strategy for Exam Timetabling.....155
6.3 Conclusions.....161
7. A Trajectory-Based Multiobjective Search.....163
7.1 Driving the Search through a Trajectory.....163
7.2 A Reference Solution Strategy.....164
7.3 Great Deluge with Variable Weights.....166
7.3.1 Description of the Method.....166
7.3.2 Investigation of Properties of Great Deluge with Variable Weights.....172
7.3.3 Experiments with Reference Points.....175
7.3.4 Evaluation of a Manageability of the Reference Point Method.....179
7.4 An Enhanced Trajectory-Based Multiobjective Optimisation Technique.....181
7.4.1 The Description of the Method for the Bi-Objective Case.....182
7.4.2 An Expansion into the Multiobjective Case.....188
7.4.3 Investigation of Dynamics of the Algorithm.....191
7.5 A Fan Search Strategy.....193
7.5.1 A Description of the Strategy.....194
7.5.2 Testing the Fan Search Strategy.....196
7.5.3 Using a Reference Point Selected from PAES Result.....201
7.6 Further Possible Strategies for the Application of the Trajectory-Based Technique.....205
7.6.1 Approximation Strategies.....206
7.6.2 An Interactive Trajectory Assessment.....208

7.7 Conclusions.....212
8. Conclusions.....214
8.1 Summary of the Presented Approaches.....214
8.2 A Comparison of Performance of Different Methods.....215
8.3 Publications.....218
8.4 Applications of the Presented Approaches in Different Areas.....219
8.4.1 An International Timetabling Competition.....219
8.4.2 An Investigation of the Protein Folding Problem.....221
8.5 Future Work.....223
References.....225

List of Figures

Figure 1.1: Exam timetabling as a Graph Colouring problem 15
Figure 3.1: Pareto-optimal solutions comprise the Pareto-front 52
Figure 3.2: Different possible shapes of Pareto-fronts 53
Figure 3.3: True and known Pareto-fronts 53
Figure 3.4: Goldberg's method of ranking the population..... 60
Figure 3.5: Ranking in Multiobjective Genetic Algorithm (MOGA)..... 62
Figure 3.6: *S*-metric for comparison of non-dominated sets..... 71
Figure 4.1: An example of calculation of the sum in formula (4.1)..... 79
Figure 4.2: An example of calculation of the number of adjacent conflicts.... 83
Figure 4.3: An example of calculation of the number of same day conflicts .. 84
Figure 4.4: An example of calculation of the number of adjacent days conflicts 85
Figure 4.5: An example of calculation of the number of overnight conflicts .. 86
Figure 5.1: The extended Great Deluge algorithm 101
Figure 5.2: Cost progress diagrams for time-predefined Simulated Annealing on CAR-F-92 dataset.....107
Figure 5.3: Cost progress diagrams for the Great Deluge algorithm on CAR-F-92 dataset 109
Figure 5.4: Time-cost diagrams for PUR-S-93 problem (120690 enrolments, 43 timeslots, search speed 82000 moves/sec).....111
Figure 5.5: Time-cost diagrams for UTA-S-92 problem (58981 enrolments, 35 timeslots, search speed 87000 moves/sec) 111
Figure 5.6: Time-cost diagrams for CAR-S-91 problem (56877 enrolments, 35 timeslots, search speed 73000 moves/sec) 112
Figure 5.7: Time-cost diagrams for CAR-F-92 problem (55552 enrolments, 32 timeslots, search speed 89000 moves/sec) 112
Figure 5.8: Time-cost diagrams for EAR-F-83 problem (8108 enrolments, 24 timeslots, search speed 99000 moves/sec) 113
Figure 5.9: Time-cost diagrams for YOR-F-83 problem (6029 enrolments, 21 timeslots, search speed 44000 moves/sec) 114
Figure 5.10: Time-cost diagrams for STA-F-83 problem (5751 enrolments, 13 timeslots, search speed 82000 moves/sec) 114
Figure 5.11: Time-cost diagrams for RYE-F-91 problem (45052 enrolments, 23 timeslots, search speed 245000 moves/sec).....115
Figure 5.12: Time-cost diagrams for KFU-S-93 problem (25118 enrolments, 20 timeslots, search speed 265000 moves/sec).....115
Figure 5.13: Time-cost diagrams for TRE-S-92 problem (14901 enrolments, 23 timeslots, search speed 119000 moves/sec).....116
Figure 5.14: Time-cost diagrams for HEC-S-92 problem (10632 enrolments, 18 timeslots, search speed 53000 moves/sec).....116
Figure 5.15: Time-cost diagrams for LSE-F-91 problem (10919 enrolments, 18 timeslots, search speed 245000 moves/sec) 116
Figure 5.16: Time-cost diagrams for UTE-S-92 problem (11796 enrolments, 10 timeslots, search speed 203000 moves/sec).....117

Figure 5.17: Time-cost diagrams for KFU-S-93 problem (search speed 260000 moves/sec) 122
Figure 5.18: Time-cost diagrams for Nott-94 problem (search speed 174000 moves/sec) 123
Figure 5.19: Time-cost diagrams for CAR-F-92 problem (search speed 87000 moves/sec) 123
Figure 5.20: The example of uncertainty in comparison of the time-cost diagram with the single solution. 128
Figure 5.21: Comparison of Threshold Acceptance and Great Deluge algorithms for CAR-F-92 problem 129
Figure 5.22: Comparison of Hill-Climbing and Great Deluge algorithms for YOR-F-83 problem 131
Figure 6.1: Mapping from the criteria space into the preference space.....137
Figure 6.2: A final population produced by NSGA.....151
Figure 6.3: A comparison of trade-off surfaces produced by NSGA with and without elitism. 154
Figure 6.4: The comparison of trade-off surfaces produced by NSGA and PAES 157
Figure 6.5: Trade-off surface produced by PAES for LSE-F-91 problem.....158
Figure 6.6: Trade-off surface produced by PAES for RYE-S-93 problem.....159
Figure 6.7: Trade-off surface produced by PAES for YOR-F-83 problem 159
Figure 7.1: Search along the defined trajectory.....166
Figure 7.2: Borderline in the weighted sum Great Deluge algorithm 167
Figure 7.3: The increase of w_1 169
Figure 7.4: The increase of w_2 169
Figure 7.5: The selection of increased weight.....169
Figure 7.6: The multiobjective Great Deluge algorithm with variable weights 171
Figure 7.7: The progress diagram for the Nott-94 problem 173
Figure 7.8: Time-cost diagram of Great Deluge algorithm with variable weights.....174
Figure 7.9: Relocation of the borderline 182
Figure 7.10: Rotation of a borderline.....185
Figure 7.11: Shift of the borderline 187
Figure 7.12: The enhanced multiobjective Great Deluge algorithm. 190
Figure 7.13: The algorithm follows a trajectory with two branches.....193
Figure 7.14: The Fan Search strategy 195
Figure 7.15: Trade-off surfaces produced by the Fan Search 198
Figure 7.16: Further iterations of the Fan Search.....199
Figure 7.17: Trade-off surfaces produced by the Fan Search 200
Figure 7.18: Trade-off surfaces produced by PAES and Fan Search. 201
Figure 7.19: The approximation of a reference point 206
Figure 7.20: Approximation of a secondary trade-off surface 208
Figure 7.21: An interactive trajectory assessment (phase 1).....209
Figure 7.22: An interactive trajectory assessment (Phase 2) 210
Figure 7.23: An interactive trajectory assessment (complete scheme).....211
Figure 8.1: Comparison graph of the performance of discussed algorithms ..216

List of Tables

Table 1.1: Conflict matrix, proposed by Cole 16
Table 4.1: An example of conflict matrix *C*..... 77
Table 4.2: Allocation of exams to timeslots..... 78
Table 4.3: The matrix of proximity coefficients *prox*(t_i,t_j) 78
Table 4.4: Number of students in timeslots..... 81
Table 4.5: The matrix of coefficients *adjs* (t_i,t_j) 83
Table 4.6: The matrix of coefficients *sday* (t_i,t_j)..... 84
Table 4.7: The matrix of coefficients *adjd* (t_i,t_j)..... 85
Table 4.8: The matrix of coefficients *ovnt* (t_i,t_j) 86
Table 4.9: An example of preassignment matrix *A* 88
Table 4.10: An example of before/after matrix *G* 89
Table 4.11: An example of immediately before/after matrix *H* 90
Table 4.12: A description of objectives 91
Table 4.13: The parameters of Carter's collection of examination datasets 93
Table 4.14: The details of constraints for Nott-94 problem..... 94
Table 5.1: Published and produced results for proximity cost 119
Table 5.2: Additional characteristics of problems.....122
Table 5.3: Published and produced results for weighted sum of adjacent and overnight conflicts.....125
Table 5.4: The best results, obtained for Nott-94 problem by Great Deluge search.....126
Table 6.1: The maximum number of conflicts when a student has exams in adjacent periods 136
Table 6.2: Solutions when all criteria are of the same importance.....141
Table 6.3: Solutions when X_2 is of higher importance than the other criteria.142
Table 6.4: Solutions for different distance measures, *W*=(1,1,1,1,1,1,1,1,1)..143
Table 6.5: Parameters for Non-Dominated Sorting Genetic Algorithm 150
Table 6.6: Comparison of results produced by NSGA and A Priori techniques 152
Table 6.7: Characteristics of results of PAES algorithm 160
Table 6.8: Results of A Priori techniques 161
Table 7.1: Reference and produced solutions for the bi-criteria case.....176
Table 7.2: Reference and produced solutions for nine-criteria case.....178
Table 7.3: Results of a questionnaire.....180
Table 7.4: Parameters and processing times of the first series of experiments 197
Table 7.5: Parameters and processing time of the second series of experiments 199
Table 7.6: The comparison of *S*-metrics of primary and secondary sets 204

Abstract

The research work presented in this thesis aims to provide effective methods for solving university exam timetabling problems. The goal is to automatically produce high quality timetables which are easy and practical to use. Several ideas are introduced, which could increase the overall performance of timetabling algorithms. The primary idea is to employ parameters which are clearly understandable for the user and which therefore make the timetabling procedure more transparent. The second idea is to consider the expected processing time as one such parameter, which allows the user to increase the quality of final solutions by the efficient management of computational resources. In addition to this, special attention is paid to the development of multiobjective approaches, which can help the user to manage a high variety of constraints (which are of different nature) and simultaneously, generate a solution, which mostly suits his/her preferences. The author also introduces the idea of a trajectory-based multiobjective approach which enables the search process to move along defined trajectories. A number of different strategies for the application of trajectory-based search are proposed, which can enable the easy expression of user preferences.

In this thesis five new exam timetabling algorithms are presented (2 single-objective and 3 multiobjective ones) which (in different ways) incorporate the ideas that are outlined above. A comprehensive series of experiments demonstrate the effectiveness of the proposed techniques. In most cases the presented approaches significantly outperform other techniques on established benchmark exam timetabling problems.

Acknowledgements

To my two supervisors: Dr. Sanja Petrovic and Prof. Edmund Burke. I appreciate their sound advice throughout the course of my PhD study.

To James Newall for his advice and encouragement during the first two years of my PhD.

To the other members of the ASAP group, past and present, for the atmosphere of mutual support and friendly co-operation in the group.

Chapter 1.

Introduction

1.1 Timetabling Problems

Scheduling can be thought of as a decision making process which involves the allocation of limited resources to tasks over time. The resources may be machines, people or other objects, while tasks can denote the separate operations in a process in which the resources are employed [Pin95]. One of the definitions of scheduling is given by Wren [Wre96], who stated that "Scheduling is the arrangement of objects into a pattern in time or space in such a way that some goals are achieved, or nearly achieved". He also described and discussed different types of problems: Rostering, Sequencing and Timetabling. He says that:

- *Rostering* is the placing of resources into slots in a pattern.
- *Sequencing* is simply an order in which activities are carried out.
- *Timetabling* shows WHEN particular events are to take place.

There is no strict borderline between these types as some scheduling problems conform to more than one of the above definitions. The vast published literature on scheduling shows that the same or similar approaches can be successfully used in a wide variety of different scheduling problems. Thus, although the research work presented in this thesis is focused on timetabling (and more precisely on university examination timetabling), the author believes that a similar methodology can be used in solving other related problems.

In addition to the formal definition, the notion of timetabling is intuitively clear from our everyday life experience. Various timetables repeatedly regulate people's actions in transport (bus, railway, air flight timetables), in work (personnel, conference timetables), at study (school, university timetables), in healthcare (nurse timetables), in entertainment (sport timetables, festival timetables) as well as in many other situations. In other words, timetables aim to help people to be "in the right place at the right time".

Higher education is one of the fields where the employment of good timetables has became increasingly important in recent years. In the UK the introduction of a modular course structure where each student can chose a set of subjects increased the complexity of timetabling problems. Usually educational institutions involve special members of staff (timetabling officers) or whole departments responsible for the construction of timetables of different types. Educational timetabling is often divided into three categories [Sch99a] and [Whi00]. The basic problems can be defined as follows:

- *School Timetabling*: the weekly scheduling of high school classes, avoiding teachers meeting two classes at the same time and vice versa.
- *Course Timetabling*: the weekly scheduling of lectures and laboratories of a number of university programmes, while minimizing the overlap of courses having common students;
- *Examination Timetabling*: the scheduling of the exams of a set of university courses so that no students must sit two or more exams simultaneously while spreading the exams out for the students as much as possible.

It should be noticed that these types of timetables have differences. For example, the school and course timetables are usually constructed on a weekly basis, while the period of an examination session is variable for different institutions.

1.2 University Exam Timetabling and its Automatisation

At its most basic, the exam timetabling is concerned with distributing a collection of university exams among a limited number of timeslots (periods). This is, of course, subject to a set of regulations and limitations (often termed constraints), which vary widely from institution to institution. There are certain *constraints* which must be satisfied under any circumstances such as the requirement that no student can sit two exams simultaneously (*clash-free requirement*), or that exam rooms have a certain physical capacity which must not be exceeded. Such constraints are known as *hard*. Solutions, which satisfy all the hard constraints, are often called *feasible solutions*.

In addition to the hard constraints there are usually various constraints that are considered to be desirable but not essential. These are often called *soft*. Of course, there is significant difference across institutions as to which constraints they consider important and which they do not. The situation in British universities is discussed in more detail in [Burk96] which analyses the responses of over 50 British universities to a questionnaire on exam timetabling. Examples of commonly occurring soft constraints can reflect the situation where students prefer to spread exams evenly throughout the examination session or at least have some time interval between exams. On the

other hand, the institution often wants to schedule large exams earlier (in order to leave more time for marking). Specific preferences may also be expressed by particular members of staff concerning, for example, invigilation duties.

The increased number of regulations for exam timetables together with a demand for their flexibility (due to the modular structure) make the traditional methods of construction ineffective. Early operational research/computing scientists noted that the use of computer-based methods can significantly improve the effectiveness of the timetabling process. A large number of timetabling algorithms have been proposed in the intervening decades. However, the real figures about computer use in exam timetabling (circa 1996) are revealed in [Burk96]. It was reported that 42% of respondents included in questionnaire still did not use a computer in timetabling at all, 37% used computer as an assistance tool (for data preparation, storage and representation including printing reports) and only 21% used the computer for actual timetabling.

The author believes that the popularity of timetabling software could be increased by developing systems which more adequately meet the real world university timetabling infrastructure. Here Carter's observation that "timetabling is 10% graph theory, and 90% politics" [Car01] should be taken as guidance. Carter was referring to lecture timetabling but the comment still has some relevance for exam timetabling. Although a powerful kernel algorithm plays a significant role in timetabling software, it should consider a spectrum of aspects, including interfaces, data formats, compatibility with

other software, maintenance and upgrading, providing a user support including user's manuals online help, users training, etc.

These tasks require the joint efforts of specialists from different fields including operations research, artificial intelligence, software development, programming languages, databases, user interfaces, and university administration. Indeed all of them have their own impact on the timetabling problem. Although the author recognises that all aspects of exam timetabling are important, the presented research is mostly aimed at providing a good kernel algorithm for exam timetabling software. Therefore, in the remaining part of this thesis the term "Exam Timetabling Problem" is considered as a combinatorial optimisation[1] problem whose solution is expressed in a numerical form. In this work, different approaches to timetabling are compared from a mathematical modelling point of view only, while the representation of results or any other human factor is not considered.

1.3 Exam Timetabling as Graph Colouring Problem

The construction of a feasible timetable, which satisfies only a clash-free requirement conforms to the solving of the well-known Graph Colouring Problem. It was described in 1941 by Brooks [Bro41]. This problem considers a connected graph comprised N vertices where every vertex is linked by an edge (edges) to some other vertex (or vertices). The vertices should be

[1] The combinatorial optimisation field comprises problems of splitting and/or ordering finite sets of elements with the aim of achieving a certain goal. Such problems have been known from antiquity (e.g. Latin squares, etc.). In modern history several classes of such problems have been introduced starting from Euler's (Leonard Euler (1707-1783) "problem of Kennigsberg bridges" which later was expanded into the Travelling Salesman Problem.

coloured into a number of colours so, that any two vertices connected by an edge should have different colours. The following question arises: is it possible to colour this graph into *p* colours and if yes, how do we do it? The other task is to find a minimum possible number of such colours. The modelling of the timetabling problem as a Graph Colouring Problem is described by Carter in the following way [Car86]:

- Each course is represented by a vertex;
- An edge connects two vertices if the corresponding courses have at least one student in common and, hence, cannot be scheduled in the same time period.

Carter completed this analogy "by associating the *p* available exam periods with *p* colours". The process of solving of such a problem is illustrated in Figure 1.1.

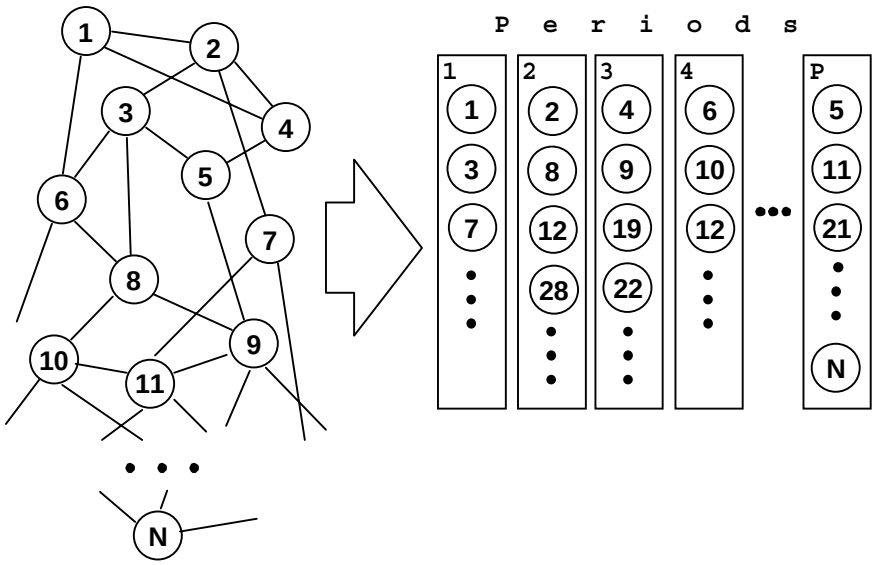

Figure 1.1: Exam timetabling as a Graph Colouring problem

Another representation of the exam timetabling problem was proposed by Cole [Col64] as a symmetrical bit matrix NxN (where N is the number of exams), which contains 1 in cell i,j, (where $i,j \in \{1,...,N\}$), if there is a conflict between exams i and j; and 0 - otherwise. Both representations (graph and matrix) can be easily transformed into each other, which can be helpful while developing algorithms for timetabling problems.

Table 1.1: Conflict matrix, proposed by Cole

j \ i	1	2	3	4	5	6	. . .	N
1	-	1	0	1	0	1		0
2	1	-	1	1	0	0		0
3	0	1	-	0	1	1		1
4	1	1	0	-	1	0		1
5	0	0	1	1	-	0		0
6	1	0	1	0	0	-		1
. . .							-	
N	0	0	1	1	0	1		-

The minimum number of colours into which a graph can be completely coloured defines its *chromatic number* χ introduced by Welsh and Powell in [WP67]. It is proved that χ should be greater or equal to the largest clique in the graph (the clique is a subgraph whose vertices are connected with each other). However, in general there is no method for exactly calculating this number as well as for actual colouring. For a more detailed overview of graph colouring in timetabling problems see [BKW03].

The difficulty of the Graph Colouring Problem is illustrated by its membership of the class of so-called *NP-complete* problems, which was

defined by Karp [Kar72]. The solution (*global optimum*) for such a problem can be obtained by a finite number of steps, but the number of these steps grows as an exponential function of the size of the problem. For example, the increment of the problem size by 1 leads to a twofold (or more) increase in the computing time. Thus, algorithms which completely enumerate can be applied only to very small size problems (for the larger-size problems the computing time becomes far too large).

More than 300 classes of NP-complete problems are known which can be polynomially transformed into each other [Gor89]. However, there is (still) no exact algorithm for solving any NP-complete problem where the processing time is a polynomial function of the problem's size (polynomial algorithm). Moreover, there is no proof of its existence or non-existence. Usually these problems are solved by some inexact algorithms (often termed as *heuristics*), which aim to produce near-optimal solutions in a reasonable processing time. The heuristics suitable for graph colouring are discussed in more detail in Section 2.1 and are overviewed in [BKW03].

An alternative formalisation of exam timetabling was recently proposed by Yanes and Ramirez in [YR03] as a "Robust Colouring Problem". Here the graph should be coloured into a fixed number of colours c (which is greater than or equal to chromatic number χ), but the goal is to minimise the equal colouring of nodes connected by so-called "complementary edges". In such a way the second-order constraints are taken into account. The authors have

proved that this problem is self-contained (not an external generalisation of the Graph Colouring Problem) and in its turn it is NP-hard.

1.4 Motivation of Presented Research

Satisfying the clash-free requirement is usually considered as a necessary but not sufficient demand for real-world timetables. The quality of the timetable is defined by the satisfaction/violation of other constraints (usually soft ones). These constraints depend on the particular institution. The ones, most widely used in British Universities were collected by Burke et al. [Burk96] and described in the following way:

- There should not be more students scheduled to a room than there are seats.
- Exams with questions in common should be scheduled in the same period.
- Some exams should be scheduled only within a particular set of periods.
- Only exams of the same length should be scheduled in the same room.
- Exams with a large number of participants should be scheduled early in the timetable.
- Some exams should only take place in particular rooms.
- Large exams have a higher importance in scheduling than the smaller ones.
- Exam A should be scheduled before exam B.
- No student should be scheduled to exams in two consecutive periods.
- No student should be scheduled to more than one exam in any particular day.
- Each student’s exams should be evenly spread through the timetable.
- No student should be scheduled to exams in two consecutive days.

- Exams should be scheduled to rooms near to the relevant departments

Of course, in any real world situation it would be extremely rare if it were possible to satisfy all the soft constraints. Therefore, a useful measure of the quality of a timetable can be taken to be the number of violations of these constraints. Minimising these violations is often one of the over-riding objectives for the development of timetabling software systems. Traditionally the violated soft constraints are aggregated (usually as a weighted sum) into an *objective function* (cost function or "fitness" or "penalty"), which serves as an index of the solution quality. Thus, the goal of the examination timetabling process can be taken to be that of producing the feasible timetable of the highest possible quality (minimum value of the particular cost function under consideration).

Managing a high variety of different constraints is quite a difficult task. Every additional constraint can increase the total complexity of the problem and can make the solution more resource-consuming. Therefore, in the real-world, there is often a high level of user-intervention and relaxation of constraints. Often, it becomes clear during the timetabling process that the particular problem instance in hand is over-constrained. This observation could motivate an argument that interactive (semi-automatic) systems are preferable to batch (full-automatic) ones. In this connection, Schaerf wrote in his survey on automated timetabling: "many authors believe that the timetabling problem cannot be completely automated" [Sch99a]. He gave two reasons for it: firstly, it is difficult to express all requirements for "good" timetables and secondly,

sometimes a system cannot find the best direction of search without human interaction.

The rejection of fully-automatic methods in favour of semi-automatic might be a useful way to improve the quality of timetables. However, the author believes that the potential of automated methods is not fully explored yet. The overall motivation for the research work presented in this thesis is to discover new methods for timetabling algorithms. It is aimed at simplifying the solution procedure, to make it more transparent for the user and at the same time to make it more effective.

The conventional weighted sum approach, which is often used as an objective function in timetabling has some weaknesses. Similar values of the objective function could well have been obtained in different ways. For example, one evaluation may be the result of obtaining a “good” satisfaction of one constraint at the expense of violations for another. A very similar result might be obtained by summing two mediocre results with respect to both constraints. It might well be that one of these evaluations is much more desirable than the other, in the context of the particular problem in hand, even though these two cost functions have similar results.

This observation motivates the research into *multicriteria* approaches to examination timetabling problems. In these approaches the violations of different constraints are measured by different objective functions. A family of existing methods are aimed at producing results, where the values of the objective functions mostly correspond to the preferences expressed by the user

(*decision maker*). In this thesis this family is expanded by developing new methods for the exam timetabling problem.

1.5 Contribution

This thesis is organised in the following way. The next chapter gives an overview of the algorithmic approaches that have been applied to exam timetabling problems over the last few decades. In Chapter 3 the state-of the art in the multiobjective combinatorial optimisation field is described. The formalisation of the exam timetabling problem is stated in Chapter 4 together with a description of the benchmark problems used in conducted experiments. Chapter 5 is devoted to the investigation of the possibility of increasing the effectiveness of a local search approach to exam timetabling by developing the proper management of the computing time. Two time-predefined algorithms are presented here: the time-predefined Simulated Annealing and the Great Deluge search. The following chapters describe a number of multiobjective approaches to exam timetabling. Chapter 6 presents a new multiobjective method based on the idea of Compromise Programming. In this chapter the effectiveness of selected existing multiobjective algorithms applied to exam timetabling problems is also studied. Chapter 7 introduces a new trajectory-based multiobjective approach. It includes two variants of the algorithm and several strategies for their application to exam timetabling problems. Chapter 8 summarises this study and discusses the applications of the proposed methods in other research areas. It also outlines some future research directions.

Chapter 2.

Overview of Algorithmic Approaches to Exam Timetabling

The research presented in this thesis continues a sequence of studies devoted to the development and investigation of different optimisation algorithms for exam timetabling problems, which have been carried out over the last twenty years or so with varying levels of success. This chapter gives an overview of the different approaches presented in the scientific literature. It discusses the major tendencies of these techniques and provides an indication of further research directions in this area. A special attention is paid to published experiments with real-world exam timetabling problems. The interested reader can see a more detailed description of the various approaches that have appeared over the years in the following timetabling survey/review papers: [Burk96], [Car86], [CL96], [Burk97], [Sch99a], [BKW03].

2.1 Sequential Methods

Some early approaches to solving the exam timetabling problem which in a certain sense, underpin the research presented in this thesis were developed in the 1960's. These approaches tended to concentrate on the fundamental hard constraint which says that, "exams in the same period should not have common students". The generation of such clash-free timetables is analogous to solving the classical Graph Colouring Problem where the vertices correspond to examinations, edges between two vertices indicate the presence of common students who should attend both exams and every colour conforms to a

particular time slot. Correspondingly, the methods for solving such timetabling problems are based upon methods for solving Graph Colouring Poblems. Appleby, Blake and Newman [ABN60] investigated such approaches and ever since the family of so-called graph colouring sequencing heuristics has been widely applicable for timetabling. Indeed, in recent times such approaches have been hybridised with modern meta-heuristics to produce high quality solutions (see Section 2.5). In essence, these basic graph colouring based methods fill a blank timetable with exams, taken in a certain order. The particular methods of ordering vary according to the different heuristics employed. More details are presented in survey papers [Car86], [CL96] and [BKW03]. Of course, the ordering motivation is to place the most "difficult" exams first depending upon the measure of "difficulty". As a measure of this "difficulty", Cole [Col64] have used the degree of each vertex (the number of conflicting exams). Different modifications of this "largest degree first" algorithm were done by Peck and Williams [PW66], Welsh and Powell [WP67], Wood [Woo68]. Broder [Bro64] also proposed to produce several solutions and choose the best one among them. Moreover, this was the first attempt to take into account soft constraints (i.e. those which are desirable but not essential to satisfy).

The inverse ordering ("smallest degree last") was proposed by Matula, Marble and Isaacson [MMI72]. The algorithm recursively removed exams with the smallest degree and placed them into the list. Each remove followed by the recalculation of degree of vertices. After completing the list, the vertices were assigned to colours correspondingly to their position in the list (in reverse order so that the last placed vertex was taken first).

Another example is the sum of degrees of adjacent vertices. Williams [Wil74] put forward an idea that the most difficult vertex is not that, which has a most conflicts by itself, but which has the most conflicting neighbours. This technique is known as “largest modified degree first”.

One of the most successful idea in this respect involves the calculation of the exam's degree as the number of available slots and its recalculation after each placement. Such a heuristic (“saturation degree first”) was presented by Brelaz [Bre79]. This algorithm (originally called DSATUR) outperformed all previous techniques. In [CLL96] Carter et al. compared different sequential methods with each other and with the random ordering and showed that the “saturation degree first” algorithm in most cases produced the best results.

Although in recent years the main attention (in exam timetabling) has been paid to metaheuristic approaches, graph colouring heuristics are still under investigation. For example, they play an important role in generating initial solutions for different search methods [BNW98]. Moreover, they were extended to take into account some soft constraints.

One of the useful ways of improving the performance of graph colouring heuristics is to find the largest (maximum) clique in a graph and colour it first (e.g. [CLL96], [MT96]). The reason behind this is that all vertices in a clique must have different colours and the colouring of the largest clique can cause the most difficulties. In [CJ01] Carter and Johnson noted that real-world exam timetabling problems often contain several maximum cliques and suggested starting the algorithm from the colouring of all maximum

cliques rather than a single one. The authors expanded their method to colour fist all “near maximum” cliques and/or “quasi-cliques”, i.e. the subgraphs with minor variations from the maximum clique.

Further modifications of sequencing heuristics use *backtracking*, which involves rescheduling of certain exams in conflict situations and can consider soft constraints. A comprehensive investigation of their effectiveness with a collection of real-world examination timetabling problems is presented in [CLC94], [CLL96].

As an alternative to the backtracking approach Burke and Newall [BN03] suggested an adaptive mechanism which can improve or salvage the initial ordering of exams while running a sequential algorithm during a number of iterations. This approach also considers soft constraints and performs at the same level as the backtracking method. In [BN02] the adaptive method was applied for the initialisation of local search metaheuristics. The authors investigated the performance of local search started from initial solutions of different quality. The experiments on real-world exam timetabling problems showed the direct impact of the quality of initial solutions on the quality of final results.

2.2 Cluster Methods

Cluster methods split the set of events into groups which are conflict-free and then assign the groups to time periods to fulfil other constraints imposed on the timetabling problem [FS83], [BLW92]. A multi-phase exam-scheduling package described in [AL89] and [LC91] consists of three phases. In the first

phase, clusters of exams are formed with the aim of minimising the number of students with simultaneous exams. In the second phase, these clusters are assigned to exam days while minimising the number of students with two or more exams per day. Finally the exam days and clusters are arranged to minimise the number of students with consecutive exams.

2.3 Metaheuristics

As briefly alluded to above, in addition to a straightforward application to simplified graph colouring analogous timetabling problems, graph colouring heuristics have been incorporated into more recent *metaheuristics* techniques. This tendency was motivated by the increase of the problem's complexity (by considering a variety of constraints imposed on timetabling problems). Moreover, the variety of timetabling models became so high, that it seems impossible to find suitable heuristics for each particular case.

In contrast to the briefly discussed heuristics (which are specialised on particular problems), Osman and Laporte [OL96] defined metaheuristics as powerful techniques which can produce good results for wide range of different problems. Several metaheuristics have proved to be a valuable tool in solving exam timetabling problems, such as: Hill Climbing (e.g. [ABN60]), Simulated Annealing (e.g. [Joh90], [TD96a], [Bul98]), Tabu Search (e.g. [Her91], [WX01]), Genetic Algorithms (e.g. [CFM93], [Erg96]), Constraint Logic Programming (e.g. [BDP94], [RO99]) and Ant Colony Optimisation (e.g. [CH97], [DPT02]). These approaches will be described in more detail.

2.3.1 Local Search Metaheuristics

Local search can be defined to be a method which represents the gradual improvement of a *current solution* (or solutions) starting from initial one(s) until some stopping condition is satisfied. Every solution is characterised by an *objective function* (it is also called *cost function* or *penalty*). It is a quantitative measure of the solution quality and its minimisation (or maximisation) is often the main goal of metaheuristic algorithms. Usually the value of the objective function of the new solution highly affects the decision of whether to accept or to discard it. This decision procedure (acceptance condition) mainly determines the type of search algorithm.

During a local search every new solution is selected among the set of candidate solutions (so-called *neighbourhood*), produced from the current one by certain modifications. The structure of the neighbourhood depends on the types of permitted alterations (*moves*) and highly affects the performance of local search algorithms. The various types of moves were studied by Costa in [Cos94] who considered the preferable sets of moves (which led to the best results) being dependent on the properties of each particular problem. Schaerf defined an atomic move as a simple replacing of one exam into a new period or swapping of the pair of exams [Sch99b]. He also introduced double moves, which comprise two atomic moves in such a way that if the first move creates any infeasibility then the second one “repairs” it.

Exam timetabling neighbourhoods can be expanded by so-called *Kempe chains* (chains of adjacent vertices coloured into two given colours). As all

vertices adjacent to such a chain have colours different to the chain ones, then the swapping of the chain colours does not affect the feasibility of a solution. In [TD96b] Thomson and Dowsland investigated the effectiveness of Kempe chains on disconnected search spaces (search spaces divided into several regions without feasible path between them). The use of Kempe chains helped to overcome the disconnectivity and led to the exploration of wider regions where the better solutions could be discovered. Later this idea was expanded into the conception of *S-chains* – the ordered lists of S coloured vertices [TD98]. The recolouring of such a chain required the special procedure suggested by the authors.

Three types of neighbourhood (each for a different purpose) were suggested for the use in exam timetabling by Di Gaspero in [DG02]:

- *recolour* – the reallocation of an exam into a new timeslot. This move could improve the cost function while maintaining the feasibility of a solution.
- *shake* – the regrouping of timeslots. This could provide a starting point for a new search phase.
- *kick* – the sequence of the recolour moves. This could yield an improvement where the single recolour move is ineffective. This move could also relocate the solution into the yet unexplored regions of the search space.

It was suggested to use these neighbourhoods sequentially, which could lead to the higher quality of final results.

Several local search metaheuristics are based upon a simple approach called *Hill-Climbing*. This approach was proposed for timetabling by Appleby, Blake and Newman [ABN60]. This algorithm iteratively inspects the neighbourhood and replaces the current solution by a candidate with better fitness. Hill-Climbing rapidly achieves the nearest local optimum (this algorithm is also called "greedy"), however, most real-world problems have a colossal number of local optima and the obtained solution is usually far from the best one. It is a very fast algorithm but due to relatively poor performance it is no longer used, on its own, as a serious approach for solving real world timetabling problems except as a comparative measure [RC95]. However, the approach, along with some level of hybridisation, still has a role to play in modern research as is discussed in Section 2.5.

2.3.2 Simulated Annealing

An idea which is more fruitful than straightforward Hill-Climbing is to allow the occasional acceptance of solutions, which are worse than the current one. Of course, this prolongs the search time but can lead to better final results. This basic mechanism is implemented in *Simulated Annealing*, which is one of the most well studied metaheuristics. This stochastic variant of local search was presented by Kirkpatrick, Gellat and Vecci [KGV83] as a computational model of the physical process of the increasing of energetic stability of molecular structure by consecutive heating and cooling of a material. Here, the candidate solutions with worse objective function values than the current one are accepted with probability calculated by formula $P=e^{-d/T}$, which was derived from the Boltzmann's distribution (Ludwig Boltzmann 1844-1906), which is

well-known in thermodynamics. In this formula d is the difference of the values of the cost function between the current and the candidate solutions and T is a parameter called the "temperature" (by analogy with the thermodynamics formula), which usually gradually reduces during the search. The reduction scheme that is employed is known as the "cooling schedule". It can be defined by a simple progression formula (geometric schedule) or by other formulas or instructions, e.g. the temperature can be decreased after a certain number of moves or successful moves, etc.

Over the last few years Simulated Annealing has been investigated for examination timetabling with some level of success. In 1990 Johnson [Joh90] applied Simulated Annealing for the generation of real world exam timetabling and revealed a definite advance over the manual approaches. In 1996 Thompson and Dowsland considered, instead of the simple geometric cooling, an adaptive cooling technique, where the temperature is automatically reduced or increased depending upon the success of the move [TD96a]. Here the authors presented the wide investigation of the performance of the Simulated Annealing algorithm on exam timetabling problems. The overall results appeared to be varied with particular datasets, neighbourhoods, cost functions and cooling schedules. However, in respect of producing good results in a reasonable amount of time the adaptive cooling technique turned out to be more preferable than the simple geometric cooling approach.

In [TD96b] the same authors investigated the so-called "reachability problem", which might arise when the search is conducted only among feasible

solutions. Here the search space could be disconnected and some solutions, which might be better than the current ones, could be unreachable. The authors proposed three methods to overcome this disadvantage:

- Launching the algorithm several times starting from a random seed;
- Increasing the size of the neighbourhood using Kempe chains;
- Temporarily allowing unfeasible solutions.

The experiments were done with real university examination datasets as well as with a number of modifications: artificially tightened, artificially disconnected (Kempe chain reachable and unreachable) and reconstructed in order to contain the global optima. The presented results confirmed the fruitfulness of the idea of the Kempe chain annealing, which almost in all cases outperformed other variants. The investigation of Kempe chains together with S-chains on real examination datasets from different universities was continued in [TD98]. The authors discovered that this approach gave the preference of moving large exams, whose proper allocation significantly increased the quality of solutions.

Simulated Annealing applied to the problem, which is divided into several subproblems according to different types of constraints was investigated by Bullnheimer in [Bul98]. He showed that the variation of the problems statement could provide a reasonable compromise between administrative and student needs. This approach was regarded as being capable of providing good solutions for real-world exam timetabling problems.

2.3.3 Tabu Search

Another metaheuristic, which can be considered to be based on Hill-Climbing, is known as *Tabu Search*. The basic idea was proposed by Glover [Glo86]. The overall defining feature of this approach is the keeping of a list of previous moves or solutions (a "tabu list") in order to avoid cycling. The Tabu Search pioneers characterised this approach as an attempt to include "intelligent" features into local search [LG96]. In [Dow98] Dowsland indicated that Tabu Search is not restricted by the confines of any real-world analogy and therefore it is considered as being highly suitable (compared to Simulated Annealing) for different modifications. Dowsland suggested the following extensions of the basic Tabu Search method:

- *variation of the cost function* during the search;
- *variation of the length* of the tabu list;
- *candidate list strategies*: restricting the neighbourhood moves to those that display certain promising features;
- *strategic oscillation*: forcing the search to oscillate through different areas of the solution space;
- *ejection chains*: combining the sequences of moves into chains, so that the change is measured for the chain as a whole and not for the individual components.

The main classification of Tabu Search strategies was carried out by Glover and Laguna in [GL97]. They classified them on the basis whether they use intensification and/or diversification. Intensification assumes a more

intensive investigation of the area, which previously yielded good results. In contrast, diversification directs the search procedure towards unexplored areas.

The Tabu Search algorithm has been successfully applied to university timetabling by Hertz [Her91]. He did not consider differences between course and exam timetables and suggested to apply his method to both of them. Here the solution procedure was divided into two phases. Firstly, objects (exams/courses), which require particular timeslots, were allocated to timeslots while satisfying a “preassignment” constraint. Secondly, the students were grouped in order to minimise the number of conflicts. Tabu Search was used in both phases while regarding different neighbourhoods: all moves were allowed in the first phase, while in the second one the moves were limited to swaps between two objects where at least one of them had to be involved in a conflict. In both phases the algorithm kept a tabu list of moves (instead of the list of solutions) with the condition that it permitted the most promising tabu moves (this allowed the algorithm to be less computationally expensive). Hertz concluded that this algorithm produced the satisfactory results on real-world examination and course datasets, and therefore was likely to be better than the existing techniques. Later Hertz and Robert [RH96] proposed to divide large problems into several subproblems and to apply Tabu Search to each component separately (for small size subproblems it could be the Branch-and-Bound algorithm). Besides this, the authors suggested to determine the draft timetable initially and then gradually improve it with respect to different objectives. They considered their approach to be satisfactory in practice.

In [BN96] Boufflet and Negre compared the performance of Tabu Search with two other techniques: the Branch-and-Bound algorithm and the semi-automatic Computer Aided Design method. They demonstrated that for artificial datasets the results obtained by Tabu Search and Branch-and-Bound were almost the same if the optimal solution existed. However, for real-world exam timetabling problems, where a lot of different criteria should be taken into account the Tabu Search method was preferable. It could produce high quality solutions where the Branch-and-Bound algorithm failed.

The benefits of the use of a variable length tabu list in examination timetabling were investigated by Di Gaspero and Schaerf in [DGS01] where the authors presented a comparison of their approach with other techniques proposed by Carter, Laporte and Lee in [CLL96] and Burke and Newall in [BN99]. Even though most of results proved to be in the same range, on several datasets the authors achieved better results than their competitors. Di Gaspero continued this work in [DG02] where he reinforced the diversification strategy by the variation of the neighbourhood. The presented algorithm (named "Recolour, Shake and Kick") cyclically changed the neighbourhood after convergence on the previous one. They produced results which confirmed the relative advantages of this algorithm over the plain Tabu Search.

In [WX01] White and Xie demonstrated a frequency-based long-term memory mechanism, where together with the tabu list of accepted moves ("recency-based" short-term memory approach) they incorporated a table of the move's quantity for each exam ("frequency-based" long term memory

mechanism). This method restricted the movement of over-active exams and vice-versa forced the movement of exams with low activity. Usually the less active exams had a higher number of conflicts, and correspondingly, their replacement significantly affected the solution. The authors considered that this technique accelerated the downhill movements and diversified the search space. For the automatic determination of appropriate algorithmic parameters, the quantitative analysis method based on the distribution of exam degrees was proposed. “Tabu relaxation” (emptying the tabu list after a number of idle moves) was also suggested as a way to move the searched region into one where better solutions could perhaps be found. This technique was successfully applied to real large-scale examination datasets and results were presented which showed an effectiveness of the suggested strategies.

Two strategies of lexicographic optimisation (the ranking of objectives due to their priorities) within Tabu Search were investigated by Paquete and Stutzle in [PS02]. One strategy (“lex-seq”) presupposed the sequential optimisation (the next objective was considered when all previous ones were satisfied completely). In the other strategy (“lex-tie”) the next objective was considered in the case of the tie regarding the previous objectives. The authors compared both strategies on benchmark datasets and found that their performance was dependent on the size of the problem. The first strategy coped better with larger-sized problems and the second strategy coped better with smaller ones. It is probably the case that the overall performance of this technique can be improved by the proper combining of these two approaches.

2.3.4 Genetic Algorithms

Probably the most attention in examination timetabling over the last decade has been paid to exploring *evolutionary* solving methods. Their idea was borrowed from biology, namely from the computational model of the evolution of species described in Darwinian (Charles Darwin 1809-1882) natural selection theory. Several examples of the computer simulation of natural genetic processes were known as the predecessors of Genetic Algorithms starting from Fraser [Fra57]. In 1966 Bremermann, Roghson and Salaff [BRS66] presented a comprehensive study of "evaluation algorithms" (later they were called "genetic") and suggested to use them as a general numerical method for optimisation. The term *Genetic Algorithm* was introduced by Holland in [Hol75] where he proved several theorems, which laid the foundation of Genetic Algorithm theory.

Following the biological association, the characteristics of Genetic Algorithms are usually described in biological terminology. In these terms, Genetic Algorithms maintain the *population* (set) of sub-optimal *individuals* (solutions). At each *generation* (iteration), a number of *children* (new solutions) are produced. During the *selection* step the extended population is evaluated and individuals with worst *fitness* (objective function) are removed from the population. The individuals in Genetic Algorithms are presented by *chromosomes* (component vectors), which contain *genes* (variables). The characteristics of individuals can be regarded in a *phenotypic* sense (while considering their fitness) or in a *genotypic* sense (paying attention to the disposition of genes in a chromosome). In order to improve the quality of the

population from generation to generation, the algorithm should allow desirable features to be passed from *parents* to children and should discourage undesirable ones. For this purpose, two following *reproductive* strategies are generally useful:

- *Crossover*. A group of methods which generate the child chromosomes by the recombination of genes in (at least two) parental chromosomes.
- *Mutation*. Various techniques which produce a "new" solution by changing genes in the "old" chromosome.

In addition to genetic operators, several other strategies have an impact on the performance of Genetic Algorithms:

- *The evaluation strategy* comprises the different activities connected with fitness: from its calculation to investigation of its search space (landscape).
- *The selection strategy* aims to keep the population size invariable and prevent it from uncontrolled growth. Here the different techniques of choosing redundant solutions are useful.
- *The representation strategy* defines the way of modelling each particular problem in the "gene-chromosome" scheme. In the direct representation (basic variant), each chromosome represents the actual timetable. However, a number of indirect representation schemes were proposed.
- *The initialisation strategy* provides the *seed* (initial population) for the subsequent generations. The Genetic Algorithms often show the best performance when starting from a seed with good fitness and simultaneously high genotypic diversity between individuals. Usually, random constructed solutions have a good diversity but a poor fitness.

Otherwise, graph colouring heuristics provide a better fitness seed but can suffer from a low diversity.

Besides this, each application of Genetic Algorithms is very sensitive to problem-dependent parameters, such as: the population size, the numbers of mutated individuals, the numbers of produced children and the stopping condition.

Traditionally, Genetic Algorithms due to their exterior differences have tended to be separated from other search metaheuristics. However, some authors have advocated the internal similarity between Genetic Algorithms and other metaheuristics. Reeves in [Ree94] gave the formalisation of "Genetic Algorithm neighbourhood" and using a mathematical evidence concluded that a Genetic Algorithm could be viewed as a form of the neighbourhood search. Later, Jones and Forrest [JF95] investigated the operational landscapes of Genetic Algorithms and heuristic search and found a certain amount of common ground in these approaches.

The first report about an application of a Genetic Algorithm to an exam timetabling problem was presented in 1993 by Corne, Fang and Melish [CFM93]. In their study they introduced a number of features, which provided an improvement (percentages are given in brackets) compared with the classic Genetic Algorithm:

- *square pressure of fitness* – reinforcing the difference between good and bad individuals (at least 25%);
- *elitism* – keeping the best individuals into later generations (250%);

- *fixed point uniform crossover* – producing a child, which inherits the fixed number of genes from each parent (400%);

The same authors introduced the so-called "delta-evaluation" for fast calculation of fitness [CRF94]. When applying a genetic operator they calculated only the change ("delta") in fitness. This method reduced the computational expense and the authors suggested its use for both examination and lecture timetabling (they called this case the "General Examination/Lecture Timetabling Problem"). Later Corne and Ross introduced "peckish" initialisation (i.e. partially greedy algorithms being used in the timetabling process) [CR96]. The peckish initialisation was demonstrated as a significant aid for solving exam timetabling problems by Genetic Algorithms.

In 1994, Burke, Elliman and Weare presented an exam timetabling Genetic Algorithm with special crossover and mutation operators [BEW94]. Later they extended the list of such operators including ones based on the graph colouring heuristics [BEW95]. These operators supported different actions such as: reducing the length of the timetable, reducing the second order conflicts, maintaining the proximity of exams. The properly chosen set of operators could direct the search procedure into the particular region of the search space (where a good timetable might be most possible).

In 1996, [Erg96] Ergul improved the mutation operator with a certain mechanism for ranking the chromosomes by their fitness and defined the probability of the mutation as a linear or quadratic function of the chromosome rank ("linear mutation" and "quadratic mutation"). He considered these

operators as less sensitive to problem-dependent input parameters. In addition to this, he suggested penalising conflicts in order to better satisfy the preassignment constraints. For this purpose, weights were assigned to exams, and the conflict penalty was calculated which took the weights of conflicting exams into account. Ergul also proposed the temporal suspension of highly conflicting exams. This means that these exams should be placed into particular slots and held there for a number of generations.

A recent genetic application to exam timetabling was presented by Sheibani [She02] who proposed a method for maximisation of the interval between exams by partitioning them into a number of sets and estimating the so-called “closeness relation” between exams in different sets. The numerical measures employed in this relation played a role of weights in the fitness function. The presented technique was applied to real-world timetabling and the produced results were of satisfactory quality.

Over the years, the performance of Genetic Algorithms on exam timetabling has been investigated thoroughly by several authors. In 1994 Terashima-Marin [Ter94] presented experiments with the pure Graph Colouring Problem. He showed that the sequential heuristics (saturation or even largest degree) easily outperform the Genetic Algorithms. The performance of Genetic Algorithms in examination timetabling (without special recombinative operators) was compared with Hill-Climbing and Simulated Annealing by Ross and Corne [RC95]. Both of these techniques produced better results than the Genetic Algorithm. The weakness of Genetic

Algorithms applied to Graph Colouring Problems using direct representation was confirmed by Ross, Hart and Corne in [RHC98]. The authors investigated the performance of different algorithms on specially designed random graphs as well as real-world benchmark timetabling problems. They discovered the dependence of the performance of algorithms on the edge density of the graphs (ratio of the actual number of edges in the graph to their maximum possible number). The authors showed that most algorithms, which can easily solve both high and low constrained problems, fail when the edge density of the graph reaches some middle value (the so-called "fallible region"). To avoid such difficulties a "cataclysmic adaptive mutation" (which prefers to mutate the exams that cause a higher penalty value) was suggested.

In order to overcome the disadvantage of the "fallible region" an advanced representation of the problem's structure was presented by Erben [Erb01] who suggested employing a Grouping Genetic Algorithm for examination timetabling. In such a representation each gene corresponds to the set of edges. The presented results showed that this approach could reduce the negative effect of a "fallible region".

Although the use of Genetic Algorithms for the pure Graph Colouring Problem is questionable they still can be considered as valuable tools for exam timetabling problems where the most attention is paid to the satisfaction of soft constraints. Probably the most successful role of genetic approaches is in forming the basis for hybrid methods, which have been shown to generate excellent results (e.g. [BN99]).

2.3.5 Constraint Logic Programming

The investigation of Constraint Logic Programming (CLP) approaches to exam timetabling has attracted the attention of the timetabling community for many years. This approach is practical because a variety of universal software tools exist, which were specially developed for solving constraint satisfaction problems. Such systems are commercially available from their vendors in the form of special programming languages or as run-time libraries. Using these tools, the programmer should express his/her problem statement using some declarative language (based on classical logic notations) and then launch the solving subroutine (*solver*), which actually produces results. Thus, one can consider the term "Constraint Logic Programming" to comprise both the algorithms that are laid inside the solvers, and the methods for data preparation and for the solving procedure control.

Most Constraint Logic Programming solvers assign values to variables using exhaustive enumeration methods with backtracking and domain reduction. Therefore, the search space is represented by a tree where variables (exams) are modelled as nodes and the number of branches of every node is equal to the number of values (timeslots) in the node's domain. The programmer's goal is to provide the proper strategy for traversing the tree (so-called *labelling*), i.e. to set up correctly an order in which the values will be assigned to the variables. Existing solvers are very sensitive to the labelling procedure. White [Whi00] discusses an example where a very small change in the ordering of variables increases the processing time from two seconds to 48 hours.

The first application, which employed Constraint Logic Programming for solving exam timetabling problems was developed in 1994 by Boizumault Delon and Peridy [BDP94]. They investigated the performance of two versions of the CHIP language (v.3 and v.4) with different labelling strategies. The best performance was achieved by the latest version of the solver while employing more advanced labelling strategies.

In 1999 Reis and Olivera presented an exam timetabling system based on the ECL^{i}PS^{e} language [RO99]. They enhanced the labelling procedure with the "labelling by variables types" level (in addition to variables and their domain values). This innovation provided an improvement in the search performance because the most constrained types of variables could be labelled first. The authors tested their technique on randomly generated datasets as well as real-world university examination problems. They assumed their tests to be successful because they obtained complete timetables without violations of hard constraints.

An example where an author considered the exam timetabling problem as a constraint satisfaction problem but did not use commercial software was provided by David in [Dav98]. The reason was provided by the practical requirements of the system (e.g. limited processing time). He implemented an enumeration using two phases (while regarding different sets of constraints): "preassignment" and "final assignment" phases. Due to the time limitation, he employed an incomplete algorithm, i.e. in certain situations backtracking was not used. However, if the second phase failed to produce a solution, then

several specially designed repair procedures were applied in order to reduce the number of constraint violations. In the case of further fails, some constraints were relaxed or extra timeslots were added. This system was successfully exploited for producing real-world exam timetables.

2.3.6 Ant Colony Optimisation

This approach is based on the principle of "positive feedback" and can be illustrated by the behaviour of real ants. When choosing a path in unknown surroundings every ant relies on pheromone trails that are left by other ants and in its own turn it adds to the pheromone trail. The more ants that have passed the same path, the higher the probability that this path will be chosen by future ants. This positive feedback works in such a way as to encourage the ants to take the shortest path. The idea of employing this principle in a search metaheuristic belongs to Dorgio et al. [DMC91]. They devised an artificial Ant Colony, where ants produced solutions. The quality of each solution affected the probability of further solutions being constructed which followed its pattern.

The application of the Ant Colony metaheuristic to the Graph Colouring Problem was presented in 1997 by Costa and Hertz [CH97]. The authors regarded graph colouring as an illustration and suggested the use of their algorithm (called ANTCOL) for a wide range of problems (including exam timetabling in respect of hard constraint satisfaction). Feasible solutions were constructed by ants, which used sequential heuristics "saturation degree first" and "recursive largest first" (the second heuristic is a modification of

"largest degree first" where the degrees of uncoloured vertices are recalculated dynamically after each step). At each iteration, the population of solutions provided statistical data about the frequency of colouring each pair of vertices into the same colour. This affected the construction of solutions in the next iteration and so on. In their study, the authors tested the proposed algorithm on random graphs. The performance of the second heuristic was found to be better than the first one and the overall behaviour of the algorithm was considered as successful. However, this conclusion was criticised in [VZ00] by Vesel and Zerovnik. They showed that the ANTCOL's results were beaten by simple multistart launch of the same sequential heuristics.

The recent investigation of the advantages of Ant Colony optimisation in examination timetabling was carried out by Dowsland, Pugh and Thomson in [DPT02]. They paid special attention to the specific distinctions between exam timetabling problems and random graphs. For this purpose, the various modifications of the Ant Colony algorithm were tested. Several aspects were investigated, such as: the influence of the different measures of solution quality on the strength of the pheromone trail, the advantage of the use of candidate lists (of exams to be assigned first) and diversification strategies, etc. The overall conclusion was that this research could provide a basis for extending the Ant Colony metaheuristic to incorporate soft constraints.

2.4 Recent Innovative Methodologies for Automated Exam Timetabling

Together with Constraint Logic Programming and Ant Colony optimisation a number of other innovative approaches have been suggested for exam

timetabling in recent years. However, these applications require more investigations and it is too early to make conclusions about their suitability for exam timetabling.

An example of the application of a multi-agent approach to exam timetabling was provided by Lin in [Lin02]. The whole problem was decomposed into several subproblems and distributed between independent subroutines (*agents*). Each agent used a constraint logic approach (ECL^{i}PS^{e} package) for the partial optimisation of its own subproblem. The partially optimised subproblems were sent to a special central agent (*broker*), which optimised the remaining parts. The final solution was aggregated from the parts, solved by the agents and the broker. The authors analysed results obtained on real and randomly generated timetabling problems of different density. It was shown that on "sparse" problems the multi-agent (decentralised) algorithm could produce better results than the centralised one.

There is a school of thought within the timetabling research community which aims to increase the level of generality of timetabling methodologies. Existing metaheuristic approaches to timetabling tend to be problem specific. The idea is to develop a system, which chooses an appropriate heuristic/metaheuristic for solving a given timetabling problem instance. In 2002 Burke et al. presented a Case-Based Reasoning approach to the selection of heuristics for solving the exam timetabling problem [Burk02]. This algorithm maintains a case-base of previously solved timetabling problems including datasets, objectives, applied techniques and obtained results. The

new problem is compared (using some similarity measure) with the problems in the case-base and the most similar case is retrieved. Thus, this system suggests which approach could be employed on the given problem. In the presented study, three different search techniques were considered: Hill-Climbing, Simulated Annealing and Tabu Search.

Evolutionary algorithms for selecting the right heuristic are another modern approach to examination timetabling [TRV99]. Algorithms which select heuristics/algorithms have been termed *hyper-heuristic*. Indeed, this is a major research direction that Burke and Petrovic discuss in [BP02]. More details about hyper-heuristics can be seen in [Burk03a].

2.5 Hybridisation of Different Methods

One of the most useful ways of improving the performance of combinatorial optimisation algorithms is by hybridising several techniques. The *Memetic Algorithm* (which can be considered to be a hybrid of a Genetic Algorithm and a local search operator) was discussed in 1992 by Moscato and Norman [MN92]. This algorithm employed the concept of a *meme* as a unit of information. This is held by an individual and can be modified by the holder before being passed to other individuals. Thus, the authors pointed out that the memetic approach (in contrast to the genetic approach) emulates *cultural* evolution rather than *biological* evolution.

A Memetic Algorithm for examination timetabling problem was presented and discussed by Burke, Newall and Weare in [BNW96] who also proposed an advanced initialisation strategy, which involved the inclusion of a

random aspect into graph colouring heuristics. This particular Memetic Algorithm hybridised a “mutation only” Genetic Algorithm with Hill-Climbing. The effect of heuristic seeding was presented in a detailed investigation in [BNW98] which explored the use of different diversity measures.

In 1999, Burke and Newall [BN99] presented an approach to the exam timetabling problem, which decomposed the larger problem into a series of smaller subproblems. The subproblems were ordered by “how difficult” each exam in the subproblem was (to schedule). This difficulty measure was provided by graph colouring heuristics. In addition a “look ahead” technique was used where the current subproblem was not fixed in place until the next one had been dealt with. The motivation here was to try and avoid situations where scheduling decisions that were taken in an earlier subproblem would lead to infeasibilities in later subproblems. A Memetic Algorithm was employed on the subproblems. This approach produced the best published results on certain benchmark problems at the time.

Caramia, Dell’Ormo and Italiano consequently applied sequencing heuristics and Hill-Climbing [CDI01]. The sequencing heuristic took into consideration the priorities of exams, which were equal to largest degree at the beginning and were dynamically reassigned during the search. When no improvement was detected, the number of timeslots was automatically increased. In order to improve the performance of the algorithm the search was periodically restarted. The authors investigated different restarting schemes

together with different ways of reassigning the exams priorities. The algorithm produced good results on real-world exam timetabling problems.

Merlot et al. [Mer02] presented a hybrid algorithm, which contained three phases. In the first phase they constructed a feasible timetable using Constraint Logic Programming with the OPL package. In the second phase the initial solution was improved by Simulated Annealing using the Kempe chains neighbourhood. The final improvement was made in the third phase by the modified Hill-Climbing method. This modification iteratively inspected all exams and all available timeslots in order to find the most fruitful moves. This procedure guaranteed the best possible solution in the case when Simulated Annealing left better solutions among the neighbourhood. For the case of the remaining unscheduled exams the authors proposed a greedy heuristic, which aimed to minimise clashes when they are unavoidable. The algorithm was applied to real exam timetabling at the University of Melbourne and produced solutions which were much better than the timetabling software which was in use at that time. Besides this, it showed a promising performance on publicly available benchmark datasets.

2.6 Summary

In this chapter a number of algorithmic approaches, which were applied over the last decades to exam timetabling problems have been discussed. Obviously, the complexity of these techniques grew together with the increasing power of computing hardware. If in the 1960's the relatively easy sequential graph colouring heuristics were mostly practical, then in recent years the main

attention has been paid to more powerful metaheuristics methods, such as Hill-Climbing, Simulated Annealing, Tabu Search and Genetic Algorithms. Their performance on exam timetabling problems is relatively well studied by different authors. Moreover, a high number of extensions and modifications of the basic algorithms have been proposed with the aim of producing higher quality results. Another promising tendency is the hybridisation of different approaches, which can significantly improve the performance of given techniques. The author believes that these two conceptions (modification and hybridisation) still have a high potential for further improvement.

Additionally, several innovative approaches recently applied to exam timetabling have been outlined, such as: Constraint Logic Programming, Ant Colony Optimisation, Case-Base Reasoning, Multi-Agent Optimisation and Hyper-Heuristics. While such approaches play an important role in the examination timetabling literature, they have little direct impact on the research work that is presented in this thesis.

Chapter 3.

A Review of Multiobjective Techniques

3.1 Basic Concepts of the Multiobjective Optimisation

In real-world problems, the quality of a given solution can rarely be estimated by only one criterion. Usually, it requires several criteria, which have different natures and importance and are often in conflict with each other. In other words, these problems involve the optimisation of a number of criteria (objective functions) simultaneously and are called *multiobjective optimisation problems*.

The general *K*-objective optimisation problem can be defined (as stated by Coello Coello [Coe99]) in the following way:

- Find the vector: $\bar{x}^* = \left[x_1^*, x_2^*, \ldots, x_K^*\right]^T$ which optimises the vector objective function: $\bar{f}(\bar{x}) = \left[\, f_1(\bar{x}), f_2(\bar{x}), \ldots, f_K(\bar{x})\right]^T$,
- subject to ***m*** inequality constraints: $g_i(\bar{x}) \geq 0 \qquad i \in \{1,2,\ldots,m\}$,
- ***p*** equality constraints: $h_i(\bar{x}) = 0 \qquad i \in \{1,2,\ldots,p\}$,
- where $\bar{x} = \left[x_1, x_2, \ldots, x_K\right]^T$ is the vector of decision variables.

The main difficulty with a multiobjective approach (which distinguishes it from the single-objective one) lies in the comparison of solutions. By definition one solution outperforms another one if the values of all objective functions of the first solution are better than the second. It is also said that the second solution is *dominated* by the first one. If no solution can dominate the given solution then it can be considered to be optimal. But due to

the conflicting nature of criteria it is usually the case that there is no unique optimal solution. It is maybe possible to improve separately at least one (but not all) objective function of a given solution, which usually causes the declining of its remaining objective functions (or at least one of them). Thus, several different solutions could be thought of as "optimal", because no one dominates the other. The concept of non-dominance was firstly formulated by the French economist Vilfredo Pareto (1848-1923) and therefore such solutions are called "Pareto-optimal". All Pareto-optimal solutions compose a certain boundary between the space which contains dominated solutions and the space where no solutions exist. This boundary is called the *trade-off surface* or *Pareto-front*. It can be depicted as a surface in the K-dimensional space, where K is the number of criteria. For bi-criteria space (criteria x_1 and x_2) the Pareto-front is presented as a curve. An example is shown in Figure 3.1. In this figure the Pareto-optimal solutions are presented by grey points while the dominated ones are represented by white points.

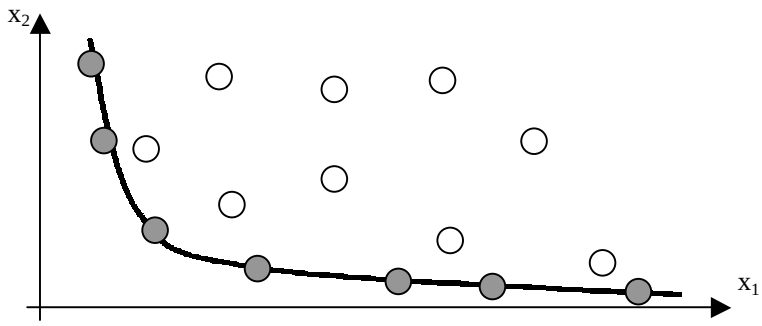

Figure 3.1: Pareto-optimal solutions comprise the Pareto-front

The shape of this surface is highly dependent on the nature of the individual problem [ZDT00]. For example, it can be shifted or inclined towards some criteria, it can intersect the axes, or even be non-convex (see Figure 3.2).

Therefore, it is generally impossible (or too difficult) to express the Pateto-front curve analytically.

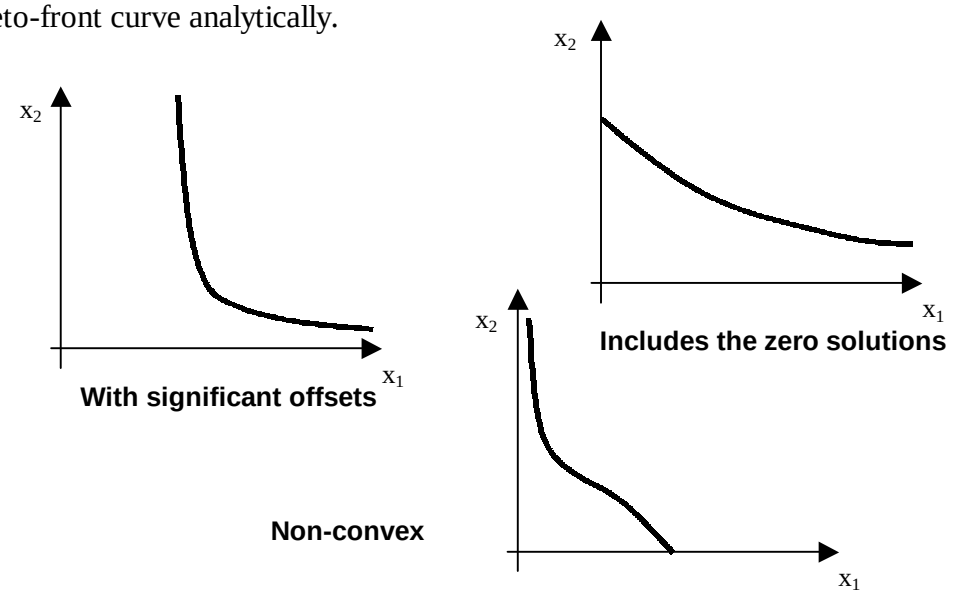

Figure 3.2: Different possible shapes of Pareto-fronts

Moreover, for the majority of problems the absolute values of Pareto-optimal solutions which define the "true" Pareto-front (line PF_{true} in Figure 3.3) are unknown. Here an algorithm can achieve some "known" Pareto-front (line PF_{known} in Figure 3.3), which comprises the already discovered solutions.

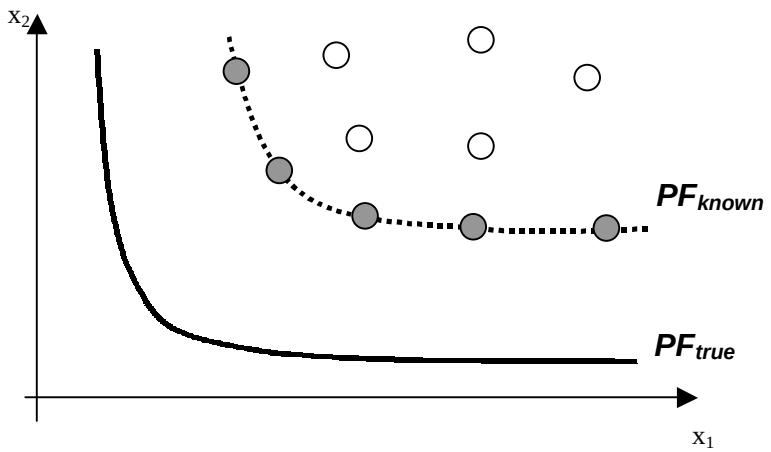

Figure 3.3: True and known Pareto-fronts

This constitutes the major difference between the two known types of multicriteria problems: multiobjective optimisation and multi-attribute decision making. Vincke [Vin92] defines them in the following way:

- *Multi-attribute decision making* studies the strategies of choosing a desirable solution among a number of known ones.
- *Multiobjective optimisation* involves designing/searching for optimal solutions.

Multiobjective reasoning often assumes a higher cooperation with the user (decision maker) than single-criteria methods. Some of these methods called *A Priori* try to achieve a single, hopefully the desirable, solution automatically. However, in many situations the decision maker should provide some additional information about his/her preferences before starting the search (which is not always easy to specify). Such methods are also known as "Decide-then-Search" [VVL00]. It is possible to produce a number of Pareto-optimal solutions, from which the decision maker should manually (or using some multi-attribute decision making technique) choose the desirable one. This type of methods is called *A Posteriori* or "Search-then-Decide" [VVL00].

Two main approaches are distinguished in both groups of methods for multiobjective optimisation: A Priori and A Posteriori. The first is the aggregation approach in which objective functions are aggregated using weighting coefficients for each of them. Therefore a multiobjective optimisation problem is transformed into a scalar one. Another approach

involves specially designed extensions of widely-used metaheuristics, such as: Genetic Algorithms, Simulated Annealing, Tabu Search and Hill-Climbing.

3.2 A Priori Methods

A Priori methods require the definition of some parameters, which (in different ways) reflect the decision maker's preferences. However the decision maker is not always able to express his/her preferences. The most useful among A Priori methods are aggregation techniques. Here the criteria vector is scalarised into an aggregation function and any of the single-objective methods can be applied.

The most often used aggregation approach is the weighted sum method. The aggregation function is calculated as the sum of objectives multiplied by their weights, which should be assigned by the decision maker. It is the first, simplest and the most popular method, which was employed in 1951 when Kuhn and Tucker applied it within non-linear programming [KT51]. This technique directs the search process approximately towards the Pareto-front. However, often the produced solutions are far from the desirable ones because the proper values for the weights are unknown. The weights do not reflect the importance or scale of the corresponding objective nor do they have any other physical meaning.

However, in spite of its weaknesses, the weighted sum approach is widely used by the multiobjective optimisation community because it is easy to apply. One of the ways of using this algorithm is to run it several times with different weights in order to produce the set of solutions, which roughly

represent the trade-off surface. An example of such a *multistart* approach within a Genetic Algorithm is presented in the work of Syswerda and Palmucci [SP91] who applied it to a resource scheduling problem.

One of the famous extensions of the weighted sum approach is Goal Programming. The origin of this technique is connected with the earlier work of Charnes et al. who expressed this idea in [CCF55] and gave it the present name in [CC61]. This algorithm aims to minimise the deviations between the current solution and some target solution, For each objective, the decision maker defines its goals. This method can achieve the goal solution if it is defined inside the feasible region. However, such a solution is generally not optimal and often it makes sense to improve it. Otherwise, if the goal occurs in an infeasible region, then the method could became inefficient [Coe99]. The ideal case seems to be when the goal is placed exactly on the Pareto-front. But this is almost impossible without previous knowledge of the Pareto-front shape.

Over the years a number of modifications of Goal Programming have been proposed. Lee and Olson described in [LO99] the following examples:

- *Least Absolute Value Regression*: The information, obtained in the previous runs of the algorithm is used for defining the deviations in the next runs.
- *MINMAX Goal Programming*: Minimizing the deviation which has the maximum value.
- *Preemptive Goal Programming*: The decision maker defines a number of goals of different priorities. After attaining the goal of one priority the

algorithm tries to reach the next priority goal. In such a way the algorithm stops at the goal of the maximum attainable priority.

- *Nonlinear Goal Programming*: The family of methods, which use different non-linear approximations of the objective functions.

One of the contemporary modifications of Goal Programming is the technique designed at the International Institute for Applied Systems Analysis [Wie99]. Here the decision maker should define the goals at two levels: the aspiration level, which is desirable to attain and the reservation level, which is probably attainable. For these two levels the authors introduced the so-called "achievement function". This function is represented by fuzzy sets and interprets the decision maker's preferences in order to define the most desirable goal.

The list of the aggregation methods can be continued by the Lexicographic approach. As it is explained in [TJR98] this technique considers the objectives of different priorities. The algorithm firstly attempts to find the solution with respect to the most prioritised objectives. Such an idea can be considered to be an alternative to the specification of weights, because the identification of the priorities can be easier for the decision maker than weights. Moreover, such an approach can be vital for timetabling problems where all constraints are usually divided by priorities into hard and soft.

The comprehensive aggregation method, known as Compromise Programming was introduced by Zeleny in [Zel73] and then refined in [Zel82]. Instead of a goal he suggested the use of the so-called *ideal* (or "utopian")

point, whose coordinates represent the optimal value of each objective. The algorithm mapped the criteria space into the preference space where the quality of the solution can be evaluated by measuring its distance to the ideal point. This distance can be expressed as *Lp* metrics:

$$L_p = \left[\sum_{i=1}^{K} (d_i)^p \right]^{1/p}, \tag{3.1}$$

where d_i is the distance between the solution and the ideal point in the preference space and the parameter p affects the compensation among criteria i.e. the offsetting of a bad value of one objective by better values of other objectives. Three values of parameter p are of particular interest: p=1, p=2 and p=∞. The value p=1 leads to the simple sum of distances or so-called "Manhattan block". It enables the absolute compensation. It can be proved that p=∞ ("Chebyshev norm") is equivalent to $L_p = \max_{i=1..K}(d_i)$. It implies no compensation among objective values of the solution. The value p=2 ("Euclidian norm") gives the solution which is geometrically closest to the ideal point.

3.3. A Posteriori Methods

A Posteriori methods do not depend on the decision maker's preferences. They are aimed at finding a set of non-dominated solutions among which the decision maker can chose the most preferable one (manually or using any multi-attribute decision making method). This group of methods is mostly presented by different modifications of Genetic Algorithms. However, a number of non-evolutionary A Posteriori approaches have also been proposed.

3.3.1 Multiobjective Versions of Genetic Algorithms

The Genetic Algorithm was the first metaheuristic which was adapted for multiobjective optimisation [VVL00]. This adaptation is significant because the population-based nature of Genetic Algorithms interacts well with the multi-solution requirement of the A Posteriori approach. In this section different variants of multiobjective Genetic Algorithms are discussed.

The first multiobjective extension of the Genetic Algorithm was developed in 1985 by Schaffer [Sch85], which he called the Vector Evaluated Genetic Algorithm (VEGA). This algorithm was different from the conventional Genetic Algorithm strategy only at the selection step. Here the population was divided into a number of subpopulations equal to the number of criteria. For each particular subpopulation the selection operator took into consideration the corresponding criterion. All subpopulations were shuffled together forming a population to which genetic operators (crossover, mutation) were applied in the common way.

Later Richardson et al. [Ric89] showed that the average influence of each objective function on a final solution is proportional to the size of the corresponding subpopulation. Therefore, the sizes of subpopulations can play a role as weights in the case when the decision maker considers the different importance of criteria. The Achilles' heel of VEGA is the tendency to produce extremal solutions, which are outstanding in one dimension, instead of compromise ones. The solutions with uniform distribution of criteria values

have less opportunity of surviving during selection. Yet in many cases such solutions are exactly the ones which we required.

The weakness of the VEGA algorithm arises because it does not provide an actual engine, which would move the population towards the true Pareto front. The idea of a non-dominated ranking was first expressed in 1989 by Goldberg [Gol89]. Initially in this method, he assigned the highest rank to all non-dominated solutions (see Figure 3.4). Then he took the next level of non-dominated solutions, excluding already ranked ones and assigned to them the next rank. In such a way, he ranked the whole population. The assigned rank was used instead of fitness for the comparison of solutions during selection.

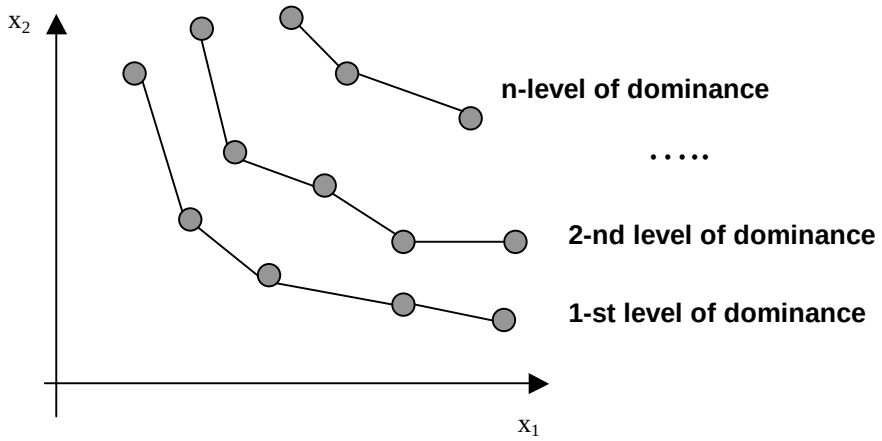

Figure 3.4: Goldberg's method of ranking the population

This method was applied to a scheduling problem by Hilliard et al. [Hil89] who confirmed that the non-dominated ranking algorithm can produce better results than VEGA. On the other hand, they found this technique to be more computationally expensive because the methods for checking the non-dominance of solutions were ineffective.

The non-dominated ranking algorithm belongs to the family of so-called Pareto-based techniques. These methods assign the fitness to an individual according to its dominance relations with other solutions in the population. In each generation, individuals are checked as to whether they are non-dominated and thus they force their way towards the Pareto-front. The main advantage of these methods is that they are independent of the shape of the Pareto-front. The two basic disadvantages are:

- *Genetic drift*: solutions tend to gather together in one part of the Pareto-front while leaving other parts empty.
- *Premature convergence*: algorithm stops to improve the solutions well before the true Pareto-front.

Goldberg proposed a way to overcome genetic drift which is known as *fitness sharing* [Gol89]. If several individuals are gathered in a group (based on the distance between them in the criteria space), then they "share" their fitness. The fitness of each individual is divided by the size of the group. Thus, the group performs as a single individual. This method is common for Pareto-based techniques. However, it requires the specification of a minimal distance, which should be considered as the threshold between separate individuals and the group.

A different way of ranking was proposed in 1993 by Fonseca and Fleming [FF93] who developed the Multi-Objective Genetic Algorithm (MOGA). Its ranking procedure is shown in Figure 3.5. Every solution obtains a rank which corresponds to the number of other solutions by which it is

dominated. In such a way the non-dominated solutions get rank 1. The solutions, which are dominated by only one other solution get rank 2, and so on. This approach helps to reduce the genetic drift because individuals grouped together have less chance of surviving (being dominated by more individuals).

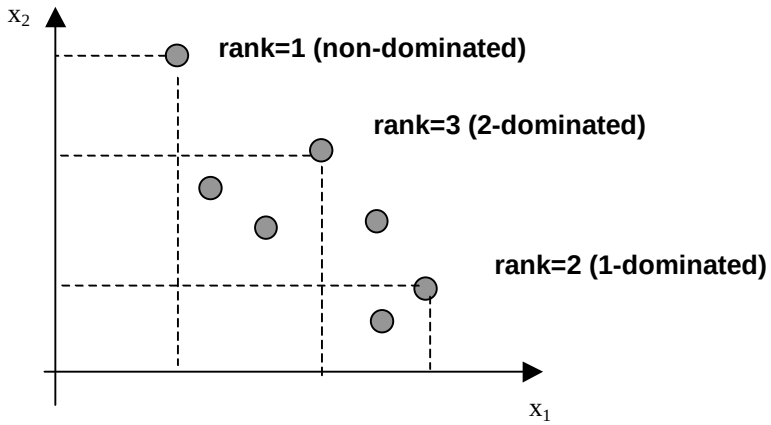

Figure 3.5: Ranking in Multiobjective Genetic Algorithm (MOGA)

In 1996, Fleming together with Shaw presented an initial study of the application of this technique to production scheduling [SF96a]. The comparison of the produced results with weighted sum ones indicated the superiority of MOGA. This algorithm produced more high quality solutions, including results, which could not be achieved by the weighted sum technique. Later, the authors (together with their colleagues) successfully applied their method to several real-world scheduling problems employing both discrete and continuous objective functions, e.g. [SF96b], [Shaw00]. The performance of MOGA was compared with other techniques and, in several problems, outperformed them. The authors underlined the significant role of the proper definition of the problem, constraints and objective functions for producing good solutions for large-scale scheduling problems.

At the same time as the development of MOGA, another variant of the Pareto-based approach called Niched Pareto Genetic Algorithm (NPGA) was being developed by Horn and Nafpliotis (under the guidance of Goldberg) [HN93]. They adapted a tournament selection approach to the multiobjective case and called it "Pareto domination tournament". The superiority between two solutions was determined by their dominance over the randomly chosen set of solutions. If two solutions dominate the same number of solutions from the chosen set, then the priority was given to the solution, which has a lower number of "neighbour" (gathered into the same group) individuals. This mechanism can significantly reduce the computational time for the identification of the non-dominated solutions. However, for good performance it requires a considerable larger population size. This algorithm had not been implemented widely. A more important fact is that this idea provides the basis for subsequent variants of the Pareto-based algorithms.

Srinivas and Deb returned to the basic Goldberg algorithm and combined it with the refined fitness sharing technique [SD94]. The authors suggested evaluating a solution by its so-called *dummy penalty*, which was calculated based on the rank of an individual and an average distance to other solutions. Such a hybrid was called the Nondominated Sorting Genetic Algorithm (NSGA) and its results usually dominated the ones produced by the previous methods. Thus, NSGA became the most popular Pareto-based technique within the multiobjective decision making community.

In 1999 Bagchi [Bag99] proposed the modification of the NSGA in order to copy the best parental individuals to later generations without changing them. He called this technique an Elitist Nondominated Sorting Genetic Algorithm and applied it to the job-shop scheduling problem. Bagchi thoroughly investigated the performance of this algorithm and compared it with a non-elitist one. The elitist version was found to be able to produce higher number of Pareto-optimal solutions in less computational time.

A very promising idea based on elitism has been expanded into a proposition of the non-generational Genetic Algorithm. Here a selection is implemented immediately after each recombination operator, and therefore the notion of generation is discarded. This conception was generally rejected for the single-objective Genetic Algorithm [Gol89] but later was found to be fruitful in the problems where solutions are highly correlated with each other. Valenzuela-Rendon and Uresti-Charre revealed the same behaviour of this method with multiobjective problems hence, in 1997 they developed the non-generational Genetic Algorithm for multiobjective optimisation [VU97]. The presented results proved the effectiveness of this method. Even though the optimal values were at the same level as the NPGA ones, the produced trade-off surface was much smoother and uniform, the dominated solutions were almost absent in a final population, and this algorithm was less time-consuming than the NPGA. While evaluating the individuals, this algorithm took into account both their dominance and “neighbour density” (the function which indicates the number of neighbour solutions), hence transforming the problem into a bi-objective form. The authors suggested solving this bi-objective

problem by using the weighted sum method. This idea was improved in 2000 by Borges and Barbosa [BB00]. To avoid the specification of weights, they defined a non-linear function of dominance and "neighbour density" measure.

Another idea for improving the performance of multiobjective evolutionary algorithms consists of the categorisation of parents for recombination ("mating restriction"). It was expressed by Goldberg [Gol89] and firstly applied within VEGA by Allenson [All92]. For a bi-objective case, he used the individuals of two types ("genders"). The member's gender (randomly assigned at birth) specified the objective by which the member should be evaluated. The crossover operated only with the individuals of different genders. In Allenson's opinion, such "biodiversity" could help to produce compromise solutions. He supposed that if such a feature exists in nature then it is worth investigating it in the algorithm. This approach was generalised by Lis and Eiben in [LE96]. They proposed to use several genders (the number of which is equal to the number of objectives) and the special multi-parent crossover. Their algorithm was tested on non-convex and discrete problems and showed a good cover of the Pareto-front.

A recent variant of the Pareto-based approach was proposed in 1999 by Zitzler and Thiele [ZT99]. They accumulated the most promising components of the previous techniques and called their method the Strength Pareto Evolutionary Algorithm (SPEA). This algorithm employed elitism by storing (separately) the subpopulation of non-dominated solutions, and a modified sharing technique. The ranking in this algorithm was implemented in a way,

which is opposite to the MOGA approach: the rank (strength) of an individual was determined by the number of members, which the given individual covers (dominates). The authors presented a detailed comparison of their algorithm with other multiobjective techniques in [ZT99] and [ZDT00]. The experiments were carried out on different shapes of Pareto-front: convex, non-convex, discrete etc. For all instances, the results (trade-off surfaces) produced by the proposed algorithm were the closest to true Pareto-front and the most uniformly distributed.

3.3.2 Non-Evolutionary Pareto-Based Techniques

The idea of employing the advantages of the Pareto-based approach within traditionally non-population techniques belongs to Ulungu et al. [UTF95] who developed the algorithm known as Multiobjective Simulated Annealing. During the search this algorithm keeps a list of non-dominated intermediate solutions, which are later compared with the selected candidates and the dominated members are replaced by the new ones. The probability of acceptance of worse solutions is calculated using the weighted sum approach. However, weights are randomly changed throughout the search in order to cover the wider region of the criteria space.

In [CJ98] Czyzak and Jaszkiewicz introduced the Pareto Simulated Annealing algorithm, which operates with a population of current solutions. At each iteration the algorithm evaluates the set of candidate solutions using a probability based on a weighted sum. To support the uniformity of the population distribution the special adaptive algorithm adjusts the weights

involved in the probability function in order to increase the distance between solutions.

The idea of keeping an archive of non-dominated solutions within Tabu Search was explored by Gandibleux et al. [GMF97]. In their Multiobjective Tabu Search algorithm they used a weighted sum method for the identification of the candidate solutions among neighbours, but the concept of the tabu list was revised. Instead of storing recent moves, it stored those objectives, which were mostly improved at recent steps. This feature helped to diversify the weights in order to distribute non-dominated solutions more uniformly.

Another variant of the Pareto-based Tabu Search was developed by Hansen [Han97], and was also called Multiobjective Tabu Search. He used a population of current solutions and every solution had its own tabu list of recent moves. Besides this, the proposed algorithm included a MOGA-like ranking and removing of worst-rank solutions in order to prevent a genetic drift.

A Pareto-based variant of Hill-Climbing algorithm was introduced in 1999 by Knowles and Corne [KC99]. The authors claimed that it was the simplest and most transparent multiobjective technique. They called this algorithm the Pareto Archived Evolution Strategy (PAES) because it maintained an archive of intermediate non-dominated solutions. The quality of every candidate solution was evaluated by using the relation of dominance to all archive members. In addition, the algorithm aimed to remove solutions from the most crowded regions of the criteria space. In [KC00a] the authors

presented a comprehensive comparison of PAES algorithm (including its extensions) with different versions of NPGA and NSGA on a wide range of benchmark problems. The figures showed little difference between the quality of results (the authors indicated the only one case where PAES was clearly beaten by NSGA). It can be concluded that the Hill-Climbing based algorithm is far less time-expensive than any based on Genetic Algorithms.

3.3.3 Hybridisation of Pareto-based Techniques

In order to improve the performance of Pareto-based Hill-Climbing Knowles and Corne combined it with a Genetic Algorithm, which lead to a memetic multiobjective algorithm [KC00b]. The new method was called Memetic-PAES. The authors proposed two archives: the global archive for storing non-dominated solutions from the current population and the local archive which was used in the same way as in PAES at the local search phase. The algorithm employed Hill-Climbing for improvement of the quality of population members while both archives were used for choosing the parents in the recombination phase. This algorithm was compared with the SPEA on Zitzler and Thiele's benchmark problems. Both algorithms (SPEA and Memetic-PAES) showed approximately the same performance on the small-sized datasets. However, as the size of the problems increased the difference in the results (in favour of Memetic-PAES) became more apparent. The authors noted that it was too early to make any conclusions because of the difference between these algorithms. Moreover, there was no information about the computational cost of the Memetic-PAES algorithm, which is expected to be very high.

3.4 The Evaluation of Trade-off Surfaces

With the number of different Pareto-based multiobjective techniques growing, the question of measuring and comparing the quality of their results becomes crucial. It is intuitively apparent that the closer the trade-off surface is to the true Pareto-front then the better it is. In addition to the closeness, the quality of solutions is affected by the number of solutions in the set, the uniformity of their distribution and the wideness of the covered sector in the criteria space. An ideal quality measure should take into account all these factors. However, it would be very difficult to determine such a measure. None of the measures proposed in the literature are comprehensive enough to consider all of the factors but they each focus on certain aspects of solution quality.

The outperformance relation is defined between two non-dominated sets. One set outperforms the other when all of its solutions "cover" (dominate or are equal to) solutions from the other set. Hansen and Jaszkiewicz in [HJ98] categorised the outperformance relation into the following classifications:

- *Weak outperformance* occurs when both sets are almost coinciding and the better set overlaps the worse one only by non-dominated points.
- *Strong outperformance* includes the properties of the weak one but at least one point from the better set should dominate the point(s) from the worse set.
- *Complete outperformance* arises when all solutions from the worse set are dominated by solutions from the better set.

In situations where a distinct outperformance cannot be detected the numerical measures of the quality of solutions (so-called *metrics*) are used. Different variants of metrics were examined in [KC02] where the authors considered that some of them could also provide the quantitative gauge for the outperformance relation. A number of simple metrics, e.g. “error ratio” - the proportion of points which do not belong to true Pareto-front, “generational distance” - average distance to the nearest point in true Pareto-front, etc. were discussed in [VV99]. All these metrics compare the given set with the true Pareto-front, and therefore, their application is limited by the problems where the true Pareto-front is known in advance.

Two other metrics were proposed in [Zit99]: the so-called *C*-metric and *S*-metric. The *C*-metric defines a degree of dominance of one set by another as the ratio of the number of covered solutions to the size of the whole set. This metric is asymmetric. This may cause the so-called “cross-cycling” effect. For example, if we have three non-dominated sets then it may be the case that the first set has the better *C*-metric value than the second one, the second better than the third, and (surprisingly) the third better than the first.

The *S*-metric is calculated as a hypervolume of a region enclosed by the trade-off surface and a chosen reference point. Such a region in two-dimensional space, where the hypervolume degrades into an area, is illustrated in Figure 3.6. Two trade-off surfaces are shown: the first contains grey points and the second contains black ones. The reference point is marked with *R*. The first region (S_1) is bounded by a dotted line and the second region

(S_2) by the solid one. The region, enclosed by the first surface is larger than the second one (S_1>S_2), which indicates the higher quality of the set of grey points over the black ones.

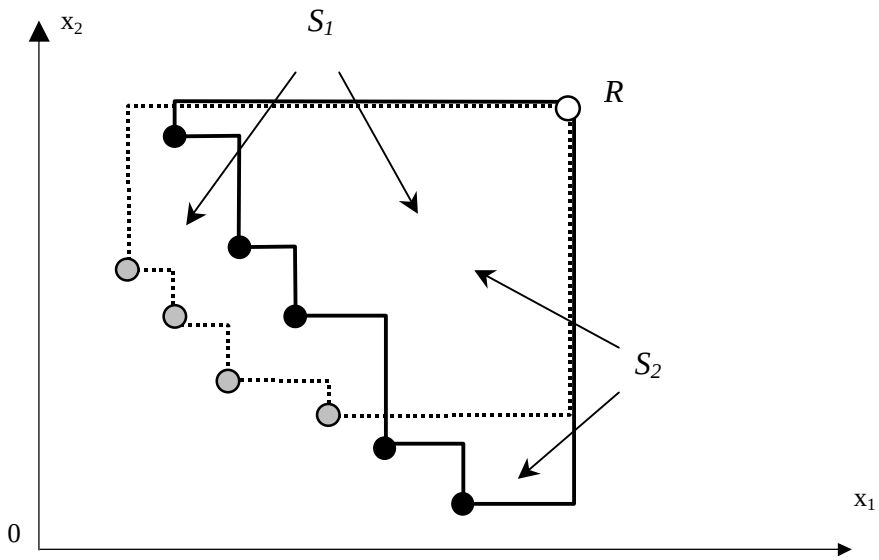

Figure 3.6: *S*-metric for comparison of non-dominated sets

Even though the *S*-metric can be computationally expensive in the high-dimensional spaces, it is free from the drawbacks of the *C*-metric. Its only weakness is the requirement for a reference point and it is not clear how to define it.

Two *D*-metrics were introduced in [CJ98] – *D1* and *D2*. Both of them required the definition of a reference set of solutions. They calculated the distances between each solution in the evaluated set to the nearest solution in the reference set. The *D1*-metric calculated the average distance while the *D2*-metric estimated the maximum one. The authors also suggested the use of the ratio *D2/D1* as a measure of uniformity of distribution of solutions in the evaluated set. In their experiments the authors employed the true Pareto-front

as a reference set. However, there were no recommendations for how to generate the reference set in the situations where the true Pareto-front is unknown.

The group of *R*-metrics was proposed in [HJ98]. This group comprises probably the most advanced techniques for comparison of non-dominated sets. They are based on the probability of the satisfaction of the decision maker preferences by the evaluated set. The decision maker's preferences are modelled by so-called "utility functions" and the probability distribution of the set of such functions is calculated. Some variants of the *R*-metric use the reference set in the same way as *D*-metrics do. However, the definition of the proper set of utility functions requires certain knowledge about the decision maker's preferences.

3.5 Multiobjective Exam Timetabling

Traditionally, exam timetabling problems are solved by A Priori approaches. The most popular method is the weighted sum. This method was applied within Simulated Annealing [TD96a], [TD98], Tabu Search [BN96], Genetic Algorithms [CRF94], Memetic Algorithms [BN99], etc. The Lexicographic approach has been applied to the examination timetabling by several authors: [LC91], [TD93], [PS02].

There are very few publications about the performance of A Posteriori methods with exam timetabling problems. Possibly, the only study of the application of a mutation-only MOGA-like ranking algorithm to examination timetabling has been presented by Paquete and Fonseca in [PF01]. Although

the numerical values of the results were not presented, the authors mentioned that they compared their technique with an aggregation approach on a real world exam timetabling problem. They concluded that the Pareto-based method produced a better cover of the trade-off surface while the aggregation method more effectively minimised the violation of soft constraints.

In addition several authors (e.g. [CP01], [TA00]) have applied A Posteriori algorithms to the class/teacher timetabling. This is class of problems, which has some similarities (but also distinct differences) with examination timetabling. However, these publications do not include a comparison of the performance of the presented algorithms with other techniques.

3.6 Summary

In this chapter the current state-of-the-art in multiobjective optimisation is described and discussed. The conventional hardship of multiobjective optimisation is the estimation of the quality of solutions. Formally, all non-dominated solutions can be considered to be optimal. However, only one solution from the non-dominated set can be selected as a final result. To select it, the decision maker has to express his/her preferences.

Two main approaches were discussed in this chapter: A Priori and A Posteriori. In A Priori methods the decision maker specifies his/her preferences regarding the solution before running the algorithm. The most popular method involves the aggregation of the problem's objectives into a single (cost) function in order to apply some single-objective metaheuristic. Usually the cost is calculated as the weighted sum of objectives or the distance

to some specified goal. In the Lexicographic approach, criteria are divided into groups which take into consideration the importance of objectives and the search is conducted sequentially starting from the group with the objectives of highest importance. The Compromise Programming technique operates with different distance measures which aggregate the objective values.

Methods belonging to the A Posteriori group aim to produce a set of non-dominated solutions, among which the decision maker can select the preferable one. This group mainly comprises different extensions of Genetic Algorithms. However, several examples of the adaptation of non-evolutionary approaches such as Simulated Annealing, Tabu Search and Hill-Climbing as well as their hybridisation are known from literature.

Different metrics for the comparison of non-dominated sets were discussed. The conclusion is that there is no unique metric which will take into consideration all aspects of the quality of solutions, the number of produced solutions, the uniformity of their distribution and the coverage of the criteria space. It appears that the problem of selecting the method for multiobjective optimisation, which produces solutions of highest quality with respect to different metrics is a multiobjective problem.

It may be concluded that little research work has been carried out on multiobjective university timetabling, especially on exam timetabling. These problems are multiobjective by their nature and this observation has served as the motivation for the research work that is described in this thesis.

Chapter 4.

Exam Timetabling Specification and Data

This chapter provides a formal mathematical statement of the university examination timetabling problem. The general problem specification is given together with additional constraints, which are widely used in real-world exam timetabling. A number of real-world university examination problems have been collected and used as benchmark problems within the timetabling research community. In the presented research they are used for experiments, therefore their description is included at the end of this chapter.

4.1 A Formalisation of Exam Timetabling Problems

Different types of examination timetabling problem can be specified depending on the chosen set of constraints. A clash-free requirement (no student can sit two exams at the same time) is the most common hard constraint for these problems. The difference in problem statements is caused by a variety of other constraints, which leads to a corresponding difference in objective functions, which provide numerical measures of violation of these constraints. Several variants of exam timetabling problems are studied in the course of this thesis in order to produce results which are comparable with published ones and therefore, objective functions are defined as was suggested in different publications. The formalisation of these problems is given in the next sections.

4.1.1 A Specification of the Basic Problem

- *The common input data* for examination timetabling is given as:

– N is the number of exams;

– M is the number of students;

– P is the given number of timeslots;

– The conflict matrix $C=(c_{ij})_{N\times N}$ where each element (denoted by c_{ij} where i , $j \in \{1,\ldots,N\}$) is the number of students that have to take both exams i and j. This is a symmetrical matrix of size N, where diagonal elements c_{ii} equal the number of students who have taken exam i.

- The solution of the problem is represented as a vector $T=(t_i)_N$, where t_i specifies the assigned timeslot for exam i ($i \in \{1,\ldots,N\}$). Each timeslot can be thought of as a non-negative integer ($1 \le t_i \le P$).

The general variant of the exam timetabling problem can be formulated as follows:

$$Minimise \quad \frac{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} c_{ij} \cdot prox(t_i, t_j)}{M}, \tag{4.1}$$

$$\text{where} \quad prox(t_i, t_j) = \begin{cases} 2^{5-|t_i - t_j|} & if\ 1 \le |t_i - t_j| \le 5 \\ 0 & otherwise \end{cases},$$

such that

$$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} c_{ij} \cdot clash(t_i, t_j) = 0 \qquad \text{where} \quad clash(t_i, t_j) = \begin{cases} 1 & if\ t_i = t_j \\ 0 & otherwise \end{cases}. \tag{4.2}$$

Equation (4.2) presents the clash-free requirement (no student can sit two exams at the same time) which is considered as a hard constraint. A feasible solution is one which completely satisfies all the hard constraints.

The statement in (4.1) represents proximity between exams, which is the most conventional soft constraint suggested in [CLL86]. If a student has two consecutive exams then a penalty value equal to 16 is assigned. Two exams with one empty period between them will be assigned a penalty value of 8. Two empty periods correspond to a penalty of 4 and so on. In order to have a relative measure, this sum is divided by the total number of students. The given soft constraint is taken into account in the single-objective version of the exam timetabling problem, which was used in the experiments presented in Sections 5.4, 5.5, 5.6.1, 5.7.1 and 5.7.2.

The proposed formalisation of exam timetabling problem can be illustrated by a small numerical example. Let us consider N=12, M=70 and P=6. An example of correspondent conflict matrix C is given in Table 4.1.

Table 4.1: An example of conflict matrix C

$i \backslash j$	1	2	3	4	5	6	7	8	9	10	11	12
1	18	5	0	6	0	5	11	0	8	4	0	5
2	5	9	8	0	3	0	8	2	0	0	8	0
3	0	8	17	2	0	12	5	0	0	8	1	0
4	6	0	2	12	0	7	1	7	9	11	2	8
5	0	3	0	0	22	0	0	7	0	7	4	15
6	5	0	12	7	0	15	4	1	5	1	0	0
7	11	8	5	1	0	4	14	0	0	3	8	4
8	0	2	0	7	7	1	0	9	2	0	2	0
9	8	0	0	9	0	5	0	2	14	0	6	8
10	4	0	8	11	7	1	3	0	0	12	4	0
11	0	8	1	2	4	0	8	2	6	4	12	0
12	5	0	0	8	15	0	4	0	8	0	0	21

One of feasible solutions to this example problem could be defined by vector T=(3,6,1,6,5,4,5,3,2,2,4,1). The illustrative representation of this solution is given in Table 4.2 where exams are sorted accordingly to assigned timeslots.

Table 4.2: Allocation of exams to timeslots

Timeslot	1	2	3	4	5	6
Exams	3,12	9,10	1,8	6,11	5,7	2,4

The remaining part of this section demonstrates the calculation of the proximity cost to this solution using formula (4.1). Here each pair of exams (i,j) gains its own proximity coefficient $prox(t_i,t_j)$ depending on assigned timeslots. All these coefficients are collected in a form of a matrix in Table 4.3. Note that formula (4.1) considers only those pairs of exams where $j>i$, therefore the values in the left-bottom triangle of the matrix are not defined.

Table 4.3: The matrix of proximity coefficients $prox(t_i,t_j)$

$i \backslash j$	1	2	3	4	5	6	7	8	9	10	11	12
1	-	4	8	4	8	16	8	0	16	16	16	8
2	-	-	1	0	16	8	16	4	2	2	4	1
3	-	-	-	1	2	4	2	8	16	16	4	0
4	-	-	-	-	16	8	16	4	2	2	8	1
5	-	-	-	-	-	16	0	8	4	4	16	2
6	-	-	-	-	-	-	16	16	8	8	0	4
7	-	-	-	-	-	-	-	8	4	4	16	2
8	-	-	-	-	-	-	-	-	16	16	16	8
9	-	-	-	-	-	-	-	-	-	0	8	16
10	-	-	-	-	-	-	-	-	-	-	8	16
11	-	-	-	-	-	-	-	-	-	-	-	4
12	-	-	-	-	-	-	-	-	-	-	-	-

Formula (4.1) contains the sum of multiplications of elements c_{ij} by corresponding coefficients $prox(t_i,t_j)$ where ($i \in \{1,\ldots,N\text{-}1\}$, $j \in \{i\text{+}1,\ldots,N\}$). The calculation of this sum for the given example is depicted in Figure 4.1. For the simplicity of representation, the intermediate sum of each row is shown separately.

5*4+0*8+6*4+0*8+5*16+11*8+0*0+8*16+4*16+0*16+5*8=444
+8*1+0*0+3*16+0*8+8*16+2*4+0*2+0*2+8*4+0*1=224
+2*1+0*2+12*4+5*2+0*8+0*16+8*16+1*4+0*0=192
+0*16+7*8+1*16+7*4+9*2+11*2+2*8+8*1=164
+0*16+0*0+7*8+0*4+7*4+4*16+15*2=178
+4*16+1*16+5*8+1*8+0*0+0*4=128
+0*8+0*4+3*4+8*16+4*2=148
+2*16+0*16+2*16+0*8=64
+0*0+6*8+8*16=176
+4*8+0*16=32
+0*4=0
Total sum=1750

Figure 4.1: An example of calculation of the sum in formula (4.1)

Following to formula (4.1) the resulting sum should be divided by the total number of students *M*. Thus, dividing 1750 by 70 one can obtain the overall penalty equal to 25.

4.1.2 A Specification of Additional Constraints

The cost function given in (4.1) is a general measure of the quality of solutions of exam timetabling problems. However, in order to take into consideration the different requirements of the participants in a real-world university

examination timetabling process, a number of additional constraints have to be taken into account instead of (or together with) the described one. In the course of this thesis nine constraints are considered which are split into three groups related to room capacities, closeness of exams, and the time and order of exams. The three groups of constraints are presented below.

- *The room capacities constraint* penalises the number of students which exceed the available room capacities. It may, of course, be the case that the same student contributes to several capacity violations. This constraint assumes the specification of the number of seats S that are available for every timeslot. Thus the total number of students in any period which exceeds S is expressed by formula (4.3).

$$\sum_{p=1}^{P}\left(S_p - S\right)\cdot exc(p) \qquad \text{where} \qquad exc(p)=\begin{cases}1 & if\ S_p > S\\ 0 & otherwise\end{cases}, \tag{4.3}$$

where S_p is the number of students taking exams in period p (which is of course a particular timeslot). This can be calculated by the next formula:

$$S_p = \sum_{i=1}^{N} C_{ii}\cdot per(t_i, p) \qquad \text{where} \qquad per(t_i, p)=\begin{cases}1 & if\ t_i = p\\ 0 & otherwise\end{cases}. \tag{4.4}$$

Let us consider that in the numerical example described in Section 4.1.1 the number of seats per timeslot S is equal to 35. Following to the allocation of exams to timeslots given in Table 4.2, one can calculate the number of students in each timeslot as illustrated in Table 4.4.

Table 4.4: Number of students in timeslots

Timeslot	1	2	3	4	5	6
Number of Students	17+21=38	14+12=26	18+9=27	15+12=27	22+14=36	9+12=21

This table shows that in two timeslots (1 and 5) the number of students is more than *S*. Thus, the total penalty for the violation of this constraint is calculated as: 38-35+36-35 and is equal to 4.

- *Closeness of exams constraints* penalise the number of conflicts where exams are not adequately spread out in time so that students do not have enough free time between two exams. These constraints require the definition of the vector $D=(d_m)_P$. Every element d_m (where $m \in \{1,\ldots,P\}$) specifies the number (for every timeslot p) which represents the day in an examination session. In further experiments the examination session is considered to start from Monday having 3 exams every day, except Saturdays (only one timeslot) and Sundays (no exams). Thus, the first three timeslots correspond to day "1" (Monday of the 1st week), the 4th,5th and 6th timeslots correspond to day "2" (Tuesday of the 1st week), etc. Day "6" appears only in one timeslot (Saturday of the 1st week), after which the second week starts. This list continues until all the given timeslots are represented. Thus the distribution of days can be expressed in the following way:

$$D_P = (1,1,1,2,2,2,3,3,3,4,4,4,5,5,5,6,8,8,8,9,9,9,10,10,10,11,11,\ldots). \qquad (4.5)$$

Note, that Sundays (for example: day "7") have a number even though it is not actually used. This is done in order to aid the calculation of adjacent days and

overnight conflicts (formulae (4.8) and (4.9)). These constraints are defined by formulae (4.6)-(4.9):

– the number of conflicts where students have exams in adjacent periods on the same day can be found by taking

$$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} c_{ij} \cdot adjs(t_i, t_j), \tag{4.6}$$

$$\text{where} \quad adjs(t_i, t_j) = \begin{cases} 1 & if \ \left(\left|t_i - t_j\right| = 1\right) \wedge \left(d_{t_i} = d_{t_j}\right) \\ 0 & otherwise \end{cases},$$

The calculation of this number could be illustrated in the same way as it was done in Section 4.1.1 for the proximity cost. When assuming that in the given example the distribution of days is represented by the vector D_p=(1,1,1,2,2,2), then all pairs of exams gain correspondent coefficients $adjs(t_i,t_j)$ shown in Table 4.5.

Table 4.5: The matrix of coefficients *adjs* (t_i,t_j)

$i \backslash j$	1	2	3	4	5	6	7	8	9	10	11	12
1	-	0	0	0	0	0	0	0	1	1	0	0
2	-	-	0	0	1	0	1	0	0	0	0	0
3	-	-	-	0	0	0	0	0	1	1	0	0
4	-	-	-	-	1	0	1	0	0	0	0	0
5	-	-	-	-	-	1	0	0	0	0	1	0
6	-	-	-	-	-	-	1	1	0	0	0	0
7	-	-	-	-	-	-	-	0	0	0	1	0
8	-	-	-	-	-	-	-	-	1	1	0	0
9	-	-	-	-	-	-	-	-	-	0	0	1
10	-	-	-	-	-	-	-	-	-	-	0	1
11	-	-	-	-	-	-	-	-	-	-	-	0
12	-	-	-	-	-	-	-	-	-	-	-	-

In contrast to the proximity coefficients all numbers presented in Table 4.5 are equal to 1 or 0. This simplifies the calculation of the sum in formula (4.6). Therefore, Figure 4.2 shows only two lines of this computation (which can be continued by the interested reader).

5*0+0*0+6*0+0*0+5*0+11*0+0*0+8*1+4*1+0*0+5*0=12
+8*0+0*0+3*1+0*0+8*1+2*0+0*0+0*0+8*0+0*0=11
+. . .
Total sum=58

Figure 4.2: An example of calculation of the number of adjacent conflicts

The final sum (equal to 58) is the sought number of adjacent conflicts.

– the number of conflicts where students have two or more exams in the same day is expressed as follows

$$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} c_{ij} \cdot sday(t_i, t_j) \qquad \text{where} \qquad sday(t_i, t_j) = \begin{cases} 1 & if\ \left(d_{t_i} = d_{t_j}\right) \\ 0 & otherwise \end{cases}, \qquad (4.7)$$

The way of calculation of this number is analogous to the described above calculation of the number of adjacent conflicts. The difference is caused only by coefficients $sday(t_i,t_j)$, which have different values than $adjs(t_i,t_j)$. These values can be also represented in the form of a matrix, which two lines are shown in Table 4.6.

Table 4.6: The matrix of coefficients $sday\ (t_i,t_j)$

$i \setminus j$	1	2	3	4	5	6	7	8	9	10	11	12
1	-	0	1	0	0	0	0	1	1	1	0	1
2	-	-	0	1	1	1	1	0	0	0	1	0
…	…	…	…	…	…	…	…	…	…	…	…	…

The calculation of the sum in formula (4.7) is illustrated in Figure 4.3 in the same way as in the previous example. Here the number of the same days conflicts is equal to 80.

5*0+0*1+6*0+0*0+5*0+11*0+0*1+8*1+4*1+0*0+5*1=17
+8*0+0*1+3*1+0*1+8*1+2*0+0*0+0*0+8*1+0*0=19
+. . .
Total sum=80

Figure 4.3: An example of calculation of the number of same day conflicts

– the number of conflicts where students have exams in adjacent days can be calculated by

$$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} c_{ij}\cdot adjd(t_i,t_j) \qquad \text{where} \quad adjd(t_i,t_j)=\begin{cases}1 & if\ \left(\left|d_{t_i}-d_{t_j}\right|=1\right)\\ 0 & otherwise\end{cases}, \qquad (4.8)$$

The calculation of this number is illustrated in the same way as previous ones. Two lines of a new matrix of coefficients $adjd(t_i,t_j)$ for the used here example problem are given in Table 4.7.

Table 4.7: The matrix of coefficients $adjd$ (t_i,t_j)

$i \setminus j$	1	2	3	4	5	6	7	8	9	10	11	12
1	-	1	0	1	1	1	1	0	0	0	1	0
2	-	-	1	0	0	0	0	1	1	1	0	1
…	…	…	…	…	…	…	…	…	…	…	…	…

Using these coefficients one can calculate the number of adjacent days conflicts as shown in Figure 4.4. Here this value is equal to 127.

5*1+0*0+6*1+0*1+5*1+11*1+0*0+8*0+4*0+0*1+5*0=27
+8*1+0*0+3*0+0*0+8*0+2*1+0*1+0*1+8*0+0*1=10
+. . .
Total sum=127

Figure 4.4: An example of calculation of the number of adjacent days conflicts

– the number of conflicts where students have exams in overnight adjacent periods (adjacent periods at adjacent days except the pair: “Saturday - first slot on Monday”) is expressed in

$$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} c_{ij}\cdot ovnt(t_i,t_j),$$
$$\text{where} \quad ovnt(t_i,t_j)=\begin{cases}1 & if\ \left(\left|t_i-t_j\right|=1\right)\wedge\left(\left|d_{t_i}-d_{t_j}\right|=1\right)\\ 0 & otherwise\end{cases}. \qquad (4.9)$$

The way of calculation of this number is similar to other constraints from this group (described above). The correspondent matrix of coefficients $ovnt(t_i,t_j)$ is presented by two lines in Table 4.8 and the example of calculation of the total sum is shown in Figure 4.5. In the presented example, this sum is equal to 8.

Table 4.8: The matrix of coefficients $ovnt$ (t_i,t_j)

$i \setminus j$	1	2	3	4	5	6	7	8	9	10	11	12
1	-	0	0	0	0	1	0	0	0	0	1	0
2	-	-	0	0	0	0	0	0	0	0	0	0
…	…	…	…	…	…	…	…	…	…	…	…	…

5*0+0*0+6*0+0*0+5*1+11*0+0*0+8*0+4*0+0*1+5*0=5
+8*0+0*0+3*0+0*0+8*0+2*0+0*0+0*0+8*0+0*0=0
+. . .
Total sum=8

Figure 4.5: An example of calculation of the number of overnight conflicts

- *Time and order of exams constraints* penalise the number of times when students are affected by inappropriate allocations of (including order of) exams. As was the case with room capacities above, one particular student may be affected at several different points in one timetable:

– the number of times that a student has an exam that is not scheduled in a time period of the proper duration. The specification of this constraint is given by two vectors. The vector $R=(r_k)_N$, whose elements r_k indicate the duration of exam k ($k \in \{1,\ldots,N\}$) and the vector $Q=(q_m)_P$ where q_m is the duration of timeslot m ($m \in \{1,\ldots,P\}$). The duration in both vectors is expressed in hours.

Expression (4.10) shows the number of students, who contribute to the violation of this constraint

$$\sum_{k=1}^{N} C_{kk} \cdot dur(k) \qquad \text{where} \qquad dur(k) = \begin{cases} 1 & if\ r_k > q_{t_k} \\ 0 & otherwise \end{cases}. \qquad (4.10)$$

In the case of the Nott-94 problem (see Section 4.3) exams last no longer than 3 hours ($1 \le r_k \le 3$). On the other hand, the duration of timeslots is distributed in the following way: the first (AM) timeslot of each day lasts 3 hours, while the duration of the second and third ones is 2 hours. As the exams of duration of 1 or 2 hours can be scheduled anywhere, the problem of fitting an exam into a timeslot of proper duration can be simplified into the one where 3-hour exams are required to be scheduled into AM timeslots.

For the used example problem, one can define these two vectors as follows: R=(1,2,2,2,3,3,2,1,3,2,3,1) and Q=(3,2,2,3,2,2). Due to such requirements exams 5 and 9 are scheduled in wrong periods, which generates a penalty equal to 36.

– the number of times that a student has an exam that is not scheduled in the preassigned time period. This constraint is defined by the matrix $A = [\ a_{kp}\ ]_{N \times P}$ where each element (denoted by a_{kp} where $k \in \{1,\ldots,N\}$ and $p \in \{1,\ldots,P\}$) is given in the following way

$$a_{kp} = \begin{cases} 1, & \text{if exam } k \text{ cannot be scheduled in time period } p \\ 0, & \text{otherwise} \end{cases}. \qquad (4.11)$$

Correspondingly, the number of students who sit exams scheduled with a violation of the "preassignment" constraint can be found by the following expression

$$\sum_{k=1}^{N} C_{kk} \cdot A_{kt_k} \ , \tag{4.12}$$

The definition of preassignment constraint could be illustrated (regarding the used example) by a matrix presented in Table 4.9.

Table 4.9: An example of preassignment matrix *A*

$p \backslash k$	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	0	0	0	0	0	0	0	0	1	0
2	0	1	0	1	0	0	0	0	0	0	0	0
3	0	1	0	1	1	0	0	0	0	0	1	0
4	0	0	0	0	1	0	0	1	0	0	1	0
5	0	0	0	0	0	0	1	1	0	0	0	0
6	0	0	0	0	0	0	1	1	0	0	0	0

In this example exams 7 and 11 are scheduled with violation of this constraint (the total penalty is equal to 26).

– the number of times that a student has an exam that is not scheduled before/after another specified exam. To consider this constraint the number of requirements U is given where two exams should be scheduled in a given sequence. The necessary data is presented by the matrix $G = [\ g_{ub}]_{U\times 2}$ where $u \in \{1,\ldots,U\}$ and $b \in \{1,2\}$ and each pair of elements g_{u1} and g_{u2} specifies two exams where the first one has to be scheduled before the second one. It is considered that improper scheduling of any pair of exams

affects the students from both given exams. Their total number can be calculated by the following expression

$$\sum_{k=1}^{U}\left(C_{g_{k1}g_{k1}}+C_{g_{k2}g_{k2}}\right)\cdot bef(k) \qquad \text{where } bef(k)=\begin{cases}1 & if\ t_{g_{k1}}\geq t_{g_{k2}}\\ 0 & otherwise\end{cases}, \quad (4.13)$$

While continuing the illustration by the same example problem, let us define U=3 and the matrix G as given in Table 4.10

Table 4.10: An example of before/after matrix G

$b^{\backslash u}$	1	2	3
1	9	8	4
2	7	2	6

Here exams 4 and 6 are scheduled in wrong order and therefore penalty = 27.

– the number of times that a student has an exam that is not scheduled immediately before/after another specified exam. The description of this constraint is analogous to the previous one. The number of pairs of exams is given as V and the matrix H = [$h_{vb}]_{V\times 2}$ contains corresponding pairs h_{v1} and h_{v2} where the first exam should be scheduled immediately before the second one ($v \in \{1,\ldots,V\}$). The number of improper scheduled pairs is expressed in

$$\sum_{k=1}^{U}\left(C_{h_{k1}h_{k1}}+C_{h_{k2}h_{k2}}\right)\cdot imb(k),$$
$$\text{where} \quad imb(k)=\begin{cases}1 & if\ \left(t_{h_{k2}}-t_{h_{k1}}\neq 1\right)\vee\left(d_{t_{h_{k1}}}\neq d_{t_{h_{k2}}}\right)\\ 0 & otherwise\end{cases}. \quad (4.14)$$

The example of calculation of this constraint is similar to the previous one. It is assumed that V=2 and matrix H is defined by Table 4.11.

Table 4.11: An example of immediately before/after matrix *H*

$b^{\backslash v}$	1	2
1	1	3
2	5	10

In this example the wrongly ordered exams are 1 and 5 and correspondent penalty is equal to 30.

4.2 A Multiobjective Statement of Exam Timetabling Problem

Soft constraints usually have very different importance for different timetable officers (decision makers). They are generally incompatible and often conflicting with each other. Generally exam timetabling problems can be considered to be multiobjective problems. This study investigates both single and multiobjective versions of exam timetabling problem. In the multiobjective variant a number of objectives (criteria) can be defined to evaluate the quality of timetables from different points of view. Each objective expresses a measure of the violation of the corresponding constraint. The following notation is introduced:

– K is the number of objectives.

– f_k is the value of criterion X_k (objective function), where $k \in \{1,...,K\}$.

The multiobjective timetabling problem can be stated analytically in the following way. Additionally to the common input data presented in Section 4.1.1 the vector $W = (w_1,\ldots,w_k,\ldots,w_K)$ is specified where w_k , $k \in \{1,...,K\}$ denotes the weight of criterion X_k. The task is to determine the

vector $T=(t_i)_N$, which makes all the elements of the vector $WF = (w_1 f_1(T),..., w_k f_k(T),..., w_K f_K(T))$ as small as possible, subject to the hard constraint expressed in formula (4.2).

In this thesis two versions of the multiobjective exam timetabling problem are investigated, which are different with respect to the sets of hard and soft constraints. In Sections 6.1 and 7.3.3 experiments are conducted with nine objectives $\{X_1, \ldots, X_9\}$ which express measures of the violation of constraints given in Section 4.1.2. The description of the objectives are given in Table 4.12.

Table 4.12: A description of objectives

Description	**Objective**
The number of students which exceed the available room capacities	X_1
The number of conflicts where students have exams in adjacent periods on the same day	X_2
The number of conflicts where students have two or more exams in the same day	X_3
The number of conflicts where students have exams in adjacent days	X_4
The number of conflicts where students have exams in overnight adjacent periods	X_5
The number of times that a student has an exam that is not scheduled in a time period of the proper duration	X_6
The number of times that a student has an exam that is not scheduled in the preassigned time period	X_7
The number of times that a student has an exam that is not scheduled before/after another specified exam	X_8
The number of times that a student has an exam that is not scheduled immediately before/after another specified exam	X_9

Sections 5.6.2, 6.2 and 7.3-7.5 are devoted to a bi-objective exam timetabling problem (formulated in the same way as in [BN99]). Here the quality of solutions is evaluated by the values of criteria X_2 and X_5, which count the number of conflicts where students have exams in adjacent periods on the same day and the number of conflicts where students have exams in overnight periods. The "room capacities" constraint is regarded as a hard one.

4.3 Benchmark Exam Timetabling Datasets

The examination timetabling research community has established a publically available set of examination timetabling data and supplementary instructions (for example: the way of calculating cost functions, etc.). All experiments discussed in this thesis were carried out with datasets, taken from the following open sources:

1. Michael Carter's collection of examination timetabling data, which can be downloaded from archive at: ftp://ftp.mie.utoronto.ca/pub/carter/testprob, comprising of 13 sets of examination data, which took place at different universities during 1983-1993. Their parameters are presented in Table 4.13.

Table 4.13: The parameters of Carter's collection of examination datasets

Data set	Institution	Exams	Students	Enrolments
CAR-F-92	Carleton University, Ottava	543	18 419	55 552
CAR-S-91	Carleton University, Ottava	682	16 925	56 877
EAR-F-83	Earl Haig Collegiate Institute, Toronto	189	1 125	8 108
HEC-S-92	Ecole des Hautes Etudes Commercials, Montreal	80	2 823	10 632
KFU-S-93	King Fahd University, Dharan	461	5 349	25 118
LSE-F-91	London School of Economics	381	2 726	10 919
PUR-S-93	Purdue University, Indiana	2419	30 032	120 690
RYE-S-93	Ryeson University, Toronto	481	11 483	45 052
STA-F-83	St Andrew's Junior High School, Toronto	138	611	5 751
TRE-S-92	Trent University, Peterborough, Ontario	261	4 360	14 901
UTA-S-92	Faculty of Arts and Sciences, University of Toronto	638	21 267	58 981
UTE-S-92	Faculty of Engineering, University of Toronto	184	2 750	11 796
YOR-F-83	York Mills Collegiate Institute, Toronto	180	941	6 029

2. The disposition of exams and students at Nottingham University in 1994 (Nott-94), which is available from: ftp://ftp.cs.nott.ac.uk/ttp/Data/Nott94-1/. This dataset includes 800 exams and 7 896 students, which compose 33 997 "student-exam" pairs (enrolments).

The investigation of the 9-objective case was carried on the Nott-94 problem. It is the only available benchmark problem, where time and order

constraints data are given. This data is useful for the calculation of objective functions f_6...f_9. The characteristics of the data are given in Table 4.14.

Table 4.14: The details of constraints for Nott-94 problem

Characteristics of the examination data	**Value**
Number of exams which are 3 hours long (require AM timeslot)	46
Number of exams which require certain timeslot(s)	9
Number of pairs of exams which have to be scheduled before/after	2
Number of pairs of exams which have to be scheduled immediately before/after	1

Chapter 5.

A Time Predefined Approach to Examination Timetabling

5.1 The Role of Computational Time in the Process of Solving Timetabling Problems

The "trade-off" between the quality of solution and the search time in examination timetabling has been discussed in several papers. In [TD96a] the authors demonstrated that a longer search produced better results. The "tabu relaxation" method presented in [WX01] is also a certain way of prolonging the search process in an attempt to improve the quality of the overall solution.

This proposition seems to be logical: the longer search allows the exploration of a greater part of the search space (using the neighbourhood defined in Section 5.3.2) and, thus, the probability of reaching a good solution is increased. The main challenge is to ensure that the approach does not converge too quickly (which hardly yields a good solution) and that all the allocated time is used to intelligently explore the search space.

The prolonging of the search process is also motivated by the progress of hardware facilities. Let us suppose that the real processing time (T_p) is calculated by the formula: $T_p = T_{mov}*N_{mov}$ (where T_{mov} is the time required for one move (iteration) and N_{mov} is the number of moves). The first factor (T_{mov}) depends on the size of a particular problem as well as the particular environment in which the algorithm is run. Factors that could affect time include computer hardware, the operating system, the compiler and

programming style. A detailed investigation of these factors exceeds the scope of this study. However, the increase in the power of computer hardware always leads to a reduction in T_{mov} . This is due not only to the processor speed, but also to an increase in the amount of RAM (avoids relatively slow dynamic reallocations of memory), to widening the set of Assembler operators for new processors, etc. Thus, while using more powerful computing facilities, the increasing of N_{mov} can be compensated by reducing T_{mov} and therefore the prolongation of the search time can be less tangible. Thus managing the search time promotes the optimal utilisation of computational resources.

In different real situations when computational time and the quality of the solution are dependent upon each other, a user can attempt to find some preferable balance between their values. In some cases the user needs an "average quality" result very quickly, but in other cases the user may want to spend more time to improve the solution. A certain estimation of the importance of computing time can be carried out in the context of attendant processes. For example, let us say that an examination timetable has to be compiled twice a year and preparation of input data and utilising the results often requires several days. In such an environment it is obvious that a computing period of 3 seconds or 3 minutes will make insignificant difference to the time taken by the timetabling process as a whole. The difference becomes important when the computing time reaches 3 hours or 3 days say, as it begins to significantly increase the time taken by the whole process. A computing time of, say, 3 weeks would mostly be regarded as unacceptable.

If computing time becomes a significant part of the time taken for the whole process of developing a timetable it should obviously be taken into account by the user when planning the complete administrative process. Thus it is safe to assume that for the purpose of improving the result a reasonable prolongation of computing time can be acceptable and indeed desirable to a user who has catered for a significant amount of time in the overall administrative plan. Of course, if the algorithm requires several hours, it is fairly painless for the user to run it overnight or over a weekend in order to obtain the result at the beginning of the next working day.

In [Burk03b] the author together with colleagues presented a mechanism that allows a user to define a certain period of time in which the algorithm should run and try to find a high quality solution. This mechanism should ensure that the algorithm searches in an intelligent manner for the specified amount of time. We do not want the algorithm to converge on a solution prematurely. We want the algorithm to use all of its specified time in trying to improve the solution. The overall motivation behind the techniques described in this study is that we want to be able to employ as much (or as little) computing resource as the user may desire to find the level of solution quality that the user is happy to pay for (in terms of computational time).

5.2 Time-Predefined Algorithms

5.2.1 The Time-Predefined Simulated Annealing

In order to run a Simulated Annealing algorithm for a given number of steps the user should precisely determine the required parameters. An approximate

indication of such parameters is not sufficient. Even a small deviation of parameter values can cause a dramatic deterioration in time spent. Also, experimental adjustment of parameter values by manual tests is not practical given the (often) high computational expense of a single run. Although different parameters have some influence on search time, they may not be suitable for its direct regulation or calculation. What is required is an additional time-predefinition mechanism, which guarantees reaching convergence in the given time.

To make Simulated Annealing run for a definite number of moves, the basic geometric cooling algorithm [Ree96] is used, which stops when reaching a certain temperature (which is called T_f). The fact that T_f can be obtained from the initial temperature T_0 by multiplying it by α (where α is a value yet to be determined) during a desired number of steps N_{mov} can be expressed by the following formula.

$$T_f = T_0 \cdot \alpha^{N_{mov}} . \tag{5.1}$$

From this equation the necessary value of α can be expressed as:

$$\alpha = e^{\frac{\ln T_f - \ln T_0}{N_{mov}}} . \tag{5.2}$$

To enable slow cooling it is desirable to have a high value of N_{mov}, which drives the exponent in formula (5.2) close to zero. Taking into account that an exponential function can be expanded in a Maclaurin series:

$$e^x = 1 + \frac{x}{1} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots ,$$

and when $x \to 0$ then $e^x \to 1+x$, formula (5.2) can be transformed into the more simple expression:

$$\alpha = 1 + \frac{\ln(T_f) - \ln(T_0)}{N_{mov}}. \tag{5.3}$$

Using either rule (5.2) or (5.3) a value for the parameter α can be now defined basing on the time interval that we want the Simulated Annealing to run for. But this algorithm still requires the determination of both temperatures. Their values are problem-specific and conventionally they are estimated while using general empiric rules. For T_0 Kirkpatrick suggested [Kir84] that its value should provide some reasonable probability for acceptance of the average-sized uphill move δ_{av}^{+} at the beginning of the search. The value of the initial temperature can be calculated by formula (5.4) which is derived from the main condition of acceptance of Simulated Annealing algorithm:

$$T_0 = \frac{-\delta_{av}^{+}}{\ln(P_0)}. \tag{5.4}$$

In this formula, P_0 denotes the chosen initial probability of acceptance. This rule seems to be is also suitable for the given time-predefined version, even though the definition of the best value of P_0 requires several runs of the algorithm.

The situation with the final temperature is less certain. The common suggestion is to choose the value of T_f to be small enough to guarantee the convergence of the algorithm. However, this does not normally take into account a restraint on the processing time. For the time-predefined variant of Simulated Annealing the determination of the proper value of T_f is more complex. This is discussed further in Section 5.4.

5.2.2 The Great Deluge Algorithm

In 1993, Dueck [Due93] introduced a local search procedure called the *Great Deluge Algorithm*. It was introduced as an alternative to Simulated Annealing. This algorithm like Simulated Annealing may accept worse candidate solutions (than the current one) during its run. The worse solution is accepted if its fitness is less than or equal to some given upper limit B (in the paper by Dueck it was called a "level"). Its value does not depend upon the current solution: at the beginning the initial value of "level" B_0 is equal to the initial cost function and at every iteration it is lowered by the fixed decay rate ΔB whose value is the only input parameter for this technique.

During the search, a particular value for B makes the corresponding part of the search space infeasible and forces the current solution to "escape" into the remaining feasible region. Thus the decreasing of B could be thought of as a control process, which drives the search towards a desirable solution. If the controlling process is relatively slow, the current "level" does not exceed the current solution - it only prohibits the longest backward moves. Thus the neighbourhood appears to be cut down from one side. The current solution has the chance to produce several successful moves (in both directions) inside the remaining neighbourhood and improve its value before the "level" comes too close. While approaching the end of the search, the number of possible forward moves in the neighbourhood (and correspondingly the chance of improving the current solution) decreases. Here a large part of the neighbourhood is cut down and the percentage of successful moves is thus reduced. This situation progresses until the "level" eventually passes the current solution.

To manage this situation, in [Burk03b] it was proposed to integrate the acceptance of all better moves into the basic Great Deluge algorithm (hybridised it with Hill-Climbing). This extension helps the current solution to jump into the unrestricted region (below the "level"). Also, at the end of the search this algorithm has a chance to make a certain improvement of the current solution independently from the current value of B until convergence (when a further improvement becomes impossible). Thus the stopping condition of this algorithm can be the same as for Hill-Climbing: no improvement during a given number of steps. The pseudocode of the final variant of the proposed extended Great Deluge Exam Timetabling algorithm is given in Figure 5.1. In this algorithm the decay rate ΔB actually defines the speed of the "level" reduction.

```
Set the initial solution s
Calculate initial cost function f(s)
Initial level B=f(s)
Specify input parameter ΔB = ?
While further improvement is impossible
   Define neighbourhood N(s)
   Randomly select the candidate solution s* ∈ N(s)
   Calculate f(s*)
   If f(s*) ≤ f(s)
   Then accept s*
   Else if f(s*) ≤ B
         Then accept s*
   Lower the level B = B –ΔB
```

Figure 5.1: The extended Great Deluge algorithm

Further experiments with this technique (discussed in this thesis in Section 5.4) revealed two main properties of the Great Deluge algorithm:

- *The profile of the process* is explicit. The search rigidly follows the decreasing of the "level". Fluctuations occur only at the beginning but later all intermediate solutions lie close to the line $f_C = B_0 - \Delta B * N_{mov}$ (N_{mov} is the desired number of moves, fc is the cost function).

- *The point of convergence* is quite recognisable. At this point the improvement of a current solution abruptly stops and the consequent process runs idle. The sharpness of this transition simplifies its detection and helps to terminate the search procedure timely.

These properties of the algorithm provide an opportunity to fit the search procedure into a certain time period. It should be taken into consideration that during the search the "level" reaches the zero value in the number of moves equal to $B_0 / \Delta B$ and the cost function of a current solution normally does not exceed the current value of "level". Thus, for problems where cost function is always positive (such as exam timetabling problem) this algorithm *guarantees* producing a result in a time, which does not exceed the value of T_P calculated by the following expression:

$$T_P = T_{mov} \cdot N_{mov} = T_{mov} \cdot \frac{B_0}{\Delta B}, \qquad \text{where } T_{mov} \text{ is the time of one move.} \qquad (5.5)$$

However, in most problems, the algorithm converges before T_P expires. The point of convergence is uncertain and problem-dependent. Therefore, if some information about the range of possible results is available, it could be used for

reducing the number of idle steps. If the user estimates the cost function of a future result as $f(s')$ (as the goal value which he/she intends to reach by prolonging the computing time), then ΔB is calculated by formula (5.6).

$$\Delta B = \frac{B_0 - f(s')}{N_{mov}}. \tag{5.6}$$

Such approximation can be done in different ways. For example, if the final cost function is completely unknown, a user can apply some quick technique (e.g. Hill-Climbing) for the same problem. Its average-quality result will give an idea of the range of possible solutions. Thus, in contrast to time-predefined Simulated Annealing, the Great Deluge algorithm allows only an approximate predefinition of the search time. However, practice shows that the possible deviation between the expected solution and the real one is insignificant (relative to the complete search interval). Therefore, the inaccuracy in the predefinition of the operational time does not usually exceed a few percent.

5.3 Experiments with Time-Predefined Techniques

Both of the described techniques were implemented with Microsoft Visual C++ 6.0 and experiments were undertaken using a PC with an Athlon 750 MHz processor and Windows 98. The overall aims of these experiments were:

- To investigate the properties of the time-predefined techniques by generating the “cost progress” diagrams (see Section 5.4) for the search processes. These diagrams track the evolution of the cost function during the search.

- To explore the manner in which the prolongation of the search can increase the quality of solution by plotting the time-cost diagrams. This can be achieved by several runs of the software with different predefined search times (number of steps).

- To evaluate the quality of the results produced by the time-predefined search in an acceptable time by comparison of its range with the outcomes of other techniques applied to the same datasets and published in the literature.

5.3.1 An Initialisation Phase

This work is not devoted to the investigation of the initialisation phase of the presented techniques. It could be the topic of a separate study. However, the initialisation strategy could have a crucial influence on the performance of the algorithms (as it can for other search methods), especially when the search space is disconnected, which is typical for examination timetabling problems. Therefore, the initial solution should be as good as possible in as little time as possible and appeared independent of the applied heuristic. In further experiments, for every problem 20 solutions were generated and the one with the minimum cost function was chosen as the initial one.

These solutions were produced by Brelaz's "saturation degree" graph colouring algorithm [Bre79], which is probably the most powerful sequential heuristic. It chooses the vertex (exam) with the least number of available colours (timeslots) and assigns the timeslot for it. In order to have different solutions with different runs, the assigned timeslot was chosen randomly among the available ones. This stochastic feature allows us to capture different

areas of the search space. The given algorithm produces feasible solutions in a few seconds, so we consider the initialisation time negligible and do not include it in the estimated search time.

5.3.2 Neighbourhood Structure

Besides the initialisation strategies, this study also does not investigate the possible effect of the utilisation of different neighbourhood structures. In order to enable comparison of different approaches the same (most common) variant of neighbourhood is used in all of them. Here all candidate solutions can be produced from the current one by a simple replacement of one exam into a different timeslot. An advantage of such a move is a very fast evaluation procedure, i.e. at each iteration, the used algorithms calculate only the difference in cost function caused by this move. This allows us to increase a number of moves produced in the same processing time and hence (in terms of this research) to improve the performance of algorithm. Otherwise, the utilisation of more advanced neighbourhood (e.g. Kempe chains) can extend the search space and produce better results in the same number of moves. However, these moves can be more computationally expensive, which can increase the search time of the algorithm. Obviously, a performance of time-predefined algorithms with different neighbourhoods requires additional detailed investigation.

5.4 Investigating the Properties of the Algorithms

In the first phase of the experiments the influence of algorithmic parameters was investigated for both of the described algorithms. The algorithms were run

on a series of benchmark datasets while employing the cost function presented by formula (4.1). In the experiments N_{mov} was determined as 3,000,000. On the graphs presented in Figures 5.2-5.3 the *y* axis represents the current cost and the *x* axis represents the number of current move together with the processing time in seconds. When the figure is labelled with "TPSA" it is describing the time-predefined Simulated Annealing approach and when it is labelled "GD" it describes the Great Deluge algorithm.

The progress diagrams for two sample runs (with different cooling schedules) of the time-predefined Simulated Annealing on the CAR-F-92 problem are shown in Figure 5.2. To define the values of initial and final temperatures the following preliminary manipulations have been carried out.

The initial temperature was the same for both schedules and was calculated by formula (5.4). The average uphill move δ_{av}^{+} was defined while recording 1000 uphill moves at the beginning of the search and calculating their average value. The initial probability of acceptance of uphill moves P_0 was defined experimentally by several runs of the algorithm. The experiments showed that for the given problems a highly suitable value of P_0 is around 0.9. In this way the initial temperatures were calculated for all given problems. In particular, for the CAR-F-92 problem, T_0 was set up to be 0.06.

In order to calculate the final temperature, the Simulated Annealing algorithm was run until no improvement was indicated during 10,000 moves (0.3% of the number of moves of the whole procedure). The algorithm converged at $T_f \approx 1.2*10^{-5}$. Based on this value the cooling rate of the first

sample run was calculated by formula (5.3). The evolution of the cost function is shown as curve *A* in Figure 5.2.

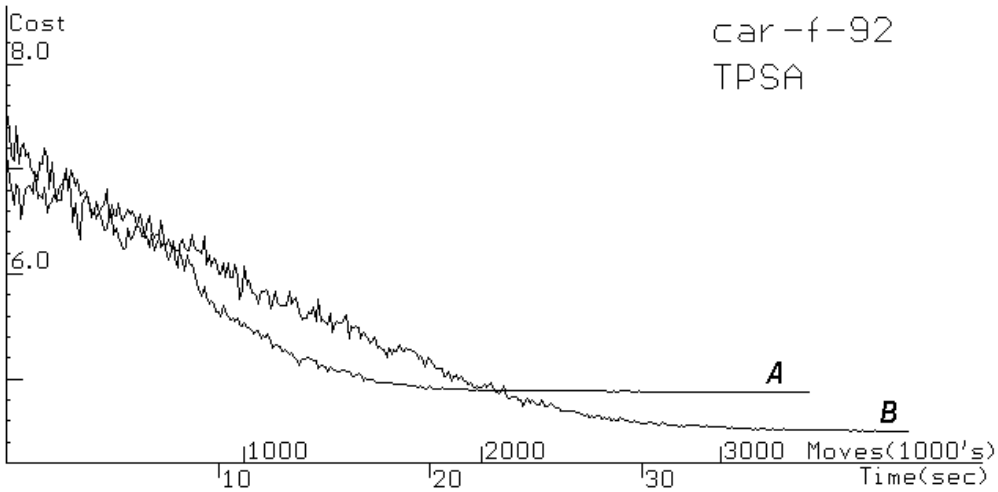

Figure 5.2: Cost progress diagrams for time-predefined Simulated Annealing on CAR-F-92 dataset

As it was proposed, the process *A* converges in approximately 3,000,000 moves. However, this diagram has a long "tail" - it spends almost half of its processing time very close to the point of convergence. To investigate this behaviour further and to look at its effect on the time-quality trade-off, a second sample run of the algorithm was carried out (the curve *B* in Figure 5.2). The initial temperature and the number of moves were the same as for the process *A*, but the final temperature was increased 35 times: $T_f = 4*10^{-4}$. This value does not provide the convergence in the given number of moves: the current solution continues to improve after passing it. Nevertheless, at the point of 3,000,000 moves the process *B* has reached a substantially better solution than the process *A*.

However, the attempts to further increase the final temperature led to a worsening of the quality of solutions at this point. It is clear that the optimal value of T_f is problem-dependent and quite difficult to determine. For the experiments presented in this study it was determined empirically by making a high number of *short-time* launches. Of course, the problem of choosing the right parameters usually presents a difficulty when employing Simulated Annealing.

In terms of setting the initial parameters, the uncertainty of the time-predefined Simulated Annealing approach is also characterised by the fact that process B continues to improve after the allocated time is expired. In circumstances when a time limit is not too strict, the user can decide not to terminate the algorithm in order to achieve a better quality solution. However, the extra time for the improvement can be quite long which can mean that even if the solution is improved, certain users may consider it too high a price to pay.

The next experiment investigated the behaviour of the Great Deluge algorithm with the same given number of moves. The decay rate ΔB was defined by formula (5.6), which utilises the estimated value of a future result $f(s')$. Its value was obtained by applying a Hill-Climbing algorithm before the first experiment. This process lasted only a few seconds and so the time is considered to be negligible. For the CAR-F-92 problem, the Hill-Climbing produced a solution with penalty value of 5.5. Of course, it is expected that the Great Deluge algorithm should achieve higher quality solutions. Therefore for

assigning *f*(*s*') in formula (5.6) the correspondent fractional decrements of this value were made. The progress of two sample runs is shown in Figure 5.3. In the first run (the curve *A*) the final "level" was determined to be 4.4 (5.5*0.8) and in the second one (the curve *B*) it was taken to be 4.95 (5.5*0.9).

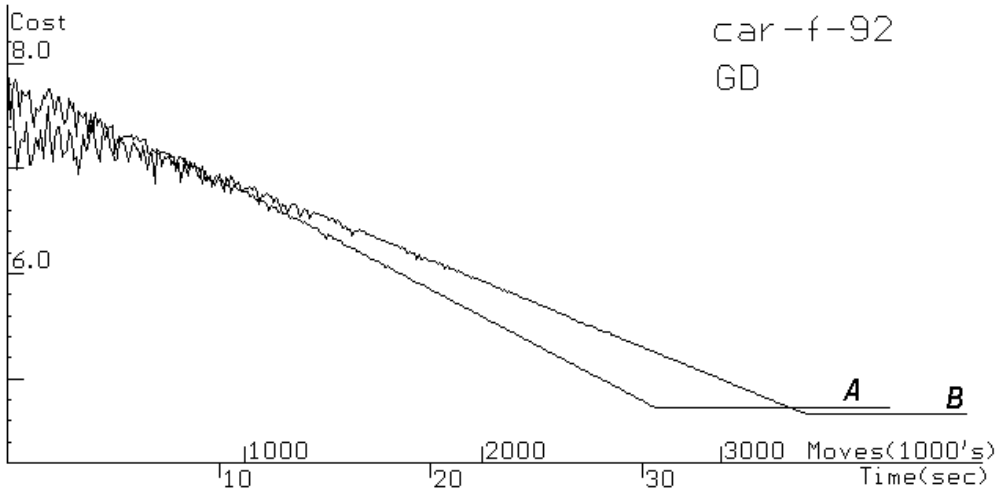

Figure 5.3: Cost progress diagrams for the Great Deluge algorithm on CAR-F-92 dataset

These diagrams display how strictly the search follows the linear movement of the "level". The fluctuations are seen only in the first half, but later all solutions become quite close to the "level" and make very small oscillations. The moment of convergence is quite recognisable (the process *A* converged at 2,700,000 moves and process *B* at 3,400,000 moves). The algorithm should be definitely terminated at the point of convergence. The presented way of assessing the final "level" causes an inaccuracy in the time predefinition of around ±13%. However, this is the only uncertainty presented by setting initial parameters for the Great Deluge algorithm.

The same experiments were also conducted on other datasets from Carter's collection. The progress diagrams show quite similar properties to the presented ones, therefore they are not presented here but can be found on the following web page: http://www.cs.nott.ac.uk/~yxb/tpls. The over-riding feature is that the behaviour of time-predefined Simulated Annealing is much more uncertain and parameter-dependent than for the Great Deluge Algorithm. It requires more preliminary work for the definition of problem-specific parameters and does not guarantee the best quality of solution in given time.

5.5 Analysis of the Relationship between Time and Cost

A second set of experiments was performed on datasets from Carter's collection, and again the cost function presented in (4.1) was employed. For all datasets, both techniques were run a number of times with different values of N_{mov}. Simulated Annealing operated with the same temperatures as in the previous experiments. In the Great Deluge algorithm f(s') of each run was assigned to be equal to the final solution of the previous launch (for the first one the Hill-Climbing method was applied). The final results of the runs were collected into 26 tables (13 datasets x 2 methods), where each table comprises from 40 to 60 results (the number of results is varied for each particular dataset because of difference in time intervals in which these highly time-expensive experiments were conducted). All these tables (in the form of diagrams) are presented and discussed below. In all following diagrams the y axis represents the final cost and the x axis represents the number of moves and computing time in seconds taken by the search session. Every point on a diagram corresponds to the final cost function and processing time of a separate

solution. The results are discussed in the context of the number of enrolments for each problem (given in the right-hand column in Table 4.1), which can be considered as a certain measure of the problem's size.

Firstly, the four "largest" problems: PUR-S-93, UTA-S-92, CAR-S-91, CAR-F-92 are considered. The resulting diagrams are shown in Figures 5.4-5.7. For every dataset the given number of timeslots, enrolments and average search speed is indicated.

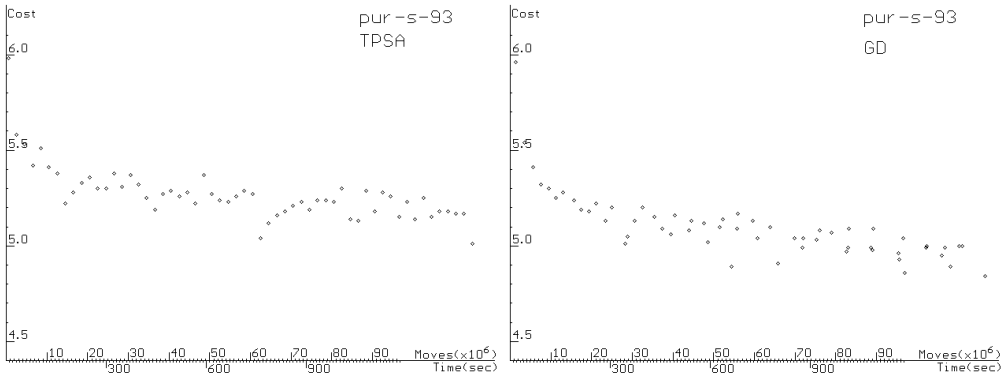

Figure 5.4: Time-cost diagrams for PUR-S-93 problem

(120690 enrolments, 43 timeslots, search speed 82000 moves/sec)

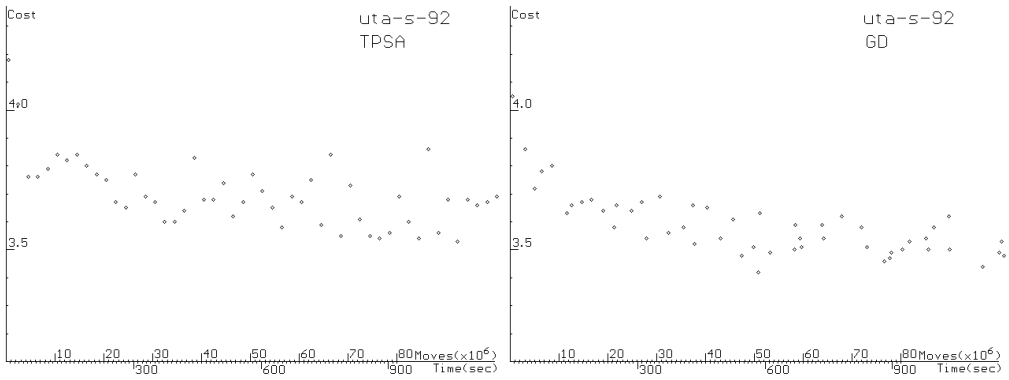

Figure 5.5: Time-cost diagrams for UTA-S-92 problem

(58981 enrolments, 35 timeslots, search speed 87000 moves/sec)

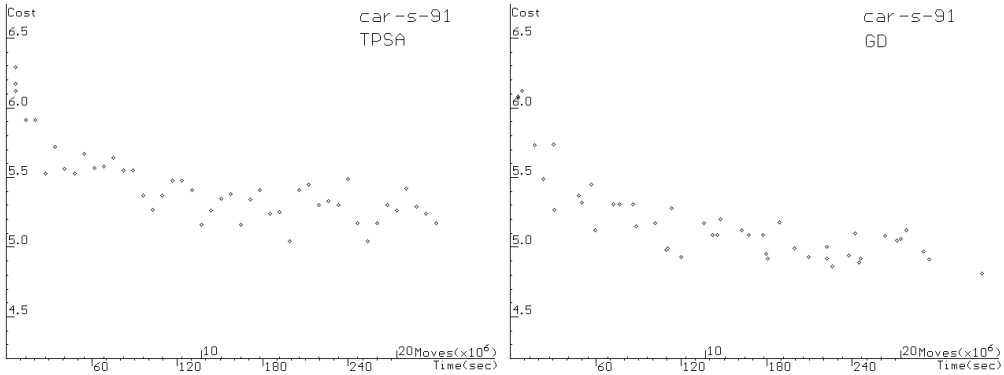

Figure 5.6: Time-cost diagrams for CAR-S-91 problem

(56877 enrolments, 35 timeslots, search speed 73000 moves/sec)

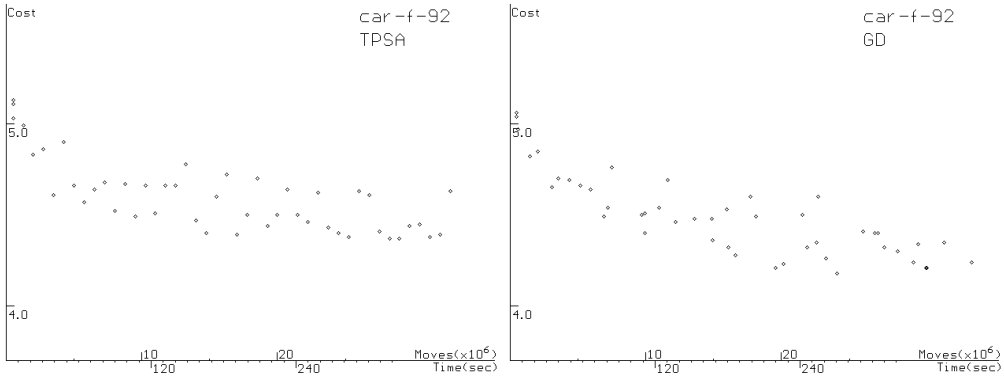

Figure 5.7: Time-cost diagrams for CAR-F-92 problem

(55552 enrolments, 32 timeslots, search speed 89000 moves/sec)

The distribution of points in these diagrams demonstrates the trade-off between the search time and overall solution quality. Even though the results are relatively scattered, there is a clear general tendency to improve the cost function as time increases. Moreover this tendency, for both techniques appears (in almost the same way) for all presented problems. Indeed all four diagrams presented here are surprisingly similar (although, the scale of both axes is individual for each problem).

The analysis of the diagrams shows that the slope of the curves is relatively steep on the left hand side of the diagrams (i.e. a small increase in

time leads to a high improvement in quality). As the search time gets longer the improvement of solutions becomes slower. Thus, the time-cost diagram of a time-predefined algorithm can be approximated as a monotonically lowered function, which asymptotically approaches some limit. This limit is obviously better than the local optimum (produced by Hill-Climbing). Possibly it is the minimum for the explored area (which can be separated in the case of a disconnected search space).

The particular shape of the time-cost curve depends on the different problem's characteristics. To investigate the influence of the problem's size the diagrams for the "smallest" problems from Carter's collection are produced: EAR-F-83, YOR-F-83, STA-F-83 (Figures 5.8-5.10).

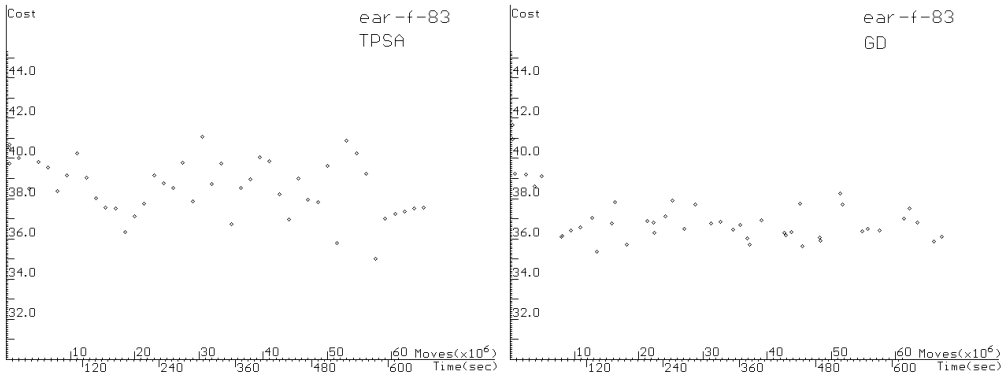

Figure 5.8: Time-cost diagrams for EAR-F-83 problem

(8108 enrolments, 24 timeslots, search speed 99000 moves/sec)

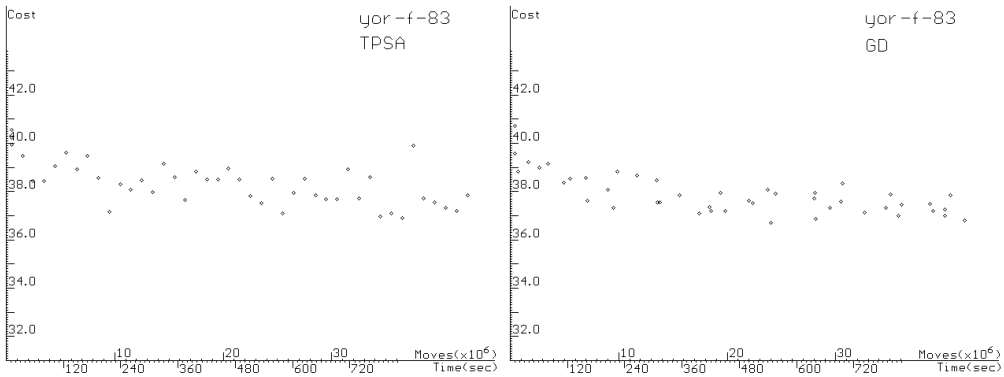

Figure 5.9: Time-cost diagrams for YOR-F-83 problem

(6029 enrolments, 21 timeslots, search speed 44000 moves/sec)

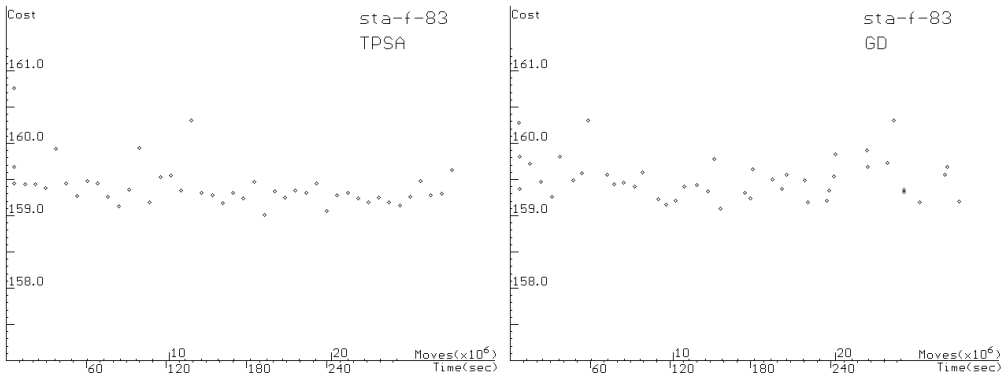

Figure 5.10: Time-cost diagrams for STA-F-83 problem

(5751 enrolments, 13 timeslots, search speed 82000 moves/sec)

These diagrams appear to be different from the ones obtained for the “largest” problems. They show a sharp rise in the final quality with relatively short computational times but further prolongation of the search has very low influence on the result (i.e. the rest of the diagram remains almost flat).

Thus the trade-off between the search time and the quality of results holds mostly for the “large” exam timetabling problems while it plays less of a role for the smaller size problems. This is in accordance with what one would expect: the larger problems need more work! With the “middle-sized” problems the presented techniques usually behave in an intermediate way

(which is also in accordance with what we would expect). Examples of such datasets (RYE-F-92, KFU-S-93, TRE-S-92, LSE-F-91, HEC-S-92, UTE-S-92) are presented on Figures 5.11-5.16.

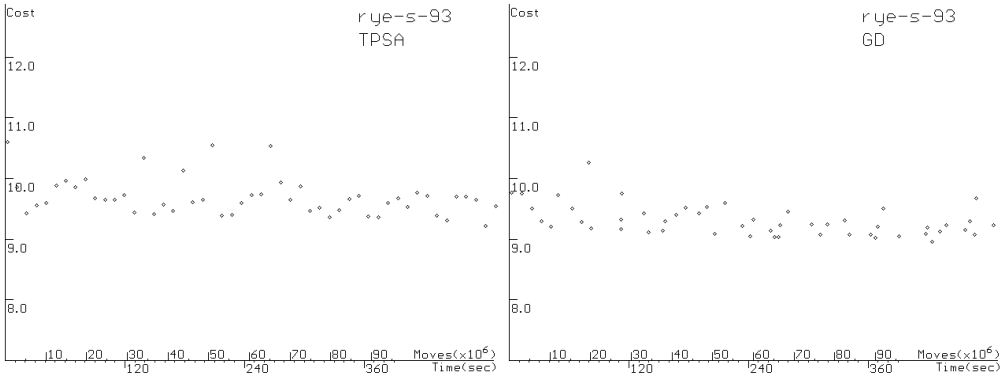

Figure 5.11: Time-cost diagrams for RYE-F-91 problem

(45052 enrolments, 23 timeslots, search speed 245000 moves/sec)

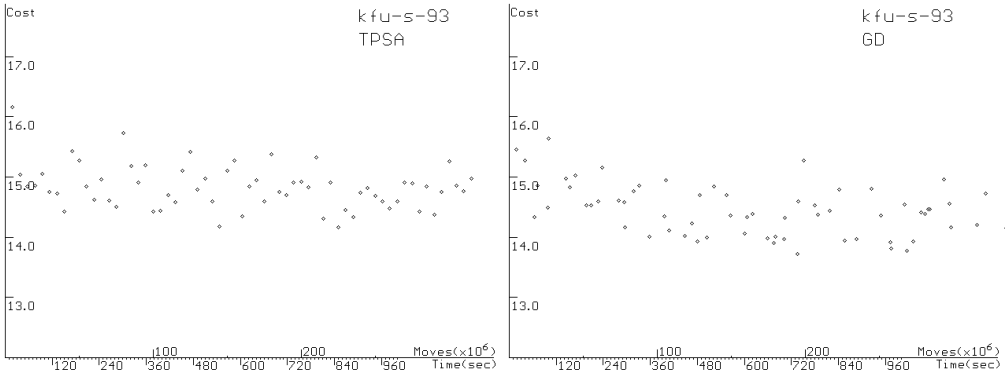

Figure 5.12: Time-cost diagrams for KFU-S-93 problem

(25118 enrolments, 20 timeslots, search speed 265000 moves/sec)

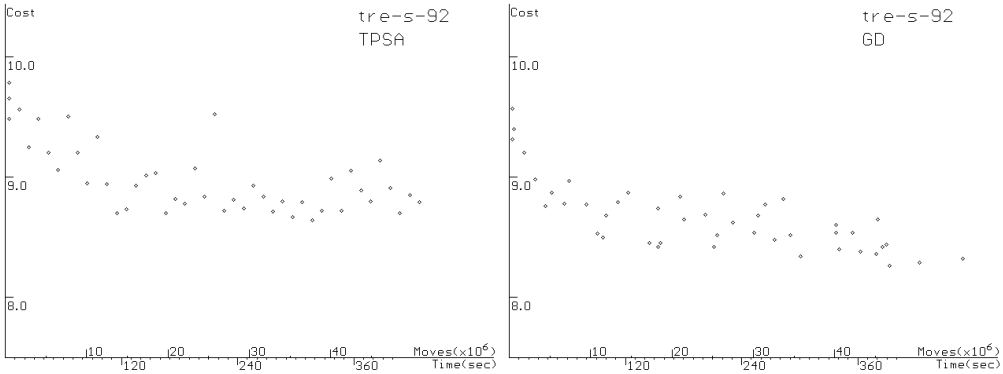

Figure 5.13: Time-cost diagrams for TRE-S-92 problem

(14901 enrolments, 23 timeslots, search speed 119000 moves/sec)

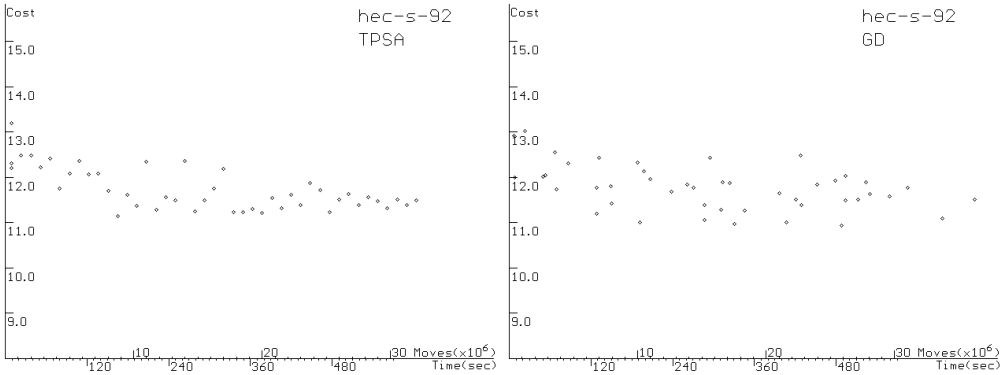

Figure 5.14: Time-cost diagrams for HEC-S-92 problem

(10632 enrolments, 18 timeslots, search speed 53000 moves/sec)

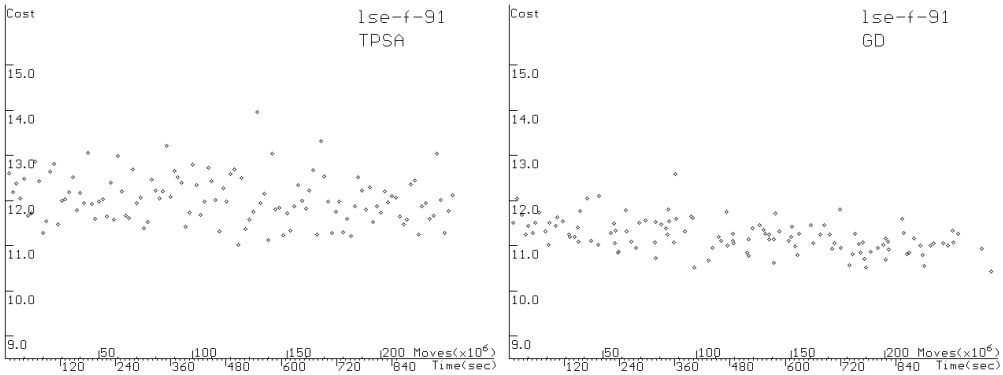

Figure 5.15: Time-cost diagrams for LSE-F-91 problem

(10919 enrolments, 18 timeslots, search speed 245000 moves/sec)

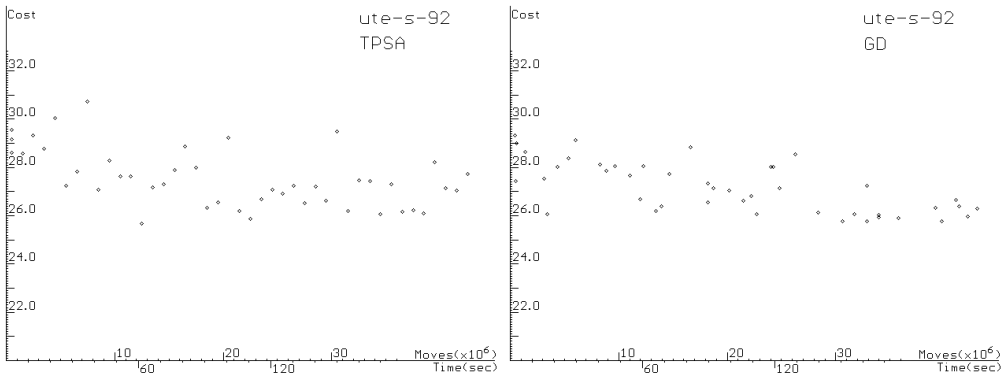

Figure 5.16: Time-cost diagrams for UTE-S-92 problem

(11796 enrolments, 10 timeslots, search speed 203000 moves/sec)

5.6 A Comparison of Time-Predefined Simulated Annealing and the Great Deluge Algorithm with the Current State-of-the-Art.

5.6.1 A Comparison on Carter's Benchmarks

To fully evaluate the presented techniques, they are compared against recently published results on the same benchmark problems. These include the results produced by Carter, Laporte and Lee [CLL96] (by employing several sequencing heuristics with backtracking) and some recent results that were produced by Di Gaspero and Schaerf [DGS01] (using Tabu Search with a variable tabu list). For every dataset Carter published four results, obtained with different heuristics. Di Gaspero and Schaerf presented their best and average values. For comparison purposes there are presented the best and worst of Carter's et al. results, the best and average results of Di Gaspero and Schaerf and the best, worst and average results for both of presented algorithms. Note that Carter's et al. results are obtained by using different algorithms whereas Di Gaspero and Schaerf's (and methods proposed in this thesis) employ the same approach.

The average values of produced results can be estimated while selecting a proper range of samples. The solutions of “short-time” searches (placed in the left hand sides of the diagrams in the previous section) are too poor and are shown mainly for illustration purposes. Therefore, as the search time (at the end of each diagram) is quite acceptable (i.e. one could consider its right hand side as within the recommended range). Thus the average values for Time-Predefined Simulated Annealing and Great Deluge are calculated as an average cost of the five rightmost points on corresponding diagrams.

All of the results are summarised in Table 5.1. Dashes show that the corresponding data was not published. The table also includes a computational time (in seconds) for each best cost. However, this is done only to give a notion about a range of useful time periods because as the published results were produced with different hardware and algorithmic solutions, the search times of the different techniques cannot be sensibly compared.

able 5.1: Published and produced results for proximity cost

Data set	**Time slots**	**Carter et al.**				**Di Gaspero and Schaerf**				**TPSA**				**GD**			
		Cost			Time for best (sec)	Cost			Time for best (sec)	Cost			Time for best (sec)	Cost			Time for best (sec)
		best	worst	aver.		best	worst	aver.		best	worst	aver.		best	worst	aver.	
CAR-F-92	32	6.2	7.6	-	47.0	5.2	-	5.6	860.6	4.4	5.1	4.5	295	**4.2**	5.1	4.3	274
CAR-S-91	35	7.1	7.9	-	20.7	6.2	-	6.5	30.2	5.0	6.3	5.3	256	**4.8**	6.1	5.0	328
EAR-F-83	24	36.4	46.5	-	24.7	45.7	-	46.7	4.6	**35.0**	41.1	37.3	589	35.4	41.7	36.7	134
HEC-S-92	18	10.8	15.9	-	7.4	12.4	-	12.6	3.7	**10.6**	13.2	11.4	1730	10.8	12.4	11.5	278
KFU-S-93	20	14.0	20.8	-	120.2	18.0	-	19.5	12.3	14.2	16.2	14.9	586	**13.7**	15.6	14.4	729
LSE-F-91	18	10.5	13.1	-	48.0	15.5	-	15.9	20.3	11.0	14.0	12.0	541	**10.4**	12.6	11.0	1030
PUR-S-93	43	**3.9**	5.0	-	21729	-	-	-	-	5.0	6.0	5.1	1395	4.8	6.0	4.9	1412
RYE-S-93	23	**7.3**	10.0	-	507.2	-	-	-	-	9.2	10.6	9.6	959	8.9	10.3	9.3	752
STA-F-83	13	161.5	165.7	-	5.7	160.8	-	166.8	3.9	**159.0**	160.8	159.4	190	159.1	160.3	159.4	157
TRE-S-92	23	9.6	11.0	-	107.4	10.0	-	10.5	16.2	8.6	9.8	8.9	326	**8.3**	9.6	8.4	392
UTA-S-92	35	3.5	4.5	-	664.3	4.2	-	4.5	50.7	3.5	4.2	3.6	906	**3.4**	4.1	3.5	585
UTE-S-92	10	25.8	38.3	-	9.1	29.0	-	31.3	42.4	**25.7**	30.7	27.2	65	**25.7**	29.3	26.2	236
YOR-F-83	21	41.7	49.9	-	271.4	41.0	-	42.1	25.2	36.9	40.6	37.5	820	**36.7**	40.7	37.2	546

Even though detailed statistical tests are not possible because of the incompleteness of the information about average published results, the figures in the table demonstrate the power of the two time-predefined approaches. Their performance is better (by all quality indices) than Di Gaspero and Schaerf's Tabu Search on all the benchmark problems. For eleven of the thirteen problems the two time-predefined algorithms achieved a better cost function value than any of the currently published ones. For five of these problems the best published results are worse than even the average outcome of the presented methods (in a sixth problem they are equal). It is interesting to note that for two of the problems (the largest one and a medium sized one) Carter et al. has an approach, which produces better quality solutions than both of presented techniques although, as mentioned above, Carter's et al. results do not represent one single approach (they employ different heuristics).

The computing time for the best produced results shows that the given solutions were obtained in quite an acceptable time (10-30 min). Even if most of the published results were produced quicker, the prolongation of time in presented algorithms is well justified by the quality of the obtained solutions.

When comparing the outcomes of two presented techniques with each other, it is evident that the Great Deluge Algorithm performs slightly better than time-predefined Simulated Annealing. In addition, for most of the problems the time-cost diagrams for the Great Deluge Algorithm are less scattered. In Table 5.1 it can be seen that the Great Deluge Algorithm outperforms the time-predefined Simulated Annealing in 9 cases by the best

values and in 11 cases by the average ones. In contrast, time-predefined Simulated Annealing is correspondingly better in terms of best values in 3 cases and in terms of average values in just one case. Moreover, it is only with small-size problems. Note that the two approaches have an equal best value for UTE-S-92 and an equal average value for STA-F-83. Perhaps the time-predefined Simulated Annealing approach does not perform as well because of imperfect values of the initial and final temperatures. However, such a situation is characteristic of a Simulated Annealing approach and the Great Deluge approach is free of this uncertainty. This alone leads us towards the conclusion that the Great Deluge algorithm is superior to the time-predefined Simulated Annealing approach. When this particular advantage is taken together with its superior performance (overall) on the benchmark problems then the evidence of its superiority over time-predefined Simulated Annealing becomes overwhelming (at least for the given benchmark examination timetabling problems).

5.6.2 Experiments with More Advanced Problems

A further series of experiments was performed to evaluate the performance of the presented approach with more complex problems (in terms of more constraints). These benchmarks were established in [BN99]. Three datasets were chosen where, in addition to the clash-free requirement (formula (4.2)), the room capacities constraint (formula (4.3)) is considered as a hard constraint. The distribution of timeslots among days (formula (4.5)) is also taken into account. Two soft constraints are considered: the number of students who have two exams in adjacent periods and in overnight periods. The

correspondent criteria values (f_2 and f_5) are calculated by formulae (4.6) and (4.9). The cost function f_c is calculated as their weighted sum (with weights 3 and 1 to represent their relative importance). It can be seen in formula (5.7) and should be minimized.

$$f_c = 3 \cdot f_2 + f_5 . \tag{5.7}$$

The characteristics of the problems are presented in Table 5.2.

Table 5.2: Additional characteristics of problems

Data	Seats	Periods	Enrolments
KFU-S-93	1955	21	25,118
Nott-94	1550	23	33,997
CAR-F-92	2000	36	55,552

These experiments were conducted in the same way as that in Section 5.5, including initialisation, neighbourhood and the methods of assigning initial parameters. Also, the same type of time-cost diagrams was used for interpretation of produced results. These diagrams are presented in Figures 5.17-5.19.

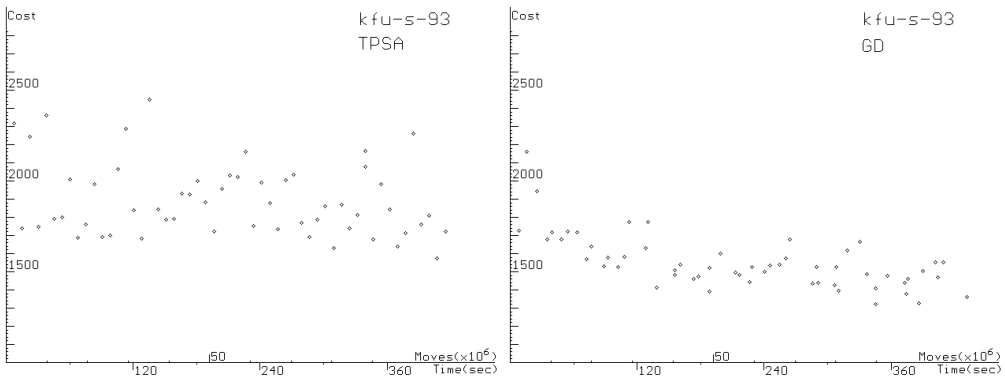

Figure 5.17: Time-cost diagrams for KFU-S-93 problem

(search speed 260000 moves/sec)

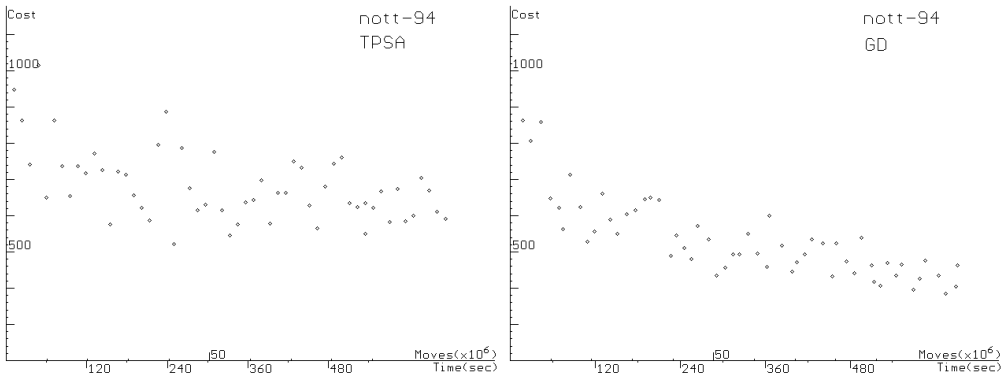

Figure 5.18: Time-cost diagrams for Nott-94 problem

(search speed 174000 moves/sec)

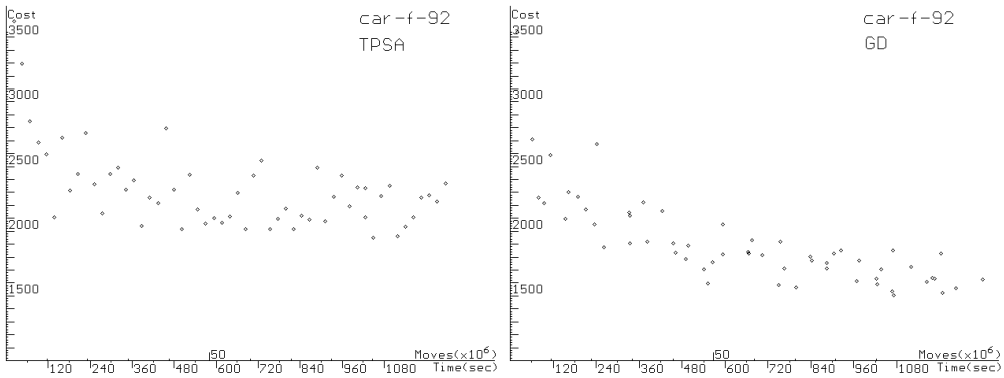

Figure 5.19: Time-cost diagrams for CAR-F-92 problem

(search speed 87000 moves/sec)

The results obtained by both described algorithms are compared with the published results obtained by the Multi-Stage Memetic Algorithm of [BN99]. There is a series of comparisons in [BN99] which establish that the Multi-Stage Memetic Algorithm had the best published results on these problems up until the development of the algorithms presented in this dissertation. In addition, to give a more complete evaluation, the Tabu Search results obtained by Di Gaspero and Schaerf [DGS01] on these 3 problems are also presented. In [BN99] and [DGS01] there is a fourth benchmark problem (PUR-S-93) that is evaluated and discussed. However, the number of timeslots

was taken to be 30, which the author strongly believes is insufficient for producing a feasible timetable. Both of the approaches in those papers publish results which are infeasible i.e. they generate high values of the cost function because a very high additional value is included when a hard constraint is broken. The problem was intentionally excluded from this comparison because two presented approaches consider only feasible solutions. If one cannot find a solution which satisfies the hard constraints then they cannot cope. However the definition of hard constraints is, of course, that they must be satisfied in all cases. The final figures are presented in Table 5.3.

able 5.3: Published and produced results for weighted sum of adjacent and overnight conflicts

Data set	**Time slots**	**MSMA**				**Di Gaspero and Schaerf**				**TPSA**				**GD**			
		Cost			Time for best (sec)	Cost			Time for best (sec)	Cost			Time for best (sec)	Cost			Time for best (sec)
		best	worst	aver.		best	worst	aver.		best	worst	aver.		best	worst	aver.	
KFU-S-93	21	1388	2662	1626	105	1733	-	1845	-	1573	2448	1825	407	**1321**	2161	1470	345
Nott-94	23	490	1022	552	467	751	-	810	-	522	1015	635	249	**384**	862	433	612
CAR-F-92	36	1665	4806	1765	186	3048	-	3377	-	1951	3622	2250	1220	**1506**	3536	1610	1268

This comparison once more justifies the power of the two time-predefined approaches. Indeed, it is clear here that the Great Deluge algorithm is far superior to time-predefined Simulated Annealing. In fact it is far superior (in terms of the quality of its results) to all previously published approaches on these problems although it is more time-consuming. However, Burke and Newall [BN99] indicated that increasing the computational time in the Multi-Stage Memetic Algorithm does not yield any improvement of the quality of final solutions. Moreover, the presented time-cost diagrams show the potential for further improvement of results (i.e. there is a clear slope in the right hand side of the diagrams). The experiments, which went beyond the right hand side of the diagrams were not continued for all 3 problems because of the extreme computational expense. However, the Great Deluge algorithm was run on the Nott-94 problem for extremely long periods to give some indication of what it could produce with enough computational time (Table 5.4).

Table 5.4: The best results, obtained for Nott-94 problem by Great Deluge search

Time (hours)	**Cost**	**Relative improvement of the cost (comparing to the best result of Multi-Stage Memetic Algorithm)**
2.5	256	1.91
67	225	2.18

As it was expected from examining the diagrams, an extremely long search can produce extremely good results. Note that although the solution with a cost of 225 took over 67 hours, it was run over a weekend (when the computer would otherwise have been idle) and produced a result which is more than twice as good as the previous best published solution. It is clear that while

this ability may not be appreciated by all users, there is a definite potential for incorporating this approach into real world examination timetabling systems (and indeed other scheduling systems).

5.7 On the Comparison of the Performance of the Time-Predefined Algorithms with other Approaches

The comparison of the presented results with previously published ones, given in Tables 5.1, 5.3 and 5.4 is done in a conventional way (as it is usually conducted with non time-predefined algorithms). It only expresses a rough idea about the range of the presented values of the cost function. However, a formal comparison (which takes into account both cost and time) of the proposed approach with non time-predefined techniques requires a more detailed study. In this section the main aspects of this investigation are sketched.

When embedding time predefinition into search algorithms it is possible to express their performance as a function: one can get any solution (within some range) as long as one pays the necessary cost in time. From this point of view the processing time can be thought of as an additional objective, which should be minimised together with the cost function. Thus, the problem becomes bi-objective where the time-cost goal comprises the typical trade-off surface. It can be investigated using several metrics as suggested in [KC02].

The formal comparison of the performance of a time-predefined and a conventional technique (which can be thought as having a single time-cost point) is quite uncertain and can be carried only within some limits. The issues are illustrated in Figure 5.20. The dotted line represents the time-cost trade-off

curve of a time-predefined algorithm “A”. A solution, produced by a traditional algorithm “B” (point *B*) has worse time and cost than “A”. Thus a complete outperformance of the algorithm “A” over “B” is evident. However, while evaluating the point *C* (the solution, produced by another non time-predefined algorithm “C”) one cannot make any general conclusion about the preference of either algorithm. Algorithm “C” partially outperforms “A”. The preference is clear between points *x* and *y* but unclear outside these points.

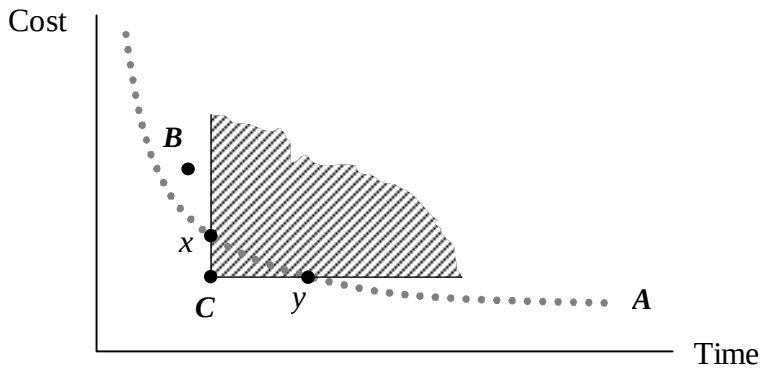

Figure 5.20: The example of uncertainty in comparison of the time-cost diagram with the single solution.

5.7.1 A Comparison with a Threshold Acceptance Method

Further comparisons of the performance of the Great Deluge algorithm with other methods are presented in [Burk03c]. This section discusses the comparison of performance of the Great Deluge algorithm with a Threshold Acceptance method. It is a deterministic variant of the Simulated Annealing approach, introduced by Dueck and Scheuer in [DS90] which accepts worse candidate solution in the case when the difference in the cost function between the current and the candidate solution does not exceed some threshold. The

threshold should be decreased during the search. However, there are no explicit recommendations concerning its initial value and the rate of decrease.

As the performance of the Great Deluge algorithm can be estimated only in respect of a time-cost diagram, it is worth making a time-cost diagram available also for the second competitor. However, the Threshold Acceptance method does not allow the direct definition of the processing time. Therefore in the following experiments the author tried to distribute its outcome evenly in the given time period by varying its parameters.

The objective function in the following experiments was defined by formula (4.1). The time-cost diagrams, obtained for CAR-F-92 dataset are shown in Figure 5.21. The Threshold Acceptance algorithm was launched several times with variations of the initial threshold in the interval: $5*10^{-5}$ - 2.5 (the results confirmed that the best value is located inside this interval). In order to get approximately the same launch time in both algorithms, rate of threshold decreasing was varied from 10^{-12} to $2*10^{-5}$.

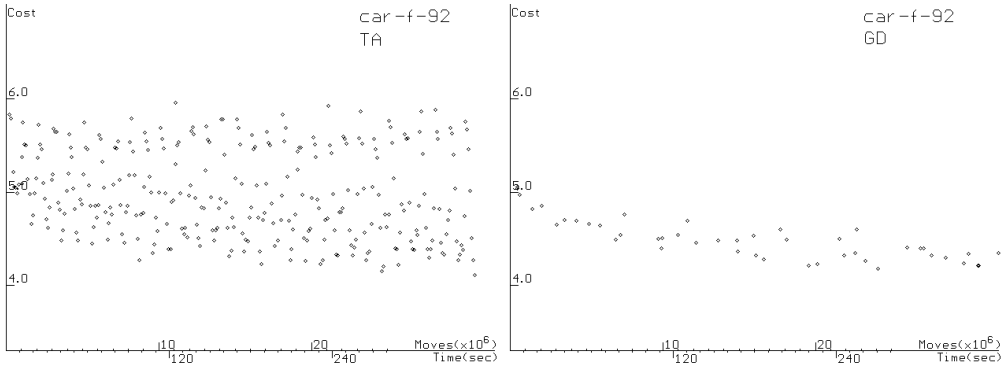

Figure 5.21: Comparison of Threshold Acceptance and Great Deluge algorithms for CAR-F-92 problem

The Threshold Acceptance diagram shows a substantially higher "scatter" of results than the Great Deluge one. A greater number of poor quality solutions are generated though the use of inappropriate parameter values. Although both methods have approximately the same values of the cost function for the best results (in the given launch time), Threshold Acceptance method *can reach* it only with properly defined parameters, while Great Deluge *does* it always.

Another advantage of the Great Deluge algorithm is the effectiveness of the search. The deriving of the best parameters requires several runs of the Threshold Acceptance method. This time should be taken into account when the total time of the solution process is calculated. Therefore its total processing time (from input to output) is several times longer than the processing time of a single run. It should be pointed out that the Great Deluge algorithm is almost free of such parameters and spends almost all of its time only in the search procedure. Hence, with respect to the total processing time, the Great Deluge is far more advantageous.

5.7.2 A Comparison with Hill-Climbing

Another comparison was undertaken with the Hill-Climbing method. In this experiment the time-cost diagrams were produced with a very short search time. The search time of Hill-Climbing depends on the stopping condition. The algorithm used as the stopping condition the given number of idle steps and it was varied in the range of 1-10000. The results for YOR-F-83 problem are presented in Figure 5.22.

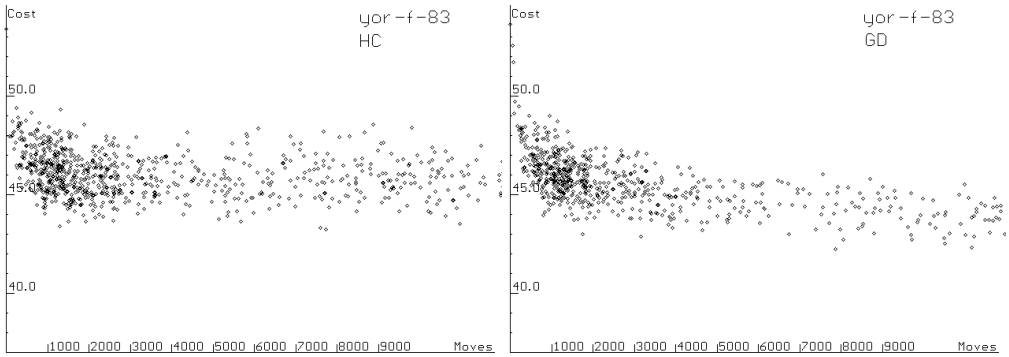

Figure 5.22: Comparison of Hill-Climbing and Great Deluge algorithms for YOR-F-83 problem

Visual inspection of these diagrams indicates that they have a broadly similar distribution of points at the beginning. This means that if the stopping condition of Hill-Climbing is chosen correctly then both algorithms have the same performance. However the behaviour of these techniques in the right hand sides of diagrams is quite different. If the chosen number of idle steps is too high - Hill-Climbing wastes this additional time, but Great Deluge uses it for improving the solution.

5.8 Conclusions

This chapter investigated a number of aspects in exploring the time-cost trade-off for improving the quality of exam timetables. In many higher educational institutions, it is quite acceptable to have examination timetable processing times of up to several hours. However, this prolongation is justified only if an algorithm uses the specified amount of time in an intelligent manner in order to produce a solution of a suitably high quality in this period. This can be achieved while embedding time-predefinition mechanisms into local search

methods. However, such mechanisms function differently for different techniques.

Two time-predefined algorithms are discussed and their properties are investigated. It could be considered that the time-cost trade-off is mostly revealed in large-scale problems. The time-predefined algorithms between them produced the best published results on all but two benchmark problems. The Great Deluge algorithm is preferred because it produces most of the best results on these problems (and all of the best results on a further three, more complicated, benchmark problems) in addition to having less uncertainty in algorithmic parameters. This chapter presents two examples of comparison of the performance of the Great Deluge algorithm with conventional methods. Both of them confirmed the superiority of the presented time-predefined approach.

Chapter 6.

Multiobjective Methods for Exam Timetabling Problems

In this chapter the multiobjective nature of exam timetabling problems is discussed. Due to the distinct characteristics of these problems (large scale, complexity, etc.), the performance of different multiobjective techniques applied to them is expected not to be similar. Considerable work needs to be carried out in order to identify the most suitable approach for multiobjective exam timetabling. As a possible tool for the real-world exam timetabling, it is presented and discussed here a new multiobjective A Priori method based on the idea of compromise programming. Besides this, a case study of an application of existing pareto-based techniques to exam timetabling problems is also included.

6.1 An Aggregation Multiobjective Technique based on Compromise Programming

Usually in A Priori multiobjective approaches an objective function is defined, which aggregates all given criteria taking into consideration decision maker's preferences. However, it is very clear that the criteria are of a very different nature. They are incommensurable due to their different units of measure with different scales. In addition, they are partially or totally conflicting. For example, in nine-criteria problem stated in Section 4.2 two exams which should be scheduled immediately before/after each other may have common students and consequently in the timetable construction, improvements of the value of criterion X_9 will lead to the degradation of the value of criterion X_2 (X_9

represents the number of students affected by the violations of "immediately before/after" constraint, while X_2 summarises the number of occurrences when a student has two exams in consecutive timeslots).

Universities will assign different priorities to the constraints imposed on their timetabling problems. The aggregation approach to timetabling enables timetabling officers to assign different relative importances (weights) to each of the criteria to reflect different institutional regulations and requirements. The weights are not related to how easy it is to satisfy the constraints. Also, the criteria which correspond to constraints that are rigidly enforced by the university (i.e. additional hard constraints that the university may set) will be assigned relatively much higher weights than those assigned to criteria that are related to the soft constraints. For example, the constraint that the resources used at any time must not exceed the resources available (represented by criterion X_1) is often considered to be hard. So one would assign it a (much) higher weight than other criteria.

6.1.1 Criteria and Preference Spaces

In [BBP01] a mathematical apparatus have been introduced, which enables different criteria to be taken into consideration simultaneously in the construction of a timetable. A *criteria space* is defined whose dimension is equal to the number of criteria. Each timetable is represented as a point in the criteria space. An ideal point, which optimises all criteria simultaneously, is defined in the criteria space as the vector $I=(f_1^*,..., f_k^*,..., f_K^*)$ where f_k^* (for $k\in\{1,...,K\}$) denotes the best value of the criteria X_k. It is generally the case

that the solution that corresponds to the ideal point is not feasible. The notion of an anti-ideal point is used to define the vector $A=(f_{1*},..., f_{k*},..., f_{K*})$, where f_{k*} (for $k\in\{1,...,K\}$) is the worst value of criteria X_k.

The difficulties caused by different units of measure for the criteria and their different scales are overcome by the use of a *preference space* and the mapping of the criteria space into the preference space introduced in [PP95]. For each criterion X_k, (where $k\in\{1,...,K\}$), a linear preference function is defined which maps the values of the criterion to the real interval $[0,w_k]$. The linear preference function, s_k , for the criterion X_k is defined below.

$$s_k = w_k \frac{f_{k^*} - f_k}{f_{k^*} - f_k^*}, \tag{6.1}$$

where f_k and s_k denote the values of the criterion X_k in the criteria space and the preference space, respectively. Note that the worst value of the criterion f_{k*} is mapped on to the value 0 and the best value of the criterion f_k^* is mapped onto w_k.

The best value of the criterion is achieved when there are no violations of the corresponding constraint in the timetable (i.e. when $f_k^*=0$ for $k\in\{1,...,K\}$). Of course, the values of f_{k*}, for $k\in\{1,...,K\}$ have to be calculated.

An example of such a calculation could be given with f_{2*} which expresses the maximum number of conflicts when students have exams in adjacent periods. First, the maximum number of conflicts is calculated for each student. The worst possible situation for the student is taken into consideration when all of the student's exams are scheduled in adjacent periods. Table 6.1

shows the maximum number of conflicts for a student as a function of the number of student's exams when there are three time periods per day.

Table 6.1: The maximum number of conflicts when a student has exams in adjacent periods

Number of exams E_s	**Day 1**			**Day 2**			**Number of conflicts adj_s^***
	Period 1	**Period 2**	**Period 3**	**Period 4**	**Period 5**	**Period 6**	
1							0
2							1
3							2
4							2
5							3
...							**...**

The value adj_s can be calculated using the following formula:

$$adj_s^* = \begin{cases} (E_s \text{ div } 3)*2 & if \quad E_s \bmod 3 = 0 \\ ((E_s - 1) \text{ div } 3)*2 & if \quad E_s \bmod 3 = 1 \\ ((E_s + 1) \text{ div } 3)*2 - 1 & if \quad E_s \bmod 3 = 2 \end{cases}, \tag{6.2}$$

where E_s is the number of the exams, taken by student s (where $s \in \{1,..., M\}$). The operators *div* and *mod* denote the quotient and remainder of two integer values, respectively.

Now f_2^* is calculated as follows:

$$f_2^* = \sum_{s=1}^{M} adj_s^*, \tag{6.3}$$

where M is the total number of students.

A mapping from the criteria space into the preference space is based on the linear preference functions for all criteria. As an illustration, Figure 6.1

depicts the mapping of the ideal point I=(0,0) from the criteria space on the point W =(w_1,w_2) in the preference space, when K=2.

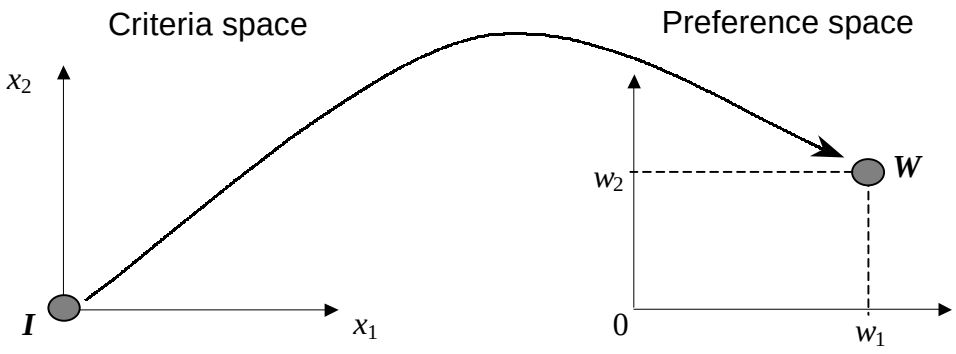

Figure 6.1: Mapping from the criteria space into the preference space

6.1.2 An Algorithm for Heuristic Search of the Preference Space

A new algorithm for heuristic search of the preference space has been developed. It is based, principally, on the idea of compromise programming - a multicriteria decision making method that attempts to determine so-called compromise solutions which are closest to the ideal point [Zel73], [Zel74]. A family of *Lp* metrics is used to measure the distance between the solutions and the ideal point W in the preference space. L_p metrics are defined by the following formula:

$$L_p(S,W) = \left\{ \sum_{k=1}^{K} [s_k - w_k]^p \right\}^{1/p} \qquad 1 \le p \le \infty , \tag{6.4}$$

where $L_p(S,W)$ is the distance between the solution S with co-ordinates s_k in the preference space and the ideal point W with the co-ordinates w_k. Three values of p are usually given a special interest in L_p : p=1, p=2, and p=∞. It can be shown that:

$$L_{\infty}(S,W) = \max_{k}[w_k - s_k]. \tag{6.5}$$

L_2 is a well-known Euclidean norm which gives the compromise solution geometrically closest to W. Smaller values of p imply compensations of criteria values (i.e. a weak value of one criterion can be offset by the strong value of another criterion). The higher values of p lead to solutions where each criterion stands on its own and there are no tradeoffs between the criteria.

The algorithm consists of two phases. The goal of the first phase is to find a set of high quality initial timetables that are good with respect of each criterion separately. These timetables will be used as the starting points in the search of the preference space in the second phase of the algorithm.

The initial timetables are constructed using the "saturation degree" sequential heuristic (described in Section 5.3.1). In each step of the timetable construction, the heuristic selects the exam which has the smallest number of remaining valid periods in the timetable constructed so far. The chosen exam is scheduled in the time period which causes the lowest increase of the criterion value. Ties are broken in the following way: for each exam and for each of its valid periods the increase of the criterion value is calculated and the exam with the highest average increase is selected and scheduled in the best valid period in terms of the criterion value. The motivation is that such an exam should be scheduled early because it contributes more significantly to the criterion value.

These timetables, which are good with respect to each criterion separately, are used as the initial points in the search of the preference space. The algorithm iteratively searches the neighbourhood of each of these

timetables trying to improve the other criteria values with the aim of getting close to the ideal point. Two operators are used to explore the neighbourhood of the timetables. They are based on Hill-Climbing and the heavy mutation operators employed in the Memetic Algorithm described in [BNW96], [BN99]. These operators are modified to take into account the distance of the timetables from the ideal point.

The Hill-Climbing operator randomly chooses an exam from the timetable and reschedules it in the valid time period which yields a maximum decrease in the distance from the ideal point. The purpose of the Hill-Climbing operator is to direct the search toward the local optima. The Hill-Climbing operator is applied until its application does not decrease the distance of the solution from the ideal point for a predefined number of times.

The Mutation operator reschedules the exams from a timeslot, which contributes the most to the distance from the ideal point. The exams from these time periods are rescheduled with respect to the distance from the ideal point. The purpose of the mutation operator is to direct the search away from local optima and to explore new areas of the preference space.

In each step of the preference space search, multiple applications of the Hill-Climbing operator are followed by an application of the mutation operator until the distance between the solution and the ideal point has not decreased for a predefined number of steps.

The search of the neighbourhood of each of the initial solutions yields one timetable. The final solution is the timetable chosen from the set of obtained timetables which has the minimum distance from the ideal point.

6.1.3 A Real Timetabling Problem: Results and Discussion

A series of experiments were carried out in order to test the presented algorithm on a real-world exam timetabling environment which has more than two criteria. A detailed specification of nine criteria is provided for Nott-94 dataset (see Section 4.3) therefore this problem instance was chosen for these experiments. The algorithm was run several times with different values for w_k and p. The processing time of each run was around 5 minutes.

The results obtained in these experiments are discussed here. The timetable officer may express his/her preference in two ways:

by assigning different weights to constraints and consequently to corresponding criteria;

by favouring solutions which permit or do not permit compensation for weak criteria values.

Two parameters are changed: w_k, for $k \in \{1,...,K\}$ and p in L_p metrics. It is considered 4 different cases with respect to the total number of time periods (i.e. 23 periods, 26 periods, 29 periods and 32 periods).

The results obtained for the case when all criteria are of the same importance and for L_2 are given in Table 6.2. This table presents the values of criteria and the distance of the solutions from the ideal point. Of course, when more time periods are available higher quality solutions can be obtained (i.e.

ones which are closer to the ideal point). The values given in the parenthesis are the percentages of the anti-ideal values of the criteria that are achieved in the solution. For example, in the solution with 23 time periods there is 5.02% of the total number of estimated possible conflicts where students have two exams in adjacent periods. In most of the cases the final solution usually significantly improves the values of the other criteria (which are not taken into consideration in the initial solution). However, this is, of course, at the expense of the light depreciation of the criterion value from the initial solution.

Table 6.2: Solutions when all criteria are of the same importance

	23 periods	**26 periods**	**29 periods**	**32 periods**
X_1 (rooms)	1038 (4.29%)	137 (0.57%)	139 (0.58%)	25 (0.10%)
X_2 (adj. per.)	1111 (5.02%)	655 (2.96%)	513 (2.32%)	314 (1.42%)
X_3 (same day)	3518 (8.49%)	2814 (6.79%)	2239 (5.40%)	1546 (3.73%)
X_4 (adj. days)	4804 (12.16%)	2759 (6.98%)	2172 (5.50%)	1646 (4.17%)
X_5 (overnight)	405 (2.70%)	265 1.77%)	231 (1.54%)	174 (1.16%)
X_6 (duration)	4 (0.28%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
X_7 (time per)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
X_8 (before/after)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
X_9 (im.bef./after)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
L_2	0.164559	0.103448	0.082128	0.058842

Let us now suppose that the timetable officer prefers timetables in which the number of conflicts where students have exams in adjacent periods is as small as possible. Therefore, the criterion X_2 is assigned a higher importance then the other criteria. In fact three cases were investigated where w_2= 2, 5, and 10 (while all the other weights were assigned value 1). This leads to solutions with a lower number of violations of the corresponding constraint (see Table 6.3). The timetable officer may choose the timetable which best satisfies his/her preferences concerning the other constraints.

Table 6.3: Solutions when X_2 is of higher importance than the other criteria

***W*=(1,2,1,1,1,1,1,1,1)**				
	23 periods	**26 periods**	**29 periods**	**32 periods**
X_1 (rooms)	982 (4.06%)	301 (1.25%)	187 (0.77%)	85 (0.35%)
X_2 (adj. per.)	713 (3.22%)	378 (1.71%)	330 (1.49%)	230 (1.04%)
X_3 (same day)	3511 (8.47%)	2922 (7.05%)	2083 (5.02%)	1562 (3.77%)
X_4 (adj. days)	5216 (13.20%)	3291 (8.33%)	2358 (5.97%)	1740 (4.40%)
X_5 (overnight)	486 (3.24%)	351 (2.34%)	221 (1.47%)	171 (1.14%)
X_6 (duration)	39 (2.70%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
X_7 (time per)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
X_8 (before/after)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
X_9 (im.bef./after)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
L_2	0.179373	0.117357	0.085153	0.062710
***W*=(1,5,1,1,1,1,1,1,1)**				
	23 periods	**26 periods**	**29 periods**	**32 periods**
X_1 (rooms)	1358 (5.62%)	418 (1.73%)	274 (1.13%)	78 (0.32%)
X_2 (adj. per.)	435 (1.97%)	220 (0.99%)	149 (0.67%)	131 (0.59%)
X_3 (same day)	3262 (7.87%)	2925 (7.05%)	2068 (4.99%)	1564 (3.77%)
X_4 (adj. days)	5701 (14.43%)	3768 (9.53%)	3219 (8.15%)	1982 (5.02%)
X_5 (overnight)	565 (3.77%)	463 (3.09%)	347 (2.32%)	221 (1.47%)
X_6 (duration)	4 (0.28%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
X_7 (time per)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
X_8 (before/after)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
X_9 (im.bef./after)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
L_2	0.203092	0.133388	0.104502	0.071009
***W*=(1,10,1,1,1,1,1,1,1)**				
	23 periods	**26 periods**	**29 periods**	**32 periods**
X_1 (rooms)	1275 (5.28%)	321 (1.33%)	192 (0.79%)	96 (0.40%)
X_2 (adj. per.)	212 (0.96%)	106 (0.48%)	90 (0.41%)	44 (0.20%)
X_3 (same day)	3525 (8.50%)	2810 (6.78%)	2240 (5.40%)	1660 (4.00%)
X_4 (adj. days)	6584 (16.66%)	4564 (11.55%)	3306 (8.37%)	2382 (6.03%)
X_5 (overnight)	855 (5.71%)	574 (3.83%)	420 (2.80%)	321 (2.14%)
X_6 (duration)	100 (6.92%)	49 (3.39%)	0 (0.00%)	0 (0.00%)
X_7 (time per)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
X_8 (before/after)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
X_9 (im.bef./after)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
L_2	0.234499	0.151720	0.111444	0.078141

The values of the parameter p significantly influence the solutions. The solutions presented in Table 6.4 are obtained when all criteria have the same weights. For p=1, and p=2, the eventual weak value of one criterion is compensated by the stronger values of all the other criteria. On the other hand for $p=\infty$ all the criteria values are reasonably strong, although in some cases not as strong as in the solutions for p=1 and p=2.

Table 6.4: Solutions for different distance measures, W=(1,1,1,1,1,1,1,1)

	23 periods	**26 periods**	**29 periods**	**32 periods**
X_1 (rooms)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
X_2 (adj. per.)	879 (3.97%)	604 (2.73%)	393 (1.78%)	316 (1.43%)
X_3 (same day)	3623 (8.74%)	2544 (6.14%)	1957 (4.72%)	1332 (3.21%)
X_4 (adj. days)	6381 (16.15%)	4571 (11.57%)	3438 (8.70%)	2482 (6.28%)
X_5 (overnight)	264 (1.76%)	164 (1.09%)	151 (1.01%)	53 (0.35%)
X_6 (duration)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
X_7 (time per)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
X_8 (before/after)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
X_9 (im.bef./after)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
L_1	0.306187	0.215262	0.162033	0.112748
X_1 (rooms)	2848 (11.78%)	2044 (8.46%)	1559 (6.45%)	1243 (5.14%)
X_2 (adj. per.)	2608 (11.78%)	1872 (8.46%)	1435 (6.48%)	1138 (5.14%)
X_3 (same day)	4886 (11.78%)	3507 (8.46%)	2688 (6.48%)	2132 (5.14%)
X_4 (adj. days)	4658 (11.79%)	3343 (8.46%)	2563 (6.49%)	2033 (5.14%)
X_5 (overnight)	807 (5.39%)	475 (3.17%)	441 (2.94%)	334 (2.23%)
X_6 (duration)	170 (11.76%)	119 (8.24%)	89 (6.16%)	74 (5.12%)
X_7 (time per)	40 (9.90%)	24 (5.94%)	24 (5.94%)	18 (4.46%)
X_8 (before/after)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
X_9 (im.bef/after)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
L_∞	0.117867	0.084592	0.064855	0.051444

6.2 A Case Study of the Application of Pareto-Based Approach to Exam Timetabling Problems

One of the attractions of Pareto-based methods is the simplicity of the expression of a decision maker's preferences. This advantage is very useful for a timetabling officer, who could obtain a set of different timetables and then chose among them for the one, which mostly conforms to the particular requirements of that university. However, very little research has been carried out concerning A Posteriori algorithms for real-world exam timetabling problems. Usually these methods have tested on small-sized optimisation problems including continuous ones where the true Pareto-front can be found analytically, and their performance in large-scale problems is not well studied. Two selected Pareto-based techniques (Non-Dominated Sorting Genetic Algorithm and Pareto Archived Evolution Strategy) are applied to benchmark exam timetabling datasets. The motivation was that the obtained results could give some notion about the possibility of the use of these methods in real examination timetabling and should provide benchmarks for further experiments with multiobjective exam timetabling problems.

6.2.1 Non-Dominated Sorting Genetic Algorithm for Exam Timetabling

Firstly, the performance of the Non-dominated Sorting Genetic Algorithm (NSGA) introduced by Srinivas and Deb in [SD94] was investigated. This is a multiobjective modification of the general single-objective Genetic Algorithm, which employs an advanced method for of the calculation of fitness. The algorithm operates with a population of individuals, whose size is set up as an

input parameter. Generally, the population size affects the quality of produced results and is limited by the used computational resources.

The NSGA algorithm operates for a given number of generations while at each generation it applies the following set of subroutines:

- mutation (SO),
- crossover (SO),
- ranking (MO),
- calculation of dummy penalty (MO),
- selection (SO).

The operations, which are the same as in the single-objective variant are marked as (SO), and specific multiobjective procedures are marked as (MO).

The mutation operator performs a sequence of atomic moves (of the same nature as in local search algorithms). Each move produces a small random perturbation of a mutated individual. In particular, the algorithm implemented for exam timetabling in this research randomly picks out an exam and replaces it into a different (also randomly chosen) timeslot. Only those moves, which generate feasible mutants are accepted. The number of undertaken successful moves affects the degree of the transformation of an individual during mutation and it can be seen as a second parameter of the algorithm. At each generation the mutation operator is applied to the given number of randomly chosen individuals. This number is specified by a third algorithmic parameter, which is called a *mutation rate* and is expressed in percents of mutated individuals in the whole population. Both described mutation parameters highly influence the

performance of the algorithm. They should have enough high values to provide a sufficient diversity of the population. However, too high values decrease the quality of final results.

Crossover. This procedure operates after mutation and generates a set of new solutions ("children"), whose number is specified by a fourth parameter of the algorithm. Every child solution is produced from two randomly chosen parents while applying the so-called "single-point crossover". Here the exam (point) Z is randomly chosen ($Z \in \{2, ..., N\}$ where N is the number of exams). In a new generated child all exams whose ordinals are less than Z are assigned to the same timeslots as in the first parent, and then starting from Z the assignment is conducted using a second parent as the pattern. The operation is accepted if it satisfies the following requirements:

a) Parents should be different from each other. This requirement is checked before the generation of a child.
b) A generated child should be different from both parents. This is checked in addition to requirement (a) because even different parents, which contain the same parts before or after the point Z can produce a child which is identical to one of the parents.
c) The child should be feasible. Requirements (b) and (c) are checked after the generation of the child.

If any of these three requirements are not satisfied then the generated child is discarded and the operation is repeated (a new pair of parents is selected, etc.). The success of the procedure depends on the diversity of the population. If

improper mutation parameters cause a low diversity, then the generation of new children becomes difficult or even the algorithm can “freeze” (no new children can be generated).

The crossover procedure is repeated until the specified number of children is produced. This parameter determines the so-called “selection pressure”, i.e. its higher value forces quicker improvement of the population, but the algorithm tends to converge with lower quality results and vice versa. Besides this, the offspring can be seen as an additional population, which together with the main population occupies certain computational resources. Thus, tuning of the sizes of both these populations should provide a reasonable balance between the processing time and the quality of the result.

A ranking procedure sorts solutions in the population and does not require the setting up of any parameters. Every individual obtains its own rank, which represents the level of its dominance. The best rank (equal to 1) is assigned to the non-dominated solutions while the dominated solutions get worse ranks (more than 1). The algorithm operates in the following way: at the beginning all individuals have the rank=1. Then the algorithm compares all individuals in pairs and assigns the rank=2 to all dominated ones. At the next iteration only individuals with the rank=2 are compared between each other and again all dominated ones get the rank=3. These iterations are repeated until no dominated solutions remain among the ones with the worst rank.

Dummy penalty is a multiobjective analogue of a fitness function, which aims to provide the uniform distribution of the final population along a trade-off

surface. Initially the dummy penalty (F_d^0) of an individual is equal to the population size divided by the rank of the individual. Then the penalty is corrected by the so-called "sharing" operator. Here distances from the given individual to all the others are calculated. The algorithm takes into account only those distances d_i, which are less than some value σ_{share}. Let N_d be a number of such distances and $i \in \{1, ..., N_d\}$. The value of σ_{share} is given as a fifth algorithmic parameter. The final dummy penalty (F_d) is calculated as follows:

$$F_d = \frac{F_d^0}{\sum_{i=1}^{N_d} \left[1 - \left(\frac{d_i}{\sigma_{share}} \right)^2 \right]}. \tag{6.6}$$

Formally speaking, the calculation of distances should include a proper normalisation of the criteria values in order to take into account the scales of criteria. However, experiments showed that even with the non-normalised values the algorithm works adequately with the given problem instances and provides a quite uniform distribution of the final population.

The "sharing" technique is highly dependent on the particular value of σ_{share}. The further experiments showed that if its value is too low, then the population tends to crowd in a small region. Otherwise, if it is too high, then the algorithm lets non-useful solutions (which have a very poor quality but which are far away from other individuals in the population) to survive for future generations.

Selection. This procedure is the same as for the single-objective Genetic Algorithms. Here the individuals with the worst dummy penalty are removed from the population. The number of removed solutions is equal to the number of children, produced by crossover. This procedure does not operate with any parameters.

The described algorithm was applied to the Nott-94 exam timetabling problem. It was considered the bi-criteria case (defined in Section 4.2) where the criterion X_1 represents the number of occurrences where a student has two exams in adjacent periods while X_2 represents the corresponding number for overnight exams. The initial population was generated as specified in Section 5.3.1.

A practical utilisation of the described algorithm included an important preliminary procedure regarding the tuning of its five problem-dependent (and probably correlated between each other) parameters. Their combination, which provides the best performance of the algorithm on the particular problem could be found by the complete enumeration of all five-dimensional "parameter vectors" and running the algorithm with each of them. However, it requires the high number of experiments and practically unrealistic. Therefore, the search for good parameter vectors was simplified by exploring the "parameter space" in two phases. Firstly, for each parameter a wide interval of values was defined and a number of test runs were launched while trying 10-20 values, uniformly distributed inside each defined interval. The parameter values, which caused the highest performance of the algorithm were selected for the second phase.

The second phase comprised the fine tuning of parameters by the slight increasing/reducing of values chosen in the first phase. The final values of parameters together with tested intervals are presented in Table 6.5 (a mutation rate is shown in percents to the population size). The author cannot guarantee that these parameters are the best ones, however they caused the best performance of the algorithm among all test launches.

Table 6.5: Parameters for Non-Dominated Sorting Genetic Algorithm

Parameter	**Tested interval**	**Selected value**
Population size	20-630	256
Number of mutation moves	1-500	15
Mutation rate (%)	10-100%	30%
Number of children	10-620	384
σ_{share}.	1-300	50

The selected values from this table were used in several final launches of the algorithm. The number of generations was set up to be 10000. This quantity is significantly higher than required because already after 5000 generations there was no visual movement of a trade-off surface. The average time of one run of the algorithm was around 5 hours. Although the algorithm with different runs produced different final populations, the distributions of their solutions in the criteria space look quite similar. A typical example of the shape of such a distribution is shown in Figure 6.2.

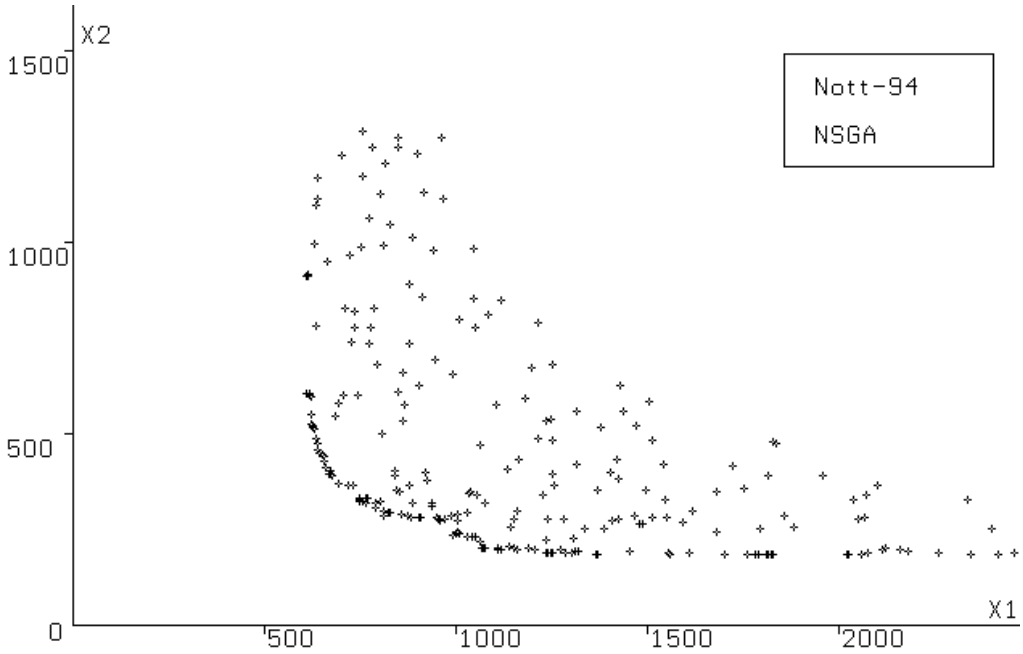

Figure 6.2: A final population produced by NSGA

In this diagram the population contains a distinct trade-off surface, which comprises 43 non-dominated solutions. Its visual examination reveals that it covers a significant sector of the criteria space, has a quite regular shape and is relatively uniformly distributed. To estimate the quality of these solutions their criteria values were compared with the results produced by A Priori techniques discussed in Section 5.6.2. For comparison, there were selected 7 samples from the set of 43 non-dominated points presented in Figure 6.2 so that they are uniformly distributed through the trade-off surface. Hence, one can consider this sample subset as an adequate representation of the complete set. The sample solutions, sorted in the ascending order of the first criterion value and marked as “NSGA” are shown in Table 6.6.

For the comparison the Table 6.6 includes three non-dominated solutions of the same problem produced by Burke and Newall in [BN99] by the

weighted sum aggregation technique within a Multi-Stage Memetic Algorithm (marked in Table 6.6 as "MSMA"). The authors varied several internal algorithmic parameters but used the same aggregation function expressed by formula (5.7), which gave a higher priority to the first criterion (in contrast, all solutions in the NSGA trade-off surface have X_2 less than X_1). Despite this, the MSMA results dominated 5 out of 7 of the NSGA sample solutions. We can see a substantial gap between their criteria values. For example, point № 3 from the MSMA set has slightly better value of X_2 than solution № 29 from the NSGA set, but 9.5 times better X_1. Although two NSGA sample points are not dominated by the MSMA ones, their minor superiority in the second criteria (1.3-1.4 times) scarcely could compensate the huge inferiority in the first one (11-17 times).

Table 6.6: Comparison of results produced by NSGA and A Priori techniques

Algorithm	**Point**	X_1	X_2	**Time (sec)**
NSGA	1	610	605	≈18000
	8	631	511	
	15	660	413	
	22	748	323	
	29	957	272	
	36	1109	198	
	43	1779	183	
MSMA	1	65	324	156
	2	76	282	340
	3	100	255	607
GD	1	117	167	383

Table 6.6 also includes one point from the middle of the time-cost diagram produced by the Great Deluge algorithm in Section 5.6.2 and shown in

Figure 5.18. This solution (marked as "GD") was achieved in 383 seconds using the same aggregation function as MSMA. This single point completely outperforms (see Section 3.4) the whole trade-off surface produced by NSGA. It has 5.2-15.2 times better value of X_1 and 1.1-3.6 times better value of X_2.

In assessing the performance of NSGA, it should be noticed that its processing time (the last column in Table 6.6) is 30-115 times longer than the corresponding search time of A Priori methods (the time of the Great Deluge result was taken within a conventionally acceptable interval). Besides this, the necessity of the definition of 5 problem-dependent algorithmic parameters is considered as a serious difficulty with respect to practical use. Their tuning requires an exhaustive search in a 5-dimensional parameter space. Thus, in spite of the quite high popularity of NSGA, its described basic variant cannot be considered as a realistic tool for multiobjective exam timetabling.

Elitism. One of the suggested ways of improving the performance of NSGA is the employment of so-called "elitism" [Bag99]. This technique keeps the best members of the population for future generations. Such an idea is plausible because the mutation of relatively good individuals often leads to a worsening of their quality.

A variant of NSGA for exam timetabling, which employs elitism has also been developed in the course of this thesis. Here the mutation operator avoids the mutation of the non-dominated individuals (it begins from the solutions with rank>1). A required number of non-dominated solutions is mutated if all dominated solutions have already been mutated, but the mutation

rate is still not achieved. This can happen when the number of non-dominated solutions (which varies from generation to generation) becomes too high, which makes impossible to achieve the mutation rate with dominated solutions. However, with the mutation rate set to be 30% (see Table 6.5) this situation occurs quite rarely.

The “elitist” variant of NSGA was also tested on the same problem. The random seed, the number of generations and all algorithmic parameters were the same as in the experiments without elitism. The behaviours of both variants of NSGA were quite similar to each other, as well, as the distribution of a final population in the criteria space. The comparison of trade-off surfaces produced by both algorithms is presented in Figure 6.3

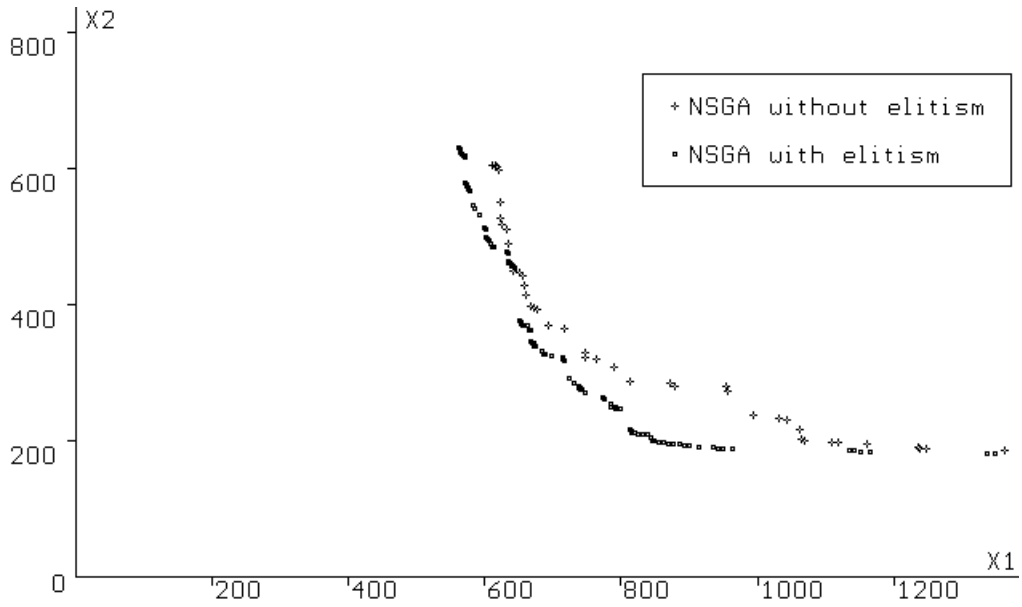

Figure 6.3: A comparison of trade-off surfaces produced by NSGA with and without elitism.

This figure shows a slight improvement of the quality of the trade-off surface produced by the “elitist” variant of NSGA relative to the “non-elitist” one.

However, this improvement is negligible in relation to the huge gap between the performance of NSGA and A Priori techniques. It is unlikely that with further modifications or when applied to different exam timetabling datasets the performance of NSGA might be improved so that it can be seriously compared with other approaches. The author considers the produced results to be so poor that there is no reason to use them as a benchmark for the estimation of the performance of other techniques. Therefore, further investigation did not carry out with this algorithm in order to pay attention to more powerful techniques. This decision was also reinforced by the unsuitability of NSGA for the practical use. This algorithm is enormously time-consuming and requires significant preliminary work comparing to other useful approaches.

6.2.2 Pareto Archived Evolutionary Strategy for Exam Timetabling

Apart from NSGA another well-known A Posteriori multiobjective technique called Pareto Archived Evolution Strategy (PAES) can also be applied to exam timetabling problems. It is a greedy local search, which archives non-dominated current solutions. This method was proposed in [KC99] as an alternative to Genetic Algorithms. Furthermore, the authors suggested it as a good benchmark technique for testing the performance of other methods.

The basic variant of the algorithm known as “(1+1)-PAES” is a multiobjective extension of the Hill-Climbing method. It inherits all its features and additionally keeps the archive of intermediate non-dominated solutions. One of these solutions is maintained as the current one. At each iteration a

feasible candidate solution is chosen among neighbourhood solutions and evaluated in the following way:

- If the current solution is dominated by the candidate one or has the same values of criteria, then the candidate solution replaces the current one.
- If the candidate solution is dominated by the current one or by any other member of the archive, then it is discarded.
- If the dominance cannot be determined, the candidate solution is added to the archive. Here this solution can also become the current one (while the previous current solution remains as an ordinary archive member).

The choice of the current solution is made so that it should be placed in the less crowded region of the criteria space. The evaluation of the solution's region is started from calculating the lowest and the highest values of each criterion among all archive members. After that, for every criterion an interval between the calculated lowest and highest values is divided into two equal parts, then both remaining parts are divided once more and so on. The number of these divisions l (depth of the binary division) is set up as an input parameter of this algorithm. This procedure divides the criteria space into a number of hypercubes (in bi-criteria case a hypercube is represented by a rectangle). The so-called "grid location" of a solution is calculated as the number of the archive members in the same hypercube and shows how crowded the region is.

The algorithm starts with the empty archive and continues to add non-dominated solutions into it until the archive becomes filled in. When this happens, the grid locations are calculated for all archived solutions and a

solution with the highest grid location is removed from the archive (if the highest grid location belongs to the candidate, then it is discarded). The archive size is set up as the second parameter of the algorithm.

The described algorithm has been evaluated on the same problem as in the previous section. The archive size was set up to be 150 and depth of binary division l =5. A number of runs of the algorithm were done starting from different random solutions. PAES is a quite fast algorithm and the processing time of each run was in an interval of 3-5 minutes. However, the algorithm has no parameters, which can influence the processing time and correspondingly the quality of results. The typical resulting trade-off surface is depicted in Figure 6.4 (the result of NSGA from the previous section is also shown to enable a comparison of these two techniques).

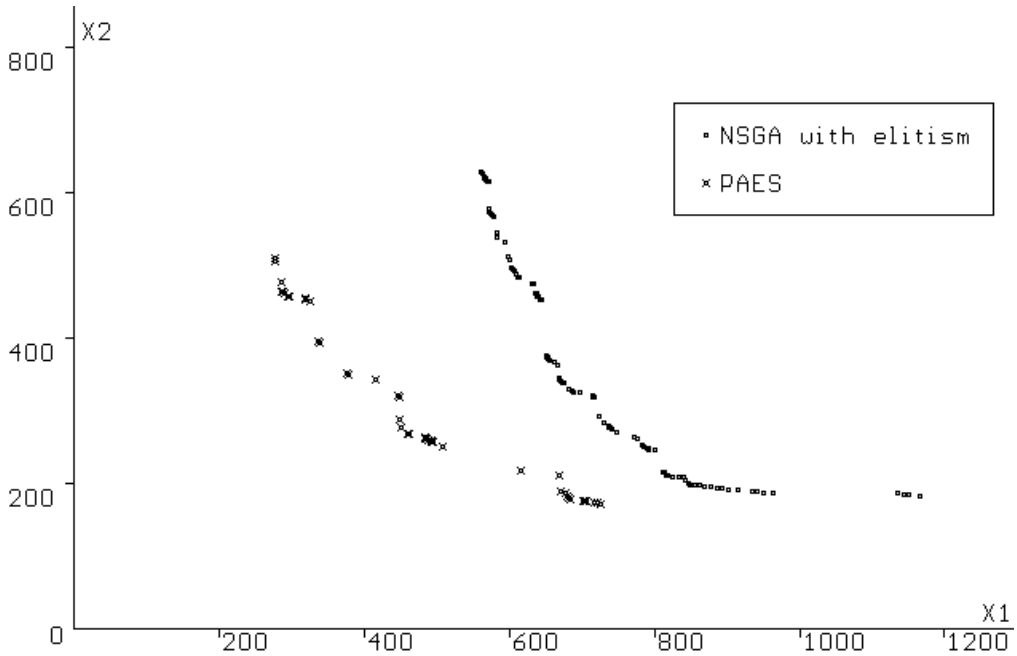

Figure 6.4: The comparison of trade-off surfaces produced by NSGA and PAES

The figure shows that the trade-off surface produced by PAES completely outperforms the NSGA one, i.e. each NSGA solution is dominated by at least one PAES result. One can consider the PAES results to be much higher quality than the NSGA ones, so that they can be taken as benchmarks for the further further experiments, presented in Section 7.5.3. With this aim, PAES was applied to all datasets from the University of Toronto archive. The defined values of the parameters (and resulting search time) were the same as in the previous experiment. For each dataset the number of trade-off surfaces were produced. Three typical examples (for LSE-F-91, RYE-S-93 and YOR-F-83 problems) are given in Figures 6.5-6.7.

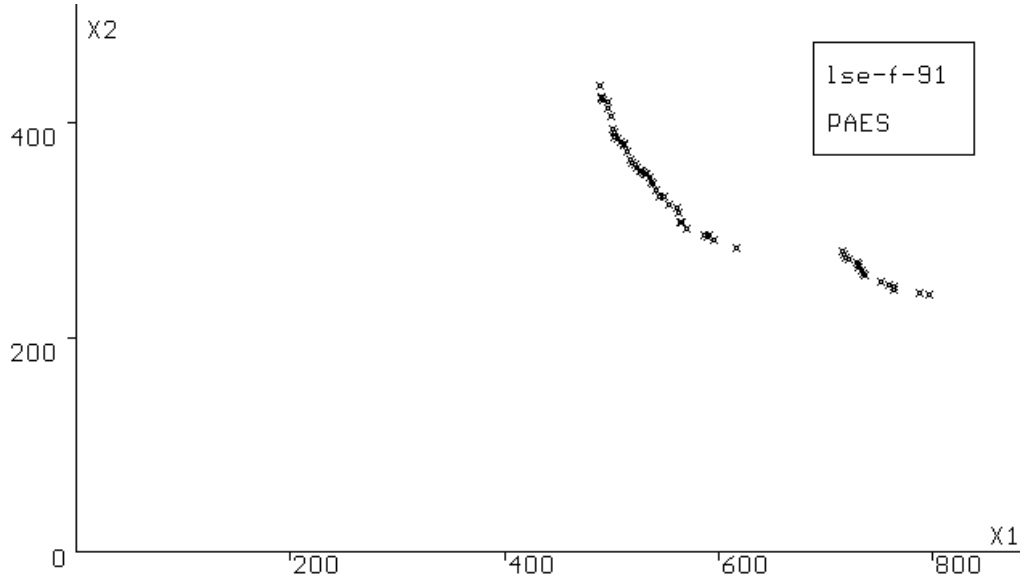

Figure 6.5: Trade-off surface produced by PAES for LSE-F-91 problem

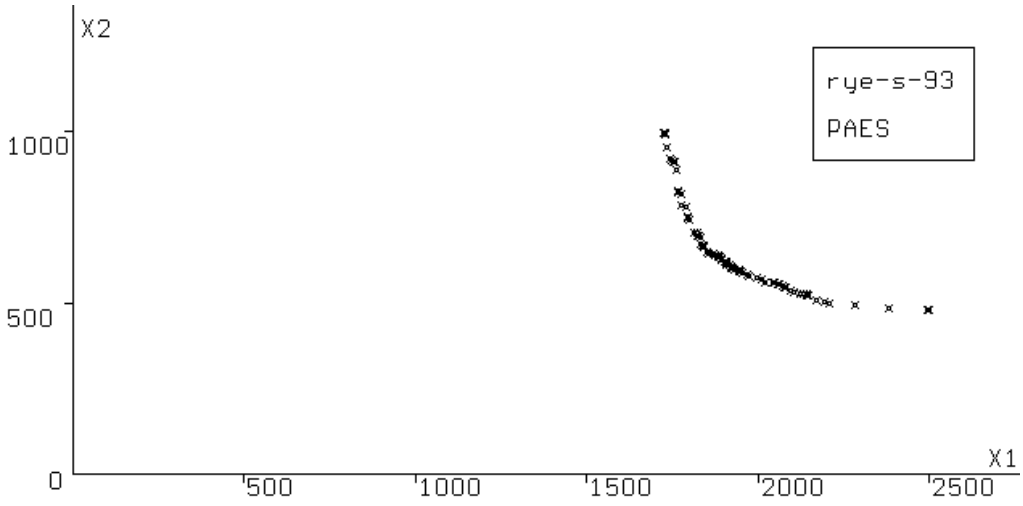

Figure 6.6: Trade-off surface produced by PAES for RYE-S-93 problem

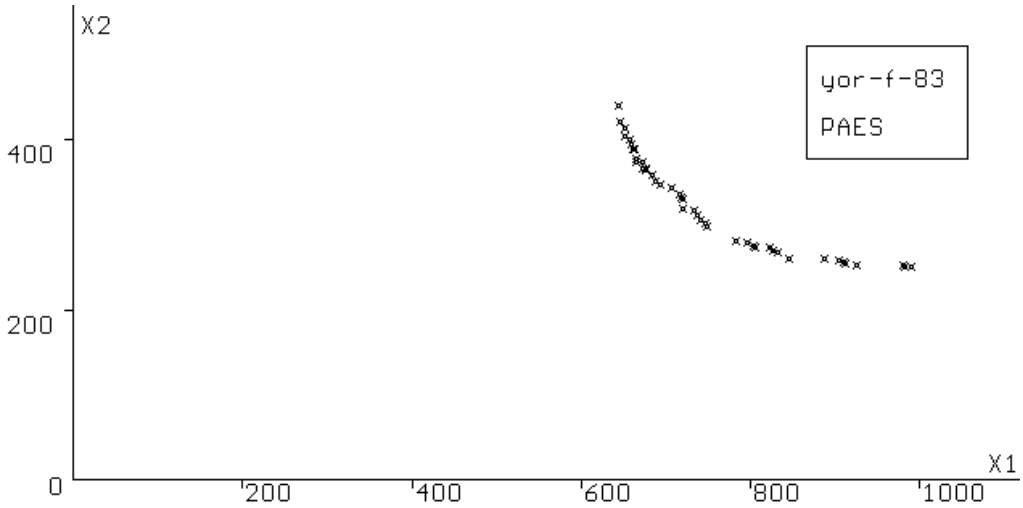

Figure 6.7: Trade-off surface produced by PAES for YOR-F-83 problem

The depicted trade-off surfaces have fairly regular shapes and uniform distributions of points among them. The common characteristics of the best produced results are given in Table 6.7: the number of solutions in the final sets and their minimum and maximum values of both criteria (which could provide an idea about the covered sectors of the criteria space).

Table 6.7: Characteristics of results of PAES algorithm

Data set	**Set size**	X_1 ***min***	X_1 ***max***	X_2 ***min***	X_2 ***max***
CAR-F-92	30	2129	2602	738	925
CAR-S-91	117	2056	2469	565	857
EAR-F-83	59	715	1039	248	434
HEC-S-92	34	478	654	202	302
KFU-S-93	44	1539	1836	841	1233
LSE-F-91	57	489	797	240	435
PUR-S-93	99	2124	2463	829	1147
RYE-S-93	79	1725	2499	479	998
STA-F-83	73	2156	2422	1065	1331
TRE-S-92	43	802	982	419	562
UTA-S-92	53	1609	1984	501	730
UTE-S-92	50	745	1141	420	747
YOR-F-83	44	645	990	250	440

However, these results are still dominated by the ones obtained with A Priori techniques. For comparison Table 6.8 shows the criteria values of selected results produced for the CAR-F-92 and KFU-S-93 datasets by the Multi-Stage Memetic Algorithm in [BN99] and the Great Deluge search in Section 5.6.2 marked in the same way as in Table 6.6 (the reason of exclusion of the third PUR-S-93 problem is described in Section 5.6.2). While comparing figures given in these tables, one can concluded that the PAES method is also scarcely useful in real-world exam timetabling.

Table 6.8: Results of A Priori techniques

Data set	**MSMA**			**GD**		
	X_1	X_2	Time (sec)	X_1	X_2	Time (sec)
CAR-F-92	363	576	186	335	590	547
KFU-S-93	222	838	73	401	477	34

6.3 Conclusions

This chapter presents an application of three multiobjective algorithms to exam timetabling: a new A Priori aggregation technique based on the idea of Compromise Programming and two existing A Posteriori Pareto-based methods: Non-dominated Sorting Genetic Algorithm and Pareto Archived Evolution Strategy.

The presented aggregation method employs the concept of an ideal point that corresponds to the solution, which does not violate any of the stated constraints. Such a solution does not usually exist in real-world timetabling problems but the aim is to try to approach it. The method evaluates timetables according to their distances from the ideal point (taking the relative importance of the constraints into account). The timetable officer may express his/her preference by altering the weights of the criteria (which correspond to the relative importance of the constraints) and by choosing a distance measure.

The initial results have confirmed that such an approach can provide a flexibility in the handling of different types of constraints which is not possible using a single objective function. It enables constraints of a fundamentally different nature to be handled together and makes an appropriate compromise

between them according to the regulations and requirements of particular universities.

Both investigated A Posteriori techniques produce sets of non-dominated solutions (trade-off surfaces) among which the user can select preferable ones. The trade-off surface, produced by NSGA is relatively uniformly distributed, covers a large sector of the criteria space and has a regular shape. The quality of these results can be slightly improved by employing elitism in the basic variant of NSGA. However, all these results are much worse than the ones produced by A Priori approaches.

In the benchmark problem instances the trade-off surface produced by PAES evidently outperforms both (non-elitist and elitist) NSGA ones. Moreover, the author have found PAES to be more easy in use. It works substantially faster than NSGA and requires less effort in tuning two parameters, than NSGA, which involves a higher number of parameters, namely five of them. Although the results of PAES are still worse than A Priori ones, they are considered to be benchmarks for testing the new multiobjective algorithms which are presented in the next chapter.

Chapter 7.

A Trajectory-Based Multiobjective Search

7.1 Driving the Search through a Trajectory

This chapter introduces a new A Priori approach for multiobjective optimisation which is based on the idea of driving current solutions through a predefined trajectory. One could consider that during the search an initial solution and all the following current solutions conform to points in the criteria space whose number of dimensions is equal to the number of criteria. The trajectory can be thought of as a set of points, corresponding to all current solutions during the search. The real search trajectory is a quite complex curve. However, the decision maker can set a line (predefined trajectory), which sketches the region in the criteria space where the search should be carried out. At each iteration the algorithm should provide a gradual improvement of the current solution while keeping it close to the defined line. Assuming, that all points on the trajectory correspond to solutions of different quality, our aim is to reach the trajectory point (or its vicinity) of maximum possible quality.

The trajectory-based approach is principally different to other well-known A Priori techniques:

- *The aggregation approach* aims to improve an aggregation function without considering the particular values of each criteria. On the other hand, the trajectory-based approach aims to improve each criterion separately while observing its effect on the other criteria values.

- *Goal Programming*. The trajectory-based approach does not aim to achieve a given point in the criteria space but to find the best possible solution within the confines of the given trajectory.
- *Lexicographic ordering* can be viewed as a special case of the trajectory specification where the trajectory consists of branches drawn parallel to the axes. While the Lexicographic ordering operates with criteria sequentially, the proposed approach operates with all of them simultaneously and takes into consideration their interaction.

In this chapter the trajectory-based search will be explained in more detail. Two variants of a trajectory-based algorithm will be discussed, basic and enhanced ones, which can be used in multiobjective optimisation. Moreover, several strategies, which might help the decision maker to express his/her preferences throughout the search of the criteria space will be suggested.

7.2 A Reference Solution Strategy

In [PB03] it was introduced a variant of an application of the trajectory-based approach while expanding the idea of the reference timetable expressed by Paechter et al. in [Pae98]. As the reference they considered a timetable produced either manually or automatically using a different dataset. The authors suggested an evolutionary algorithm, which obtains a solution genotypically similar to the reference one while penalising the differences between the reference and new timetables. They also pointed out that the

reference solution may already be located in a local optimum, and therefore it is worth starting the search for the new solution from scratch.

For the purpose of multiobjective optimisation the reference solution can be considered in a phenotypic sense (i.e. in the sense of criteria values). Thus, the decision maker should specify the criteria values of some attainable solution which to a certain degree meet his/her preferences. This solution can be produced manually or selected from the set of solutions generated by some automated method. It is assumed that the decision maker is not satisfied completely with this solution, but this choice gives information that is helpful for a further search for a better solution.

Having a more or less preferable reference solution, one can consider that all further solutions which dominate the reference one (where all reference criteria are outperformed by the new ones) will be even more preferable. In order to find these solutions, the following method could be suggested. The reference solution is represented as a point in the criteria space and a trajectory is drawn through this point and the origin. In such an approach the reference solution is used only for drawing the appropriate trajectory (as a benchmark for assessing the final solution), but does not affect the further search process. If the reference solution was produced by some search method, then it is likely that such a result already lies in a local optimum and cannot be used for the initialisation purpose. Generally, local search techniques show the best performance when they start from a random solution. Therefore, it can be

suggested keeping this practice also for the presented approach. For the bi-criteria case (criteria x_1 and x_2) the method is illustrated in Figure 7.1.

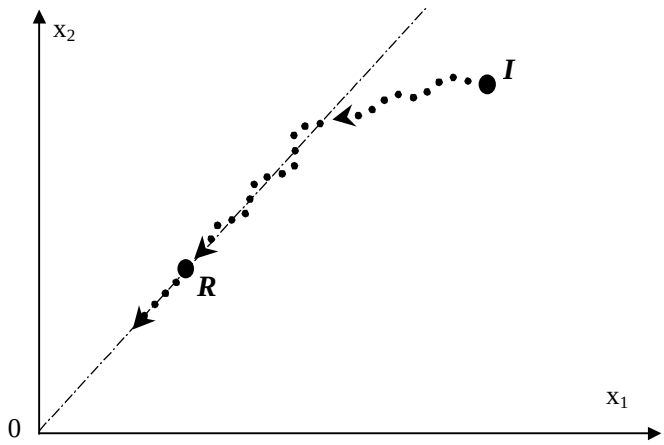

Figure 7.1: Search along the defined trajectory

In this figure the trajectory is depicted as a dash-dotted line. The search starts from a randomly generated initial solution (point *I*) and at first approaches the trajectory (generally, the initial solution does not lie on the trajectory). The search then follows the trajectory until it reaches the vicinity of the reference solution (point *R*). Passing the reference point the search continues along the trajectory and stops when there is no improvement of any criterion value for a predefined number of iterations. The point of convergence will be obviously superior to the reference point.

7.3 Great Deluge with Variable Weights

7.3.1 Description of the Method

This section presents a technique which enables driving the search through a predefined trajectory drawn through the origin. It operates with a weighted sum cost function, but the weights are varied dynamically during the search. A

special procedure for weights variation has been developed in order to regulate the direction of the search.

The explanation of the proposed method is illustrated with a bi-criteria case (the goal is to minimise criteria x_1 and x_2). Let us consider a weighted sum aggregation function with weights w_1 and w_2 within the Great Deluge algorithm. The condition of acceptance of a candidate solution $S=(s_1, s_2)$ in each iteration can be expressed by the following inequality:

$$s_1 w_1 + s_2 w_2 \leq B . \tag{7.1}$$

This formula states that the algorithm accepts any solution in the space bounded by axes x_1 and x_2 (as the criteria values are always positive) and the line

$$x_1 w_1 + x_2 w_2 = B . \tag{7.2}$$

In Figure 7.2 this borderline is marked as G_1G_2. The points where it intersects the axes can be calculated as: $G_1=B/w_1$; $G_2=B/w_2$. The space of acceptance is denoted by the shadowed triangle.

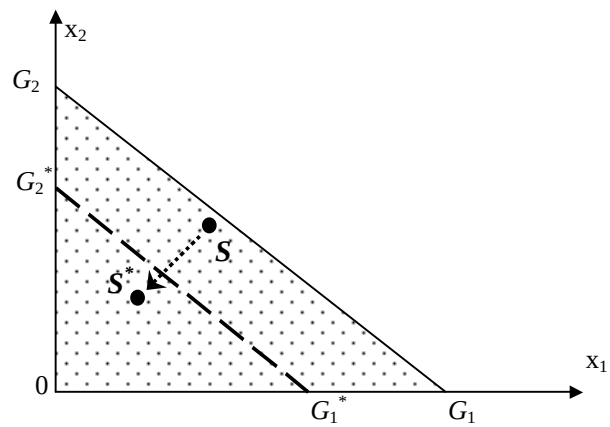

Figure 7.2: Borderline in the weighted sum Great Deluge algorithm

The lowering of the level value B at each step corresponds to a shift of the borderline towards the origin. The new borderline $G_1^*G_2^*$ is expressed by

$$x_1 w_1 + x_2 w_2 = B - \Delta B. \tag{7.3}$$

The new intersection points are calculated as $G_1^*=(B-\Delta B)/w_1$ and $G_2^*=(B-\Delta B)/w_2$. The shifting of the borderline means that the new current solution (S^*) will be closer to the origin.

Let us define $\Delta w = \Delta B / (B-\Delta B)$. Consequently, the equation (7.3) can be transformed into the following form:

$$x_1 w_1 (1+\Delta w) + x_2 w_2 (1+\Delta w) = B. \tag{7.4}$$

Due to this formula the decrease of B in each iteration can be replaced with the appropriate increase of both weights as it causes the same effect (shifting of the borderline).

Hence, finally, each increase of a single weight induces a rotation of the borderline such that the new solution improves the corresponding criterion more than the other one. Thus, equation (7.5) corresponds to the line $G_1^*G_2$ in Figure 7.3 and equation (7.6) corresponds to the line $G_1G_2^*$ in Figure 7.4.

$$x_1 w_1 (1+\Delta w) + x_2 w_2 = B, \tag{7.5}$$

$$x_1 w_1 + x_2 w_2 (1+\Delta w) = B. \tag{7.6}$$

Increasing the weight w_1 causes the current solution S to move into a new position S^* so that the value of criterion X_1 is improved more than the value of

X_2 . This is illustrated in Figure 7.3. The increase of w_2 causes the opposite effect (Figure 7.4).

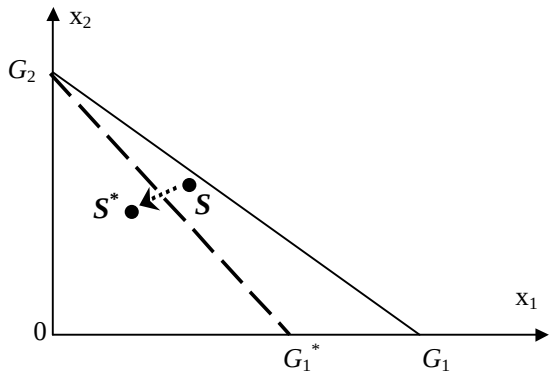

Figure 7.3: The increase of w_1

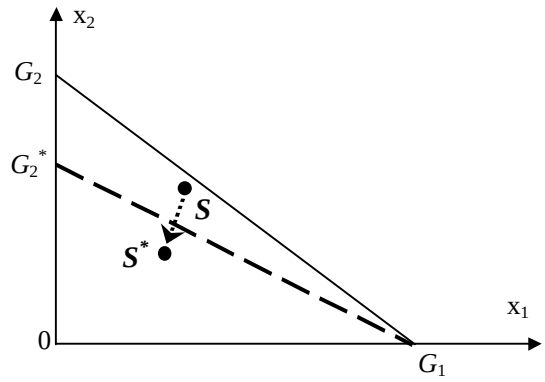

Figure 7.4: The increase of w_2

Thus, instead of reducing a level B at each step, the proposed algorithm increases a single weight. Although the value of B is invariable during the search, this technique is considered as a multiobjective extension of the Great Deluge algorithm because it incorporates the same principles.

In order to force the current solutions to follow the given trajectory the rules were developed for selecting the weight to be increased. The rules are illustrated for the bi-criteria case in Figure 7.5.

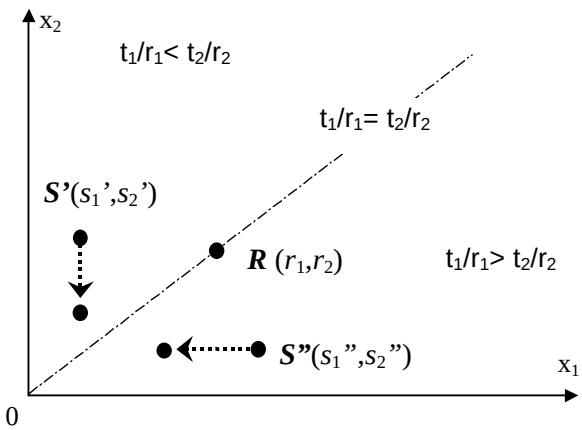

Figure 7.5: The selection of increased weight

Every point (t_1, t_2) on the trajectory which is drawn through the reference point R=(r_1, r_2) and the origin (presented by a dash-dotted line in Figure 7.5) satisfies the following equation

$$\frac{t_1}{r_1} = \frac{t_2}{r_2}. \tag{7.7}$$

Thus, the trajectory divides the criteria space into two halves: one where $t_1 / r_1 < t_2 / r_2$ and another where $t_1 / r_1 > t_2 / r_2$. Obviously, if the point S' (the current solution) is placed in the first half (above the trajectory), the search will be directed towards the trajectory by decreasing s_2' (it implies increasing w_2). Similarly, if the point S'' is placed in another half (below the trajectory) we have to decrease s_1'' (increase w_1) to direct the search towards the trajectory.

The proposed rule can be expanded into a K-criteria space as well. Here, we define the vector (s_1 / r_1, s_2 / r_2, … , s_n / r_n) and find its maximum element s_m / r_m ($m \in \{1,...,K\}$). It determines the criterion whose value will be decreased (its weight will be increased). The pseudocode for the algorithm is given in Figure 7.6.

Set the reference solution $\boldsymbol{R} = (r_1, r_2, .., r_K)$
Set the randomly constructed initial solution $\boldsymbol{S}^0 = (s_1^0, s_2^0, .., s_K^0)$
Specify the initial weights $(w_1^0, w_2^0, .., w_K^0) = ?$
Calculate initial cost function $f(S) = s_1^0 w_1^0 + s_2^0 w_2^0 + \ldots + w_K^0 s_K^0$
Level $B = f(S)$
Specify the input parameter $\Delta w = ?$
While not stopping condition do
  Define neighbourhood $N(S)$
  Randomly select the candidate solution $S^* \in N(S)$
  If $(f(S^*) \leq f(S))$ *or* $(f(S^*) \leq B)$
    Then accept candidate $S = S^*$
  Find m *correspondent to:* $\max_{i=1\ldots K} (s_i / r_i)$
  Increase the weight $w_m = w_m (1 + \Delta w)$

Figure 7.6: The multiobjective Great Deluge algorithm with variable weights

In this algorithm the value of the input parameter Δw affects the computing time in the sense that greater values of Δw cause the faster search. However, in contrast to the basic single-objective variant of the Great Deluge algorithm, the search speed is not steady and therefore, convergence in the given number of iterations cannot be guaranteed.

Apart from Δw, this algorithm requires the specification of initial weights $(w_1^0, w_2^0, \ldots, w_K^0)$. They determine the angle of an initial borderline, which passes through the initial solution. During experimental tests of different methods of weight initialisation it was found that they affect the duration of the first phase of the search: proper definition of initial weights allows the current solution to reach the trajectory more expeditiously. The best values of initial weights are probably problem-dependent. However, in the following experiments a fairly good performance was achieved when setting w_i^0 equal to

s_i^0/r_i , i=(1…K). If the value of some reference criterion r_i is equal to 0 (the correspondent constraint in the reference solution is satisfied), then the algorithm operates, instead of r_i, with some small value which is less than half of the measurement unit of the criterion. For example, when a criterion has an integer value, it is enough to set r_i to be less than 0.5.

7.3.2 Investigation of Properties of Great Deluge with Variable Weights

This section discusses the properties of the variable weights Great Deluge algorithm. In the first series of experiments this algorithm was tested on the bi-criteria case of the Nott-94 exam timetabling problem. The problem formulation was the same as the one given in Section 6.2: the first objective represents the number of conflicts where students have to sit two exams in adjacent periods, and the second objective represents the number of conflicts where students have exams in overnight adjacent periods.

The aim of the first experiment was to investigate the ability of the algorithm to follow the defined trajectory. Both reference criteria values were specified to be equal to 300. This means that the trajectory is a 45^0 angled line (dash-dotted line in Figure 7.7). In addition Δw was set to be 10^{-6} (which led to a processing time of around 3 minutes). In order to follow the progress of the search process, after each 50 000 steps the current solution was plotted as a dot in the criteria space, and after each 500 000 steps the current borderline is drawn as a dotted line. The complete diagram is presented in Figure 7.7.

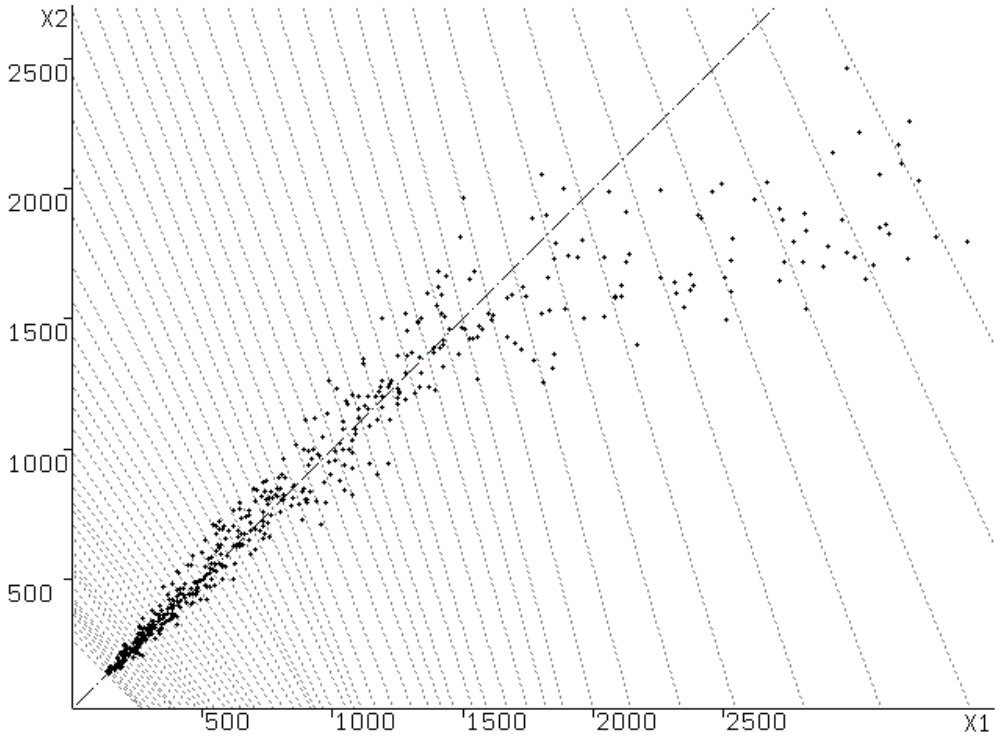

Figure 7.7: The progress diagram for the Nott-94 problem

Starting from a random solution the search is first directed towards the trajectory and then follows it producing solutions, which are very close to it. The "scatter" is relatively high at the beginning of the search and then becomes very low. Looking at the dynamics of the borderline it should be noticed that the interval between borderlines becomes shorter to the end of the search. This means that the improvement of the current solutions is unsteady and the search time cannot be predefined in advance.

The purpose of the next experiment is to show that this algorithm inherits the main property of its basic single-objective variant, i.e. that the longer search yields higher quality results. This algorithm does not allow setting up of the search time in advance but it can be varied it by specifying different values of Δw (even if their relationship is not well-defined). In this

experiment, the algorithm was launched a number of times through the same trajectory as in the previous experiment while varying Δw within interval $[10^{-4},\ldots,10^{-7}]$. Formally speaking the results of such an experiment should be represented in the form of a 3-D diagram. The representation can be simplified (while converting the diagram into the 2-D form) by using the fact that all solutions, which belong to such a trajectory (which lies under the angle of 45^0 to the criteria axes) have approximately the same values of both criteria. Thus, these values can be displayed on the same axis of a time-cost diagram. This diagram is shown in Figure 7.8 where the values of both criteria are displayed on the vertical axis and the search time (number of moves) on the horizontal axis. Each point in this diagram represents the result of the separate launch of the algorithm.

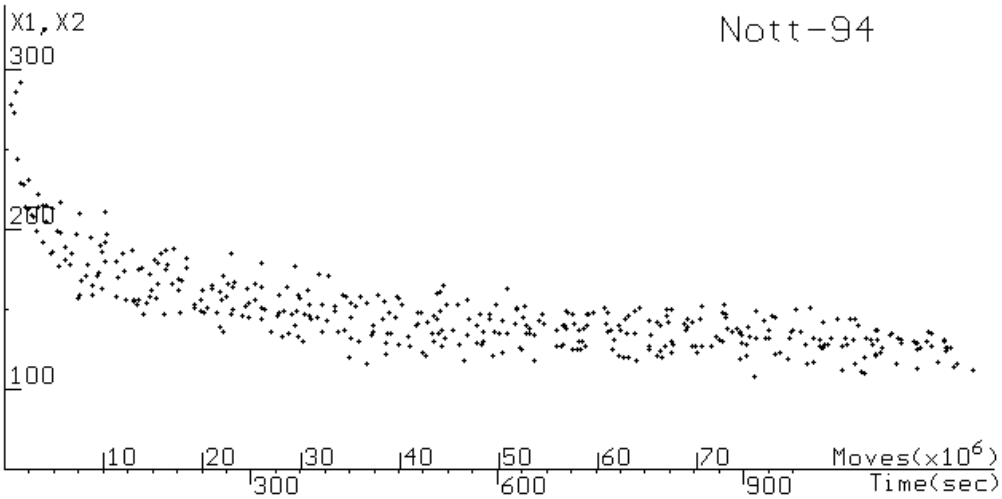

Figure 7.8: Time-cost diagram of Great Deluge algorithm with variable weights

The distribution of points in this diagram has the same shape as for the single-objective Great Deluge algorithm (see Section 5.5). The quality of the final results is relatively poor for the short runs and becomes better with the

prolongation of the processing time. This can be considered as a strong reason to incorporate the time-predefinition feature into this algorithm (this will be described in Section 7.4) in order to enable the decision maker to choose a longer search if the quality of solution is crucial or a shorter search, if a result of an average quality has to be produced quickly. Moreover, the processing time can be thought of as an additional criterion, which could also be taken into consideration by the decision maker.

This diagram can also clarify the question about the limitations of the presented algorithm, i.e. whether the search converges before or after reaching the reference solution. The obvious answer is that the time-cost curve represents the minimum criteria values of the reference solution reachable in the given time. For example, the reference point used in the first experiment (300,300) was reached and improved by any run presented in this diagram. However, in order to reach and improve the solution (150,150) the search should last longer than 300 seconds.

7.3.3 Experiments with Reference Points

The next series of experiments were carried out using, as reference points, a number of high quality solutions produced by other either single or multiobjective methods. Firstly, it was done with the results produced by the Multi-Stage Memetic Algorithm presented in [BN99], where the cost function is the weighted sum expressed by formula (5.7). Three datasets were used in the experiments: CAR-F-92, KFU-S-93 and Nott-94. Among the best published results for each dataset three non-dominated solutions were selected

(which were produced using different parameters values) to be the reference ones (9 reference points in total). For all reference points, the corresponding trajectories were drawn and the proposed algorithm was launched through every trajectory starting from random solutions. The results are shown in Table 7.1 where X_1 and X_2 present criteria values of solutions obtained by Multi-Stage Memetic Algorithm (MSMA) and the trajectory-based approach (TBA).

Table 7.1: Reference and produced solutions for the bi-criteria case

		CAR-F-92 (36 periods)		**KFU-S-93 (21 periods)**		**Nott-94 (23 periods)**	
		MSMA	TBA	MSMA	TBA	MSMA	TBA
1st point	X_1	302	**282**	222	**204**	65	**53**
	X_2	804	**799**	838	**743**	324	**271**
2nd point	X_1	313	**286**	228	**218**	76	**57**
	X_2	766	**706**	704	**608**	282	**187**
3rd point	X_1	363	**327**	307	**258**	100	**59**
	X_2	576	**541**	589	**562**	255	**149**

All produced final results dominate the reference points. This confirms the ability of the proposed algorithm to drive the search through different trajectories, and to produce high quality solutions, which are better than the reference ones.

Each run of the trajectory-based algorithm lasted around 30-40 minutes, and the major part of this time was spent on approaching the reference point (approximately 95% of the total time). This happened because for the given problem instances the distances between initial solutions and reference points were much longer (more than 10 times) than the distances between the

reference points and the origin (see for example in Figure 7.7). Although, only a minor part of the search was spent on the actual improvement of the reference criteria values, it was of crucial importance for the algorithm to traverse the part of the trajectory between initial and reference solution slowly. Only in this case could the algorithm provide good final solutions (which dominate the reference points) and therefore the time spent on approaching the reference point is justified.

The series of experiments were carried out in order to investigate the effectiveness of the proposed technique when the number of criteria is greater than two. The Nott-94 dataset was considered with 9 objectives which are defined in Section 6.1. Again, the solutions presented in that section are used as reference points. The Great Deluge algorithm with variable weights was launched for each of these reference points. All the launches lasted approximately 20-25 minutes. The results are compiled in Table 7.2.

Table 7.2: Reference and produced solutions for nine-criteria case

		23 periods		**26 periods**		**29 periods**		**32 periods**	
		CPA	TBA	CPA	TBA	CPA	TBA	CPA	TBA
1st point	X_1	1038	**795**	137	**0**	139	**0**	25	**0**
	X_2	1111	**651**	655	**476**	513	**360**	314	**184**
	X_3	3518	**3360**	2814	**2795**	2239	**2059**	1546	**1353**
	X_4	4804	**4185**	2759	**2494**	2172	**1687**	1646	**1390**
	X_5	405	**54**	265	**45**	231	**43**	174	**104**
	X_6	4	**0**	0	**0**	0	**0**	0	**0**
	X_7	0	**0**	0	**0**	0	**0**	0	**0**
	X_8	0	**0**	0	**0**	0	**0**	0	**0**
	X_9	0	**0**	0	**0**	0	**0**	0	**0**
2nd point	X_1	0	**0**	0	**0**	0	**0**	0	**0**
	X_2	879	**778**	604	**353**	393	**292**	316	**190**
	X_3	3623	**3524**	2544	**2174**	1957	**1482**	1332	**1104**
	X_4	6381	**6221**	4571	**3661**	3438	**2518**	2482	**2028**
	X_5	264	**152**	164	**38**	151	**48**	53	**2**
	X_6	0	**0**	0	**0**	0	**0**	0	**0**
	X_7	0	**0**	0	**0**	0	**0**	0	**0**
	X_8	0	**0**	0	**0**	0	**0**	0	**0**
	X_9	0	**0**	0	**0**	0	**0**	0	**0**
3rd point	X_1	2848	**1734**	2044	**889**	1559	**670**	1243	**1**
	X_2	2608	**1367**	1872	**802**	1435	**703**	1138	**488**
	X_3	4886	**3760**	3507	**2127**	2688	**1481**	2132	**1210**
	X_4	4658	**2289**	3343	**1922**	2563	**1201**	2033	**1073**
	X_5	807	**332**	475	**190**	441	**128**	334	**155**
	X_6	170	**0**	119	**0**	89	**0**	74	**0**
	X_7	40	**0**	24	**0**	24	**0**	18	**0**
	X_8	0	**0**	0	**0**	0	**0**	0	**0**
	X_9	0	**0**	0	**0**	0	**0**	0	**0**

As in the previous experiments the presented algorithm (TBA) has produced the solutions, which dominate the reference ones (CPA) by all criteria.

7.3.4 Evaluation of a Manageability of the Reference Point Method

The evaluation of properties of the reference point approach was extended with a small pilot test regarding a manageability of the proposed technique. In the course of this thesis the properties of two A Priori multiobjective techniques were compared, namely:

- the Great Deluge algorithm with conventional weighted sum aggregation function;
- the reference point approach introduced in this chapter.

The test was conducted on Nott-94 benchmark problem with bi-criteria case (the specification of criteria was the same as in the previous section).

The test was organised in the following way: eight researchers and PhD students, who were familiar with multiobjective optimisation or/and timetabling were asked to imagine themselves in the role of the timetabling officers, who have to produce an exam timetable. They were asked to use specially developed software which allowed the application of either of the compared techniques starting from the same initial solution and being launched for approximately the same time interval. The participants were able to make any number of runs (while varying input parameters) of compared techniques in order to formulate an opinion of how comfortable the techniques were to use.

Opinions were collected in the form of questionnaire comprising two questions, the answers of which were represented by marks (in the range from 1 to 5). The questions and possible answers were the following:

Question 1. How difficult it was to express the preferences in a numerical form (as weights or reference points)? The marks represented the following answers:

1. Definitely difficult;
2. It seems to be difficult;
3. Between difficult and easy;
4. It seems to be easy;
5. Definitely easy.

Question 2. How well the produced results conformed to the expectations about future solutions? The assigned marks corresponded to the following:

1. Definitely contradicted;
2. It seems that contradicted;
3. Between conformed and contradicted;
4. It seems that conformed;
5. Definitely conformed.

Eight responses were collected, which marks are shown in Table 7.3. The marks, labelled as "WS" correspond to the weighted sum technique, and ones, labelled as "RP" – to the reference point method.

Table 7.3: Results of a questionnaire

Question	Method	Marks								Average mark
1	WS	3	2	5	2	5	4	3	4	3.5
	RP	4	5	5	4	5	4	3	4	4.3
2	WS	3	3	4	3	4	3	2	3	3.1
	RP	5	4.5	5	4	3	5	4	5	4.4

The collected data is not sufficient to carry out a detailed statistical analysis and make definitive conclusions. For illustration purposes, the average values

of marks are shown in the last column of Table 7.3. However, from this preliminary evaluation, it would appear that the proposed technique is worthy of attention as a realistic multiobjective optimisation tool.

7.4 An Enhanced Trajectory-Based Multiobjective Optimisation Technique

The aim of the algorithm presented in the previous section was to produce a solution, which improves (more or less) proportionally all the criteria values of the reference point. The presented algorithm has two weaknesses. It does not allow a higher improvement of a selected criterion (or criteria) than the other one(s). Besides this, it does not allow for prediction of the search time.

The decision maker can improve some of the reference criteria values more than the others when orienting the trajectory into directions which are different from the origin. Formally speaking, the definition of a line in K-dimensional space can be done in different ways (e.g. by specifying the angle or deriving the system of linear equations) and all of them are potentially useful in the trajectory-based approach. However, the author believes that the most transparent variant of such a definition is the specification of two points through which the necessary trajectory should be drawn. As a second point to be used in addition to the reference one, it can be suggested the current solution (at the start of the search it is equivalent to the initial one). Here the purpose of the reference solution can be quite flexible. It can be used in the same way as in previously described strategy (where the goal is to reach it and to continue the improvement of criteria values following the same trajectory). In addition it can be the target solution (which should be just achieved) or serve as the ideal

point (in order to approach it as much as possible while keeping the given trajectory). The different strategies, which help to express the decision maker's preferences in terms of reference points are discussed in Sections 7.5 and 7.6.

In this section an enhanced trajectory-based multiobjective Great Deluge algorithm is presented. Here the trajectory can be drawn from a current (initial) point into some reference point placed in the sector of dominance (sector where all points dominate the given one). The movements of the current solutions through the trajectory are steady, which allows the algorithm to traverse the segment between two points in a specified number of moves.

7.4.1 The Description of the Method for the Bi-Objective Case

The enhanced trajectory-based algorithm is illustrated using a bi-objective problem. Its geometric interpretation is given in Figure 7.9. Let S^0R be a defined trajectory determined by an initial solution S^0 and a reference solution R. It is represented by a dash-dotted line in Figure 7.9. At each iteration the current solution S with the criteria values (coordinates) s_1 and s_2 should be moved into a new position $S^*(s_1^*,s_2^*)$, which dominates the solution S and at the same time is closer to the trajectory.

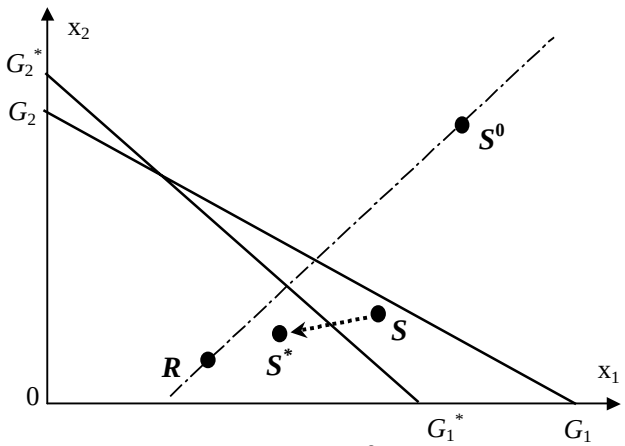

Figure 7.9: Relocation of the borderline

In contrast to the technique described in Section 7.3.1 in this algorithm the relocation of the borderline G_1G_2 into the new position $G_1^*G_2^*$ cannot be defined by increasing a weight of a single criterion. The equation of the new borderline contains new weights for both criteria and a new acceptance level for the weighted sum of criteria values. Namely, the new solution has to satisfy the following condition

$$s_1^* w_1^* + s_2^* w_2^* \leq B^* . \tag{7.8}$$

The task is to calculate values for w_1^*, w_2^* and B^* which will lead to the new borderline $G_1^*G_2^*$. In order to achieve this, two atomic transformations are carried out on the initial borderline G_1G_2.

- *Rotation*, which changes the slope of the initial borderline producing a new line $G_1'G_2'$ (see Figure 7.10). It determines the values of w_1^* and w_2^*.
- *Parallel shift* of the line $G_1'G_2'$ which moves it into position $G_1^*G_2^*$ (depicted in Figure 7.11). It determines the value of B^*.

These two atomic transformations are explained in more detail below.

The first decision to make is to choose the direction of the rotation of the initial borderline G_1G_2. In the example in Figure 7.9 the current solution is placed below the trajectory. Therefore the borderline should be rotated clockwise so that the new solution S^* is closer to the trajectory. This means that the value of x_1 will be decreased more than the value of x_2. If the current solution is placed above the trajectory, the rotation should be directed anti-clockwise.

All points, which belong to the trajectory S^0R satisfy the following equation

$$\frac{x_1 - r_1}{s_1^0 - r_1} = \frac{x_2 - r_2}{s_2^0 - r_2}. \tag{7.9}$$

Consequently the points below the trajectory satisfy the condition given by inequality

$$\frac{x_1 - r_1}{s_1^0 - r_1} > \frac{x_2 - r_2}{s_2^0 - r_2}. \tag{7.10}$$

If condition (7.10) is satisfied, then a clockwise rotation of the borderline is performed. If the opposite inequality holds, then the anti-clockwise rotation takes place. If the current solution is placed exactly on the trajectory, no rotation will be performed. However, this is unlikely to happen when dealing with problems with integer values of objectives, because trajectories generally consist of points which have a real number for at least one coordinate.

Formula (7.10) is correct when both right and left denominators have positive values. This is guaranteed when the point R dominates the point S^0 (the trajectory is drawn in the sector of dominance). Note that the trajectories must not be drawn perpendicular to the axes. This case presupposes no improvement of the corresponding objective and turns the problem into the single-objective form. Therefore, the denominators in (7.10) cannot have zero values.

Once the direction of the rotation is determined, the new values of weights w_1^* and w_2^* have to be calculated. The procedure for the clockwise rotation is illustrated in Figure 7.10. It is considered the rotation around the

point of intersection of the borderline and the trajectory. The point is denoted by $T=(t_1,t_2)$. The line G_1G_2 is rotated and a new line $G_1'G_2'$ is produced.

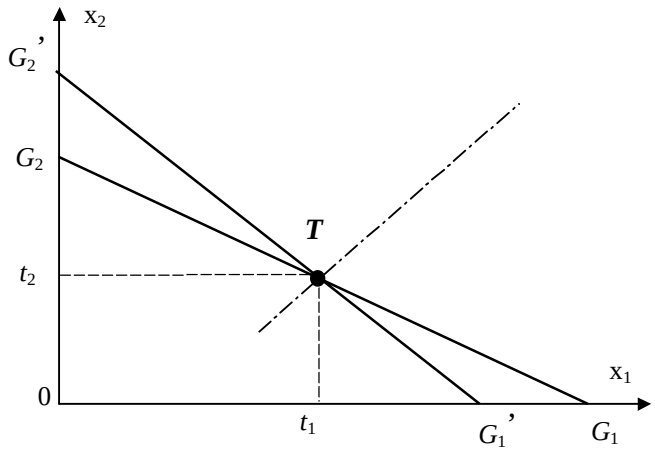

Figure 7.10: Rotation of a borderline

The angle of rotation can be defined in different ways. However, the angle has a significant impact on the performance of the algorithm. For example, the use of the constant angle makes the algorithm dependent on the criteria scale. The author have found that a fairly good performance is provided by the rotation of the borderline in such a way that the ratio of weights is decreased by the constant rotation rate λ :

$$\frac{w_2^*}{w_1^*} = \frac{w_2}{w_1}(1-\lambda)\,. \tag{7.11}$$

The rotation rate λ should take as values some small numbers (in further experiments $\lambda \in [10^{-6}...10^{-4}]$). It could be noticed that too small values of λ cause imprecise following of the trajectory while too large values make the algorithm unstable. However, its influence on the behaviour of the algorithm needs further investigation.

The formulae for new values of weights (w_1^* and w_2^*) are denoted below. Triangles given in Figure 7.10 lead to following relations:

$$\frac{G_1}{G_2} = \frac{G_1 - t_1}{t_2}, \qquad \frac{G_1^{'}}{G_2^{'}} = \frac{G_1^{'} - t_1}{t_2}. \tag{7.12}$$

Taking into account that intersection points are defined as: $G_i=B/w_i$ and $G_i^{'}=B/w_i^*$. (for i=1,2) we can substitute the weights in (7.11) and transform it into the following expression

$$\frac{G_1^{'}}{G_2^{'}} = \frac{G_1}{G_2}(1-\lambda). \tag{7.13}$$

Formulae (7.13) and (7.12) give the following formula

$$G_1^{'} - t_1 = (G_1 - t_1)\cdot(1-\lambda). \tag{7.14}$$

In order to find the value of w_1^* using the given above expressions for G_1 and $G_1^{'}$ equation (7.14) can be transformed into

$$w_1^* = \frac{B}{\frac{B}{w_1}(1-\lambda) + \lambda t_1}. \tag{7.15}$$

Weight w_2^* is calculated from expression (7.11) as:

$$w_2^* = w_2 \frac{w_1^*}{w_1}(1-\lambda). \tag{7.16}$$

Correspondingly, the rotating of the borderline anti-clockwise yields the expressions

$$w_2^* = \frac{B}{\frac{B}{w_2}(1-\lambda)+\lambda t_2}, \tag{7.17}$$

$$w_1^* = w_1 \frac{w_2^*}{w_2}(1-\lambda).$$

After rotation, the obtained borderline $G_1'G_2'$ is shifted parallel into new position $G_1^*G_2^*$. The shifting is illustrated in Figure 7.11.

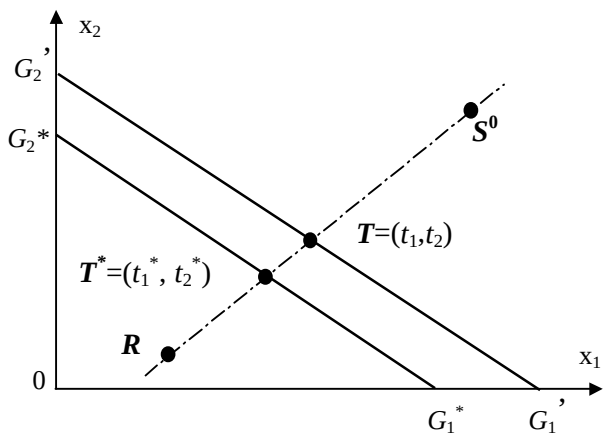

Figure 7.11: Shift of the borderline

In order to calculate the value for B^* we use equation (7.2). If we consider that the new borderline intersects the trajectory in the point $T^*=(t_1^*,t_2^*)$, the value of B^* can be found by

$$B^* = t_1^* w_1^* + t_2^* w_2^*. \tag{7.18}$$

The intersection point T^* can be determined using the following reasoning. During the search at each iteration the intersection point between the borderline and the trajectory is moved from T into T^*. The initial intersection point, in the first iteration, is the initial solution $S^0=(s_1^0,s_2^0)$ while the last intersection point is the reference solution $R=(r_1,r_2)$. Following the idea of time-predefined Great Deluge algorithm we can organise the search in order to

have the constant shift of the borderline $|T\ T^*| = \Delta T$ and guarantee to traverse the segment $|S^0R|$ in defined number of moves N_{mov}. Here ΔT is analogous to the decay rate in the single-objective Great Deluge algorithm and can be expressed as follows

$$\Delta T = \frac{|S^0 R|}{N_{mov}} = \frac{\sqrt{\left(s_1^0 - r_1\right)^2 + \left(s_2^0 - r_2\right)^2}}{N_{mov}}. \tag{7.19}$$

The coordinates of the point T^* at each iteration can be calculated based on the given value ΔT by expressions

$$\begin{aligned} t_1^* &= t_1 - \Delta T \frac{(s_1^0 - r_1)}{|S^0 R|}, \\ t_2^* &= t_2 - \Delta T \frac{(s_2^0 - r_2)}{|S^0 R|}, \end{aligned} \tag{7.20}$$

or basing on the predefined number of moves by formulae

$$\begin{aligned} t_1^* &= t_1 - \frac{(s_1^0 - r_1)}{N_{mov}}, \\ t_2^* &= t_2 - \frac{(s_2^0 - r_2)}{N_{mov}}. \end{aligned} \tag{7.21}$$

The iterative use of formulae (7.20) or (7.21) requires the definition of the initial values of t_1 and t_2, which are equivalent to the coordinates of the initial solution S^0.

7.4.2 An Expansion into the Multiobjective Case

The described algorithm can be generalised to handle K-objective problems where K>2. Here instead of inequality (7.10) the position of the current

solution relative to the given trajectory can be evaluated by the following vector

$$\left(\frac{s_1 - r_1}{s_1^0 - r_1}, \frac{s_2 - r_2}{s_2^0 - r_2}, \ldots, \frac{s_K - r_K}{s_K^0 - r_K}\right). \tag{7.22}$$

The elements of this vector are the projections of the current solution's criteria values ($s_1, s_2, \ldots, s_K$) on the trajectory. They can be considered as measures of the "distance" between the current criteria values and the trajectory. In order to approach the trajectory, the author suggests that the largest distance is decreased. Thus we find the maximum element of the vector (7.22), denote its index by m and update w_m by formula (7.23), which corresponds to (7.15).

$$w_m^* = \frac{B}{\frac{B}{w_m}(1-\lambda) + \lambda t_m}. \tag{7.23}$$

The remaining weights can be updated in different ways. In the given approach weight w_n of the criterion n, which corresponds to the minimum element of vector (7.22) is updated. The value of w_n is calculated by (7.24), which corresponds to (7.16).

$$w_n^* = w_n \frac{w_m^*}{w_m}(1-\lambda). \tag{7.24}$$

All the other weights remain the same ($w_i^* = w_i$, $i \neq m$, $i \neq n$).

Finally, the new elements of the vector T^* and the value of B^* are calculated by formulae

$$t_i^* = t_i - \frac{(s_i^0 - r_i)}{N_{mov}}, \tag{7.25}$$

$$B^* = \sum_{i=1}^{n} t_i^* w_i^* . \tag{7.26}$$

The pseudocode for the algorithm for *K*-criteria case is given in Figure 7.12.

Specify the number of moves $\boldsymbol{N_{mov}}$ *= ?*
Specify the initial weights $(\boldsymbol{w_1^0, w_2^0, .., w_K^0}) = ?$
Specify the rotation rate $\boldsymbol{\lambda} = ?$
Set the initial solution $\boldsymbol{S^0} = (\boldsymbol{s_1^0, s_2^0, .., s_K^0})$
Set the reference solution $\boldsymbol{R} = (\boldsymbol{r_1, r_2, .., r_K})$
Calculate initial cost function $\boldsymbol{f(S^0)} = \boldsymbol{s_1^0 w_1^0 + s_2^0 w_2^0 + \ldots + w_K^0 s_K^0}$
$\boldsymbol{T = S^0}$
$\boldsymbol{B = f(S^0)}$
$\boldsymbol{S = S^0}$
While not stopping condition do
  Define neighbourhood $\boldsymbol{N(S)}$
  Randomly select the candidate solution $\boldsymbol{S^*} \in \boldsymbol{N(S)}$
  If $(\boldsymbol{f(S^*)} \leq \boldsymbol{f(S)})$ *or* $(\boldsymbol{f(S^*)} \leq \boldsymbol{B})$
    Then accept candidate $\boldsymbol{S = S^*}$
  Find index $\boldsymbol{m}$ *correspondent to:* $\boldsymbol{max_{i=1...K}} (\boldsymbol{s_i - r_i / s_0 - r_i})$
  Find index $\boldsymbol{n}$ *correspondent to:* $\boldsymbol{min_{i=1...K}} (\boldsymbol{s_i - r_i / s_0 - r_i})$
  Calculate $\boldsymbol{w_m}$ *by formula (7.23)*
  Calculate $\boldsymbol{w_n}$ *by formula (7.24)*
  Calculate vector $\boldsymbol{T}$ *by formula (7.25)*
  Calculate $\boldsymbol{B}$ *by formula (7.26)*

Figure 7.12: The enhanced multiobjective Great Deluge algorithm.

This algorithm requires the specification of three input parameters:

The number of moves N_{mov}. This parameter indicates that the presented algorithm is time-predefined. It can be set up taking into account available computational resources and expected processing time.

The vector of initial weights (w_1^0,w_2^0, .., w_n^0). Contrary to the basic Great Deluge algorithm with variable weights, the initial values of the weights do not significantly affect the performance of this technique. It was found that in the beginning of the search they become tuned relatively quickly. In further experiments all initial weights were set to be equal to 1. Also, the changing of the trajectory does not require any additional update of the weights. A search in a new direction should be started with previous weights.

The rotation rate λ. The issue of tuning of this parameter requires further investigation. In further experiments its value was adjusted manually. However, a certain dependence exits (and should be studied in future) between λ and N_{mov}; namely larger N_{mov} requires smaller λ.

The presented algorithm also requires a specification of the initial and the reference solutions. They will be discussed in Section 7.5. The stopping condition can be defined in different ways, which are often used in local search techniques (for example, “no improvement during a given number of moves”).

7.4.3 Investigation of Dynamics of the Algorithm

The behavior of the enhanced variant of the trajectory-based technique was tested on the same benchmark problem as the basic variable weights Great Deluge algorithm (bi-objective variant of Nott-94 problem), whose problem statement was given in Section 7.3.2.

The experiment aimed to demonstrate the ability of the proposed technique to follow the defined trajectories. In addition, the author wants to show the flexibility of the given method, which enables decision maker to define different trajectories and to change the trajectory during the search. As an example, it was taken a trajectory, which consists of two branches (segments). The first branch was laid from initial solution into the point (700,1200). At this point the direction of the trajectory was changed and the second segment was directed into the point (300,0). The initial values of all weights were set up to be equal to 1 and the rotation rate $\lambda = 10^{-5}$ (its value was defined empirically by several launches of the algorithm). The shift of the borderline in the first segment was defined by (7.21) where the number of moves was set up to be $1.5*10^6$. Using this number, the value of ΔT was calculated by (7.19) and used to define the shift of the borderline in the second branch by formulae (7.20). In such a way the ΔT was kept constant during the whole search process. The algorithm started from a random solution and traversed the first segment of the trajectory in 23 seconds. After reaching the first reference point the search was redirected along the second branch. The second segment was not traversed completely because the search converged before reaching the second reference point. Each $10*10^3$ iterations a position of the current solution was plotted and each $100*10^3$ iterations the current borderline was drawn. The complete progress diagram of this search is shown in Figure 7.13 where the trajectories are drawn by dash-dotted lines.

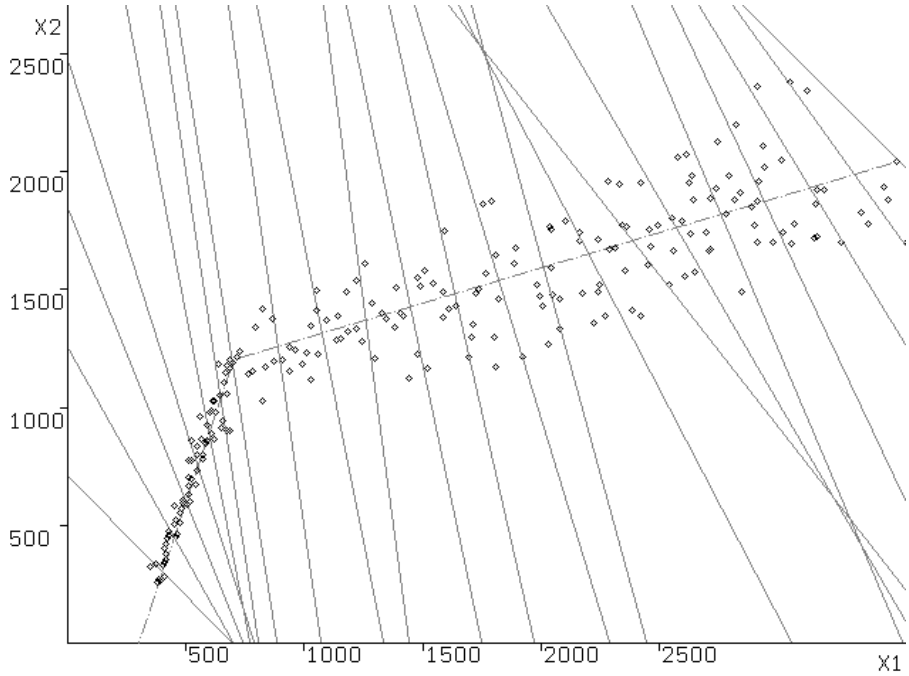

Figure 7.13: The algorithm follows a trajectory with two branches

In this diagram one can observe that the current solutions follow the trajectory in the same manner as was the case with the basic Great Deluge algorithm with variable weights. At the beginning it shows a relatively high "scatter", but towards the end of the search all current solutions are placed very close to the trajectory. It can be noticed that the rotations of the borderline depend on the current circumstances. However, in contrast to the first trajectory-based algorithm the points of intersection between the borderlines and the trajectory (defined as T in previous section) are placed on constant intervals on the trajectory (equal to $\Delta T \cdot 10^5$). This means that the search through the trajectory is conducted with constant speed.

7.5 A Fan Search Strategy

The enhanced multiobjective trajectory-based Great Deluge algorithm can be applied for the same purpose as the basic variant (reference point strategy).

However, its flexibility in the definition of the trajectory allows for the development of a wide range of more advanced strategies. The author presents here a strategy, which he named a “Fan Search” due to the shape of branches of its trajectories. It keeps the advantages of A Priori approaches and simultaneously allows the decision maker to express his/her preferences in such an easy way (as provided by A Posteriori methods).

7.5.1 A Description of the Strategy

The aim of this strategy is to improve the results produced by any Pareto-based technique. Initially a Pareto-based algorithm is applied to obtain a primary surface of non-dominated solutions. The decision maker chooses the most preferable (reference) solution from the obtained set. In this way, the decision maker implicitly expresses his/her preferences. Our aim is to improve the reference solution but in contrast to the previous strategy (described in Section 7.2) the improvement here is carried out in different directions in the criteria space having the form of a fan shape. In other words, the algorithm produces a secondary set of solutions, all of which dominate the chosen one. Again, as in previously described strategy, it is not suggested the reference solution to be taken as a starting point for further improvement, because it is assumed that it already lies in a local minimum.

The graphical representation of the Fan Search strategy is shown in Figure 7.14. The gray points indicate a primary trade-off surface where the chosen reference point is marked with *R*. The proposed technique consists of two phases. In the first phase the initial trajectory is determined by a randomly

generated initial solution *I* and the reference solution *R*. The search is carried out until the borderline reaches the point *R*. If the search is slow enough, then at this moment the current solution is situated in the vicinity of the reference one. This current solution is memorized in order to be used as a starting point for the second phase.

In the second phase several trajectories are drawn from the memorised point into different directions with the aim of producing solutions which dominate the memorised one. These trajectories are branched in such a way as to form a fan shape, dividing the sector of dominance into equal parts. The decision maker decides on the number of branches and angles between them (the angles between the branches are the same to have equal parts in the sector of dominance). The secondary set of solutions marked with **F** comprises the convergence points of each of the searches along the branches (coloured black in Figure 7.14). All these points dominate the reference one, and the decision maker can choose the most preferable one as a final solution.

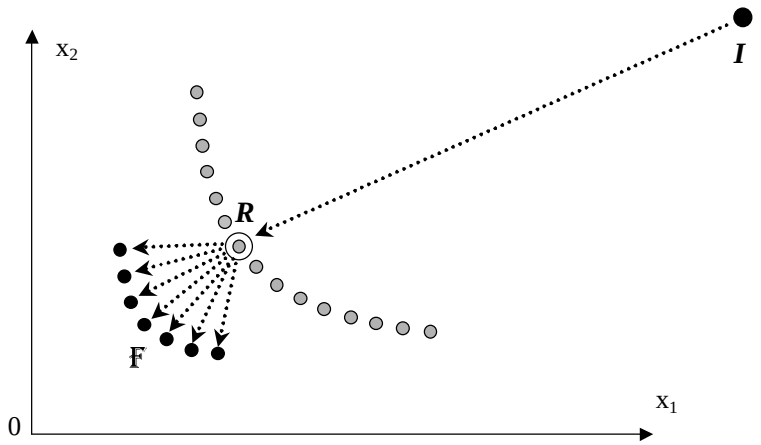

Figure 7.14: The Fan Search strategy

Taking into account that the search is conducted with constant ΔT, the processing time is proportional to the length of the trajectory. The relative lengths of the first part of the trajectory and the consequent branches are dependent on the positions of the initial and the reference points. However, in the discussion about experiments on real-world exam timetabling problems presented in Section 7.3.3 it was mentioned that the search spent the vast majority of time (95%) on approaching the reference point. Thus (in real-world situations) the fan branches are relatively short and launching the search through a number of branches does not increase the total processing time significantly.

The author considers the described property as one of the major advantages of this algorithm. While most of the population-based techniques can produce a set of solutions in the time equal to the time spent on one solution multiplied by the population size, the Fan Search does it substantially faster.

7.5.2 Testing the Fan Search Strategy

This series of experiments aims to test the ability of the Fan Search strategy to produce the secondary trade-off surfaces. In addition, the dependence of the quality of the final solutions on the search speed was investigated. The problem instance that was used in experiments was the one given in Section 7.4.3. Firstly three experiments were carried out which used the same reference point R=(500,500) but comprised a different number of moves defined for the first branch and different values of rotation rate. The values of these parameters are

shown in Table 7.3. These three different search environments produced three trade-off surfaces depicted in Figure 7.15. The total processing time of each run is also shown in Table 7.4.

Table 7.4: Parameters and processing times of the first series of experiments

Trade-off surface	1st	2nd	3rd
Reference point	r_1=500; r_2=500	r_1=500; r_2=500	r_1=500; r_2=500
N_{moves} **of a first part**	$2*10^6$	$20*10^6$	$200*10^6$
Rotation rate λ	$5*10^{-5}$	10^{-5}	$2*10^{-6}$
Processing time (sec)	96	1048	12490

At each experiment the search was conducted from the same random initial solution S^0 = (3719,1928) toward the same reference point (marked with *R* in Figure 7.15). When the borderline reached the reference point, the current solution was memorized in order to keep the starting solution for the consequent branches. In total 19 branches were defined in the following way: one branch was drawn toward the point (0,0); 9 branches toward the points $((r_1 / 10)\cdot i\ ,\ 0)$, where $i \in \{1\ldots9\}$ and 9 branches toward the points $(0\ ,\ (r_2/10)\cdot i)$, where $i \in \{1\ldots9\}$. In this way, the sector of dominance is divided into almost equal parts. The shift of the borderline in the branches was calculated in order to keep ΔT constant during the whole search process in the same way as in Section 7.4.3. Starting from the same memorised solution the search was launched through all these trajectories, which yielded 19 secondary solutions.

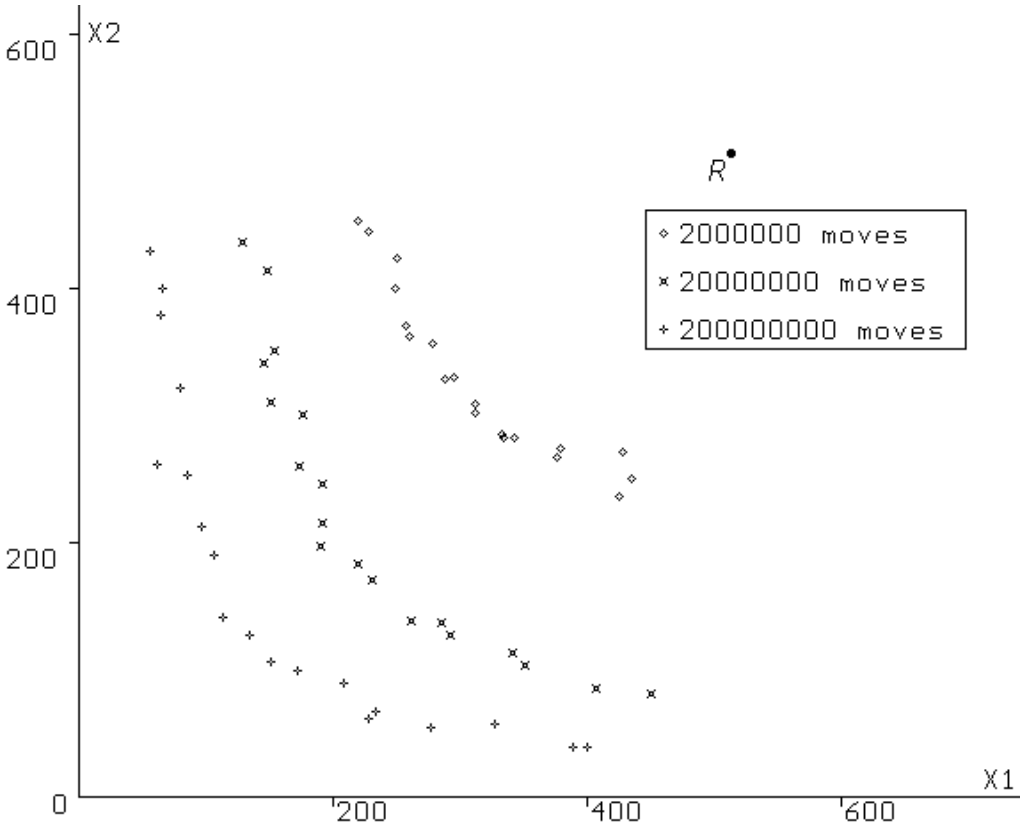

Figure 7.15: Trade-off surfaces produced by the Fan Search

This diagram shows three explicit trade-off surfaces. The slower searches produce the surfaces, which lie close to the origin. Every solution on the surface produced in shorter time is dominated by at least one solution on the higher time surface. Thus, if the decision maker is not satisfied by any of the solutions produced by the relatively fast Fan Search, then one of the secondary solutions can be used once more as a new reference point (*R*'). The next set of solutions can be produced by slower search, and so on, until no sensible improvement is obtained (Fig. 7.16).

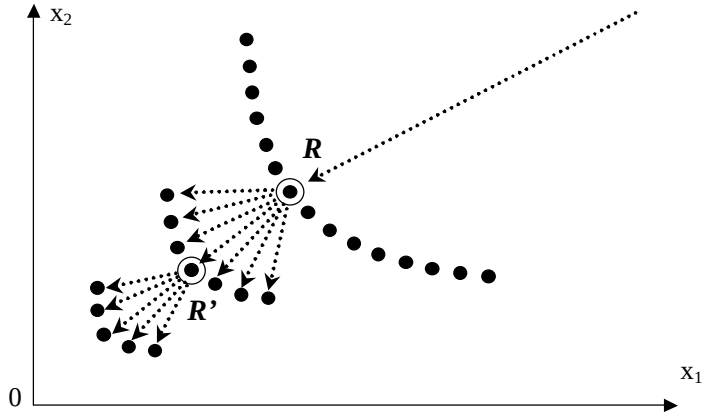

Figure 7.16: Further iterations of the Fan Search

This proposition was tested in the second series of experiments. The parameters and processing times are summarised in Table 7.5 and the resulting trade-off surfaces are depicted in Figure 7.17. All parameters of the algorithm were the same as in the previous experiments except reference points, which are marked in Figure 7.17 with *R*, *R'* and *R"*. The first trade-off surface (produced from the reference point *R*=(500,500) in $2*10^6$ moves) was taken from the previous series of experiments. One point *R'*=(335,283) was chosen from this set to be a reference one for the search performed in $20*10^6$ moves. When the second trade-off surface was produced, one of its points *R"*=(220,184) was once more selected and the algorithm was launched for $200*10^6$ moves using the selected point as a reference one.

Table 7.5: Parameters and processing time of the second series of experiments

Trade-off surface	1st	2nd	3rd
Reference point	r_1=500; r_2=500	r_1'=335; r_2'=283	r_1"=220; r_2"=184
N_{moves} **of a first part**	$2*10^6$	$20*10^6$	$200*10^6$
Rotation rate λ	$5*10^{-5}$	10^{-5}	$2*10^{-6}$
Processing time (sec)	96	732	5772

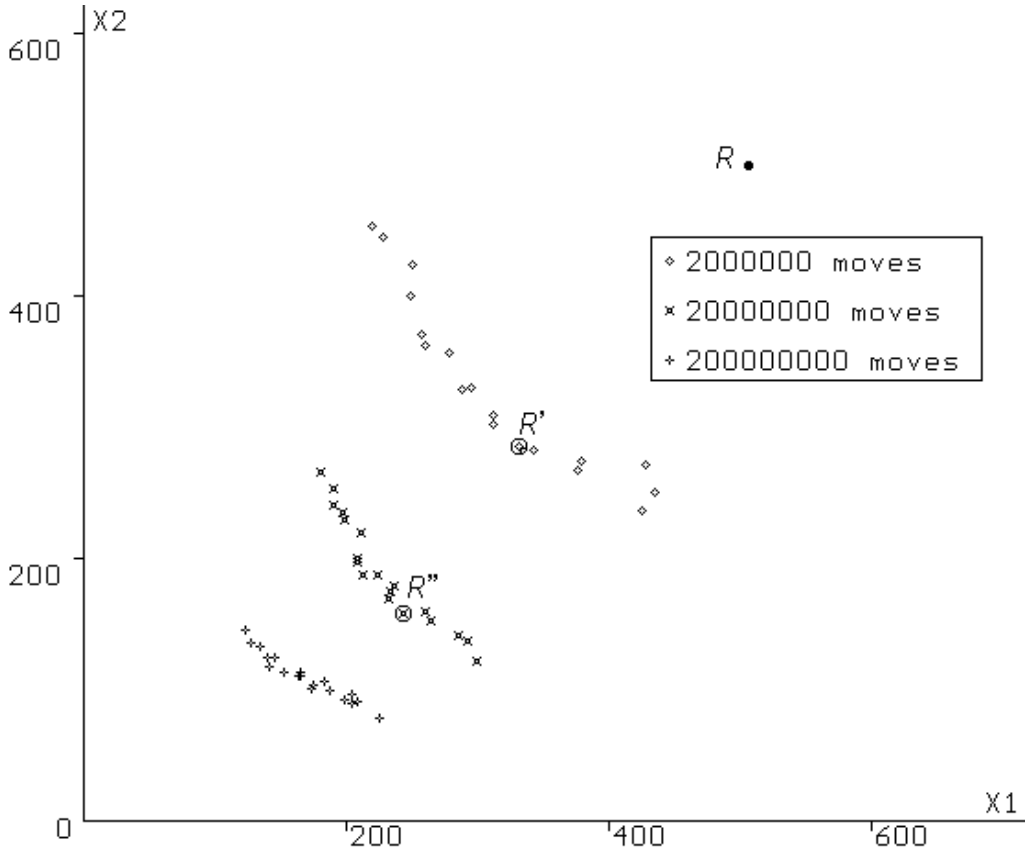

Figure 7.17: Trade-off surfaces produced by the Fan Search

In this diagram the second and third trade-off surfaces occupy a more narrow (than in Figure 7.15) sector of the criteria space because the corresponding reference points are placed close to the origin (this causes more a narrow sector of dominance). This also leads to a total time which is approximately twice as short as that of the total time from the previous experiments. However, one can see that the position of the reference point does not influence the quality of the achieved results. Both trade-off surfaces produced in the second series of experiments can be considered as just short segments of the ones produced in the first series of experiments.

7.5.3 Using a Reference Point Selected from PAES Result

The experiments are continued while using the Fan Search strategy for improving trade-off surfaces produced by PAES method applied to exam timetabling and described in Section 6.2.2. Both primary and secondary sets of solutions for Nott-94 problem are shown in Figure 7.18. Among the primary set (marked as "PAES") one point R= (336,397) is chosen to be the reference one and the algorithm is launched from a random initial solution. The number of moves for the first branch N_{mov}=10*10^6 and rotation rate λ=2*10^{-5}. The location of fan branches and their ΔT were defined in the same way as in the previous experiments. The produced set of secondary solutions is marked with "FSS" in Figure 7.18.

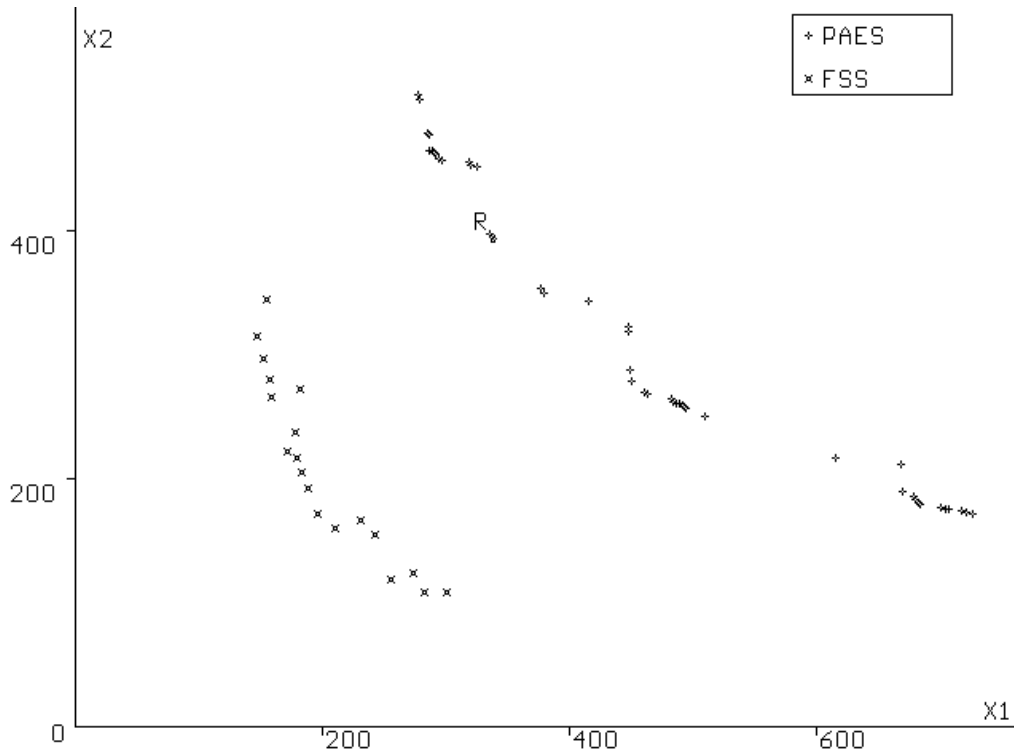

Figure 7.18: Trade-off surfaces produced by PAES and Fan Search.

Even though some of the secondary solutions dominate each other, they are distributed quite uniformly and almost completely cover the sector of dominance. Some solutions in the secondary set have both criteria values which are twice as good as the reference one. Other secondary solutions have approximately the same (as in reference solution) values of one criterion while 2-3 times better values of another one. One can see that every point of the first set is dominated by at least one point from the secondary set. Moreover, some of these solutions dominate all of the solutions produced by the PAES algorithm. Although the algorithm aimed to outperform only the reference point, it showed a complete outperformance over the whole set of solutions.

It should be also noticed that the first secondary solution was achieved in 325 sec, while the algorithm was completed in 772 sec. Thus, the processing time necessary to produce the set of 19 solutions (which cover the whole sector of dominance) was only 2.4 times longer than the time in which the algorithm can produce a single solution.

Additional experiments were performed on 13 timetabling problems from the University of Toronto archive. The definition of criteria and the algorithm's parameters were the same as in the previous experiments. As primary sets the results produced by PAES (in Section 6.2.2) were used among which the reference points were randomly chosen. The secondary sets were obtained using the Fan Search strategy. For all theser experiments the presented approach achieved a complete outperformance of the secondary sets over the primary ones.

The formal comparison of the quality of the primary and the secondary sets can be made using metric measures, discussed in Section 3.4. The properties of several metrics were examined and the *S*-metric was chosen for the comparison because it is independent of the true Pareto-front and the decision maker's preferences and provides a quantitative measure for the outperformance relation. In the bi-objective case it is computationally inexpensive and only requires the definition of one reference point (here the term "reference point" has different meaning from the one assumed in this chapter). In these calculations the coordinates of the *S*-metric reference point were considered to be equal to the maximum values of the correspondent criteria among the both compared sets. The resulting values of the *S*-metric are presented in Table 7.6. Here S_{PAES} denotes the *S*-metric of the trade-off surface, produced PAES algorithm; S_{FSS} is the *S*-metric of the produced secondary set.

Table 7.6: The comparison of *S*-metrics of primary and secondary sets

Data set	S_{PAES}	S_{FSS}	S_{FSS}/S_{PAES}
CAR-F-92	55485	**740581**	13.3
CAR-S-91	101609	**749320**	7.4
EAR-F-83	104891	**275234**	2.6
HEC-S-92	105555	**172594**	1.6
KFU-S-93	69355	**585263**	8.4
LSE-F-91	42411	**148976**	3.5
PUR-S-93	72741	**619391**	8.5
RYE-S-93	325646	**990056**	3.0
STA-F-83	78995	**97323**	1.2
TRE-S-92	51793	**301503**	5.8
UTA-S-92	62254	**410718**	6.6
UTE-S-92	70584	**398837**	5.7
YOR-F-83	50676	**188546**	3.7

This table shows that the Fan Search results outperform the PAES ones substantially on all datasets. The last column in the table shows the ratio of the obtained values of S-metric. It can be noticed that the least outperformance was achieved for the relatively small sized problems (e.g. STA-F-83 and HEC-S-92). On the other hand the highest outperformance was obtained for the largest problems (such as CAR-F-92 and PUR-S-93). This may indicate that the presented technique is especially useful for large-scale exam timetabling problems.

The computational time of the presented experiments was around 10-20 minutes. This time limitation was set because it is quite acceptable for

university exam timetabling. However, giving a longer acceptable time the time-predefined multiobjective algorithm could produce results of even higher quality. As an additional advantage of the presented technique, it should be noticed its simplicity in use. Only one parameter (λ) is not very straightforward (where PAES has two such parameters). All other input data of the proposed algorithm is quite transparent and does not require any efforts for tuning.

7.6 Further Possible Strategies for the Application of the Trajectory-Based Technique

The trajectory-based approach provides different opportunities for the decision maker to express his/her preferences while solving an optimisation problem. Two basic strategies were discussed and investigated in the previous sections of this chapter. In this section the author outlines some of further possible strategies of the application of the trajectory-based multiobjective approach. The aim is not to present a detailed study but to lay the ground for possible future research.

Following the principles of two described strategies the aim of further ones is to simplify the expression of the decision maker's preferences and simultaneously reduce the time expense required for reaching high quality preferable solutions. Here the useful information can be obtained by any other approach or by the short-time launches of the presented algorithm. Taking into account that the search time can be varied from several seconds to several hours, the main attention should be paid to the proper specification of the trajectories for the most time-expensive launches.

7.6.1 Approximation Strategies

The previously described strategies assume that a reference point corresponds to some known solution. However, if there is an insufficient number of known solutions or if they are not distributed uniformly then the decision maker can suppose that a number of unidentified solutions, which are placed between the known ones, could be (potentially) obtained and each of these solutions could be considered as an approximated reference point. An example of such an approximation for the two-criteria case is shown in Figure 7.19 where the trade-off surface comprises N solutions (black points) and has an unusually long interval between points i and i+1. If the decision maker considers that the solutions (1…i) have too high a value of the second criterion and the solutions (i+1…N) have too high a value of the first one then he/she might prefer some solution placed in the region between points i and i+1. In this situation these points can be connected with a line, any point of this line can be considered as a reference one (point R) and any trajectory can be drawn through it (dash-dotted lines).

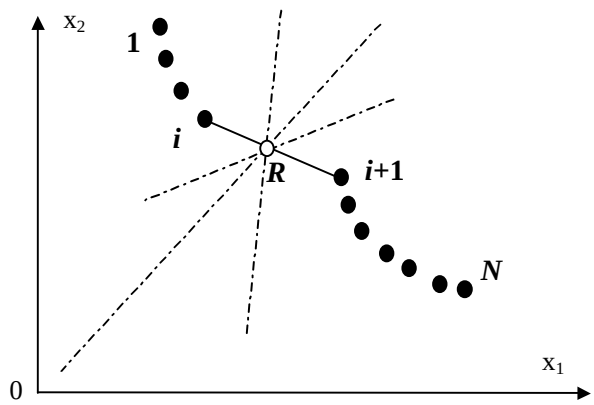

Figure 7.19: The approximation of a reference point

The approximation of another type can be made under an assumption that a secondary trade-off surface follows the shape of a primary one by a certain scale. At least for the used problem instance such an assumption is justified by the shapes of the surfaces in Figures 7.15 and 7.17. Thus, the expected secondary trade-off surface could be drawn as the dilation of the primary one and the approximated reference point could be chosen on this curve. This strategy can be used instead of the Fan Search in situations when the scale of possible dilation is known (for example, when the decision maker already acquired a single (or several) secondary solution(s) but prefers to obtain another one).

The idea of this method is explained by Figure 7.20 where the primary trade-off surface is depicted by black points. Let us assume that the decision maker also knows one solution (point S) and supposes that it belongs to the expected secondary trade-off surface (for example, this solution was produced by a relatively long search). The point S^* can be defined to be the projection of the point S onto the primary surface, while using the origin as a focal point. The scale is calculated as a ratio of distances: $|0S|/|0S^*|$. The criteria values of the primary points are multiplied by this scale to get an expected surface (grey curve). The decision maker can choose two reference points: R' from the primary set and R'' from the expected secondary set and launch (from initial solution I) the same procedure as the Fan Search but comprising only one secondary branch ($R'R''$). Obviously, in order to reach the expected secondary trade-off surface the speed of this launch should be the same as one which yielded solution S.

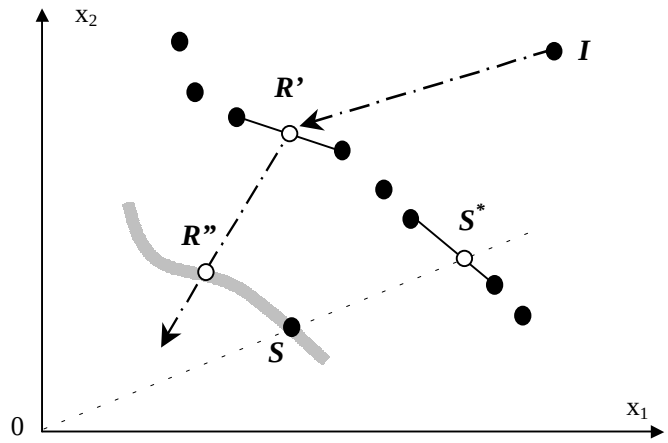

Figure 7.20: Approximation of a secondary trade-off surface

7.6.2 An Interactive Trajectory Assessment

The trajectory-based algorithms have a high potential for use in the interactive multiobjective approach. This approach allows the decision maker to achieve the solution, which precisely matches his/her preferences. It is also useful in the case when the decision maker would like to correct the previously expressed preferences after the search started (based on the analysis of the current information). One of the possible ideas of such an application is presented in this section. Like the Fan Search, this procedure also involves the restarting of the search from memorised solutions but enables the decision maker to control the direction of further branches.

The interactive trajectory assessment is based on the following idea. When the decision maker obtains a solution, he/she indicates a criterion, which should be reduced or increased. At the same time, the decision maker defines the expected amount of this reduction/increase. Using this information the algorithm calculates a new branch of the trajectory. To reserve starting points for the next branches the algorithm memorises intermediate solutions every

given number of iterations. The explanation of this procedure is given by one example, whose first phase is shown in Figure 7.21.

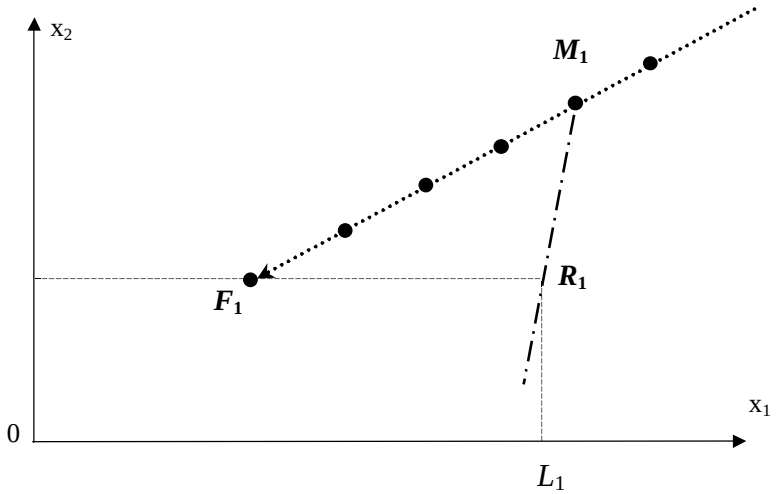

Figure 7.21: An interactive trajectory assessment (phase 1)

In the first phase an initial trajectory can be drawn into any specified point (in the given example it is the origin). During the search a number of intermediate solutions are memorised (black points). Let us assume that when the search converges to the point F_1, the decision maker is not satisfied by the acquired solution and prefers to increase x_1 up to the value L_1, which is acceptable for him/her (in order to improve x_2). The algorithm draws a new branch of the trajectory through the point R_1. The new starting point is selected among the memorised ones. This selection should meet two conditions:

- the starting point should be dominated by R_1 ;
- the new branch of the trajectory should be the shortest.

In Figure 7.20 this point is M_1. After the selection the consequent search is launched through the trajectory M_1R_1.

Thus, every phase of the interactive trajectory assessing comprises the following steps:

- obtaining the final solution of the previous branch (F_i);
- specifying the increase/decrease of a criterion (L_i);
- calculating the reference point (R_i);
- selecting the starting point (M_i);
- launching the algorithm through the new branch;

An example of the second phase of the search is given in Figure 7.22 (the black points represent available memorised solutions).

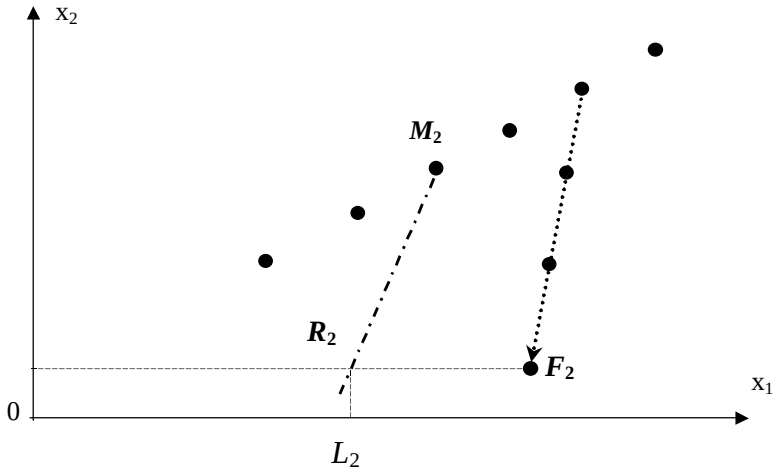

Figure 7.22: An interactive trajectory assessment (Phase 2)

The decision maker continues controlling the increase/decrease of criteria values of produced solutions until he/she is satisfied with them. The complete scheme of the process is represented by a tree, shown in Figure 7.23.

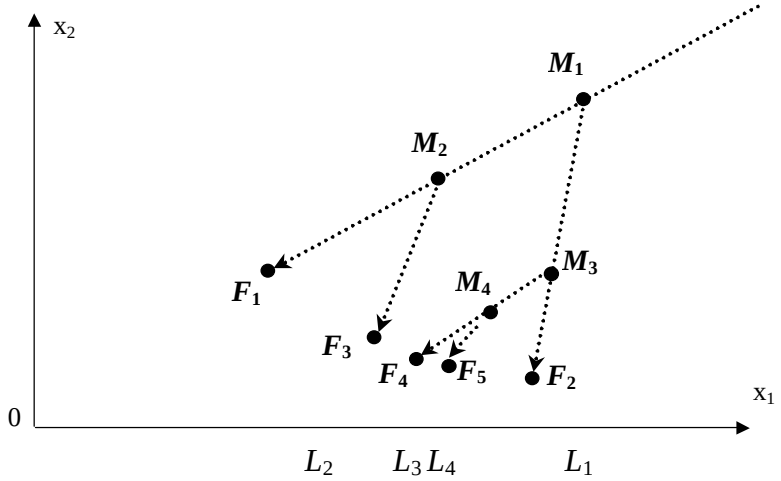

Figure 7.23: An interactive trajectory assessment (complete scheme)

In this scheme the length of each later branch is usually shorter than the previous ones. Therefore, the total length of the secondary branches (and correspondingly, a time expense) is relatively low.

The proposed approach provides to the decision maker a set of tools for real-time steering the search process. He/she can:

- Adjust the search speed.
- Regulate the search direction.
- Divide the search into several branches.

At any point the decision maker can temporarily suspend the search process, analyse the information about the overtaken path and choose new search parameters (the direction and the speed). He/she can also indicate the memorised points and later use them as initial points for a search in different directions.

The described strategies enable the decision maker to effectively explore the criteria space. For example, while suspending the search at some

point the decision maker can launch a high-speed fan search from the current solution in order to explore the forthcoming trade-off surface. Of course, such exploration requires a certain experience, but it may lead to the better satisfaction of the decision maker's preferences.

7.7 Conclusions

In this chapter the idea of trajectory-based multiobjective approach is presented. Here a local search algorithm drives the search of the criteria space following the trajectory assigned by the decision maker. This approach is principally different from other well-known multiobjective techniques.

There are presented two variants of the multiobjective extension of the Great Deluge algorithm. These algorithms use the weighted sum aggregation of criteria as an objective function and modify the aggregation function dynamically in order to drive a search through the given trajectory.

It was described the simple mechanism of a weights variation, which is able to drive the search through the straight line drawn from the origin to some reference point. In addition an enhanced algorithm was suggested, which conducts the search from the current point into any direction in the sector of dominance. This technique allows the reassigning of a trajectory during the search that provides an opportunity for different strategies of the application of the method.

Two strategies of the application of the trajectory-based approach were discussed. The first strategy requires the specification of a reference solution.

The author believes that such a specification may be quite transparent for the decision maker. The second strategy (the Fan Search) is suitable for improving the results from Pareto-based techniques. It provides the same flexibility for the decision maker as A Posteriori approaches.

Using these strategies, the comparison of both proposed algorithms with several existing multiobjective techniques was presented while solving university exam timetabling problems. It was shown that the proposed algorithms can produce results of higher quality (dominating by all criteria) than other approaches. It can be considered that the trajectory-based approach combines the power of aggregation methods and the transparency of Pareto-based ones.

Besides this, the presented experiments showed that the multiobjective extension of the Great Deluge algorithm inherits its main property and that a longer search yields better results. The enhanced trajectory-based algorithm allows the specification of the number of moves during the search and therefore employs the advantages of the idea of the time-predefinition.

A number of further strategies for navigation through the search space along the defined trajectories were also outlined. They provide a wide range of opportunities for the decision maker for obtaining the most preferable solutions. Of course, the list of possible strategies can be extended.

Chapter 8.
Conclusions

8.1 Summary of the Presented Approaches

The University Examination Timetabling Problem has attracted significant research interest over the years. Its various instances usually appear as large-scale, highly constrained and difficult to solve NP-hard problems. These problems are often varied in their structure, which can contribute to making them a difficult class of problems, offering some serious research challenges. From a practical point of view they are also very important problems. The quality of solutions to these problems often has a significant impact on the institutions concerned.

This thesis introduces new examination timetabling algorithms which consider computational expense as a major input parameter. In particular, it introduces an adapted version of the Great Deluge Algorithm for exam timetabling. This algorithm requires just two input parameters: computational time (that the user is willing to spend) and an approximation of the objective function value that would be desirable. This employment of just two input parameters represents a significant achievement in itself, particularly as the parameters are measures that real world users can readily understand (time and desired solution quality) rather than abstract concepts (such as number of generations in a Genetic Algorithm, or cooling rate for Simulated Annealing etc.).

The second contribution of this research is that an exhaustive investigation into the multiobjective nature of exam timetabling problems was carried out. It was presented and discussed a new multiobjective algorithm based on the idea of Compromise Programming and studied the performance of existing Pareto-based methods on exam timetabling problems.

In addition to this, the trajectory-based approach was introduced as an alternative to conventional multiobjective methods. The author's contribution in this area comprises two versions of trajectory-based local search algorithm based on the Great Deluge method and several strategies of their application in practice. The presented approach adheres to practice of employing straightforward parameters.

8.2 A Comparison of Performance of Different Methods

The results of the presented experiments on real-world exam timetabling problems demonstrate that the performance of a local search algorithm can be significantly improved by incorporating a controlled management of the processing time into the approach. Moreover, this behaviour is also characteristic to the proposed multiobjective techniques. This is evidenced by a series of presented comparisons of the performance of these algorithms on a range of well-known exam timetabling benchmark problems.

In this study the author developed 5 different algorithms and, in addition, applied 5 well-known algorithms to benchmark exam timetabling problems while considering single and multi-objective cases. A number of selective comparisons were made between the results produced in this research

and the ones published in the literature. All presented comparisons are shown in Figure 8.1, where the nodes contain the algorithms. The compared pairs of algorithms are connected by edges, where arrows indicate the superiority of results. Each arrow points to the algorithm, whose performance was found to be evidently better than another one. The arrow is not drawn for the pairs where the superiority is not evident (the performance is different for different problems). The algorithms whose results were taken from the literature are shown in the left part of the graph and are separated from the ones obtained in the course of this research.

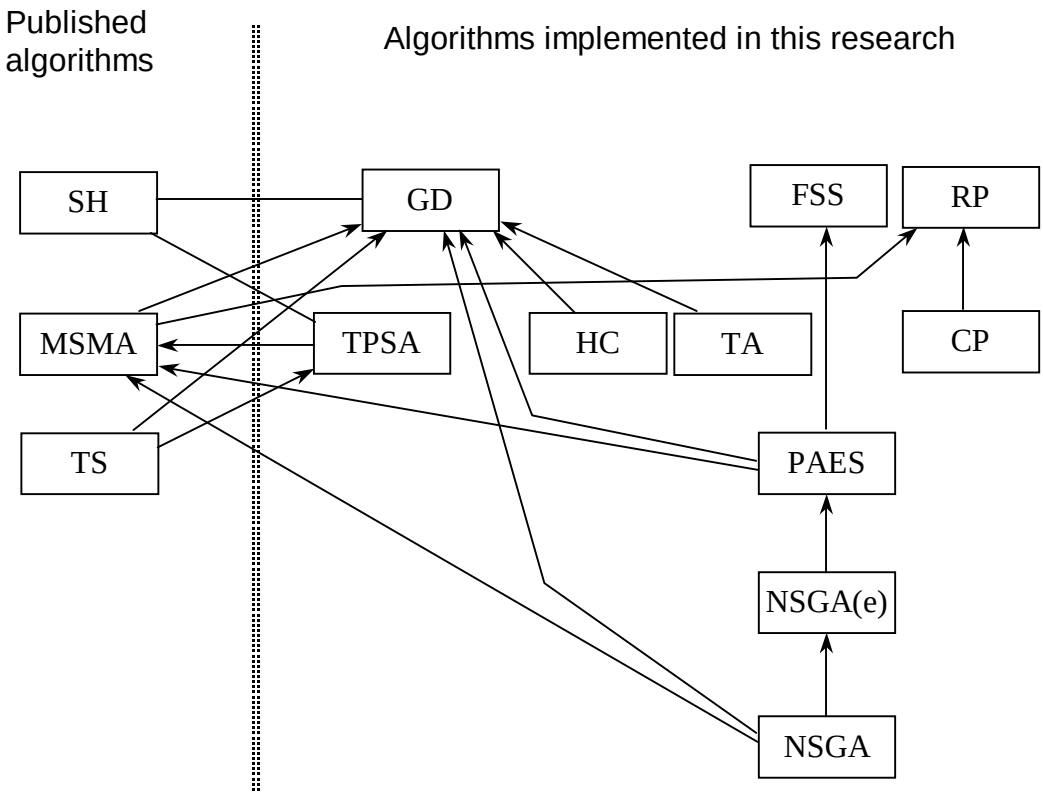

Figure 8.1: Comparison graph of the performance of discussed algorithms

From this graph, the performance of the discussed algorithms can be summarised in the following way:

- The Great Deluge algorithm (GD) was compared with the sequential heuristics (SH) of Carter's et al. [CLL96] in single-objective case where produced results were better than the published ones in 10 cases and worse in 2 cases. Besides this the Great Deluge algorithm was compared to and evidently outperformed: Hill-Climbing (HC) and the Threshold Acceptance method (TA) in the single-objective case. It also outperformed: the Non-Domianted Sorting Genetic Algorithm (NSGA) and the Pareto-Archived Evolution Strategy (PAES) in the bi-objective case. In both the single and bi-objective cases it outperformed: Multi-Stage Memetic Algorithm (MSMA) published in [BN99] and Tabu Search (TS) published in [DGS01].

- The Time-Predefined Simulated Annealing (TPSA) produced 8 results better and 4 worse than SH in a single objective case, but all the results were worse than MSMA in the bi-objective case. In both cases (single and bi-objective), TPSA produced better results than TS.

- The multiobjective Great Deluge algorithm with variable weights applied with the reference point strategy (RP) produced better results than MSMA in the bi-objective case and Compromise Programming approach (CP) in the nine-objective case.

- The enhanced variant of the multiobjective Great Deluge search used with the Fan Search strategy (FSS) outperformed the PAES method. In its own turn, PAES outperformed Non-dominated Sorting Genetic Algorithm with elitism (NSGA(e)), which outperformed plain NSGA. However, both PAES and

NSGA were outperformed by MSMA. This chain of comparisons was carried out for the bi-objective case.

In order to reflect the relative performance of the different approaches, the nodes of the graph are depicted while trying to orient the arrows in an upper direction (the inferior algorithms are placed below the superior ones). The upper level contains the presented variants of the Great Deluge algorithm because in most cases they produced results of higher quality than other techniques.

8.3 Publications

During the research work presented in this thesis the following papers were published:

Journal papers:

- E. K. Burke, **Y. Bykov**, J. P. Newall, S. Petrovic. "A Time-Predefined Local Search Approach to Exam Timetabling Problems". Accepted for publication in *IIE transactions on Operations Engineereing*.

- E. K. Burke, **Y. Bykov**, J. P. Newall, S. Petrovic. "A New Local Search Approach with Execution Time as an Input Parameter". Accepted for publication in *YUJOR - Yugoslav Journal of Operational Research*. Selected as one of 5 best papers from the 6th Balkan Conference on Operations Research, Thessaloniki, Greece, 20-24 May 2002.

Volumes with selected refereed conference papers:

- E. K. Burke, **Y. Bykov**, S. Petrovic. "A Multicriteria Approach to Examination Timetabling". E. Burke, W. Erben (eds.), *The Practice and Theory of Automated Timetabling III: Selected Papers (PATAT 2000)*. Lecture Notes in Computer Science 2079 Springer-Verlag, Berlin, Heidelberg, New York, 2001, 118-131.

- S. Petrovic, **Y. Bykov**. "A Multiobjective Optimisation Technique for Exam Timetabling Based on Trajectories". Accepted for publication in E. Burke, P. De Causmaecker (eds.), *The Practice and Theory of Automated Timetabling IV: Selected Papers (PATAT 2002)*. Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, New York, 2003, 179-192.

Journal papers in preparation for submission:

- **Y. Bykov**, S. Petrovic. "Trajectory-Based Search of the Criteria Space".

8.4 Applications of the Presented Approaches in Different Areas

The presented approaches concentrate upon the exam timetabling problem but there is significant scope to investigate these approaches for other optimisation problems. The brief description of the application of the proposed approaches to two different domains, namely course timetabling and bio-informatics is given in the next sections.

8.4.1 An International Timetabling Competition

The described Great Deluge algorithm was applied to the university course timetabling problem while participating in the International Timetabling

Competition organized by Metaheuristics Network and sponsored by the Practice and Theory of Automated Timetabling IV (PATAT IV) conference in 2003. The algorithm obtained third place among 21 participants (even though it was initially designed for exam timetabling rather than course timetabling).

Each participant of the competition had to present solutions to 20 problem instances. Every problem comprised the assignment of the given number of courses to timeslots and rooms. Two hard constraints were considered:

- courses with common students had to be scheduled in different timeslots (a clash-free requirement),
- rooms had to be suitable for assigned courses (facilities required for corresponding courses had to be available).

Three soft constraints generated a penalty. These constraints are:

- a student had only one lecture in a day,
- a student had more than three consecutive lectures in a day,
- a student had lecture in last timeslot of a day.

The objective function was calculated as a sum of the violations of these constraints.

All solutions had to be obtained in a limited time interval. In order to synchronise the time intervals on different hardware, the special test program was provided by the organising committee. In particular, on a PC Athlon 750MHz the processing time was limited to 15 minutes.

Although the processing time was relatively short, which was not suitable for the Great Deluge algorithm where longer available time leads to better results, the author's results were the best among all participants in 7 from 20 cases. Moreover, among all the registered participants, only the Great Deluge algorithm has provided a solution with a zero value of the objective function and hence has reached the global optimum of the problem. It should be noticed that the first and second place participants also used local search algorithms (Simulated Annealing and Tabu Search). However, they paid more attention to neighbourhood structures. Indeed, this is one of the most promising ways of improving the performance of timetabling algorithms and can be considered as an important direction of the future research.

8.4.2 An Investigation of the Protein Folding Problem

At the time of writing this thesis the trajectory-based Great Deluge algorithm was successfully applied in the bio-informatics area. It showed a very promising performance when used for the investigation of Protein Folding problem.

The Protein Folding problem considers a chain of residues of different types (aminoacids), which fold in a 3-dimentional conformation. Given the energy of interactions between each pair of aminoacids as a function of the distance between them, the goal is to find the conformation, which imposes the minimum sum energy for the whole chain. Thus, the solution of this problem can be represented with coordinates of all residues, subject to the hard constraint:

- The distance between every pair of consequent residues must be equal to a given constant.

The objective function is the total energy of conformation and should be minimised.

However, the difficulties in solving this problem arise because the particular interactions between different aminoacids are not well-defined. Thus, the major research challenge involves the validation of the model rather than the actual solving of the problem. The tuning of the model parameters can be done while running the algorithm a number of times using existing real proteins (with known conformations) as problem instances.

The motivation for the use of trajectory-based approaches can be confirmed as follows. This problem has a highly disconnected search space, which obstructs the conducting of local search among feasible solutions. Much better results can be obtained while temporarily relaxing the hard constraint. Conventionally this is done using an objective function, which is the weighted sum of violations of the hard and soft constraints and by assigning a very high weight to the hard constraint. However, when applied to the Protein Folding problem this method requires a very precise specification of the weights, which is practically impossible. Too high a weight assigned to the hard constraint results in a poor quality of final solution, while too low a weight leads to an infeasible one.

The trajectory-based method was found to be very advantageous in such a situation. Here the coordinates of the reference point are known (value

zero for the hard constraint violations and a real-world value for the objective function) and a necessary trajectory can be easily drawn. The search is conducted through the trajectory exactly into the given point while avoiding the drawbacks of the weighted sum approach.

The further validation of the model demands taking into consideration other constraints. The third objective (which considers the angle between consequent residues) was included into the model and the trajectory-based algorithm successfully managed the three-objective Protein Folding problem.

8.5 Future Work

The author intends to expand the practical benefits of the presented approaches. It would be worth more accurately investigating their properties, such as the dependence on a problem's size (and on other characteristics of a problem). Another direction would be to investigate the influence of an initial solution on the overall result while exploring different initialisation methods. Also this research did not touch on the question about neighbourhood variation, which probably influences the result as well as questions about disconnected search spaces, relaxation of a problem, Kempe or S-chains, etc. The described Great Deluge algorithm employs a linear reduction of the level of acceptance of worse solutions as the simplest variant. It seems reasonable to investigate other possible reductions.

In the future research work the proposed multiobjective algorithms will be evaluated in other domains with a different number of objectives and on different shapes of trade-off surfaces. Additional issues will be investigated:

how to initialise the weights in Great Deluge with variable weights and how to choose a “good” rotation rate in the enhanced multiobjective Great Deluge algorithm. A detailed comparison of the presented approach with other multiobjective approaches (e.g. Goal Programming and Lexicographic ordering) will be carried out together with an investigation of the proposed trajectory-based strategies, especially – interactive ones.

In addition to this, the idea of time-predefined algorithms requires more attentive study. Possibly, the multiobjective approach can be expanded while considering the quality of solution and computational time as two user objectives. Additionally, some methods suitable for the formal comparison of the time-cost indices of different algorithms could be defined and investigated.

Also, it is certainly the case that the proposed methods are open to different extensions and hybridisations. In particular, the family of time-predefined techniques is not limited to the suggested methods. The author believes that the predefinition of time can indeed be embedded into other techniques. In the multiobjective case, more advanced mechanisms of the search directing in the criteria space together with new strategies of the assigning proper trajectories will be developed.

References

[ABN60] J. S. Appleby, D. V. Blake, E. A. Newman. "Techniques for Producing School Timetables on a Computer and their Application to other Scheduling Problems". *The Computer Journal* 3, 1960/61, 237-245.

[AL89] T. Arani, V. Lotfi. "A Three Phased Approach to Final Exam Scheduling". *IIE Transactions* 21, 1989, 86-96.

[All92] R. Allenson. "Genetic Algorithms with Gender for Multi-Function Optimisation". Technical Report EPCC-SS92-01, Edinburgh Parallel Computing Centre, Edinburgh, Scotland, 1992.

[Bag99] T. P. Bagchi. "Multiobjective Scheduling by Genetic Algorithms". Kluwer Academic Publishers, Boston, Dordrecht, London, 1999.

[BB00] C. C. H. Borges, H. J. C. Barbosa. "A Non-Generational Genetic Algorithm for Multiobjective Optimization". In *Proceedings of 2000 Congress on Evolutionary Computation, San Diego, California* vol.1, 2000, 172-179.

[BBP01] E. K. Burke, Y. Bykov, S. Petrovic. "A Multicriteria Approach to Examination Timetabling". E. Burke, W. Erben (eds.), *The Practice and Theory of Automated Timetabling III: Selected Papers (PATAT 2000)*. Lecture Notes in Computer Science 2079, Springer-Verlag, Berlin, Heidelberg, New York, 2001, 118-131.

[BDP94] P. Boizumault, Y. Delon, L. Peridy. "Planning exams using CLP". In *Proceedings of 2nd ICPAP'94, London*, 1994.

[BEW94] E. K. Burke, D. G. Elliman, and R. F. Weare. "A Genetic Algorithm for University Timetabling". T. C. Fogarty (editor), *Evolutionary Computing: AISB Workshop, Leeds, UK, April 1994; Selected Papers.* Lecture Notes in Computer Science 865, Springer-Verlag, Berlin, Heidelberg, New York, 1994.

[BEW95] E. K. Burke, D. Elliman, R. Weare. "Specialise Recombinative Operators for Timetabling Problem". In *Proceedings of the AISB Workshop on Evolutionary Computing, University of Sheffield, UK*, 1995, 75-85.

[BKW03] E. K. Burke, J. Kingston and D. De Werra. "Applications to Timetabling". To appear in J. Gross, J. Yellen (eds.), *Handbook of Graph Theory*, to be published by Chapman Hall/CRC Press, 2003.

[BLW92] N. Balakrishnan, A. Lucena, R. T. Wong. "Scheduling Examinations to Reduce Second-Order Conflicts". *Computers and Operations Research* 19, 1992, 353-361.

[BN96] J. P. Boufflet, S. Negre. "Three Methods Used to Solve an Examination Timetabling Problem". E. Burke, P. Ross (eds.), *The Practice and Theory of Automated Timetabling: Selected Papers (ICPTAT '95)*. Lecture Notes in Computer Science 1153. Springer-Verlag, Berlin, Heidelberg, New York, 1996, 327-344.

[BN99] E. K. Burke, J. P. Newall. "A Multi-Stage Evolutionary Algorithm for the Timetable Problem". *The IEEE Transactions on Evolutionary Computation* 3(1), 1999, 63-74.

[BN02] E. K. Burke, J. P. Newall. "Enhancing Timetable Solutions with Local Search Methods". In *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT2002)*, 2002, 336-347.

[BN03] E. K. Burke, J. P. Newall. "Solving Examination Timetabling Problems through Adaptation of Heuristic Orderings". Accepted for publication in *Annals of Operations Research*, 2003.

[BNW96] E. K. Burke, J. P. Newall, R. F. Weare. "A Memetic Algorithm for University Exam Timetabling". E. Burke, P. Ross (eds.), *The Practice and Theory of Automated Timetabling: Selected Papers (ICPTAT '95)*. Lecture Notes in Computer Science 1153. Springer-Verlag, Berlin, Heidelberg, New York, 1996, 241-250.

[BNW98] E. K. Burke, J. P. Newal, R. F. Weare. “Initialization Strategies and Diversity in Evolutionary Timetabling”. *Evolutionary Computation* 6(1), 1998, 81-103.

[BP02] E. K. Burke, S. Petrovic. “Recent Research Directions in Automated Timetabling”. *European Journal of Operational Research - EJOR* 140(2), 2002, 266-280.

[Bre79] D. Brelaz. “New Methods to Colour the Vertices of a Graph”. *Communication of the ACM* 22(4), 1979, 251-256.

[Bro41] R. L. Brooks. “On Colouring the Nodes of a Network”. *Proceedings of Cambridge Philosophical Society*, 1941, 194-197.

[Bro64] S. Broder. “Final Examination Scheduling”. *Communications of the ACM* 7, 1964, 494-498.

[BRS66] H. J. Bremermann, J. Roghson, S. Salaff. “Global Properties of Evolution Processes”. H. H. Pattee, E. A. Edelsack, L. Fein, A. B. Calahan (eds.), *Natural Automata and Useful Simulations*, Macmillan, 1966, 3-42.

[Bul98] B. Bullnheimer. “An Examination Scheduling Model to Maximize Students' Study Time”. E. Burke, M. Carter (eds.), *The Practice and Theory of Automated Timetabling: Selected Papers (PATAT '97)*. Lecture Notes in Computer Science 1408. Springer-Verlag, Berlin, Heidelberg, New York, 1998, 78-91.

[Burk96] E. K. Burke, D. Elliman, P. Ford, R. Weare. "Examination Timetabling in British Universities: a Survey". E. Burke, P. Ross (eds.), *The Practice and Theory of Automated Timetabling: Selected Papers (ICPTAT '95)*. Lecture Notes in Computer Science 1153. Springer-Verlag, Berlin, Heidelberg, New York, 1996, 76-90.

[Burk97] E. K. Burke, K. S. Jackson, J. H. Kingston and R. F. Weare. "Automated Timetabling: The State of the Art". *The Computer Journal* 40(9), 1997, 565-571.

[Burk02] E. K. Burke, A. J. Eckersley, B. McCollum, S. Petrovic, R. Qu. "A Case Based Heuristic Selection Investigation of Hill Climbing, Simulated Annealing and Tabu Search for Exam Timetabling Problems" (abstract). In *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT2002)*, 2002, 408-410.

[Burk03a] E. K. Burke, E. Hart, G. Kendall, J. P. Newall, P. Ross, S. Schulenburg. "Hyper-Heuristics: An Emerging Direction in Modern Search Technology". Chapter 16 in F. W. Glover, G. A. Kochenberger (eds.), *Handbook of Metaheuristics* . International Series in Operations Research and Management Science 57, Kluwer Academic Publishers, Boston, Dordrecht, London, 2003, 457-474.

[Burk03b] E. K. Burke, Y. Bykov, J. P. Newall, S. Petrovic. "A Time-Predefined Local Search Approach to Exam Timetabling Problems". Accepted for publication in *IIE transactions on Operations Engineereing*, 2003.

[Burk03c] E. K. Burke, Y. Bykov, J. P. Newall, S. Petrovic. "A New Local Search Approach with Execution Time as an Input Parameter". Accepted for publication in *YUJOR - Yugoslav Journal of Operational Research*, 2003.

[Car86] M. W. Carter. "A Survey of Practical Applications of Examination Timetabling Algorithms". *Operations Research* 34(2), 1986, 193-201.

[Car01] M. W. Carter. "A Comprehensive Course Timetabling and Student Scheduling System at the University of Waterloo". E. Burke, W. Erben (eds.), *The Practice and Theory of Automated Timetabling III: Selected Papers (PATAT 2000)*. Lecture Notes in Computer Science 2079. Springer-Verlag, Berlin, Heidelberg, New York, 2001, 64-82.

[CC61] A. Charnes, W. W. Cooper. "Management Models and Industrial Applications of Linear Programming". Wiley, New York, 1961.

[CCF55] A. Charnes, W. W. Cooper, R. Ferguson. "Optimal Estimation of Executive Compensation by Linear Programming". *Management Science* 1, 1955, 138-151.

[CDI01] M. Caramia, P. Dell'Olmo, G. F. Italiano. "New Algorithms for Examination Timetabling". S. Nher, D. Wagner (eds.), *Algorithm Engineering 4th International Workshop, WAE 2000, Saarbrucken, Germany, September 2000. Proceedings*. Lecture Notes in Computer Science 1982, Springer-Verlag, Berlin, Heidelberg, New York, 2001, 230-241.

[CFM93] D. Corne, H. L. Fang, C. Mellish. "Solving the Module Exam Scheduling Problem with Genetic Algorithms". P. W./H. Chung, G. Lovergrove, M. Ali (eds.), *Proceedings of the Sixth International Conference in Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*. Gordon and Breach Science Publishers, 1993, 370-373.

[CH97] D. Costa, A. Hertz. "Ants can Colour Graphs". *Journal of Operational Research Society* 48, 1997, 295-305.

[CJ98] P. Czyzak, A. Jaszkiewicz. "Pareto Simulated Annealing – a Metaheuristic Technique for Multiple-Objective Combinatorial Optimization". *Journal of Multi-Criteria Decision Analysis* 7, 1998, 34-47.

[CJ01] M. W. Carter, D. G. Johnson. "Extended Clique Initialisation in Examination Timetabling". *Journal of the Operational Research Society* 52, 2001, 538-544.

[CL96] M. W. Carter, G. Laporte. "Recent Developments in Practical Examination Timetabling". E. Burke, P. Ross (eds.), *The Practice and Theory of Automated Timetabling: Selected Papers (ICPTAT '95)*. Lecture Notes in Computer Science 1153. Springer-Verlag, Berlin, Heidelberg, New York, 1996, 3-21.

[CLC94] M. W. Carter, G. Laporte, J. W. Chinneck. "A General Examination Scheduling System". *Interfaces* 24, 1994, 109-120.

[CLL96] M. W. Carter, G. Laporte, S. Y. Lee. "Examination Timetabling: Algorithmic Strategies and Applications". *Journal of Operational Research Society* 47(3), 1996, 373-383.

[Coe99] C. A. Coelo Coelo. "A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques". *Knowledge and Information Systems. An International Journal* 1(3), 1999, 269-308.

[Col64] A. J. Cole. "The Preparation of Examination Timetables Using a Small-Store Computer". *The Computer Journal* 7, 1964, 117-121.

[Cos94] D. Costa. "A Tabu Search Algorithm for Computing an Operational Timetable". *European Journal of Operational Research* 76, 1994, 98-110.

[CP01] M. P. Carrasco, M. V. Pato. “A Multiobjective Genetic Algorithm for the Class/Teacher Timetabling Problem”. E. Burke, W. Erben (eds.), *The Practice and Theory of Automated Timetabling III: Selected Papers (PATAT 2000)*. Lecture Notes in Computer Science 2079. Springer-Verlag, Berlin, Heidelberg, New York, 2001, 3-17.

[CR96] D. Corne, P. Ross. “Peckish Initialisation Strategies for Evolutionary Timetabling”. E. Burke, P. Ross (eds.), *The Practice and Theory of Automated Timetabling: Selected Papers (ICPTAT '95)*. Lecture Notes in Computer Science 1153. Springer-Verlag, Berlin, Heidelberg, New York, 1996, 227-240.

[CRF94] D. Corne, P. Ross, H. L. Fang. “Fast Practical Evolutionary Timetabling”. T. C. Fogarty (editor), *Evolutionary Computing: AISB Workshop, Leeds, UK, April 1994; Selected Papers.* Lecture Notes in Computer Science 865, Springer-Verlag, Berlin, Heidelberg, New York, 1994, 250-263.

[Dav98] P. David. “A Constraint-Based Approach for Examination Timetabling Using Local Repair Techniques”. E. Burke, M. Carter (eds.), *The Practice and Theory of Automated Timetabling: Selected Papers (PATAT '97)*. Lecture Notes in Computer Science 1408. Springer-Verlag, Berlin, Heidelberg, New York, 1998, 169-186.

[DG02] L. Di Gaspero. "Recolour, Shake and Kick: a recipe for the Examination Timetabling Problem" (abstract). In *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT2002)*, 2002, 404-407.

[DGS01] L. Di Gaspero, A. Schaerf. "Tabu Search Techniques for Examination Timetabling". E. Burke, W. Erben (eds.), *The Practice and Theory of Automated Timetabling III: Selected Papers (PATAT 2000)*. Lecture Notes in Computer Science 2079. Springer-Verlag, Berlin, Heidelberg, New York, 2001, 104-117.

[DMC91] M. Dorigo, V. Maniezzo, A. Colorni. "Positive Feedback as a Search Strategy". Technical Report 91-016, Politecnico di Milano, Italy, 1991.

[Dow98] K. Dowsland. "Off-the-Peg or Made-to-Measure? Timetabling and Scheduling with SA and TS". E. Burke, M. Carter (eds.), *The Practice and Theory of Automated Timetabling: Selected Papers (PATAT '97)*. Lecture Notes in Computer Science 1408. Springer-Verlag, Berlin, Heidelberg, New York, 1998, 37-52.

[DPT02] K. A. Dowsland, N. Pugh, J. Thompson. "Examination Timetabling with Ants" (abstract). In *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT2002)*, 2002, 397-399.

[DS90] G. Dueck, T. Scheuer. "Threshold Accepting: a General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing". *Journal of Computational Physics* 90, 1990, 161-175.

[Due93] G. Dueck. "New Optimization Heuristics. The Great Deluge Algorithm and the Record-to-Record Travel". *Journal of Computational Physics* 104, 1993, 86-92.

[Erb01] W. Erben. "A Grouping Genetic Algorithm for Graph Colouring and Exam Timetabling". E. Burke, W. Erben (eds.), *The Practice and Theory of Automated Timetabling III: Selected Papers (PATAT 2000)*. Lecture Notes in Computer Science 2079. Springer-Verlag, Berlin, Heidelberg, New York, 2001, 132-156.

[Erg96] A. Ergul. "GA-Based Examination Scheduling Experience at Middle East Technical University". E. Burke, P. Ross (eds.), *The Practice and Theory of Automated Timetabling: Selected Papers (ICPTAT '95)*. Lecture Notes in Computer Science 1153. Springer-Verlag, Berlin, Heidelberg, New York, 1996, 212-226.

[FF93] C. M. Fonseca, P. J. Fleming. "Genetic Algorithms for Multiobjective Optimisation: Formulation, Discussion and Generalization". In *Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, California*, 1993, 416-423.

[Fra57] A. S. Fraser. “Simulation of Genetic Systems by Automatic Digital Computers”. *Australian Journal of Biological Sciences*, 1957.

[FS83] J. G. Fisher, D. R. Shier. “A Heuristic Procedure for Large-Scale Examination Scheduling Problems”. Technical Report 417, Department of Mathematical Sciences, Clemson University, 1983.

[GL97] F. Glover, M. Laguna. “Tabu Search”. Kluwer Academic Publishers, Boston, Dordrecht, London, 1997.

[Glo86] F. Glover. “Future Paths for Integer Programming and Links to Artificial Intelligence”. *Computers and Operational Research* 5, 1986, 533-549.

[GMF97] X. Gandibleux, N. Mezdaoui, A. Freville. “A Tabu Search Procedure to Solve Multiobjective Combinatorial Optimization Problems”. *Advances in Multiple Objective and Goal Programming*. Lecture Notes in Economics and Mathematical Systems 455, Springer-Verlag, Berlin, Heidelberg, New York, 1997, 291-300.

[Gol89] D. E. Goldberg. “Genetic Algorithms in Search, Optimisation and Machine Learning”. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[Gor89] E. Gordeev. “Decision Problems and their Solutions”. *Cybernetics - Unlimited Possibilities and Possible Limits*, USSR Academy of Science, 1989, 5-48 (in Russian).

[Han97] M. P. Hansen. “Tabu Search for Multiobjective Optimization: MOTS”. In *Proceedings of MCDM’97, Cape Town, South Africa*, 1997.

[Her91] A. Hertz. “Tabu Search for Large Scale Timetabling Problems”. *European Journal of Operational Research* 54, 1991, 39-47.

[Hil89] M. R. Hillard, G. E. Lliepins, M. Palmer, G. Rangarajen. “The Computer as a Partner in Algorithmic Design: Automated Discovery of Parameters for a Multiobjective Scheduling Heuristic”. *Impacts of Recent Computer Advances on Operations Research*, North-Holland Publishing Company, New York, 1989.

[HJ98] M. P. Hansen, A. Jaszkiewicz. “Evaluating the Quality of Approximations to the Non-Dominated Set”. Technical Report IMM-REP-1998-7, Technical University of Denmark, 1998.

[HN93] J. Horn, N. Nafpliotis. “Multiobjective Optimization Using the Niched Pareto Genetic Algorithm”. Thechnical Report IlliGAl 93005, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1993.

[Hol75] J. H. Holland. “Adaptation in Natural and Artificial Systems”. University of Michigan Press, 1975.

[JF95] T. Jones, S. Forrest. “Genetic Algorithms and Heuristic Search”. In *Proceedings of International Joint Conference on Artificial Intelligence*, 1995.

[Joh90] D. G. Johnson. “Timetabling University Examinations”. *Journal of the Operational Research Society* 40, 1990, 39-47.

[Kar72] R. M. Karp. “Reducibility among Combinatorial Problems”. *Complexity of Computer Computations*, Plenum press, New York, 1972, 85-104.

[KC99] J. D. Knowles, D. W. Corne. “The Pareto-Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimisation”. In *Proceedings of the 1999 Congress on Evolutionary Computation, Washington, DC*, 1999, 98-105.

[KC00a] J. D. Knowles, D. W. Corne. “Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy”. *Evolutionary Computation*, 8(2), 2000, 149-172.

[KC00b] J. D. Knowles, D. W. Corne. “M-PAES: A Memetic Algorithm for Multiobjective Optimization”. In *Proceedings of the 2000 Congress on Evolutionary Computation, Piscataway, New Jersey*, vol.1, 2000, 325-332.

[KC02] J. D. Knowles, D. W. Corne. "On Metrics for Comparing Non-Dominated Sets". In *Proceedings of the 2002 Congress on Evolutionary Computation Conference (CEC02)*, IEEE Press, 2002, 711-716.

[KGV83] S. Kirkpatrick, J. C. D. Gellat, M. P. Vecci. "Optimization by Simulated Annealing". *Science* 220, 1983, 671-680.

[Kir84] S. Kirkpatrick. "Optimization by Simulated Annealing – Quantitative Studies". *Journal of Statistical Physics* 34, 1984, 975-986.

[KT51] H. W. Kuhn, A. W. Tucker. "Nonlinear Programming". In *Proceedings of the Second Berkley Symposium on Mathematical Statistics and Probability, Berkley, California*, 1951, 481-492.

[LC91] V. Lotfi, R. Cerveny. "A Final Exam-Scheduling Package". *Journal of Operational Research Society* 42, 1991, 205-216.

[LE96] J. Lis, A. E. Eiben. "A Multi-Sexual Genetic Algorithm for Multiobjective Optimization". In *Proceedings of the 1996 International Conference on Evolutionary Computation, Nagoya, Japan*, 1996, 59-64.

[LG96] M. Laguna, F. Glover. "What is Tabu Search?". *Colorado Business Review* LXI(5), 1996.

[Lin02] S. L. M. Lin. "A Broker Algorithm for Timetabling Problem". In *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT2002)*, 2002, 372-386.

[LO99] S. M. Lee, D. L. Olson. "Goal Programming". *Multicriteria Decision Making: Advances in MCDM Models, Algorithms, Theory and Applications*, Kulwer Academic Publishers, Boston, Dordrecht, London, 1999, 8.1-8.33.

[Mer02] L. T. G. Merlot, N. Boland, B. D. Hughes, P. J. Stuckey. "A Hybrid Algorithm for the Examination Timetabling Problem". In *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT2002)*, 2002, 348-371.

[MMI72] D. W. Matula, G. Marble, I. D. Isaacson. "Graph Colouring Algorithms". *Graph Theory and Computing*, Academic Press, New York, 1972.

[MN92] P. Moscato, M. G. Norman. "A "Memetic" Approach for the Travelling Salesman Problem - Implementation of a Computational Ecology for Combinatorial Optimisation on Message-Passing System". In *Proceedings of the International Conference on Parallel Computing and Transputer Applications*. IOS Press, Amsterdam, 1992.

[MT96] A. Mehrotra, M. A. Trick. "A Clique Generation Approach to Graph Coloring". *INFORMS Journal on Computing* 8, 1996, 344-354.

[OL96] I. Osman, G. Laporte. "Metaheuristics: A Bibliography". *Annals of Operations Research* 63, 1996, 513-623.

[Pae98] B. Paechter, R. C. Rankin, A. Cumming, T. C. Fogarty. "Timetabling the Classes of an Entire University with an Evolutionary Algorithm". T. Beck, M. Schoenauer (eds.), *Parallel Problem Solving from Nature - PPSN V*. Springer-Verlag, Berlin, Heidelberg, New York, 1998.

[PB03] S. Petrovic, Y. Bykov. "A Multiobjective Optimisation Technique for Exam Timetabling Based on Trajectories". Accepted for publication in E. Burke, P. De Causmaecker (eds.), *The Practice and Theory of Automated Timetabling IV: Selected Papers (PATAT 2002)*. Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, New York, 2003, 179-192.

[PF01] L. F. Paquete, C. M. Fonseca. "A Study of Examination Timetabling with Multiobjective Evolutionary Algorithms". In *Proceedings of 4th Metaheuristics International Conference (MIC 2001), Porto*, 2001, 149-154.

[Pin95] M. Pinedo. "Scheduling Theory, Algorithms, and Systems". Prentice-Hall, Inc. A Simon & Schuster Company, New Jersey, USA, 1995.

[PP95] S. Petrovic, R. Petrovic. "Eco-Ecodispatch: DSS for Multicriteria Loading of Thermal Power Generators". *Journal of Decision Systems* 4, 1995, 279-295.

[PS02] L. Paquete, T. Stutzle. "Empirical Analysis of Tabu Search for the Lexicographic Optimisation of the Examination Timetabling Problem" (abstract). In *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT2002)*, 2002, 413-420.

[PW66] J. E. L. Peck, M. R. Williams. "Algorithm 286 - Examination Scheduling". *Communication of the ACM* 9, 1966, 433-434.

[RC95] P. Ross, D. Corne. "Comparing Genetic Algorithms, Simulated Annealing, and Stochastic Hillclimbing on Timetable Problems". In *Proceedings of the AISB Workshop in Evolutionary Computing*, Lecture Notes in Computer Science 993, Springer-Verlag, Berlin, Heidelberg, New York, 1995.

[Ree94] C. R. Reeves. “Genetic Algorithms and Neighbourhood Search”. T. C. Fogarty (editor), *Evolutionary Computing: AISB Workshop, Leeds, UK, April 1994; Selected Papers.* Lecture Notes in Computer Science 865, Springer-Verlag, Berlin, Heidelberg, New York, 1994, 115-129.

[Ree96] C. R. Reeves. “Modern Heuristic Techniques”. *Modern Heuristic Search Methods*, John Willey & Sons Ltd., 1996.

[RH96] V. Robert, A. Hertz. “How to Decompose Constrained Course Scheduling Problems Into Easier Assignment Type Subproblems”. E. Burke, P. Ross (eds.), *The Practice and Theory of Automated Timetabling: Selected Papers (ICPTAT '95)*. Lecture Notes in Computer Science 1153. Springer-Verlag, Berlin, Heidelberg, New York, 1996, 364-373.

[RHC98] P. Ross, E. Hart, D. Corne. “Some Observations about GA-Based Exam Timetabling”. E. Burke, M. Carter (eds.), *The Practice and Theory of Automated Timetabling: Selected Papers (PATAT '97)*. Lecture Notes in Computer Science 1408. Springer-Verlag, Berlin, Heidelberg, New York, 1998, 115-129.

[Ric89] J. T. Richardson, M. R. Palmer, G. Liepins, M. Hillard. “Some Guidelines for Genetic Algorithms with Penalty Functions”. In *Proceedings of the Third International Conference on Genetic Algorithms, George Mason University*, 1989, 191-197.

[RO99] L. P. Reis, E. Oliveira. "Constraint Logic Programming Using Set Variables for Solving Timetabling Problems". In *Proceedings of INAP'99 – 12th International Conference on Applications of Prolog, Tokyo, Japan*, 1999.

[Sch85] J. D. Schaffer. "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms". In *Proceedings of the First International Conference on Genetic Algorithms*, 1985, 93-100.

[Sch96] A. Schaerf. "Tabu Search Techniques for Large High-School Timetabling Problems". Technical Report CWI, Amsterdam, 1996.

[Sch99a] A. Schaerf. "A Survey of Automated Timetabling". *Artificial Intelligent Review*, 13, 1999, 87-127.

[Sch99b] A. Schaerf. "Local Search Techniques for Large High School Timetabling Problems". *IEEE Transactions on Systems Man and Cybernetics* 29(4) part A, 1999, 368-377.

[SD94] N. Srinivas, K. Deb. "Multiobjective Optimization using Non-Dominated Sorting Genetic Algorithms". *Evolutionary Computation* 2(3), 1994, 221-248.

[SF96a] K. J. Shaw, P. J. Fleming. "An Initial Study of Practical Multiobjective Production Scheduling, Using Genetic Algorithms". In *Proceedings of the International Conference on Control'96, University of Exter, UK*, 1996.

[SF96b] K. J. Shaw, P. J. Fleming. "Initial Study of Multi-Objective Genetic Algorithms for Scheduling the Production of Chilled Ready Meals". In *Proceedings of the Second International Mendel Conference on Genetic Algorithms, Brno, Czech Republic*, 1996.

[Sha00] K. J. Shaw, P. L. Lee, H. P. Nott, M. Thompson. "Genetic Algorithms for Multiobjective Scheduling of Combined Batch/Continuous Process Plants". In *Proceedings of the 2000 Congress on Evolutionary Computation, Piscataway, New Jersey*, 2000, 293-300.

[She02] K. Sheibani. "An Evolutionary Approach for the Examination Timetabling Problems" (abstract). In *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT2002)*, 2002, 387-396.

[SP91] G. Syswerda, J. Palmucci. "The Application of Genetic Algorithms to Resource Scheduling". In *Proceedings of the Fourth International Conference on Genetic Algorithms, San Mateo, California*, 1991, 502-508.

[TA00] M. Tanaka, S. Adachi. "Request-Based Timetabling by Genetic Algorithm with Tabu Search". In *Proceedings of the Third International Workshop on Frontiers in Evolutionary Algorithms*, 2000, 999-1002.

[TD93] J. M. Thompson, K. A. Dowsland. "Multi-Objective University Examination Scheduling". Technical report EBMS/1993/12, European Business Management School, Swansea University, UK, 1993.

[TD96a] J. M. Thompson, K. A. Dowsland. "General Cooling Schedules for a Simulated Annealing Based Timetabling System". E. Burke, P. Ross (eds.), *The Practice and Theory of Automated Timetabling: Selected Papers (ICPTAT '95)*. Lecture Notes in Computer Science 1153. Springer-Verlag, Berlin, Heidelberg, New York, 1996, 345-363.

[TD96b] J. M. Thompson, K. A. Dowsland. "Variants of Simulated Annealing for the Examination Timetabling Problem". *Annals of Operations Research* 63, 1996, 105-128.

[TD98] J. M. Thompson, K. A. Dowsland. "A Robust Simulated Annealing Based Examination Timetabling System". *Computers and Operational Research* 25(7/8), 1998, 637-648.

[Ter94] H. Terashima-Marin. "A Comparison of GA-Based Methods and Graph-Colouring Methods for Solving the Timetabling Problem". Master's thesis, Department of AI, University of Edinburgh, 1994.

[TJR98] M. Tamiz, D. Jones, C. Romero. "Goal Programming for Decision Making: An Overview of the Current State-of-the-Art". *European Journal of Operational Research* 111, 1998, 569-581.

[TRV99] H. Terashima-Marin, P. M. Ross, M. Valenzuela-Rendon. "Evolution of Constraint Satisfaction Strategies in Examination Timetabling". In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*. Morgan Kaufmann, 1999, 635-642.

[UTF95] E. L. Ulungu, J. Teghem, P. Fortemps. "Heuristics for Multi-Objective Combinatorial Optimisation Problem by Simulated Annealing". In *Proceedings of the 6th International Conference on MCDM, Beijing, China*, 1995, 228-238.

[Vin92] P. Vincke. "Multicriteria Decision-Aid". John Wiley & Sons, Chichester, New York, Brisbane, Toronto, Singapore, 1992.

[VU97] M. Valenzuela-Rendon, E. Uresti-Charre. "A Non-Generational Genetic Algorithm for Multiobjective Optimization". In *Proceedings of the Seventh International Conference on Genetic Algorithms, San Mateo, California*, 1997, 658-665.

[VV99] D. A. Van Veldhuizen. “Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations”. PhD Thesis, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1999.

[VVL00] D. A. Van Veldhuizen, G. B. Lamont. “Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art”. *Evolutionary Computation* 8(2), 2000, 125-147.

[VZ00] A. Vesel, J. Zerovnik. “How Good Can Ants Color Graphs?”. *Journal of Computing and Information Technology* 8(2), 2000, 131-136.

[Whi00] G. M. White. “Constraint satisfaction, Not So Constrained Satisfaction and the Timetabling Problem”. In *Proceedings of the Third International Conference on the Practice and Theory of Automated Timetabling (PATAT2000)*, 2000, 32-47.

[Wie99] A. P. Wierzbicki. “Reference Point Approach”. *Multicriteria Decision Making: Advances in MCDM Models, Algorithms, Theory and Applications*, Kulwer Academic Publishers, Boston, Dordrecht, London, 1999, 9.1-9.39.

[Wil74] M. R. Williams. “Heuristic Procedures - if they Work, Leave Them Alone”. *Software Practice and Experience* 4, 1974, 237-240.

[Woo68] D. C. Wood. “A System for Computing University Examination Timetables”. *The Computer Journal* 11, 1968, 41-47.

[WP67] D. J. Welsh, M. B. Powell. “An Upper Bound for the Chromatic Number of a Graph and its Application to Timetabling Problem”. *The Computer Journal* 10, 1967, 85-86.

[Wre96] A. Wren. “Scheduling, Timetabling and Rostering - a Special Relationship?”. E. Burke, P. Ross (eds.), *The Practice and Theory of Automated Timetabling: Selected Papers (ICPTAT '95)*. Lecture Notes in Computer Science 1153. Springer-Verlag, Berlin, Heidelberg, New York, 1996, 46-75.

[WX01] G. M. White, B. S. Xie. “Examination Timetables and Tabu Search With Longer Term Memory”. E. Burke, W. Erben (eds.), *The Practice and Theory of Automated Timetabling III: Selected Papers (PATAT 2000)*. Lecture Notes in Computer Science 2079. Springer-Verlag, Berlin, Heidelberg, New York, 2001, 85-103.

[YR03] J. Yanez, J. Ramirez. “The Robust Coloring Problem”. *European Journal of Operational Research* 148, 2003, 546-558.

[ZDT00] E. Zitzler, K. Deb, L. Thiele. “Comparison of Multiobjective Evolutionary Algorithms: Empirical Results”. *Evolutionary Computation* 8(2), 2000, 173-195.

[Zel73] M. Zeleny. “Compromise programming”. J. L. Cochrane, M. Zeleny (eds.), *Multiple Criteria Decision Making*, University of South Carolina Press, Columbia, 1973, 262-301.

[Zel74] M. Zeleny. “A Concept of Compromise Solutions and the Method of Displaced Ideal”. *Computers and Operational Research* 1, 1974, 479-496.

[Zel82] M. Zeleny. “Multiple Criteria Decision Making”. McGrow-Hill Book Company, 1982.

[Zit99] E. Zitzler. “Evolutionary Algorithms for Multiobjective Optimisation: Methods and Applications”. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1999.

[ZT99] E. Zitzler, L. Thiele. “Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach”. *IEEE Transactions on Evolutionary Computation* 3(4), 1999, 257-271.