University of Exeter
Department of Computer Science

University Examination Timetable Optimisation: Analysis, Initialisation, and Effective Heuristic Optimisation

Amjad Alasmar Alsuwaylimi

November, 2020

Supervised by Professor Jonathan Fieldsend & Dr Alberto Moraglio

Abstract

In higher education institutions, particularly universities, the task of scheduling examinations is a heavily constrained problem involving the allocation of exams, and corresponding enrolled students, to examination rooms over a limited number of periods. This is commonly performed by heuristic optimisers, as the task is NP-complete. The research work presented in this thesis focuses on examination timetabling with the aim of investigating three main areas: (i) initialising efficient seeded solutions for starting examination timetabling heuristics, (ii) developing a novel genetic algorithm-based examination timetabling optimiser, and (iii) analysing and comparing published works from the literature.

The new iterative initialisation algorithm presented here attempts to generate legal and high-quality solutions, to feed into a heuristic optimiser. Subject to satisfying hard constraints, it schedules as many conflicting examinations as possible in the early and late periods of a timetable, whilst managing the soft constraints. The proposed initialisation strategy is empirically verified on problem instances from two different benchmark sets: ITC 2007 and Yeditepe, and compared to a number of popular initialisation approaches. The effectiveness of this approach is also evaluated via incorporation in an exemplar evolutionary algorithm.

This thesis also investigates a novel genetic algorithm that incorporates new operators to avoid various types of violations that occur in the exam timetable optimisation task. It utilises the initialisation approach developed earlier in the thesis, as well as specialised operators for search. It is validated on a range of problems and is seen to produce results that place its performance amongst the state-of-the-art.

The third area of investigation of this thesis is concerned with issues in the comparison between published works on examination timetabling in the literature. Such comparison is often difficult and can be misleading because results obtained differ significantly in run-times — even when variations in computational power are accounted for. Consequently, a multi-objective comparison scheme based on (uncertain) Pareto dominance is presented and utilised with the aim of comparing published exam timetabling approaches on the Toronto benchmark sets to identify the (probabilistic) Pareto set of optimisers.

Acknowledgements

I would like to take this opportunity to personally thank a number of people for their help and support during my PhD study. First and foremost, I would like to express my profound gratefulness and deep honours to my first supervisor, Professor Jonathan Fieldsend, for his constant guidance, support, help, encouragement, and constructive comments throughout this study. I really appreciate his dedicated commitments and professionalism in supervising my research study.

Also, I would like to express my gratitude to Dr Alberto Moraglio, my second supervisor, for all the positive, constructive comments in my work that he has always given to me despite his busy schedule and commitments. I also profoundly thank Dr Ke Li and Dr Christine Mumford as my examiners, who already gave valuable feedback to improve the thesis.

I would also like to thank the Northern Border University of Saudi Arabia for the doctoral scholarship and other financial support throughout the course of my PhD study.

Special thanks to my lovely wife Reham Alanazi, my kids Feras and Dana, who always bring joy and laughter; thanks for being there for me. Finally, thank you very much to my parents, brothers and sisters for their support, love, and prayers.

Contents

List of Tables

List of Figures

List of Algorithms

Nomenclature and Abbreviations

List of Acronyms

$\mathcal{P}$	Pareto set *(p. 161)*
AVG RK	Average Ranking *(p. 143)*
EGD	Extended Great Deluge *(p. 46)*
FD	Flex-Deluge *(p. 46)*
NLGD	Non-Linear Great Deluge *(p. 46)*
NP	Non-deterministic Polynomial *(p. 12)*
NSGA-II	Non-dominated Sorting Genetic Algorithm-II *(p. 62)*
PAES	Pareto Archived Evolution Strategy *(p. 62)*
PESA-II	Pareto Enveloped based Selection Algorithm-II *(p. 62)*
PESA	Pareto Enveloped based Selection Algorithm *(p. 62)*
SPEA2	Strength Pareto Evolutionary Algorithm2 *(p. 62)*
SPEA	Strength Pareto Evolutionary Algorithm *(p. 62)*
ACO	Ant Colony Optimisation *(p. 52)*
BL	Back List *(p. 82)*
BS	Back Section *(p. 79)*
CD	Conflict Density. *(p. 16)*
CP	Constraint Programming *(p. 27)*
DM	Deep Mutation *(p. 124)*
EAs	Evolutionary Algorithms *(p. 3)*
ELM	Elite Light Mutation *(p. 129)*
ERW	Enhanced Roulette Wheel Selection Strategy *(p. 131)*
ESGA	Exam Specialised Genetic Algorithm *(p. 110)*
FL	Front List *(p. 82)*
FS	Front Section *(p. 79)*
GAs	Genetic Algorithms *(p. 48)*
GC	Graph Colouring *(p. 29)*
GDA	Great Deluge Algorithm *(p. 46)*
GRASP	Greedy Randomised Adaptive Search Procedure *(p. 31)*
HC	Hill-Climbing *(p. 38)*
HH	Hyper-heuristics *(p. 58)*

HM	Heavy Mutation *(p. 127)*
ILP	Integer Linear Problem *(p. 28)*
ILS	Iterated Local Search *(p. 66)*
IP	Integer Programming *(p. 27)*
ITC 2007	International Timetabling Competition 2007 *(p. 19)*
LD	Largest Degree *(p. 77)*
LE	Largest Enrolment *(p. 77)*
LM	Light Mutation *(p. 129)*
LWD	Largest Weighted Degree *(p. 77)*
MAs	Memetic Algorithm *(p. 55)*
MILP	Mixed-Integer Linear Programming *(p. 28)*
MISTA	Multidisciplinary International Conference on Scheduling: Theory and Applications *(p. 9)*
ML	Middle List *(p. 82)*
MM	Moderate Mutation *(p. 121)*
MOEAs	Multi-objective evolutionary algorithms *(p. 59)*
MOPs	Multi-objective optimisation problems *(p. 59)*
MS	Middle Section *(p. 79)*
MSSH	Multi-Start Stochastic Hill-climbing *(p. 76)*
OBSI	Ordering-Based Scheduling Initialisation *(p. 86)*
OR	Operations Research *(p. 1)*
PATAT	Practice and Theory of Automated Timetabling *(p. 9)*
PBDM	Period Based Deep Mutation *(p. 126)*
PBM	Period Based Mutation *(p. 123)*
PL	Period List *(p. 86)*
PSO	Particle Swarm Optimisation *(p. 47)*
RD	Random Schedule Allocation *(p. 79)*
RK	Ranking *(p. 143)*
SA	Simulated Annealing *(p. 41)*
SCE	Shuffled Complex Evolution *(p. 56)*
SD	Saturation Degree *(p. 77)*
SI	Swarm Intelligence-based algorithms *(p. 3)*
SPL	Selected Period List *(p. 86)*
TS	Tabu Search *(p. 39)*
VNS	Variable Neighbourhood Search *(p. 44)*

List of Symbols

α	schedule of exams to rooms at periods *(p. 86)*
R_p	available room in period. *(p. 86)*
rp	get a random period *(p. 86)*
α	history coefficient (influence of pheromone) *(p. 53)*
β	heuristic coefficient (influence of heuristic information) *(p. 53)*
ΔE	the difference in the objective value (energy) between the current solution and the generated neighboring solution *(p. 41)*
η_{ij}	the heuristic information on edge (i, j) *(p. 53)*
ρ	evaporation rate *(p. 53)*
τ_{ij}	the amount of pheromone on edge (i, j) *(p. 53)*
$\|E\|$	number of exams *(p. 14)*
$\|P\|$	number of periods available *(p. 14)*
c	soft constraints cost *(p. 125)*
$C = \|E\| \times \|E\|$	Conflict Matrix *(p. 16)*
c_{ij}	the number of students enrolled for both exam i and j *(p. 16)*
E	set of exams *(p. 14)*
$eList$	selected exams list *(p. 121)*
$f(Sol)$	the fitness value of the initial solution *(p. 39)*
$f(Sol^*)$	the fitness value of a neighbouring solution *(p. 39)*
K_B	the Boltzmann constant *(p. 41)*
m	total number of ants *(p. 53)*
mp	number of periods to be mutated *(p. 123)*
Mut_{size}	the size of mutation *(p. 124)*
$MutL$	mutation list *(p. 119)*
$N(Sol)$	set of neighbours of the solution Sol *(p. 40)*
N_i^k	set of neighbourhood of ant k in node i *(p. 53)*
$N_k(Sol)$	set of solutions in the k^{th} neighborhood of solution *(p. 44)*
ne	number of exams *(p. 121)*
$Next\_Pop$	next population *(p. 132)*
np	number of periods *(p. 121)*
P	set of periods available *(p. 14)*
p_i	the period scheduled to exam i *(p. 18)*
P_{ij}^k	the probability of ant k location i choosing a location j *(p. 53)*
pc	crossover probability *(p. 48)*
pe_i	a penalty value imposed to the violation of a specific constraint *(p. 125)*
PL	period list *(p. 119)*
pm	mutation probability *(p. 48)*
Pop	population *(p. 49)*
$Popsize$	population size *(p. 48)*

R_p	set of room lists, available each period *(p. 120)*
$R_{exclusive}$	room exclusive (hard) constraint *(p. 119)*
R_{seats}	the available seats of room *(p. 119)*
$random[0,1)$	random variable function returning value in range [0,1) *(p. 41)*
RL	room list *(p. 119)*
rn	number of runs *(p. 130)*
S	a set of students *(p. 21)*
s	a student *(p. 21)*
SL	solutions list ordered by their cost values *(p. 124)*
Sol	initial solution *(p. 39)*
Sol'	best admissible neighbour *(p. 40)*
Sol^*	neighbour solution of the initial solution *(p. 39)*
sp	selection probability value *(p. 123)*
sr	solution rank *(p. 123)*
T	timetable *(p. 122)*
t	iteration *(p. 49)*
TE	current temperature *(p. 41)*
TE_0	initial temperature *(p. 41)*
$temp$	temporary list *(p. 121)*
w	weight associated with soft constraint *(p. 22)*
w^{2D}	weight for "Two Exams in a Day" *(p. 21)*
w^{2R}	weight for "Two Exams in a Row" *(p. 21)*
w^{FL}	weight for "Front Load penalty" *(p. 21)*
w^{NMD}	weight for "No Mixed Duration" *(p. 21)*
w^{PS}	weight for "Period Spread" *(p. 21)*
w_i	an attached weight *(p. 125)*
X_{ij}	one if exam i clashes with exam j (i.e. exam i and j have common students), and zero otherwise *(p. 14)*
X_p	the maximum number of exams that can be scheduled at period p *(p. 15)*
Y_{ip}	one if exam i is allocated to period p; otherwise Y_{ip} equals zero *(p. 14)*
Y_{jp}	one if exam j is allocated to period p; otherwise Y_{jp} equals zero *(p. 14)*
Z_{ip}	the cost of scheduling exam i in period p *(p. 14)*
Max	maximum number reached by random *(p. 121)*
offs	new solutions (offspring) *(p. 121)*
$TBList$	tabu list *(p. 40)*
UP	parameter that represents the rain speed *(p. 46)*
$LEVEL$	the water level *(p. 46)*

Chapter 1

Introduction

1.1 Background and Motivation

When solving a problem, people usually search for the best possible solution from those available, using background information and choosing between candidate solutions from a *search space* (i.e. the number of possible solutions to the problem). This work is concerned particularly with solving timetabling problems. Timetabling problems occur in various fields, including in healthcare (e.g. Burke et al. (2004c)), education (e.g Kristiansen and Stidsen (2013)), sports (e.g. Easton et al. (2004) and Trick (2011)), and transportation (e.g. Leake (1996)). Timetabling problems have become a research area in Operations Research (OR) and Artificial Intelligence (AI), with a number of conferences having been dedicated to the topic. Within the education domain, educational timetabling can be classified into three categories (Schaerf, 1999): School timetabling (Abdullah et al., 2005), Course timetabling (Abdullah et al., 2005), and Exam timetabling (Qu et al., 2009b). The similarities and differences between these categories are discussed in Chapter 2.

In the field of higher education, the problems are known as University Timetable Problems. These consist of allocating meetings between teachers and students within a given period of time while satisfying various constraints. University timetabling problems are often large as there are more programmes, classes, and resources than in secondary or primary education scenarios. There are also more interactions because students have more choices, while different programmes require similar teaching. Higher education timetabling has therefore been proven to be difficult tasks faced by educational institutions (Bardadym, 1995). The need for automated scheduling to build a feasible timetable that satisfies the requirements of all students and teachers has become increasingly difficult in recent years due to the increasing proliferation in the number of students and courses as universities expand, and as the options available within degree programmes increase. Matched with this, there are limitations in some other facilities like rooms and periods, etc., which tend to lag in their corresponding increase (if they increase at all). This has motivated many researchers and commercial software providers to use artificial intelligence, mathematical models, meta-heuristic algorithms, and other algorithms to assist staff in building high-quality solutions to their timetabling problems.

In this thesis, the focus is the university examination scheduling problem. This problem involves an administratively taxing task, which is frequently repeated in various academic institutions for each course session. Furthermore, in recent years, this problem has become increasingly challenging because of the rise in the number and complexity of students' examination enrolments (i.e. students are given the flexibility to enrol in different modular courses across faculties) (Burke et al., 1994a; McCollum, 2007).

Recently, evolutionary algorithms and meta-heuristic approaches have attracted a lot of attention as potential solutions to such examination timetabling problems. Such problems involve many constraints that must be satisfied simultaneously, such as meeting regulations, using resources properly, and meeting people's requirements to which evolutionary algorithms are particularly well-suited (Silva et al., 2004). The efficiency of these approaches is often dependent on the initial population, which is optimised from, however. An initial solution in examination timetabling problem is normally constructed using an appropriate heuristic (or set of initial solutions), then evolutionary algorithms or meta-heuristics are employed to improve this initial solution. Generally speaking, therefore, high quality and diverse initial solutions will improve the performance within an evolutionary timetabling algorithm. In the current state-of-the-art timetabling solutions, however, while a lot of work has been directed towards developing effective search heuristics, there has been less consideration of the task of generating the initial solutions (therefore identifying more promising solutions in the space search to start from) for the timetabling problem, in a way that meets as many constraints as possible.

Common approaches to solving examination timetabling problems typically include two phases: a construction phase and an improvement phase (Hertz, 1991). The basis of the construction phase are approaches such as Graph-Colouring (Carter and Laporte, 1996; Burke et al., 2007), Fuzzy Logic (Asmuni et al., 2005b) and Neural Networks (Corr et al., 2006a) that start with an empty solution and then attempts to construct a complete (final) solution. In contrast, in the improvement phase, the focus is on repeatedly improving the complete solution until an optimal solution is produced. Some examples of improvement approaches are: Tabu Search (Di Gaspero and Schaerf, 2000; White and Xie, 2000; Kendall and Hussin, 2005b), Simulated Annealing (Burke and Newall, 2003; Thompson and Dowsland, 1998) Memetic Algorithms (Burke et al., 1995c; Burke and Silva, 2005), Genetic Algorithms (Burke et al., 1995b), Ant Colony Optimisation (Dowsland and Thompson, 2005; Eley, 2007), Particle Swarm Optimisation (Chu et al., 2006), the Great Deluge Algorithm (Burke et al., 2004a), as well as hybridisations of distinct heuristic methods (Caramia et al., 2001; Merlot et al., 2003).

The other well-known objective of examination timetabling in the literature is to solve this problem and obtain high-quality timetables. To this end, many meta-heuristic approaches have been proposed and applied by researchers (Burke et al., 1996; Carter et al., 1996; Qu et al., 2009b). These research have been devoted to investigate various approaches, develop new methods in order to solve the problem effectively and produce promising findings. However, due to the inherent complexity of the problem, there are still opportunities for improvements in the current state-of-the-art.

Some of these approaches are population-based. In these approaches, an algorithm works on several solutions, attempting to enhance them. The population-based approaches can be characterised into Swarm Intelligence (SI)-based algorithms and Evolutionary Algorithms (EAs) (Yang, 2010; Dréo et al., 2006).

SI depends on mimicking the cooperative problem-solving behaviour exhibited in social colonies' collective intelligence within a self-organised system (Burke and Newall, 2004; Kirkpatrick et al., 1983; Clerc and Kennedy, 2002), which arises from the local communication between individuals and the environment (Burke and Bykov, 2008). These characteristics of SI motivated researchers to use such behaviour to tackle optimisation problems (Blickle and Thiele, 1995). SI-based algorithms, which include the Ant Colony Optimisation algorithm (Dowsland and Thompson, 2005), Fish Swarm Optimisation algorithm (Turabieh and Abdullah, 2011a), and Artificial Bee Colony algorithm (Alzaqebah and Abdullah, 2011), have been widely employed to solve the examination timetabling problems in the existing literature.

Evolutionary algorithms, meanwhile, comprise several heuristics that can solve optimisation problems through the imitation of some natural evolutionary aspects. An example of an evolutionary algorithm is the Genetic Algorithm (GAs) (Holland, 1992). Genetic algorithms have been the evolutionary algorithms most commonly used by researchers to solve combinatorial optimisation problems (Mohammed et al., 2017). Several researchers have used GAs to solve examination timetabling problems (Corne et al., 1993, 1994b; Paechter et al., 1994; Burke et al., 1994a; Pillay and Banzhaf, 2010). They concluded that conventional GAs had not generated good results compared to a number of approaches developed for the examination timetabling problem. However, modified basic genetic operators have been seen to enhance the algorithm's performance (Zhong et al., 2013). This observation, therefore, motivated us to establish a further goal for this research, to focus on enhancing conventional genetic algorithms to improve initial solutions obtained during the construction phase.

The construction of an examination timetabling problem is a challenging task and quite often time-consuming. It is concerned with assigning exams to a specific number of periods so as to satisfy a given set of constraints (Balakrishnan, 1991; Schaerf, 1999; Qu et al., 2009a). Many approaches have been investigated in an attempt to solve examination timetabling problems (as we shall see later in Chapter 2). These approaches often employed ordering strategies (i.e. different graph colouring heuristics) in order to construct examination timetables in the construction phase. Since none of the ordering strategies provides a guarantee of successful scheduling, there have been extensive studies on constructive approaches reported in scientific literature. Most of these incorporated graph colouring heuristics have employed some adaptive strategies (Rahman et al., 2009; Burke and Newall, 2004), fuzzy techniques (Asmuni et al., 2009; Pais and Burke, 2010), decomposition (Abdul-Rahman et al., 2014a; Qu and Burke, 2007), and neural networks (Corr et al., 2006b). As a result, approaches related to these ordering strategies are used to construct a feasible timetable before proceeding to improve the solution quality. Furthermore, several studies have shown that a high quality initial solution influenced the performance of the search algorithm e.g. Burke and Newall (2003); Gogos et al. (2012). However, most previous researches published that have been surveyed do not emphasise the importance of generating good quality solutions where administration requirements are considered in the construction phase without taking into account markers' and students' preferences. Therefore, this has motivated this research to investigate whether the use of a new initialisation method could be of benefit in the construction phase and the influence of this method on the solution quality when an improvement approach is employed, as it is an under-explored area.

In addition, an observation of the recent examination timetabling literature shows that there are a number of issues encountered in comparison between approaches proposed to solve examination timetabling problems. Research published in this area often proposes new optimisers, which are evaluated against common test problems but run for very different durations (and thus potentially computation times) than the prior work they are compared to. Additionally, a variable range of repeats is also used (with the best found across repeats taken as a measure of quality). Such issues mean it is difficult to interpret and rank optimisers fairly based on previously published results alone. Therefore, this observation also motivated us to address some of these concerns, through developing a comparison framework.

1.2 Research Objectives

As mentioned above, this thesis focuses on the examination timetabling problem. As such, besides aiming to propose a strategy that could be used to establish a good starting point for subsequent exam timetabling optimisation heuristics, we are aiming to produce feasible solutions for all instances, with better quality and diversity while also being competitive with other algorithms specially developed for solving examination timetabling problems.

To be specific, the research aims at proposing a new initialisation approach, whose main novelty stems from the way it manages allocation among three interacting lists of exams. These lists are arranged and processed in a step-wise fashion in order to provide a good satisfaction of hard and soft constraints. In addition to the development of a new initialisation approach (as mentioned earlier), we aim to investigate a meta-heuristics approach to minimise the total cost of soft constraints. An improvement methodology involves an exam specialised genetic algorithm which incorporates novel operators for this problem. The exam specialised genetic algorithm employs our developed initialisation method in order to take advantage of good initial search populations. Furthermore, it is designed with the aim of effectively exploring and efficiently exploiting the solution space. Moreover, the crossover operator is eliminated in order to produce a high-quality examination timetable in a shorter time compared to the conventional genetic algorithms. The study also aims to address a number of issues identified with the common comparison between published works on examination timetabling in the literature and derive an uncertain Pareto analysis to mitigate uncertainties in effective computation capabilities between published works.

The objectives listed below sum up the scientific aim of the work reported in this thesis. The first objective concentrates on the construction of high quality solutions while the second objective seeks to further improve the quality of these solutions. The third objective analyses results from the last 25 years of exam timetable optimisation, comparing and contrasting these using a specially developed framework. Each of the objectives is discussed fully in the corresponding chapters.

- To propose a novel initialisation algorithm to seed the initial population in order to provide a good starting point for subsequent optimisation of the exam timetabling solution by means of a meta-heuristic algorithm.

- To propose an optimisation method that incorporates novel operators (directed and undirected mutations and an enhanced selection strategy) which are applied to examinations or periods to improve the quality of the final solution.

- To develop a framework to compare the published exam timetabling results on the Toronto benchmark sets to identify optimal approaches whilst explicitly accounting for the uncertainly in timing due to potential differences in effective computing power used across studies.

1.3 Research Contributions

The main contributions of the research reported in this thesis are that it proposes a novel heuristic for constructing examination timetabling and that it is the first study to investigate an uncertain multi-objective analysis of published examination timetabling approaches. The specific contributions are described below:

- The presentation of a new initialisation method for constructing examination timetables. This shows an improvement in terms of the quality and diversity of the solution when compared with other initialisation methods from the literature (i.e. different graph colouring heuristics and random scheduling).

- Application of an exam specialised genetic algorithm adapted with novel operators to solve capacitated and uncapacitated examination timetabling problems. The approach is incorporated with the new initialisation method to improve the solutions obtained within reasonable computational times. Notably, the study introduces a number of novel mutation operators as well as an enhanced roulette wheel selection strategy. It is concluded that the combination of the novel initialisation method proposed in this thesis with the proposed algorithm adapted with the new operators can significantly improve solutions and avoid various types of violation. Moreover, the finding of this research is that the proposed algorithm is competitive with many state-of-the-art schedulers from the literature (after comparison with 16 algorithms in respect to an uncapacitated problem (the Toronto dataset), and 27 algorithms in respect to capacitated problems (both the ITC 2007 and Yeditepe datasets).

- The investigation, for the first time, of a bi-objective comparison scheme based on (uncertain) Pareto dominance in order to identify the uncertain Pareto front, and dominated algorithms, for published approaches designed to solve the Toronto benchmark sets. Although many approaches have been developed and proposed in the last few decades, this analysis suggests that among the top-ranked approaches are our proposed initialisation approach and an approach from Burke and Bykov (2008).

1.4 List of Publications

The following publications have been conducted as a result of this research. The chapter in this thesis which describes the content of the publication, is also presented.

- Amjad. A. Alsuwaylimi and Jonathan. E. Fieldsend, "A New Initialisation Method for Examination Timetabling Heuristics", 2019 IEEE Symposium Series on Computational Intelligence (SSCI), Xiamen, China, 2019, pp.1636-1643 (Chapter 3).
- Amjad. A. Alsuwaylimi and Jonathan. E. Fieldsend, "An uncertain multi-objective analysis of 25 years of exam timetable optimisation". Submitted to *the Journal of Information Sciences* (Chapter 5).

1.5 Overview of the Thesis

This thesis is organised into six chapters. This chapter has presented the introduction, objectives, and the contribution claimed for the research and an outline of the dissemination of the research work. The remainder of this thesis is structured as follows:

Chapter 2 provides details of the fundamental aspects of the research area that will be tackled, introduces various educational timetabling problems and concentrates upon particular research issues concerned with university examination timetabling problems. It describes the benchmark datasets widely used in the examination timetabling community and reviews the different algorithms and approaches investigated in respect to examination timetabling problems, with a focus on the recent approaches applied to the benchmarks.

Chapter 3 elaborates on the initialisation approach proposed in this research. In the course of this chapter, popular initialisation approaches from the literature and all the main producers involved in constructing initial examination timetables are described. Furthermore, this chapter presents comparisons between our proposed initialisation and other popular initialisation strategies in the initialisation phase, as well as when they are incorporated within a simple Evolutionary Algorithm. The implementation and experimental analysis of these strategies are discussed in respect to two well-known capacitated benchmark datasets.

Chapter 4 presents an exam specialised genetic algorithm that can utilise the higher quality seed solutions that have been generated by our proposed initialisation method (see Chapter 3) to optimise a problem more effectively. The proposed approach and mutation operators are presented in this chapter. A comparison is undertaken between the proposed approach and a basic Genetic Algorithm for both capacitated and uncapacitated benchmark datasets. In addition, a comparison is made with state-of-the-art approaches.

Chapter 5 presents practical issues in examination timetabling and performance assessment. It outlines the fundamental problems that arise when comparing the algorithm performances published in the literature. It also summarises the performance of popular exam timetabling optimisation approaches in respect to common exam timetabling benchmarks. The popular capacitated and uncapacitated benchmarks used in this study are briefly described. This chapter also describes a multi-objective analysis of the results that can address some of the issues that have been identified, and presents a comparison of 16 different approaches using the uncapacitated benchmark.

Finally, the overall conclusions of the work presented in this thesis, and opportunities for possible future research are presented in Chapter 6.

Chapter 2

Background and Literature Review

2.1 Introduction

This chapter provides details of the fundamental aspects of the research area. First, it describes the general timetabling problem, specifically the university timetabling problem and the related constraints that need to be considered in the problem. Next, it provides an overview of examination timetabling and identifies methods used in it. Then, it describes the literature relevant to the university examination timetabling problems, focusing on automated solutions to these problems and the most influential works for the scope of the current study.

Broadly, the remaining sections of the chapter fall into two parts. The first part, in Sections 2.2 through to 2.3, describes the timetabling problem, moving from the general articulation of what timetabling involves through to the specific issue of university examination timetabling. The second part, in Sections 2.4 to 2.5, introduces the various techniques that have been applied to the examination timetabling problem, moving from the distinction between exact, constructive, and improvement techniques with advantages and disadvantages of each of these techniques in Subsections 2.4.1, 2.4.2, and 2.4.3, through to the detail of specific meta-heuristic algorithms in Subsection 2.4.3, and the specific research in relation to hybrid meta-heuristics, hyper-heuristics and multi-objective approaches in Subsections 2.4.4 to 2.4.6. Section 2.5 discusses which of these various approaches are the current state-of-the-art in university examination timetabling. Section 2.6 presents the insights and motivations obtained by this background study and literature review. Finally, section 2.7 summarises the chapter.

2.2 Overview of Educational Timetabling Problems

Timetabling problem is considered to be a highly constrained scheduling problem. It is a challenging and complex problem in the AI and OR and has been actively researched since the 1960s. During that time, many conferences have been held to discuss the best possible ways to tackle this kind of problem, such as Practice and Theory of Automated Timetabling (PATAT), which specifically addressed the topic at the centre of this thesis. Other conferences explore timetabling in theory and practice, such as the Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA). In general, timetabling can be classified into many different categories, such as educational timetabling, transportation timetabling, nurse rostering, and sports timetabling. Educational timetabling is the problem that has been most extensively explored among those interested in artificial intelligence (Lewis, 2008; Qu et al., 2009b). Figure 2.1 shows a diagram of the university timetabling problem.

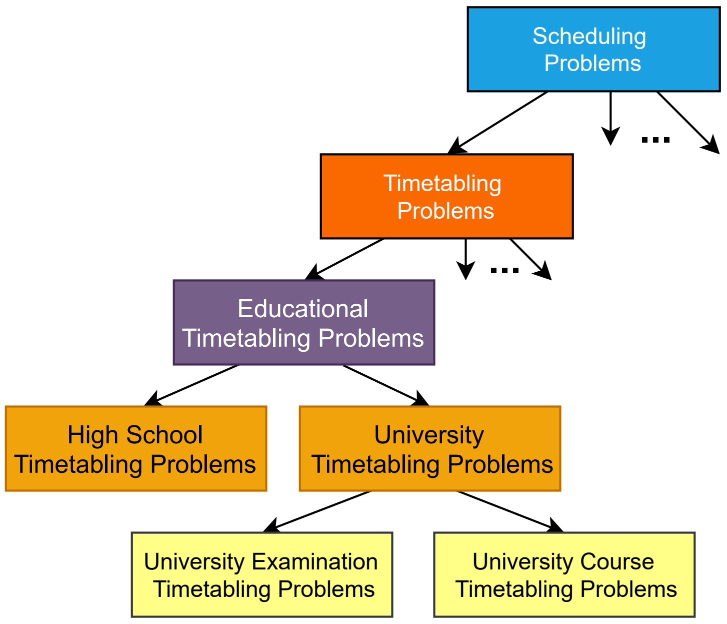
Scheduling Problems
...
Timetabling Problems
...
Educational Timetabling Problems
High School Timetabling Problems
University Timetabling Problems
University Examination Timetabling Problems
University Course Timetabling Problems

Figure 2.1 Diagram of the university timetabling problem (Babaei et al., 2015a).

There are some conflicting views related to the terms *timetabling* and *scheduling* that can be found in the literature. For instance, Wren (1996) defined the term *timetabling* as the process of assigning given resources into a limited number of available periods and locations in order to achieve the highest potential satisfaction of a set of stated objectives, while the term *scheduling* is referred to the process of assignment of certain resources into periods and places with the aim of decreasing the total cost of resources employed. On the contrary, Carter (2001) emphasised that the term *timetabling* determines when events will occur but does not usually entail allocating resources as *scheduling* does. For instance, the process of creating a university course calendar typically does not include identifying which professors will be assigned to which specific course. This information is usually determined upon well before creating the course timetable. However, we believe that minor distinctions between various perspectives on what is meant by the terms *timetabling* and *scheduling* are not significant. The critical point is that scientific progress undertaken in addressing and understanding timetabling problems from many application domains will be made by depending on the state-of-the-art in other scheduling problems and vice versa. Hence, both terms are considered equivalent in this work and will be used interchangeably in the text.

In general, Burke et al. (2004d) defined the timetabling problem as the problem that consists of a limited number of periods, a set of resources, a given number of meetings, and a set of pre-defined schedule requirements (i.e. constraints). The main objective is that periods and resources are assigned to the meetings while constraints should be satisfied as much as possible. The following terms are commonly used in the timetabling problem:

- Event: the activity that will be scheduling.
- Period: the duration of time during which the event can be allocated.
- Resources: the distinct resources that the event needs, such as rooms.
- Constraints: restriction of an event in terms of where and when it must be scheduled. There are two types of constraints: soft (which are preferred to be satisfied) and hard (which must be satisfied).
- Individual: a person who must be present at an event.
- Conflict: the scheduling of two events that have at least one individual in common in the same period.
- Instance: the problem that will be solved – it often includes number of exams, total number of students enrolled, number and capacity of available rooms, and number of periods.
- Examination session: this term involves a fixed length of examination timetable (i.e. number of periods over a specified length of time) where all exams should be scheduled into those periods. Universities often organise two examination sessions throughout the year, which often last for two or three weeks.

As mentioned above, timetable constraints are identified as hard and soft. Hard constraints are those that must be achieved under any situation; soft constraints, however, can be broken when necessary. The fitness of a solution to hard constraints is specified by the degree of complete fulfilment. While in soft constraints, the degree of satisfaction is used to determine the number of satisfied soft constraints. Based on a survey (Qu et al., 2009b), the most studied area for the timetabling problem is educational timetabling, which includes school timetabling and university timetabling (course and examination). Schaerf (1999) and Melício et al. (2004) also categorised educational timetabling problems into three types:

1. School timetabling: the classes are distributed across the week, ensuring that no teacher has two classes at the same time and that no class has two teachers.
2. University examination timetabling: the university exams are scheduled in a manner that avoids a conflict between exams for courses sharing some students while trying to expand the time between students' exams as much as possible.

3. University course timetabling: the scheduling of lectures across the week while minimising overlaps between lectures for courses that share some students.

There are, however, distinctions in the three kinds of timetabling addressed; for example, the duration of examination sessions differs based on the institution. Whereas the course and school timetable are usually organised based on weekly rules. A more detailed description of the differences between these types can be found in (Carter and Laporte, 1996). For a full description of the differences between university examination and course timetabling problems, see (McCollum, 2007)

2.2.1 University Timetabling Problems

The timetabling problems in universities are known to be highly constrained optimisation problems. Finding a feasible and optimal timetable is a challenging task due to the extreme constraints imposed by limited rooms and periods, alongside the complex mix of course options available to students. In recent years, the need for automated timetabling has become very important at universities due to the increase in the number of students, courses, and also teachers, and the limitation of some other facilities like rooms, labs, etc. This combination makes it impractical to construct a timetable manually, and this has motivated many researchers to attempt to apply artificial intelligence, mathematical models, meta-heuristic algorithms and other algorithms to build high-quality models to solve university timetabling problems.

In itself, the task of optimising timetabling problems is not a new one, with research stretching back at least five decades. Gotlieb (1962) provided the earliest research in this field. The author tried to address curriculum-based course timetabling and considered that each class could be processed by only one teacher at a time for a group of students, while the number of periods was freely chosen.

The general timetabling problem can be expressed as follows: numerous events must be timetabled by allocating them to specific periods. Such problems occur in a variety of domains such as universities and schools. In a university timetabling problem: a set of events, courses or exams are scheduled within a fixed number of rooms and periods within a week or semester. University timetabling problems involve the allocation of courses or exams, students, teachers and rooms within a fixed number of periods while accounting for specific constraints.

A solution is either feasible or valid if it satisfies all of the hard constraints, and a feasible solution is the best solution if it satisfies all of the soft constraints as well (Silva et al., 2004). It is very difficult, or may even be impossible to satisfy all of the soft constraints, however. This complexity requires scheduling of a class or exam timetable to be treated as a solution over hard constraints and for optimisation over soft constraints (Rudová and Murray, 2002). Because the number of variables differ between different universities, however, according to their specific class or exam structures and constraints, it is difficult to form a general solution that can meet the requirements of all universities.

Many researchers have attempted to simplify the problem by considering only simple exam structures. A general technique, however, should consider different aspects to users when simplified. Instead, it should include all possibilities that can be simplified based on individual user requirements. Because the numbers and types of constraints differ between universities, they should be classified carefully to generalise the problem.

Rather than considering a particular university with only specific constraints, other constraints should be considered, which would satisfy most universities. These should fall under the category of hard constraints. All other constraints can be explicitly categorised as soft constraints. In other words, soft constraints should not be a part of the problem so that adding or deleting them should not affect the main problem or solution.

University examination timetabling	University course timetabling
A number of exams may be scheduled in one room or an exam may be divided between several rooms.	One course should normally be scheduled within one room.
Aims to decrease the number of student who have exams within adjacent periods.	Usually, it is preferred that students have two or more consecutive courses.

Table 2.1 The key differences between examination and course timetable problems.

As shown earlier in Figure 2.1, university timetabling problems can be classified into two main categories: university examination and course timetabling. Each category has its own requirements, constraints, and timetabling structure (Schaerf, 1999; Schaerf and Di Gaspero, 2001). The main significant differences between them are briefly summarised in Table 2.1. The main focus of this thesis is on the university examination timetabling problem. Therefore, the following sections will discuss the university examination timetabling problems in greater detail.

2.2.2 University Examination Timetabling Problems

Examination timetabling is a time-consuming, recurring and important administrative task in academic institutions (Qu et al., 2009b). It is often difficult and demanding, and it affects many people (administrators, academic staff, and students) (Prida Romero, 1982). The process of finding a feasible or optimal timetable for exams is often a challenging task due to the extremely constrained nature of these problems. This calls for the use of heuristic and meta-heuristic algorithms that do not guarantee an optimal examination timetable but are in many cases able to obtain a "good enough" timetable for practical purposes. Thus, it is impossible to carry out an exhaustive search for the examination timetable in a reasonable time (i.e. polynomial-time) because of the exponential growth of this problem. This, in turn, has been proved to lie in the set of NP-complete problems, according to some researchers, e.g. Karp (1972), de Werra (1985), de Werra (1997), Cooper and Kingston (1995), and Schaerf (1999).

Before defining an NP-complete problem, it is crucial first to introduce some notions related to computational complexity theory in order to understand the NP-complete problem. In computational complexity (which is a sub-field of theoretical computer science and mathematics), a finite size problem that can be solved by an algorithm in "polynomial time" and can be tracked belongs to the P problems that are said to be "easy problems". In contrast, a problem is called NP (which stands for Non-deterministic Polynomial) when its solution can be guessed and verified in non-deterministic polynomial time. In the NP problem, the execution times of the latter grow much more rapidly as the problem size increases. NP-Hard is the set where problems are at least as hard as NP. This means NP-Hard problems are equally or more complex than any problem in NP set. NP-Complete is the intersection set of NP and NP-Hard. A problem in the NP-Complete set can be reduced to any other NP-Complete problem. That means if any of the NP-Complete problems would have an efficient solution, then all of the NP-Complete problems could be solved with the same solution (Cook, 1971; Karp, 1972).

Examination timetabling is essentially the problem of scheduling the exams of courses avoiding conflicts between exams that have common students and ensuring that, as far as possible, individual students' exams are spread through the examination period (Schaerf, 1999). Additional factors that have to be taken into account are the availability of rooms, and the preferences of lecturers or markers of the exams (e.g., do they expect to have enough time for marking, especially for exams with a large number of students). As with the course timetabling problem, the exams timetabling problem, therefore, has both hard and soft constraints, and the quality of the timetable is defined by the extent to which soft constraints are met (Qu et al., 2009b). This feeds through to two "qualities" of solution: a feasible solution, where the hard constraints are achieved, and an optimised solution where there is an attempt to build on the feasible solution by meeting as many of the soft constraints as possible (Schaerf, 1999).

Many universities are witnessing a growing number of student enrolments in a wide variety of courses with more combined degree courses. This adds to the challenge of developing examination timetabling software. Burke et al. (1996) researched examination timetabling problems in universities in more than fifty universities. They found that the constraints vary significantly from one university to another. Some examples of hard constraints for examination timetabling are:

- Certain examinations have to be successive or take place in a specific sequence (after/before each other).
- Exams having the largest number of students should be scheduled first and earlier in the timetable to allow more time for marking.
- Examinations which are given by the same instructor, if scheduled in the same periods, must be assigned to neighbouring classrooms.
- No student should have two exams at the same time.
- Certain examinations have to take place in a specific room.
- There has to be enough seating capacity in the room for the number of students scheduled for it.

In general, the accepted hard constraints for examination timetabling problems are:

1. There must be enough seating capacity.
2. No student should have two exams at the same time (simultaneously).

Examination timetabling problems have the following characteristics (different from course timetabling problems) (Schaerf, 1999):

- There is only one exam for each subject.
- The conflicts condition is strict. It is acceptable that a student is forced to skip a lecture due to time clashes but not an exam.
- There are various types of constraints such as no more than one exam should take place per day for each student, or there should not be too many consecutive exams for each student.
- The number of periods may vary, unlike course timetabling where it is fixed.
- There can be more than one exam per room.

Various academic institutions impose a variety of constraints (Burke et al., 1996). Such variations make the examination timetabling problem more complex and challenging, since the variety of constraints may require different formulations of objective functions or evaluation functions. Hence, several models and formulations for examination timetabling problems have been presented by various researchers (as described later). For instance, de Werra (1985) provided a formal approach to the examination timetabling problem, which is based on a mathematical programming model. To clarify this approach, let us consider the following notations:

- E is a set of exams, E=$\{e_1, e_2, \ldots, e_{|E|}\}$.
- $|E|$ is the number of exams.
- P is a set of periods available, $P = \{p_1, p_2, \ldots, p_{|P|}\}$.
- $|P|$ is the number of periods available.
- Z_{ip} is the cost of scheduling exam i in period p.
- Y_{ip} is one if exam i is allocated to period p; otherwise Y_{ip} equals zero.
- Y_{jp} is one if exam j is allocated to period p; otherwise Y_{jp} equals zero.
- X_{ij} is one if exam i clashes with exam j (i.e. exam i and j have common students), and zero otherwise.

An examination timetable is considered feasible if the following hard constraints are fulfilled: (i) each exam must be assigned once, and (ii) any conflicting exams must not be assigned in the same period. The problem aims to produce an exam timetable using P

periods without violating these hard constraints. The objective is to minimise the cost of scheduling examination i in period p. This examination timetabling problem can be represented as adopted from Terashima-Marín (1998).

$$\min \sum_{i=1}^{|E|} \sum_{p=1}^{|P|} Z_{ip} Y_{ip} \tag{2.1}$$

Subject to:

$$\sum_{p=1}^{|P|} Y_{ip} = 1, \qquad \forall\ i \in \{1, \ldots, |E|\} \tag{2.2}$$

$$\sum_{i=1}^{|E|-1} \sum_{j=i+1}^{|E|} \sum_{p=1}^{|P|} Y_{ip} Y_{jp} X_{ij} = 0 \tag{2.3}$$

Equations (2.2) and (2.3) illustrate the hard constraints that must be satisfied. Equation (2.2) denotes that the timetable must be feasible where all examinations must be scheduled, and each examination must be scheduled only once. Equation (2.3) denotes that no student should sit for more than one examination in the same period. If examination i and examination j are scheduled in period p, the number of students sitting both examination i and j (X_{ij}) must be equal to zero, and this should be true for all examinations already scheduled. The other hard constraint to consider is room capacity. If X_p is the maximum number (capacity) of examinations that can be scheduled in period p, then the hard constraints can be represented as:

$$\sum_{i=1}^{|E|} Y_{ip} \leq X_p \qquad p \in P = \{p_1, p_2, \ldots, p_{|P|}\} \tag{2.4}$$

It can be said that the ultimate aim of the solution to the examination timetabling problem is to guarantee that all students can take any exams they are required to take. In addition, however, it is also desirable to use resources (such as rooms, periods and more) effectively. In order to achieve this, certain types of constraints must be satisfied, and this level of satisfaction gives an indication of quality for the timetable.

In the examination timetabling research community, some previously published works have shared benchmark datasets. Two extensively used benchmark datasets that many researchers have used, and continue to use, are the Toronto (Carter et al., 1996), and International Timetabling Competition 2007 (ITC 2007) benchmark datasets (McCollum et al., 2007, 2012a). The details of the most popular benchmark datasets are discussed in the following section.

2.3 Benchmark Examination Timetabling Datasets

This section introduces the benchmark datasets most widely used to evaluate the efficiency and effectiveness of proposed algorithms. These are from Toronto (Carter et al., 1996), Nottingham (Burke et al., 1995c), Melbourne (Merlot et al., 2003), and Yeditepe (Parkes and Ozcan, 2010). The Toronto dataset, among these four datasets, has been the most widely used by the examination timetabling community, for instance, it is used in many papers that can be found in the PATAT conference, such as Burke and Carter (1998), Burke and Erben (2003), Burke and Erben (2001), Burke and Trick (2005), and Özcan et al. (2019). More recently, the Second International Timetabling Competition (ITC 2007) dataset has gained prominence. This test suite is taken from anonymised institutions for the purpose of competition use (McCollum et al., 2007). It adds additional real-world constraints to the problem, as well as new forms of evaluation and data. The introduction of these datasets has encouraged researchers to develop many approaches.

Examination timetabling constraints vary from institution to institution because each one has different requirements and constraints to suit its business model. Moreover, the stakeholders of the examination timetable can be affected due to them having different preferences in terms of a high-quality timetable. For instance, an administrator might require scheduling all the exams avoiding conflicts (i.e. no student should be assigned to sit two exams at the same day); students prefer exams to be spread as much as possible to allow for revision time between exam papers. Commonly-used constraints in the examination timetabling problem are considered in this section.

Examination timetabling problems can be categorised as Capacitated or Uncapacitated (Kahar and Kendall, 2010; Aldeeb et al., 2019). With regard to the capacitated problem, room capacity is considered as a hard constraint. On the other hand, room capacities are not considered in the uncapacitated problem. The Toronto dataset is uncapacitated problems whilst the Nottingham, Melbourne, Yeditepe, and ITC 2007 datasets are capacitated problems.

In the examination scheduling literature, the characteristics of all the benchmark problems are presented as data for each problem (i.e. instance or problem instance) involved in tables which are organised based on the name of institution, followed by the name of each instance, the total number of students enrolled for the examination session, a number of exams exists in the instance, a number of total enrolments of students for the courses, the density of conflict, and required number of periods for each instance.

The Conflict Density (CD) of each problem instance demonstrates the difficulty with respect to exams in conflict. In order to calculate the density of the conflicting exams in each of the instances, a Conflict Matrix, $C = |E| \times |E|$ where exams i,$j \in \{e_1, e_2, \ldots, e_{|E|}\}$($|E|$ is the number of exams). Element c_{ij} indicates the number of students registered for both exam i and exam j. In the matrix, each element c_{ij}= 1 if exam i has at least one common student with exam j (i.e. conflicting exams); c_{ij}= 0 otherwise (Broder, 1964). The CD represents the ratio between the number of elements of value “1” to the total number of elements in the conflict matrix. Formally, CD is presented by the formulation (Carter et al., 1996):

$$CD = \frac{\sum_{i=1}^{|E|-1} \sum_{j=i+1}^{|E|} (\frac{\beta_{ij}}{|E|})}{|E|} \tag{2.5}$$

As shown in Equation above, $|E|$ denotes the number of exams and β_{ij} is a decision variable assuming value one if exam i and exam j have at least one student in common while taking value zero otherwise. We now detail each benchmark dataset in turn.

2.3.1 University of Toronto Dataset

Problem Description

The Toronto benchmark set is one of the most widely used datasets in the literature for testing examination timetable optimisation algorithms. The dataset consists of 13 real-world exam timetabling problems introduced by Carter et al. (1996) that are collected from different academic institutions. Three instances are from Canadian highs schools, five instances from Canadian institutions, one instance from the London School of Economics, one instance from King Fahd University (in Dhahran) and the last instance from Purdue University. Table 2.2 shows the specifications for the Toronto datasets (Carter et al., 1996).

Hard Constraints

The dataset enforces only one hard constraint that prevents any student from taking two exams in the same period.

Soft Constraints

It has only one soft constraint that requires expanding the students' exams by five periods at least.

University	Problem Instance	No. of Exams	No. of Students	No. of Enrolments	No. of Periods	Conflict Density
Carleton University, Ottawa	car91	682	16,925	56,877	35	0.13
Carleton University, Ottawa	car92	543	18,419	55,522	32	0.14
Earl Haig Collegiate, Toronto	ear83	190	1125	8109	24	0.27
Ecole des Hautes Etudes Commercials, Montreal	hec92	81	2823	10,634	18	0.20
King Fahd University, Dharan	kfu93	461	5349	25,113	20	0.06
London School of Economics	lse91	381	2726	10,918	18	0.06
Purdue University, Indiana	pur93	2419	30,029	120,681	42	0.03
Ryerson University, Toronto	rye92	486	11,483	45,051	23	0.07
St. Andrews High School, Toronto	sta83	139	611	5751	13	0.14
Trent University, Peterborough	tre92	261	4360	14,901	23	0.18
University of Toronto, Arts & Science	uta92	622	21,266	58,979	35	0.13
University of Toronto, Engineering	ute92	184	2749	11,793	10	0.08
York Mills Collegiate Institute, Toronto	yor83	181	941	6034	21	0.29

Table 2.2 The characteristics of problem instances from the Toronto benchmark dataset.

Problem Formulation

The formulation of the Toronto problem is defined by Carter et al. (1996). This formulation is simplified as the only hard constraint taken into account is the exam conflict and one soft constraint (as discussed above). In order to clarify this formulation, several notations are considered as follows:

- $E = \{e_1, e_2, \ldots, e_{|E|}\}$, the exams set.
- $P = \{p_1, p_2, \ldots, p_{|P|}\}$, the set of predefined periods.
- $C = |E| \times |E|$, the conflict matrix, where each element, in the matrix, c_{ij} denotes the number of common students in exam i and exam j, where exams i, $j \in E$.
- $|S|$, the total number of students in a given problem instance.
- c_{ij}, the total number of students enrolled for both exam i and j. $c_{ij} = 1$ if exam i conflict, i.e. have at least one common student, with exam j or $c_{ij} = 0$, otherwise.
- p_i, the period scheduled to exam $i(i \in \{1, \ldots, |E|\})$ within the set of predefined periods $(1 \leq p_i \leq |P|)$.

The soft constraint is defined by the function f_c as shown in Equation (2.6).

$$f_c = \frac{1}{|S|} \sum_{i=1}^{|E|-1} \sum_{j=i+1}^{|E|} c_{ij} \times \mathit{proximity}(i,j) \tag{2.6}$$

where:

$$\mathit{proximity}(i,j) = \begin{cases} \frac{2^5}{2^{|p_i - p_j|}} & \text{if } 1 \leq |p_i - p_j| \leq 5, \\ 0 & \text{otherwise.} \end{cases} \tag{2.7}$$

subject to:

$$\sum_{i=1}^{|E|-1} \sum_{j=i+1}^{|E|} c_{ij} \times \delta_{(p_i,p_j)} = 0, \quad \delta_{(p_i,p_j)} = \begin{cases} 0, & p_i \neq p_j \\ 1, & p_i = p_j \end{cases}. \tag{2.8}$$

Equation (2.6) represents the soft constraint that aims to reduce the number of exams allocated in adjacent periods in such a way as to minimise the number of students sitting exams in close *proximity*. Equation (2.7) determines the penalty that can be incurred by scheduling the exams i and j in periods p_i and p_j, respectively. The weight of the penalty is 16, 8, 4, 2, and 1, for exams scheduled within one, two, three, four, and five periods, respectively. If the period spread is more than five, the penalty weight is equal to zero. Equation (2.8) defines the only hard constraint, requiring that there can be no conflicts between exams that are scheduled in the same period.

2.3.2 International Timetabling Competition 2007 Dataset

Problem Description

The second international timetabling competition (ITC 2007) included three tracks, namely Track 1 — examination timetabling, Track 2 — post-enrolment-based course timetabling, and Track 3 — curriculum-based course timetabling (Mccollum et al., 2010). Here we concentrate on its examination datasets, which are derived from real-world timetabling problems. The exam timetabling track involves eight publicly available instances and four hidden instances. Each instance has distinct characteristics and constraints (i.e. soft and hard constraints) that are similar to those constraints encountered in practice. Table 2.3 illustrates the characteristics of these problem instances.

Problem Instance	No. of Exams	No. of Students	No. of Enrolments	No. of Periods	Conflict Density	Tot.Room capacity
Exam1	607	7891	32,380	54	0.05	802
Exam2	870	12,743	37379	40	0.01	4076
Exam3	934	16,439	61,150	36	0.03	5212
Exam4	273	5045	21,740	21	0.15	1200
Exam5	1018	9253	34,196	42	0.009	2395
Exam6	242	7909	18,466	16	0.06	2050
Exam7	1096	14,676	45,493	80	0.02	2530
Exam8	598	7718	31,374	80	0.05	922
Exam9	169	655	2532	25	0.08	170
Exam10	214	1577	7853	32	0.05	1914
Exam11	934	16,439	61,150	26	0.03	4924
Exam12	78	1653	3685	12	0.18	1525

Table 2.3 The characteristics of problem instances from the ITC 2007 benchmark dataset.

All exams have to be scheduled (a complete solution), and exams cannot be divided between rooms and periods. A feasible solution must satisfy all the hard constraints, whereas the quality of an examination timetable is determined by the soft constraint violations subject to this. The examination timetabling problem specified in ITC 2007 is an extension to the previous model, which is formulated in Toronto specification (Leite et al., 2018), in which novel hard constraints, as well as soft constraints, are introduced. The following is the extended set of hard constraints, as well as soft constraints (McCollum et al., 2012b).

Hard Constraints

The hard constraints are listed below (Mccollum et al., 2010):

H1: No Conflicts – A conflict can exist between a pair of exams, which are attended by a specified student when these exams are scheduled in a same period, or in the same period. The hard constraint necessitates that such conflicts do not exist within the scheduled exams of students.

H2 : Room Occupancy – For each specified room and each specified time slot, the number of the allocated seats must be less than or equal to the number of the available seats in the specified room.

H3: Period Utilisation – For each of the exams, the exam duration must be less than or equal to the duration of the period. Period-related constraints, which are a group of time-ordering requirements between exams' pairs must be observed, particularly for any pair ($exam_1$, $exam_2$) of exams, these constraints are specified as follows:

H4: After Constraint – $exam_1$ must occur firmly after $exam_2$.

H5: Exam Coincidence – $exam_1$ must occur simultaneously to $exam_2$.

H6: Period Exclusion – $exam_1$ must not occur simultaneously to $exam_2$.

H7: Room Related – Satisfaction-of-room-related hard constraints (e.g. $exam_A$ must be completely arranged in $room_X$).

In addition, all the specified exams must be arranged, and must not be divided between specified periods or specified rooms. Therefore, an examination timetable could be obtained, which satisfies the entire hard constraints.

Soft Constraints

The soft constraints for the ITC 2007 examination track are (Mccollum et al., 2010):

S1: Two Exams in a Row – The number of times, whereby examinations are consecutively arranged for a student, must be minimised.

S2: Two Exams in a Day – This the case, where there are three periods or more on a specified day. This involves the number of occurrences of a student, who has two exams on a specified day that are not adjacent directly, which means that students must have at least a free period between exams; such a case must be minimised.

S3: Period Spread – This soft constraint necessitates that the examinations' set taken by students is spread over a fixed number of time slots.

S4: Mixed Durations – A penalty should be incurred when there are exams in the same room and in the same period with the existence of mixed durations.

S5: Front Load – It is necessary to distribute exams with the largest number of students, and distribution should be performed at the commencement of the examination session.

S6 & S7: Room and Period Penalties – A utilisation penalty for the rooms, as well as the periods are specified so that the utilisation of given rooms or some periods is reduced to a minimum.

Problem Formulation

The key objective of tackling the problem of these datasets involves satisfying the entire set of hard constraints and minimising the total soft constraint violation (or penalty value) (Mccollum et al., 2010). In other words, the number of hard constraint violations must equal zero, whereas the quality of an examination timetable is determined by the soft constraint violations subject to this. The best quality solution is that with the lowest weighted sum of soft constraint violations. Formally, subject to all hard constraints are satisfied, the objective function is to minimise the total penalty as result of soft constraint violations defined by as given in Equation (2.9) for ITC 2007, where the total penalty for the soft constraint violations is calculated for each student s.

$$f_c(x) = w^{NMD}C^{NMD} + w^{FL}C^{FL} + C^P + C^R + \sum_{s \in S}(w^{2R}C_s^{2R} + w^{2D}C_s^{2D} + w^{PS}C_s^{PS}). \quad (2.9)$$

Where 'x' represents a complete timetabling solution. 'S' refers to students' set. 'w' represents the weighting applied to each of the individual penalties. These weights are defined in the 'institutional model index' file. Each problem instance has its own weight, and each associate weight is described as follows:

- w^{2R}: weight for "Two Exams in a Row". w^{2R} corresponds to the number of students that take two exams, that are scheduled back to back on the same day, multiplied by this weight.
- w^{2D}: weight for "Two Exams in a Day". w^{2D} corresponds to the number of students sitting two exams that are allocated not back to back but on the same day, multiplied by this weight.
- w^{PS}: weight for "Period Spread". w^{PS} corresponds to the sum of occurrences of students who are sitting exams within a fixed period spread defined in the problem instance.
- w^{NMD}: weight for "No Mixed Duration". w^{NMD} corresponds to all the exams that have the same duration we count 0. Otherwise we count the number of different durations minus "1" multiplied by this weight.
- w^{FL}: weight for "Front Load penalty". w^{FL} corresponds to the number of large exams allocated in the latter periods multiplied by this weight. The number of periods considered as latter periods and the number of students constituted "large examinations" are given.

Table 2.4 summaries the weights associated with soft constraints for the ITC2007 dataset. Weights are not included with the period C^P as well as the room-related soft constraint C^R in the objective function because they were already covered in their definition found in McCollum et al. (2012b). Table 2.5 illustrates the detailed penalties of Equation (2.9) where some penalties (C_s^{2R},C_s^{2D},and C_s^{PS}) are directly associated to each student s and other penalties (C^{NMD},C^{FL},C^{P}, and C^{R}) are not directly associated to each student.

Problem Instance	w^{2D}	w^{2R}	w^{PS}	w^{NMD}	w^{FL}	w^{P}	w^{R}
Exam1	5	7	5	10	100	30	5
Exam2	5	15	1	25	250	30	5
Exam3	10	15	4	20	200	20	10
Exam4	5	9	2	10	50	10	5
Exam5	15	40	5	0	250	30	10
Exam6	5	20	20	25	25	30	15
Exam7	5	25	10	15	250	30	10
Exam8	0	150	15	25	250	30	5
Exam9	10	25	5	25	100	10	5
Exam10	0	50	20	25	100	10	5
Exam11	50	10	4	35	400	20	10
Exam12	10	35	5	5	25	5	10

Table 2.4 The weight of ITC 2007 examination benchmark sets.

Soft Constraint	Mathematical Symbol	Description
S1	C_s^{2R}	"Two Exams in a Row" penalty for student s.
S2	C_s^{2D}	"Two Exams in a Day" penalty for student s.
S3	C_s^{PS}	"Period Spread" penalty for student s.
S4	C^{NMD}	"No Mixed Duration" penalty.
S5	C^{FL}	"Front Load" penalty.
S6	C^{P}	"Period" penalty.
S7	C^{R}	"Room" penalty.

Table 2.5 A list of penalties in relation to ITC 2007 soft constraints.

2.3.3 University of Yeditepe Dataset

Problem Description

The Yeditepe dataset is comprised of a number of datasets from the Faculty of Engineering and Architecture of Yeditepe University (Parkes and Ozcan, 2010). In general, they are smaller than the ITC 2007 problems with regards to a number of exams, students, and periods. Also, this dataset is less constrained compared with the ITC 2007 as it contains two hard constraints and only one soft constraint (in ITC 2007, there are seven hard constraints and seven soft constraints). The characteristics of the Yeditepe dataset are given in Table 2.6.

Hard Constraints

The Yeditepe examination timetabling problems' hard constraints are:

- **Examination conflict**: Any student must not have more than one exam at any given time period.
- **Capacity**: It is not possible to exceed the number of seats in any given room at any given time period.

Soft Constraints

Two exams in a row (C_s^{2R}): Two exams should not be scheduled in two adjacent periods on the same day.

Problem Instance	No. of Exams	No. of Students	No. of Rooms	Days	No. of Periods	Conflict Density
YUE20011	126	559	2	6	18	0.18
YUE20012	141	591	2	6	18	0.18
YUE20013	26	234	2	2	6	0.25
YUE20021	162	826	2	7	21	0.18
YUE20022	182	869	2	7	21	0.17
YUE20023	38	420	1	2	6	0.2
YUE20031	174	1125	2	6	18	0.15
YUE20032	210	1185	2	6	18	0.14

Table 2.6 The characteristics of problem instances from Yeditepe dataset.

Problem Formulation

The objective function of Yeditepe is given below in Equation (2.10):

$$\text{(Minimise)} \sum_{s \in S} w^{2R} C_s^{2R}. \tag{2.10}$$

Two exams in a row (C_s^{2R}) is a penalty given whenever a student s has to attend two distinct exams scheduled in two consecutive periods on the same day. The weight (w^{2R}) for this penalty is "1" for all problem instances of Yeditepe (Parkes and Ozcan, 2010).

2.3.4 Other Benchmark Examination Timetabling Datasets

Other examination timetabling problems datasets in the literature include those of the University of Nottingham, University of Melbourne, Suleyman Dermirel University, MARA University Malaysia dataset (Kendall and Hussin, 2005b), Universiti Kebangsaan Malaysia (UKM) (Ayob et al., 2007a), Universiti Malaysia Pahang dataset (UMP) (Kahar and Kendall, 2010), KAHO Sint-Lieven (Demeester et al., 2012), Middle East Technical University (METU) (Ergül, 1995), École de Technologie Supérieure (ETS) (Wong et al., 2002), Hubei University of Technology (HUT) (OuYang and Chen, 2010), University of the Thai Chamber of Commerce (Innet, 2013), Universiti Utara Malaysia (UUM) (Abdul-Rahman et al., 2014), Universiti Sains Islamic Malaysia (USIM) (Aldeeb et al., 2015). The properties of these datasets are summarised in Table 2.7.

Dataset	No. of Students	No. of Exams	No. of Rooms	No. of Periods	Days	No. of Enrolments	Conflict Density
Mara (UiTM)	84675	472201				470793	
UKM	14047	818	7	42	15	75857	0.05
UMP	3550	157	20	10	12731		0.05
KAHO	135 groups	336	71	40			
METU	16000	1467	1	39	13		
ETS	370	173	32	11			
HUT	2134	224	18	14			
UTCC	135	71	10	33	11		
UUM	13359	639	40	22	11		0.04
USIM	854	18	20	10	6140		

Table 2.7 The characteristics of the examination timetabling problems benchmark datasets from different universities.

Burke et al. (1995c) introduced the Nottingham dataset which included 800 exams with 10,034 student conflicts among them, 7,896 students and 33,997 distinct enrolments. The exams have to be distributed into 26 periods or less. The following constraints have to be achieved:

1. No student should have two exams at the same period.
2. Occupancy of exam rooms cannot exceeded 1550 students because the maximum number of seats is specified as 1550 seats.
3. If any student has two exams on the same day, there must be at least one complete period between them.

Any examination timetable must meet the first two constraints to be at least feasible. The scale of the problem is considered a challenge for finding a complete solution within a feasible number of trials. The problem is made more complex by the size and the number of the rooms available for each period. Table 2.8 presents a description of Nottingham dataset (Burke et al., 1995c).

Problem Instance	No. of Exams	No. of Students	Tot.Room Capacity	No. of Enrolments	No. of Periods	Conflict Density
NOTT	800	7896	1550	33997	23	0.03
Objective: minimise consecutive exams on the same day.						

Table 2.8 The characteristics of University of Nottingham benchmark dataset.

Merlot et al. (2003) provided two datasets for the exam timetabling problem that were gathered from Melbourne University at the PATAT conference in 2002. The first dataset included 521 exams, 28 sessions, 20656 students and 62248 enrolments. The second dataset included 562 exams, 31 sessions, 19816 students and 60637 enrolments. In the Melbourne datasets, there are two exam sessions on each of five workdays, and the sessions' capacities varied. This dataset aims to minimise the occasions in which two exams are scheduled consecutively for a given student, either overnight or on the same day. Some exams were restricted to particular sessions. In one problem instance, this denied all feasible solutions. The characteristics of each these datasets are given in Table 2.9 (Merlot et al., 2003).

Problem Instance	No. of Exams	No. of Periods	No. of Students	No. of Enrolments	Objective
I	521	28	20656	62248	Minimise adjacent exams on the same day or overnight
II	562	31	19816	60637	Minimise adjacent exams on the same day or overnight

Table 2.9 The characteristics of the Melbourne benchmark datasets.

The SDU dataset is from Suleyman Demirel University and was first introduced by Altıntas et al. (2014). The SDU examination timetabling problem here is a capacitated problem which means room capacities are considered as a hard constraint. During exams in each period, there is a maximum capacity of seating available. So, the number of students taking any exam should not exceed the capacity of the room (i.e. the number of students taking an exam in a particular room cannot exceed the capacity of that room). The SDU dataset consists of a set of five real-world exam timetabling problem instances and has been formulated in the same way as the ITC 2007 benchmark instances. The characteristics of the SDU instances are concisely given in Table 2.10, where Density indicates the percentage of conflict density whilst HC shows the number of hard constraints.

Problem Instance	No. of Exams	No. of Students	No. of Rooms	No. of Periods	Conflict Density	Period HC
SDU01	212	10953	17	50	3.24	142
SDU02	236	11012	26	61	5.08	123
SDU03	430	24867	29	80	1.37	317
SDU04	166	8028	18	59	12.60	61
SDU05	269	12091	33	46	3.59	173

Table 2.10 The characteristics of problem instances from Suleyman Demirel University dataset.

From the literature survey, we observe that most of the current studies over examination timetabling problems concentrate on benchmark problems. Using the same examination benchmark datasets in different studies conducted is crucial to assess the efficiency and efficacy of a particular approach accurately. Depending on the reported results, it can also provide a quick understanding and generalisation of the strength or capability of a particular approach for solving this problem effectively. Thus, three different examination benchmark datasets (i.e. the Toronto, ITC 2007, and Yeditepe) are used in this thesis to test the proposed approaches and compare the results against existing literature on these benchmark datasets. These benchmark problems chosen have been in use for a number of years now. However, from the recently published works, it is clear that these benchmark datasets are still used extensively (Aldeeb et al., 2019). Researchers are still interested in finding (near) optimal solutions for them (Bellio et al., 2021).

The organisation of international timetabling competitions has highly influenced the research in educational timetabling (Müller et al., 2018). The overall aim of this organisation was to provide a better understanding between researchers and practitioners by enabling emerging approaches to be trailed and tested on real-world models of timetabling problems (Mccollum et al., 2010). The success of this organisation is definitive evidence of the importance and complexity of the timetabling problem. In 2002, the first International Timetabling Competition (ITC 2002) was prepared by the European Metaheuristics Network (Paechter et al., 2002), which concentrated on a simplified version of the university course timetabling problem. The next (i.e. second) competition was in 2007, where the ITC 2007 consists of three tracks on curriculum-based timetabling (Di Gaspero et al., 2007; Bonutti et al., 2012), post-enrollment timetabling (Lewis et al., 2007), and examination timetabling (McCollum et al., 2007). The ITC 2007 competition aimed to narrow the gap between practice and research by providing a significant degree of complexity in the tracks by adding more hard and soft constraints so that the employed formulations are closer to the real world needs (Mccollum et al., 2010). In 2011, research in another area of educational timetabling was encouraged by the ITC 2011 competition (Post et al., 2016), where the focus was on the important area of High School Timetabling. This is a complex common problem faced by thousands of educational institutions around the world. The fourth competition was in 2019, aiming to motivate further research on complex university course timetabling problems (Müller et al., 2018). The most recent competition is ITC 2021, which focuses on sport timetabling (Van Bulck et al., 2021).

Because the focus of this thesis is on solving university examination problems, a particular track of the ITC 2007 has been chosen, which is dedicated to examination timetabling problems for universities. This track was developed to reflect the challenges of tackling real-world examination timetabling problems. It is complex and rich with many considerations related to hard and soft constraints. This provided us with valuable test data and insight into our proposed approaches. We believe that the problem model, as described in ITC 2007, can capture most of the requirements usually encountered in reality. It is interesting to note that competitors' ITC 2007 examination track results are still maintained at the competition website, where all problem instances are available. Many researchers are still interested in investigating their approaches using this track in a similar manner as the Toronto datasets did a decade ago, as demonstrated in (Bellio et al., 2021).

For the Yedtiepe dataset, although few prior works use this dataset, there is more clarity, consistency, and standard format than other datasets. Thus, this is the reason why this dataset is chosen for testing in this thesis. Furthermore, this dataset is in the same format as ITC 2007 format, allowing testing this dataset by the same optimiser Muklason et al. (2017). The following section presents different algorithmic techniques that have been successfully applied to the university examination timetabling problem.

2.4 Algorithmic Techniques for the University Examination Timetabling

Since the 1990s, the examination timetable problem has been extensively investigated by the optimisation community. Surveys conducted by Carter et al. (1996) and Schaerf (1999) reviewed early approaches used to tackle the examination timetabling problem. From the more recent surveys by Qu et al. (2009b) and Gashgari et al. (2018), the timetabling solution methods can be typically classified into these categories: exact algorithms, graph colouring techniques, single-solution based meta-heuristics, population-based algorithms, hyper-heuristics, and techniques of decomposition or clustering. Hybrid algorithms, which combine the features of several algorithms, are generally viewed as the state-of-the-art in the application domain (Caramia et al., 2001; Merlot et al., 2003). Because of the complexity of the examination timetabling problem, precise methods (i.e. exact techniques) can be applied only on small-sized problem instances. Since the real problem instances that exist in practice are typically large scale, precise methods are often impractical in the time available. As a result, heuristic and meta-heuristic algorithms are used to tackle this problem.

Most of the existing algorithmic techniques utilised to tackle examination timetabling problems are based on single objective models. There are, however, a few studies that focus on the multi-objective approaches in the literature. Other approaches related to constructing exam timetables are graph colouring heuristics, fuzzy-based, decomposition, and neural networks.

2.4.1 Exact Techniques

Heuristics and meta-heuristics are approximation algorithms that only enumerate the search space partially and therefore do not guarantee an optimal solution. Exact techniques, however, can perform an implicit enumeration of the search space. Hence, these techniques are named complete techniques (i.e. algorithms) since they guarantee their optimality where the encountered solution is optimal (Talbi, 2009). These techniques can have an excessive time overhead, however, which is prohibitive for larger practical problems (Chen and Bushnell, 1996). Examples of these techniques involve Constraint Programming (CP) (Boizumault et al., 1996; Freuder and Wallace, 2005) and Integer Programming (IP) (Bosch and Trick, 2005).

Boufflet and Nègre (1996) applied exact approaches to tackle exam timetabling problem. In this work, three approaches were proposed (i.e. Tree Search, Tabu Search, and Computer-Aided Design System) and then an exact method based on the GC method was employed to explore the path in the tree. The tree search method was capable of tackling this problem. The proposed approach has only been applied to solve the examination timetabling problem of the University of Technology of Compiègne, however.

CP permits direct programming with constraints (Boizumault et al., 1996). This makes it possible to solve problems like the timetabling problem easily and flexibly. Two important characteristics of the technique include using backtracking, as well as logical variables. CP varies from other kinds of programming in that it stipulates the steps necessary for execution. In CP, however, only the properties which should or should not be in the solution can be stipulated (i.e. the hard constraints) (Qu et al., 2009b).

Merlot et al. (2003) proposed a hybrid three-phase approach involving CP, Simulated Annealing, and Hill-Climbing. The CP implemented in this study was similar to that proposed in Boizumault et al. (1996). In the first phase, CP was performed in order to generate an initial feasible solution within a short time. In the second and third phases, Simulated Annealing and Hill-Climbing methods were employed to improve the quality of the solution. Kempe-chain neighbourhood structures were used to encourage the diversification of the search space (Thompson and Dowsland, 1996b). The aim also is that chains of conflicting examinations are swapped between two or more feasible periods. This hybridisation approach was tested over three different datasets (i.e. Melbourne, Toronto, and Nottingham).

IP involves a mathematical programming technique (Al-Yakoob et al., 2010). In this technique, the optimisation problem that should be solved must be created as an Integer Problem. When the objective function, as well as the constraints, are linear, and the entire problem variables are integer-valued, the IP problem is referred to as an Integer Linear Problem (ILP) . When both integer and continuous variables are present, however, the problem is referred to as a Mixed-Integer Linear Programming (MILP) (Feng et al., 2017). Schaerf (1999) surveyed a number of approaches utilising the MILP technique to school, course, and examination timetabling problems.

It is worth noting that the major advantages of the exact techniques are that they have a good performance in many problems. They can provide an appropriate solution to simple optimisation problems (Jansson and Knüppel, 1995). Furthermore, these techniques are easy to design and implement (Festa, 2014). Furthermore, they can easily incorporate assumptions to solve problems (Beheshti and Shamsuddin, 2013; Aladağ and Hocaoğlu, 2007). On the other hand, the significant disadvantage is that these techniques are inefficient when addressing large-scale and complex combinatorial optimisation problems such as examination timetabling problems (Beheshti and Shamsuddin, 2013). This is due to the techniques' inability to search exhaustively in search spaces that grow exponentially with the given problem size.

2.4.2 Constructive Heuristic Techniques

In the literature the most successful approaches exclusively dedicated to solving timetabling problems consist of two phases: the construction phase and improvement phase (Hertz, 1991). In the construction phase, an empty solution is constructed by an iterative approach, and this construction process is repeated until a final solution is completed. This process is often performed by utilising some heuristics (Burke et al., 2010c). In contrast, in the improvement phase, an improvement approach is employed to improve the quality of the solution initially constructed in the construction phase until an optimal solution is obtained.

The next subsections describe the most widely-used approaches in the construction phase. Improvement heuristic techniques, meanwhile, are discussed in Section 2.4.3.

2.4.2.1 Graph Colouring Heuristics

Graph colouring (GC) heuristics are the most commonly used method in the construction phase. It is defined as the problem of colouring vertices of a graph with the fewest number of colours possible while ensuring that no two neighbouring vertices have the same colour (Carter et al., 1996). Examination timetabling problem, in its simplest form (without soft constraints), can be represented as a graph colouring problem where the vertices represent the examinations, and the edges represent the conflict between pair examinations, and the colour of the vertices represent different periods that are available in the exam timetable (Carter, 1986). A definition of the concepts and terms related to a graph can be found in (Burke et al., 2004b). An undirected graph $G = (V, E)$ is a representation that comprises a set of vertices, $V = v_1, v_2, \ldots v_n$ (representing the events) and a set of edges E where every edge connects two distinct vertices (Garey and Johnson, 1979). If (v_i, v_j) is an edge in a graph $G = (V, E)$, then vertex v_i is adjacent to vertex v_j (Burke et al., 2004b). In such a way, every vertex in the graph represents a distinct exam, and every edge represents a conflict between the two exams (connected vertices). The rule is that every adjacent vertex is coloured with a different colour where every colour represents a period in the timetable, and the number of colours used should be minimal (Garey et al., 1974).

To convert a simple graph colouring solution to a valid examination timetable, Figures 2.2 (a) and (b) show an example of a graph model for a simple examination timetabling problem. Figure 2.2 (a) presents a simple timetable which is converted to equivalent undirected graph colouring, i.e. in this example we attempt to schedule ten exams (denoted as e_1 to e_{10}). Figure 2.2 (b), represents the solution found using an optimal number of colours (periods), which is four periods (denoted as p_1, p_2, p_3 and p_4). Table 2.11 represents the solution (in timetable form) converted from Figure 2.2 (b). This figure shows that events e_1, e_9 and e_5 are scheduled in period p_1 events e_2, e_3 and e_{10} are scheduled in period p_2 and so on.

p_1	p_2	p_3	p_4
e_1	e_2	e_6	e_4
e_9	e_3	e_7	
e_5	e_{10}	e_8	

Table 2.11 The graph colouring solution as a timetable.

Several graph colouring heuristics (i.e. Largest Degree, Weighted Largest Degree, Largest Enrollment, and Saturation Degree) have been proposed in order to construct a conflict-free timetable. These heuristics prioritise events according to the level of difficulty in the

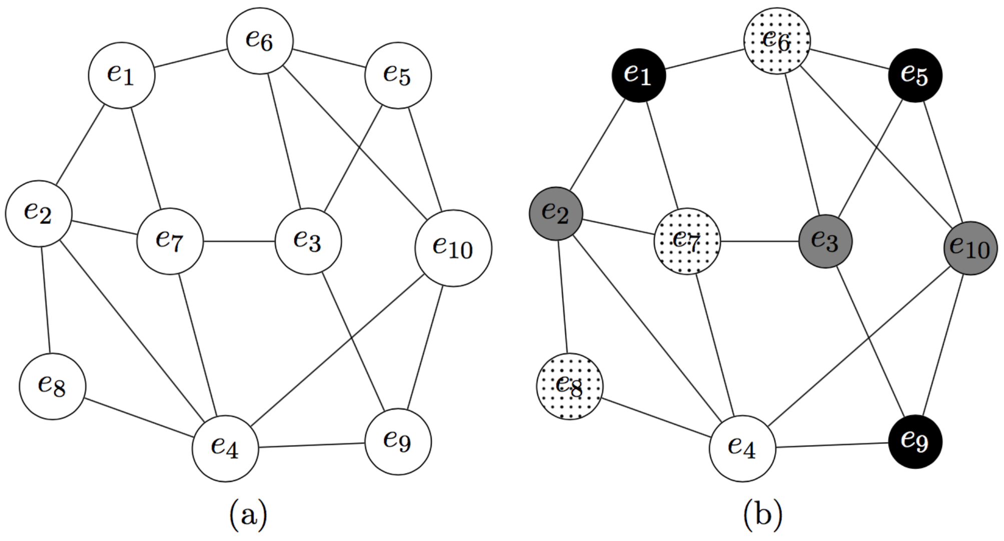

Figure 2.2 A graph model for a simple university timetabling problem (a) An undirected graph; (b) A solution for the university timetable problem.

scheduling process. The rationale behind this is to ensure that the most difficult exam is scheduled first (Burke et al., 2004b). Examples of common graph colouring heuristics are in Asmuni et al. (2005a) and Burke et al. (2007). For more details on these heuristics (see chapter 3). Table 2.12 shows a brief description of the most common GC based heuristic orderings applied to examination timetabling problems.

Ordering heuristic	Description
Largest Degree	Exams with a higher degree of conflicts are scheduled first.
Least Saturation Degree	Exams with least number of available periods are scheduled first.
Largest Enrollment	Exams with a higher degree of students' enrollment are scheduled first.
Weighted Largest Degree	Exams with a higher degree of conflicts and a higher number of students involved in the conflict are scheduled first.
Random Ordering	Exams are scheduled randomly.

Table 2.12 The graph colouring heuristics in examination timetabling.

The graph colouring heuristics mentioned above were extensively studied in early scheduling research and are still in use today as an initialisation approach for meta-heuristics, or by completely integrating them with meta-heuristics in various ways. For instance, Appleby et al. (1961) applied graph colouring heuristic to create school timetables. Since that time, it has been extendedly used graph-based heuristic orderings to solve other types of scheduling problems. Largest Degree was the most commonly heuristic ordering employed in the earlier examination timetabling research (Broder, 1964; Cole, 1964; Welsh and Powell, 1967). The heuristic orderings Largest Enrollment and Largest Degree were used by Wood (1968), where exams that require the room with the largest capacity were chosen first. Thus, these exams were then ordered decreasingly by the number of exams in conflict. The same procedure was used for the second largest room and so on. Johnson (1990) also combined heuristic orderings Largest Enrollment, and Largest Degree, where it has been considered them simultaneously through the simple linear combination of Largest Degree, with Largest Enrollment, multiplied by a varied weighted factor w_{LE}.

A study on the performance of these four heuristic orderings (i.e. Largest Degree, Saturation Degree, Weighted Largest Degree, and Largest Enrolment) was presented by Casey and Thompson (2002). The heuristic orderings were utilised to produce the initial solutions in the construction phase of a Greedy Randomised Adaptive Search Procedure (GRASP) algorithm. Roulette wheel selection was then used to select the next exam to be allocated from the top n exams in the exam ordering. The appropriate value for n was experimentally chosen depending on the total number of exams in the problem instance. The selected exam was then allocated into the first period that met all the hard constraints.

Foxley and Lockyer (1968) used a priority formula to order exams. This formula used all information concerning the exams extracted from the problem. Furthermore, a manual special priority setting was used in order to override other soft constraints. For example, final exams were given special priority and were timetabled first.

One of the main advantages of graph-based heuristics is they are capable of producing initial solutions in a shorter time and are easy to apply (Carter, 1986). Scheduling exams are conducted smoothly in a sequential manner by considering certain criteria to ensure that the scheduling process ends by allocating all the exams in the first (i.e. construction) phase. In some cases, however, it is difficult to assign the exams in the first attempt. This can have a significant effect on the initialisation process. Thus, the main disadvantage of graph-based heuristics is that a feasible examination timetable is not always achievable during the first phase. Furthermore, these heuristics are not sufficient to produce high quality initial solutions (Burke et al., 2007; Ayob et al., 2007b; Qu et al., 2009b). In the literature, different heuristic orderings have been examined. It has been demonstrated that determining which heuristic ordering would be most appropriate for any given problem is challenging (Carter et al., 1996). In addition, the investigation presented in Burke and Newall (2004) indicated that better solutions could be obtained during the initialisation phase by adaptively changing the heuristic ordering. A common observation in many approaches indicates that constructing high quality solutions in the construction phase should not be precluded from consideration. A study by Burke et al. (1998a) suggested that the use of an appropriate initialisation method for generating the initial solutions of an evolutionary algorithm for timetabling problems could substantially improve performance. Therefore, this observation led to the conjecture that this thesis concentrates on studying a new initialisation method that can simultaneously consider hard and soft constraints during the construction phase, which might improve the quality of the solutions obtained.

2.4.2.2 Fuzzy-based Techniques

The concept of fuzzy logic was first presented in 1965 by Zadeh based on the fuzzy set theory (Zadeh, 1965). Fuzzy logic is not logic that is fuzzy (i.e. not clear or vague), but it is the logic that aims to describe fuzziness by using fuzzy techniques. Basically, fuzzy techniques are utilised to represent and employ knowledge that is uncertain, ambiguous, or imprecise and relates reasoning with linguistic terms.

In the more traditional propositional logic, each proposition or fact in real-life situations is represented by two values of the truth. These values are *true* or *false*. However, there is still some degrees of uncertainty that is not involved. The limitations of the traditional (i.e. crisp) logic are in describing uncertainties. On the other hand, the power of fuzzy logic is in handling such the limitations where it can measures the degree to which the proposition is correct. In fuzzy logic, numeric values between 0 and 1 are defined to each proposition or fact in order to present uncertainty where 0 is equated with completely *false* value and 1 is equated with completely *true*. Values between (0,1) indicate partially *false* and partially *true*, and are described by linguistic expressions.

Zadeh (1975) proposed the term 'linguistic variable' to refer to a variable whose values are in the form of "linguistic expressions" instead of numerical values. For example, 'speed' is a linguistic variable with linguistic value 'fast'. 'medium' and 'slow' are other possible linguistic values for the linguistic variable 'speed'. These linguistic values are characterised by a fuzzy set (membership function) in a universe of discourse U = [0, 120], they might be interpreted as:

- 'slow' to be a speed below about 30 mph.
- 'medium' to be a speed around 65 mph.
- 'fast' to a speed above about 85 mph.

These linguistic values would be associated with fuzzy sets whose membership functions are are shown in Figure 2.3.

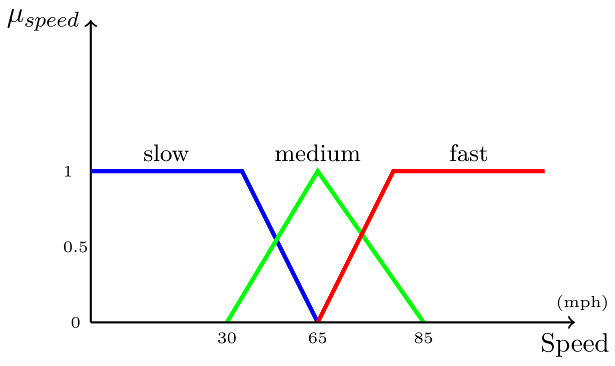

Figure 2.3 Membership functions for the linguistic variable 'speed'.

Fuzzy methodologies have been investigated for other timetabling problems such as aircrew rostering by Teodorović and Lučić (1998), driver scheduling by Li and Kwan (2003) and nurse rostering by Auf'm Hofe (2001). Fuzzy-based techniques have been applied successfully to timetabling problems (i.e. examination and course timetabling problems). In the specific context of examination timetabling, Asmuni et al. (2005b) used a fuzzy technique to solve the examination timetabling problem. In their work, several sets of two graph colouring heuristics were combined, and the examinations were ordered based on the difficulty of timetabling them. The approach is based on three ordering heuristics derived from GC strategies (i.e. largest enrolment, saturation degree, and largest degree) and three combinations of two heuristics. The fuzzy approach is used in Asmuni et al. (2005b) to evaluate input variables, which are the knowledge obtained from the heuristics, and these then help to generate an exam weight as an input variable. In order to obtain a feasible examination timetable, the examinations were ordered based on decreasing examination weight values and allocated in the exam timetable in such a way as to satisfy all hard constraints. The approach used a strategy named 'bumped back'. This strategy is used to 'unschedule' ('bump back') any exams that were already assigned earlier in order to maintain feasibility and avoid an infinite loop. In this work, conflicting exams are moved from the selected period (i.e. a period that has been selected randomly from the list of periods) into another valid period and inserting the current unscheduled exam (i.e. skipped exam) into the selected period or the need to use the bump back strategy where it 'bumps' the scheduled exams (exams which cause conflict) back to the unscheduled exam list in order to make periods available for the skipped exams (i.e. exams that could not be assigned in the first attempt and are recorded on a 'skipped list' due to no valid free-conflict period was available). The work showed that a tuning procedure needed to be employed to improve the quality of solutions.

Petrovic et al. (2005) applied a fuzzy technique to meet two hard constraints. The first constraint was the front-load constraint, where examinations with the largest number of students must be timetabled earlier. The second constraint was that students should have a sufficient break time between two adjacent exams. Fuzzy sets and fuzzy rules were employed for determining the degree of constraint satisfaction and acquiring the satisfaction degree for each constraint. The approach was verified on the Toronto benchmark sets.

Asmuni et al. (2009) also implemented a fuzzy technique to construct solutions for examination timetabling problems. Three criteria were presented in this work in order to show the performance of the approach. The first criterion was the penalty weight compared with other construction heuristics. The second criterion was the number of bump-back strategies required for each dataset. The last criterion was the processing time for a static and dynamic heuristic for each combination. The results showed that the approach was capable of obtaining a better result than other approaches on the Toronto benchmark set.

Generally, researchers who have applied fuzzy-based techniques have frequently enumerated the many advantages and disadvantages that these techniques offer Behrooz et al. (2018). Most studied examination timetabling problems revealed that these techniques have the ability to mimic how human timetabling experts (i.e. (timetabling officers) construct real-world exam timetables (Asmuni et al., 2005a). When soft constraints are considered during the construction phase, the satisfaction of a soft constraint is often satisfied or not satisfied when binary logic is applied. Thus, another advantage of fuzzy techniques is that the degree of satisfaction of these constraints can be measured (Petrovic et al., 2005). Each soft constraint is satisfied to a certain degree. In addition, these techniques also help to maximise the satisfaction degree of any constraints where the quality of constructed solutions is considered higher if all the soft constraints have a high degree of satisfaction. As stated in Zimmermann (1996), fuzzy techniques are advantageous of combining multiple sources of information, Asmuni (2008) successfully combined multiple heuristics simultaneously in the construction process by using fuzzy methodologies in order to provide a measure of the difficulty of allocating exams.

Fuzzy techniques do have some disadvantages, however. The main disadvantage of applying the fuzzy methodologies is that producing initial solutions take more computational time. Moreover, tuning the best fuzzy model (i.e. best set of membership functions for the given set of rules) requires more time to guide the constructive algorithm used. Furthermore, the membership functions are not fixed where they differ in each problem (i.e. there is no generic fuzzy model that suits all problem instances) (Asmuni et al., 2009).

2.4.2.3 Decomposition Techniques

The basic idea behind decomposition is to "divide and conquer", as as an optimal solution could be achieved more easily and quickly for smaller sub-problems utilising relatively simple approaches (Carter, 1983). Decomposition techniques are another way to resolve complex problems, especially those with a large search space. The technique works by dividing the problem into sub-problems that can be solved more easily and then provide opportunities to improve the quality of the obtained solutions. The early assignment of certain sub-problems can affect the feasibility, however (Qu et al., 2009b).

A few studies have attempted to apply decomposition techniques to the examination timetabling problem. Burke and Newall (1999) incorporated the memetic algorithm with a decomposition method for solving the examination timetabling problem. The approach in this work is a multi-stage approach that divides problems into sub-components, effectively meaning that scheduling takes place on one group at once. The backtracking and forward-checking strategies are employed in order to verify the feasibility of solutions. Experimental results showed that the approach considerably enhanced the quality of the solutions and reduced the amount of time taken to find those solutions when tested on three instances (i.e. car92, kfu93 and pur93) of the Toronto dataset and one large instance (i.e. Nott) of the Nottingham dataset.

Qu and Burke (2007) investigated the potential of an adaptive decomposition approach for producing an initial solution for the examination timetabling problem. Their approach involved two stages. In the first stage, the aim was to generate a complete solution. First, the problem was split into two groups, with the first group consisting of difficult exams (determined based on the feasibility of the exams in the prior iteration). The size of this group is always changed based on an assessment of the examinations at the end of each iteration. After the first stage (i.e. the constructive stage) an adaptive decomposition was applied. At this point, the approach proceeded to the improvement phase where the exams in the second group (the 'easy' exams) were then reordered so as to improve the quality of the generated solutions. This work also introduced the concept of a set of 'boundary examinations' between the difficult and easy sets. The approach was tested on the Toronto benchmark datasets and found to be an effective and simple technique compared with other techniques.

In Kendall and Li (2008), the decomposition technique combined a number of examinations that had common features into one examination in order to simplify the problem. This combining of examinations was based on a criterion of compatibility that measures the viability of the combination and this helped to decrease the size of the search space when constructing the examination timetable. The study demonstrated that the approach can improve the solution quality, although it requires high computation time, particularly for large-sized problem instances. This approach, so far, has been tested on only one instance (i.e. the sta83) of the Toronto benchmark datasets where it has shown competitive results.

Abdul-Rahman et al. (2014b) extended the study presented by Qu and Burke (2007). The approach separated the exam timetabling problem into two sets (i.e. a difficult set and an easy set). Initially, the scheduling process assumed that all exams could be placed easily, but the difficult set was gradually increased as infeasibility became evident during the process. The difficult set therefore consisted of those exams that can cause hard constraint violations in order to schedule them in different ways. The approach also presented a boundary set which is located between the two sets (i.e. the difficult and easy sets), with all exams in this set considered as difficult exams. Roulette wheel selection was performed within a prefixed size in order to shuffle the best exam ordering.

Many advantages can be enumerated for using the decomposition techniques. Many authors stated that the decomposition techniques require less computational work than traditional methods (Bulut et al., 2004; Momani and Odibat, 2006; Wazwaz, 2005). Other advantages include splitting up a problem into smaller sub-problems helps solve the problem more efficiently where the decomposition techniques are employed where the number of variables and constraints is often significantly reduced in the individual models and becomes less difficult to solve (Stefansson et al., 2011).

There are, however, some evident disadvantages of the decomposition algorithm. In the examination timetabling problems often have a high number of exams that have to be timetabled. Decomposition technique for timetabling might be more efficient if they consider subsets of exams, rather than the whole set of exams (Carter, 1983). The technique allocates subsets of exams sequentially, adding the exams from the current subset into an already created timetable. Each subset is small enough to be tackled by other approaches. However, The major disadvantage of such a decomposition is that feasibility often could not be achieved due to early assignment of certain sub-problems (Qu et al., 2009b).

2.4.2.4 Neural Networks

The basic idea behind neural networks is derived from systems of neurons in the human brain. Researchers were aiming to understand human behaviour and the thinking process by modelling human brain development (Porto-Pazos et al., 2011). Even now, many prominent researchers in the neural networks field have a background in psychology. In a neural network algorithm, interconnected 'brain cells' (i.e. input, hidden and output layers) inside a computer can manipulate data to allow the computer to understand patterns, learn things, and make decisions in a human-like manner (Haykin, 1999). The input layer includes neurons that transmit information to the hidden layer in the same way as sensory neurons do in biological neural networks. The hidden layers are made up of neurons that link to other neurons on a neighboring hidden layer or the last layer (i.e.

the output layer), and they are where the majority of the processing occurs. In addition, there are neurons responsible for showing the output in response to the provided inputs in the output layer (Haykin, 1999). Neural networks are capable of learning from their experiences and adapting to changes in the environment. The learning begins using two approaches, supervised learning and unsupervised learning (Ansari and Bakar, 2014).

Corr et al. (2006a) implemented a neural network for constructing an examination timetable. GC heuristics, as well as the Kohonen self-organising neural network, were employed in this approach in order to train the regularities of the defined input feature vector. In the construction process, a measure of the difficulty of scheduling examinations is determined before proceeding to ordering and assigning the examinations. The most difficult examinations must be scheduled first to a period. During the construction process, therefore, three groups of examination (i.e. early, middle, late) were categorised by the neural network using three graph heuristics. Toronto's collection of benchmark examination datasets was used to verify this approach. The work illustrated that neural networks could be used to generate feasible solutions to the examination timetabling problem.

The advantage of the neural network is that it can be self-learning, learning decision making like a human, and easy to implement by a computer (Dong et al., 2019). On the other hand, The disadvantage is that it is arduous to understand its reasoning process and reasoning basis. Moreover, when the information is insufficient, it loses the ability to work (Ansari and Bakar, 2014). Another disadvantage is that the computional time taken to obtain the solutions depends on the neural network size. High computational time are required if the size of the neural networks are large. In addition, training the large neural network are often not conducted in shorter time (Whitley et al., 1995).

2.4.3 Meta-heuristics and Improvement Heuristic Techniques

Classical search methodologies such as dynamic programming, integer linear programming (Kanit et al., 2009) and Graph Colouring (Burke et al., 1994b, 1995b) are usually ineffective and inefficient for tackling resource-constrained problems. Accordingly, meta-heuristic algorithms such as Simulated Annealing, Tabu Search, Hill-Climbing, Ant colony, and Evolutionary Algorithms have become widespread in the optimisation area in recent years because of their robustness, ability to satisfy a large number of constraints, modelling capability and effective solutions to many real-world problems such as airline crew scheduling, round-robin sports scheduling, nurse scheduling, etc. (Burke and Petrovic, 2002; Petrovic and Burke, 2004; Lewis, 2008; Babaei et al., 2015b; Kochetov, 2016; Tein and Ramli, 2010; Lewis and Thompson, 2011; Deng and Lin, 2011).

The 'meta-heuristic' term was presented for the first time by Glover in 1986 by joining the Greek prefix “meta” that means beyond or of high-level with heuristic which is from the Greek heuriskein and means “to search”. A meta-heuristic is formally defined, however, as “*an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions*” (Osman and Laporte, 1996). Many additional, similar definitions have been provided, e.g. by

Voß et al. (2012), Glover and Laguna (1993), and Glover and Laguna (1997).

Blum and Roli (2003, 2008) summarised the principal characteristics of meta-heuristics as follows:

- Meta-heuristics are considered as strategies to direct the search process.
- The objective is to explore the search space as efficiently as much as possible in order to find near optimal solutions.
- The complexity degree of meta-heuristics algorithms ranges from simple procedures to complex learning processes.
- They are considered approximate and often non-deterministic.
- Meta-heuristics can be combined with intelligent mechanisms to avoid getting trapped in confined areas of the search space.
- Meta-heuristics are problem-independent techniques.
- Nowadays, the advanced meta-heuristic techniques utilise search experience (adding memory for saving the properties of the best solution reached until that moment) in order to guide the search toward "good" regions of the search space.

In the context of the present survey, meta-heuristic techniques can be categorised into one-phase optimisation algorithms, two-phase optimisation algorithms, and algorithms which allow relaxation (Teoh et al., 2015). With regard to one-phase optimisation algorithms, both the hard and soft constraints are satisfied at the same time, whereas in the two-phase optimisation algorithms, all hard constraints are satisfied first to obtain a feasible solution (timetable) before satisfying the soft constraints. With respect to algorithms allowing relaxation, the first stage will produce a population of high-quality feasible solutions (timetables) by concentrating on a particular main criterion and relaxing the other criteria. In the second stage, the algorithm starts searching for a compromise solution that satisfies as many of the soft constraints as possible without violating the solution obtained from the first stage. Obtaining high-quality feasible timetables requires computation time, however, and this can make it very difficult to obtain an optimal or even near-optimal solution within a short time period without violating any of constraints.

Problem instances can be tackled by algorithms that utilise both heuristics, as well as meta-heuristics. Heuristic algorithms are problem-dependent. This means that they can be adapted to a specified problem, whereby an advantage of the details can be taken. GC heuristics are, for instance, utilised to generate solutions for a specified timetable problem instance, but only the hard constraints are typically employed in the improvement phase (Carter and Laporte, 1996; Burke et al., 2007). Meta-heuristics, however, are problem-independent, meaning that they can be utilised to optimise any type of problem, and this in turn means that they can typically consider both the hard and the soft constraints simultaneously.

Meta-heuristic algorithms can be divided into two sub-types: single-solution meta-heuristics and population-based meta-heuristics (Talbi, 2009). Single-solution meta-heuristics mainly aim at modifying and optimising one single solution; they maintain the search focused on the local regions. This type of meta-heuristic is, therefore, exploitation oriented. Examples include Hill-Climbing, Tabu Search, Simulated Annealing, Variable Neighbourhood Search, Great Deluge (Talbi, 2009), each of which is discussed in the following subsections. On the other hand, population-based meta-heuristics mainly aim at modifying and optimising multiple candidate solutions in parallel; they maintain the search focus on the whole space. This type of meta-heuristic is, therefore, exploration oriented. Examples include Genetic Algorithms, Ant Colony Optimisation, Memetic Algorithm, and Hybrid (Talbi, 2009). The next subsection discusses the categories of meta-heuristic approaches.

Many papers have described the advantages of the usage of meta-heuristic algorithms and how they can address the educational timetabling problems, such as Genetic Algorithm (Pongcharoen et al., 2008; Alves et al., 2017), Tabu Search (De Causmaecker et al., 2009; Islam et al., 2016), Simulated Annealing (Aycan and Ayav, 2009; Cheraitia and Haddadi, 2016), Ant Colony Optimisation (Lutuksin and Pongcharoen, 2010; Khair et al., 2018), the Great Deluge (Turabieh and Abdullah, 2011b; Burke and Bykov, 2016), Hyper-heuristics (Burke et al., 2007; Epitropakis and Burke, 2018), Evolutionary Algorithm (Sultan et al., 2008; Leite et al., 2016) and many more. In general, these algorithms show very promising results and have many advantages (Blum and Roli, 2003; Beheshti and Shamsuddin, 2013):

- They are robust and adaptive to changes in the environment and conditions.
- They can be used in tackling complex problems.
- They can incorporate other techniques to prevent becoming stuck in local optima.
- These algorithms can find promising areas of the search space in a reasonable computational time due to their ability to explore and exploit.
- They are easily applicable in parallel processing.

Although these algorithms have achieved satisfactory results in many domains, including in examination timetabling problems, they cannot provide a guarantee that optimal solutions will be found, and they have some unavoidable disadvantages. The following subsections discuss the most popular algorithms applied to the examination timetabling problems, moving from definitions and methodologies to advantages and disadvantages for each algorithm.

2.4.3.1 Hill-Climbing

Hill-Climbing (HC) is one of the simplest meta-heuristic local search methods, also called iterative improvement. The algorithm iteratively chooses the current solution and produces a neighbouring solution, If this is better than the preceding one, the method will move to that solution (Burke et al., 2005a). The termination condition for HC is applied when it is unable to find a better solution (Hansen and Mladenović, 2001). The fundamental pseudo-code of a HC for a minimisation problem is described in Algorithm 2.1 below (Hoos and Stützle, 2004) where *Sol* is an initial solution, *Sol** is any solution that can be reached by modifying one element in the initial solution (a neighbour solution of the initial solution). $f(Sol)$ and $f(Sol^*)$ indicate the fitness value of the initial solution and the neighbouring solution, respectively.

Algorithm 2.1 Pseudo-code for a Hill-Climbing (HC) algorithm.

```
1:  $Sol \leftarrow$ an initial candidate solution
2:  evaluate($Sol$)
3:  while $Sol \neq$ a local optimum do
4:      Choose an unevaluated neighbour $Sol^*$ of $Sol$
5:      evaluate($Sol^*$)
6:      if $f(Sol^*) < f(Sol)$ then
7:          $Sol := Sol^*$
8:      end if
9:  end while
10: Return $Sol$                                   ▷ the best solution found.
```

Because HC is usually trapped in local optima, its performance is somewhat poor. In order to enhance its performance, therefore many researchers have hybridised HC with another method. For example, Tam and Ting (2003), Burke et al. (2003a), Abuhamdah and Ayob (2010a), and Abuhamdah and Ayob (2010b) successfully applied an enhanced version of HC with other methods to solve course timetabling problems.

Merlot et al. (2003) presented a hybrid algorithm for solving examination timetabling problems. The hybridisation included three phases: Constraint Programming, Simulated Annealing, and HC, where HC was employed to further improve solutions obtained. Their approach was tested on the Toronto, the University of Melbourne, and Nottingham benchmark datasets. In another study, Kendall and Hussin (2005b) investigated implementing a hyper-heuristic with HC to tackle examination timetabling problems from the MARA University of Technology. They found that their approach generated good quality solutions with respect to proximity cost.

Bykov and Petrovic (2016) successfully applied HC for the examination timetabling problem. Moreover, HC is a basis for some other meta-heuristic methods such as Simulated Annealing, Tabu Search, and Memetic Algorithms (see below). Other studies that have discussed HC with respect to the examination timetabling problem can be found in Mandal and Kahar (2015).

The main advantage of HC is that it is a simple method and easy to implement and hybridise with other algorithms. However, the disadvantage is that it is easily led to fall into local optima as it only accepts better solutions within a certain neighbourhood, without exploring other neighbourhoods.

2.4.3.2 Tabu Search

The Tabu Search (TS) method was introduced by Glover (1986). Glover and Laguna (1997) defined TS as: *"A meta-heuristic that guides a local heuristic search procedure to explore the solution space beyond local optimality"*. TS has overcome the drawback of the HC method (stuck in local optima) by using a memory referred to as the "tabu list" that is used to store the visited solutions in order to prevent the same solution being revisited in the future.

Algorithm 2.2 shows the pseudo-code of a TS for a minimisation problem (Martins and Ribeiro, 2006). Sol_0 is an initial solution and a neighbourhood of a solution Sol is a set $N(Sol)$. Each solution $Sol' \in N(Sol)$ is reached from Sol by an operation called a *move*. *TBList* means the tabu list (Glover, 1989). Procedure `Select_Best_Neighbour` returns the best non-forbidden neighbour solution Sol. If the *TBList* list is full, the oldest forbidden solution is removed from it (i.e. the *TBList*). The incumbent solution is inserted into the *TBList* and is replaced by the best neighbour Sol'.

Algorithm 2.2 Pseudo-code for a Tabu Search (TS) algorithm.

```
1: Generate an initial solution $Sol_0$ and set $Sol \leftarrow Sol_0$
2: $Sol^* \leftarrow Sol$
3: $TBList \leftarrow \emptyset$
4: while stopping criterion is not reached do
5:     $Sol' \leftarrow$ Select_Best_Neighbour$(N(Sol) \setminus TBList)$
6:     if $f(Sol') < f(Sol^*)$ then
7:         $Sol^* \leftarrow Sol'$
8:     end if
9:     if $|TBList|$ = Tabu List Size then
10:        remove the oldest solution from the tabu list ($TBList$)
11:    end if
12:    $TBList \leftarrow TBList \cup Sol$
13:    $Sol \leftarrow Sol'$
14: end while
15: return $Sol^*$
```

Many researchers have applied the TS method to tackle university timetabling problems. The TS algorithm was first applied by Costa (1994) to university course timetabling problems, while Alvarez-Valdes et al. (2002) applied TS in the academic scheduling problem. The authors used two successive phases that consist of a main method employed to produce a clash-free feasible timetable and then a TS method to enhance the produced timetable. Because the TS behaviour depends on the neighbourhood structure to find the global optimum value, De Causmaecker et al. (2009) proposed four different techniques to diversify the neighbourhood structure (Swap move, Time-Swap neighbourhood, Room-Swap neighbourhood, and Time-Room Swap neighbourhood). These techniques were also called the horizontal swap because they involved content swapping within the same candidate solution.

Wilke and Ostler (2008) used TS to solve the problem of school timetabling. TS was compared to several algorithms (i.e. genetic algorithm, simulated annealing, and branch & bound) with the aim of providing a software framework that can solve various timetabling problems. Simulated annealing generally was capable of obtaining the best result. TS, however, was able to improve the solution in a shorter computational time.

Mushi (2006) developed a TS method that generates course timetables by decreasing penalties over an invalid solution (i.e. timetable). The approach was tested on a dataset from the University of Dar-as-salaam. The results were compared with a manually constructed course timetable. In addition, termination condition was determined if there is no improvement after 1000 iterations. The findings revealed that their proposed system outperformed the manual system. A set of examples of the implementation of TS methods to solve examination timetabling problems can be found in Amaral and Pais (2016).

The TS method is considered a robust local search ability of the global iterative optimisation method. The advantage of using the TS method is that it uses a flexible memory of search history to avoid roundabout and escape the trap of local optima (Abido, 2002). However, the deficiency (i.e. disadvantage) is that sometimes the TS needs to reach a previously visited solution, so that roundabout is unavoidable and a better diversification strategy is required (Fouskakis and Draper, 2002). Other advantages are that a longer tabu list may cause longer computational time. In addition, the TS depends strongly on the initial solution and serial iterative search process (Zhang, 2011).

2.4.3.3 Simulated Annealing

Another HC-based method that attempts to avoid being trapped in local optima is Simulated Annealing (SA). The origins of the method are in statistical mechanics (the Metropolis Algorithm), and it was first introduced as a search algorithm for combinatorial optimisation problems in Kirkpatrick et al. (1983) and Černỳ (1985). It simulates energy levels in cooling solids to achieve a stable crystal lattice structure with a minimum energy state (Dong et al., 2019). The main idea is to accept uphill moves that lead to worse quality solutions than the current solution in order to escape from local minima while, during the search, the size of such uphill moves deemed acceptable effectively decreases.

The pseudo-code for a SA for a minimisation problem is described in Algorithm 2.3 (Aarts and Korst, 1989), where K_B indicates to the Boltzmann constant and TE are the temperature values (i.e. the temperature of the heat bath), which are selected depending on the cooling approach. Generally, the temperature remains fixed for the first generations before it is decreased by the *annealing scheduler* (Kirkpatrick et al., 1983; Skiundefinedcim and Golden, 1983; Luke, 2009). For minimisation problems, the algorithm starts from an initial solution Sol_0 and sets the temperature at TE_0 and then randomly searches different neighbourhoods gradually decreasing the value of improvement which can help to escape from local optima. For each TE, the inner loop is performed until *thermal equilibrium* is reached. At each iteration, a neighbour solution Sol^* and the variation ΔE in the objective value are computed. The new solution Sol^* replaces the incumbent if ΔE is less than zero, (i.e. if the new solution is better). The state transition is performed with probability $e^{-\Delta E/(K_B TE)}$ when the new solution is worse than the current solution. Once *thermal equilibrium* is met, the TE is decreased depending on the *annealing schedule*. In most implementations, the new TE is geometrically decreased, by the multiplication of the current TE by a constant smaller than 1.

Algorithm 2.3 Pseudo-code for a Simulated Annealing (SA) algorithm.

```
1: Generate an initial solution $Sol_0$ and set $Sol \leftarrow Sol_0$
2: Compute the initial temperature $TE_0$
3: $TE \leftarrow TE_0$
4: while stopping criterion is not reached do
5:     while thermal equilibrium is not reached do
6:         Obtain a neighbour solution $Sol^* \in N(Sol)$ at random
7:         Compute $\Delta E = f(Sol^*) - f(Sol)$
8:         if $\Delta E < 0$ then
9:             $Sol \leftarrow Sol^*$
10:        else
11:            if $e^{-\Delta E/(K_B TE)} > random[0, 1)$ then
12:                $Sol \leftarrow Sol^*$
13:            end if
14:        end if
15:    end while
16:    decrease $TE$ according with the annealing schedule
17: end while
18: return $Sol$
```

SA has been extensively investigated and successfully applied to many areas, among them being examination timetabling. Johnson (1990) applied SA to a real-world examination timetabling problem. The SA was capable of producing a high-quality solution compared to manual techniques. Also, Thompson and Dowsland (1998) applied SA to the examination timetabling problem in order to reduce soft constraint violations. The approach solved the problem in two phases: in the first, it successfully obtained an initial feasible timetable, and then in the second optimised the generated timetable. An adaptive cooling schedule was also implemented.

Thompson and Dowsland (1998) and Dowsland and Thompson (2012) also applied SA to solve the exam timetable problem. They compared three neighbourhood operators (standard, in which the neighbourhood includes solutions produced by altering the colour of S-Chains, single vertex, and Kempe chains). They showed that the Kempe chains neighbourhood operator is the most efficient as this operator provided more flexibility to allow large and difficult exams to move within the timetable. For measuring the algorithm performance, eight examination timetable problem instances collected from various universities were used as tasted. Mühlenthaler (2015) explored the structure of the search space in the examination timetable problem and set up suitable conditions for the connectedness of clash-free timetables under the operation of Kempe-exchange.

The Kempe-chain move has been successfully employed in examination timetabling problems in order to escape from a local optimum during the search process, presented in studies by Casey and Thompson (2002), Merlot et al. (2003), Burke and Bykov (2006), Burke et al. (2010a), Shaker and Abdullah (2009), and Gogos et al. (2010). Generally, the Kempe-chain neighbourhood works over two subsets of exams, in which it selects

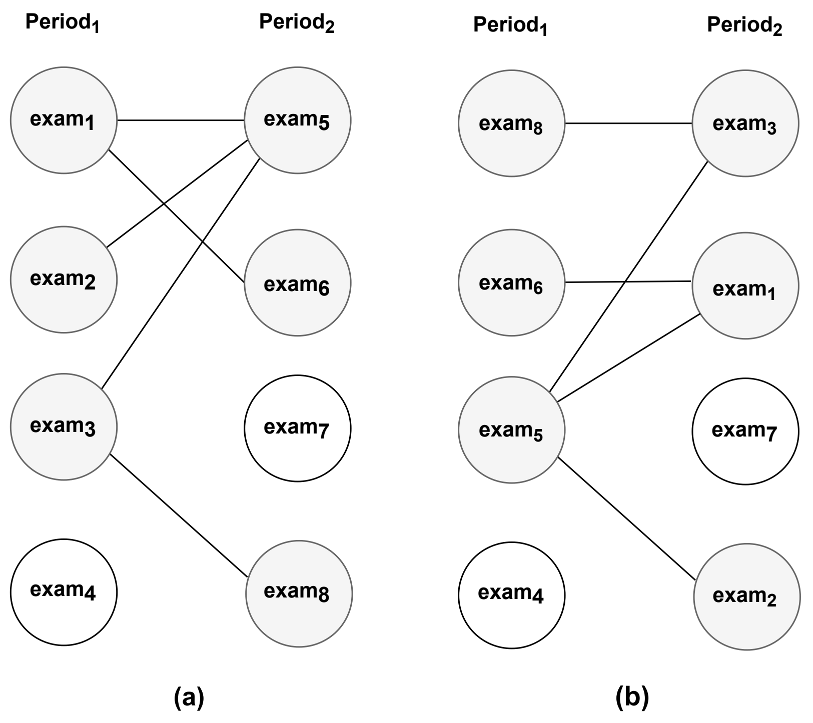

Figure 2.4 The one-pair Kempe-chain (a) before and (b) after the move.

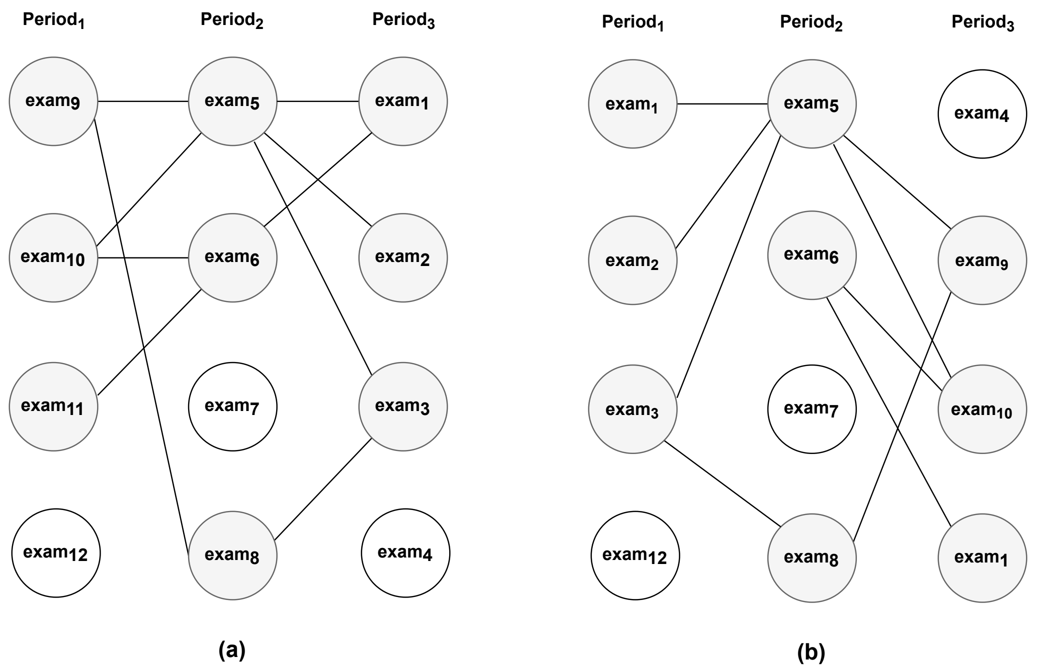

Figure 2.5 The two-pair Kempe-chain (a) before and (b) after the move.

exams randomly and attempts to swap these exams between another two valid periods. Each subset of exams is linked by edges in order to represent the clashes between the exams (Thompson and Dowsland, 1996a). Figure 2.4 illustrates an example of the standard Kempe-chain (i.e. one-pair or one-move kempe-chain). In addition, two-pair Kempe-chain is another variant of Kempe-chain that have been employed in examination timetabling where includes exams connected within k different periods (Thompson and Dowsland, 1996a; Tuga et al., 2007). Figure 2.5 presents an example of a two-pair Kempe-chain.

Frausto-Solís and Alonso-Pecina (2008) hybridised TS and SA algorithms to tackle the

ITC 2007 course timetabling problem. The method included two phases; a construction and an improvement phase. In the construction phase, SA was employed to construct a feasible course timetable. While in the improvement phase, SA was utilised to obtain a solution as close to the optimal solution as possible, within a determined time limit. The TS algorithm is employed whenever there is no improvement in the solution quality. The approach was capable of producing feasible solutions, although overall solutions quality was lacking. Further, the hybridisation of SA with other approaches in examination timetabling showed promising results. Other recent examples of the application of SA in examination timetabling problem can be found in Battistutta et al. (2017), Leite et al. (2019a), and June et al. (2019a).

The advantages of the SA algorithm are as follows (Selim and Alsultan, 1991). First, the SA algorithm could easily be hybridised as well as modified. Second, the SA does not "*stick*" to a local optimum solution. Third, it is also a high-performance algorithm, robust, and is relatively easy to code, even for complex problems (Zhang, 2011). On the other hand, it has some advantages. Application of the SA may require large amounts of computation time. Another disadvantage of SA is that more iterations are required to obtain the best solution. Moreover, it does not have a memory compared to the TS. Hence, previously visited solutions are possibly revisited (Zolfaghari and Liang, 1999).

2.4.3.4 Variable Neighbourhood Search

Variable neighbourhood search (VNS) is a local search meta-heuristic originally introduced by Mladenović and Hansen (1997) and Hansen and Mladenović (2001) for solving a set of combinatorial and global optimisation problems. During the local search, this approach changes more than one neighbourhood structure dynamically. The aim is to avoid getting trapped in local optima. This helps VNS explore the search space more effectively, jumping from the current solution to a new neighbourhood once a better new neighbourhood is found (Abdullah et al., 2005; Burke et al., 2010b). Furthermore, VNS is considered to be a descent-ascent approach that requires implementation of a shaking method and local search iteratively to obtain a high quality solution to a problem. Algorithm 2.4 below shows pseudo-code for a VNS algorithm for minimisation problem adopted by Mladenović and Hansen (1997).

A set of neighbourhood structure N_k is determined first during the initialisation process where $k = 1, \ldots, k_{max}$ indicates the total number of neighbourhood structures used in the local search. $f(Sol)$ means the quality of the solution Sol. The VNS begins from an initial solution Sol that is initialised randomly from the k^{th} neighbourhood. There are three steps in the VNS procedure (i.e. shaking, local search, and move) that are repeated until some stopping criterion is reached, such as a maximum number of iterations, a set number of non-improving iterations or a pre-set CPU time. Intensification and diversification often work together, so the main aim of VNS is to balance between them when it explores the search space. The local search step has an intensification purpose to converge to a good solution, whilst the shaking step is considered as a diversification method that helps to prevent cycling during the search process. In the shaking step, a solution Sol'' in the k^{th} neighbourhood of the current solution Sol is chosen at random, where Sol'' is considered

Algorithm 2.4 Pseudo-code for a Variable Neighbourhood Search (VNS) algorithm.

```
1: Initial a set of neighbourhood structures $N_k$, $k = 1, \ldots, k_{max}$
2: Generate the initial solution $Sol_0$ and set $Sol \leftarrow Sol_0$
3: while termination conditions not met do
4:     Set $k \leftarrow 1$
5:     while $k \leq k_{max}$ do
6:         Obtain a neighbour solution $Sol^{'} \in N_k(Sol)$ at random        ▷ Shaking step
7:         $Sol^{''} \leftarrow$ Local_Search($Sol^{'}$)                          ▷ Local search step
8:         if $f(Sol^{''}) < f(Sol)$ then
9:             $Sol \leftarrow Sol^{''}$                                          ▷ Move step
10:            $k \leftarrow 1$
11:        else
12:            $k \leftarrow k + 1$
13:        end if
14:    end while
15: end while
16: return $Sol$
```

as the starting point for the local search. Then the process continues by visiting the k^{th} neighbourhood of the current solution Sol sequentially until a local optimum $Sol^{''}$ is found. The solution $Sol^{''}$ is accepted if $f(Sol^{''})$ is better than $f(Sol)$. Note once a neighbourhood structure produces a better solution, the local search process restarts from the first neighbourhood ($k = 1$). Otherwise, k is incremented (i.e. $k := k + 1$) and a different neighbourhood is used.

The VNS approach have been widely used in many optimisation areas. For instance, VNS was implemented to tackle a nurse scheduling problem (Burke et al., 2003b), a graph colouring problem (Avanthay et al., 2003), a median cycle problem (Pérez et al., 2003), and project scheduling problem (Fleszar and Hindi, 2004). The VNS approach has also been applied successfully to solve examination timetabling problems (Wong et al., 2005; McCollum et al., 2003).

A variable neighbourhood descent employing multiple neighbourhood structures was proposed by Wong et al. (2005) to solve an uncapacitated examination problem. Exploration and exploitation of the search space of solutions are considered when a different local search operator integrates for each neighbourhood structure. This approach was tested over 16 uncapacitated examination timetabling problem datasets from the Toronto, Nottingham, and Melbourne datasets. Experimental results showed that the approach performed well compared with other published approaches that related to these benchmark datasets.

A study by Burke et al. (2010b) has proposed hybridisation of a VNS with a genetic algorithm. The genetic algorithm in this approach was considered as a neighbourhood selector within a VNS approach, which means it worked at a high-level rather than being directly applied to solve the problem. The genetic algorithm intelligently carried out the search process by choosing the list of neighbourhoods from the variable neighbourhood Search framework. The aim was to prove that the solution quality was based on the selection of a neighbourhood. The approach was capable of solving the examination timetabling problem and obtaining high-quality solutions to the Toronto benchmark problems.

One significant advantage of the VNS is that it requires relatively few parameters that must be tuned (Burke et al., 2010a; Pérez-Peló et al., 2019). However, the major disadvantage is that the VNS implementation requires more execution time of solving large and complex problems (Burke et al., 2010a).

2.4.3.5 Great Deluge Algorithm

The Great Deluge algorithm (GDA) was first proposed by Dueck (1993). The GDA is considered to be a type of local search similar to SA. The GDA is much simpler than SA, however, since it allows worse solutions to be accepted based on some given conditions. It also requires only one parameter to be tuned, which is the "*UP*" parameter. This parameter represents the 'rain speed' which influences the estimation of the quality of the obtained solution and the amount of computation time. This algorithm is simulated based on the analogy of a rising water level. Algorithm 2.5 shows a GDA algorithm pseudo-code for a minimisation problem (Talbi, 2009). In minimisation problems, solutions are accepted by the "*UP*" parameter if the cost value (objective value) is less than or equal to the current solution, which is always low in every iteration based on the decay rate. In a typical implementation of the GDA, the algorithm essentially begins with an initial water level as the quality of the initial solution, which is based on cost value. Then the level of water is decreased by an amount of decrease of the "*UP*" parameter. The best value for this parameter is between 0 and 1 (Dueck, 1993), unless the algorithm needs a long computation time to explore the uncovered area of the landscape to reach good solutions. There are many variants form of GD in the literature such as Extended Great Deluge (EGD), Flex-Deluge (FD), and Non-Linear Great Deluge (NLGD). A detail discussion of these variants can be found in Sin and Kham (2012).

Algorithm 2.5 Pseudo-code for a Great Deluge Algorithm (GDA).

```
    Input: Level $L$
 1: $Sol \leftarrow Sol_0$                          ▷ Generation of the initial solution.
 2: Choose the rain speed $UP$                                              ▷ $UP > 0$
 3: Choose the initial water level $LEVEL$
 4: Repeat
 5: Generate a random neighbour $Sol'$
 6: if $f(Sol') < LEVEL$ then
 7:     $Sol \leftarrow Sol'$                             ▷ Accept the neighbor solution.
 8:     $LEVEL = LEVEL - UP$                                ▷ Update the water level.
 9: end if
10: Until Stopping criteria satisfied
    Output: Best solution found.
```

Burke and Newall (2003) applied the GDA to examination timetabling problems to increase the quality of the initial examination timetables that could be obtained. In the initialisation phase, the examination timetables were constructed by an adaptive ordering method that was taken from Burke and Newall (2004). The performance of the GDA was compared with two local search algorithms, namely, HC and SA. In this study, several termination conditions were specified where the number of iterations for which the algorithm would be run was determined (i.e. two million iterations), stopping prematurely if there is no improvement in the solution quality after a certain number of iterations (i.e. one million iterations). The approach was tested on the Toronto problems. Experimental results showed that the GDA performance outperformed both HC and SA.

Furthermore, in Burke and Bykov (2006), in an extension of their work in Burke et al. (2004a), the flex-deluge algorithm was also proposed and implemented for solving the examination timetabling problems. Some modifications were made to the GDA and HC that helped to propose novel acceptance criteria are based on a flexibility coefficient. Updating the flexibility coefficient helped to explore the landscape effectively and guide the approach to avoid certain moves. Kempe chain neighbourhood (Thompson and Dowsland, 1996b, 1998) was used in order to fulfil all hard constraints. The approach has been verified on the Toronto benchmark datasets, achieving some of the best results in the literature at this time.

McCollum et al. (2009) investigated an extended GDA for solving examination timetabling problems from the Second International Timetabling Competition 2007 (ITC 2007). This approach consisted of two phases. In the first phase, the work was focused on employing the adaptive ordering heuristics proposed by Burke and Newall (2004) where the initial solution was constructed. The solutions were improved in the second phase (i.e. the improvement phase using a reheating mechanism). The approach was capable of returning good solutions compared to other published results.

Turabieh and Abdullah (2011b) incorporated a heuristic operator (i.e. an electromagnetic-like mechanism) with the GDA in order to solve capacitated and uncapacitated examination timetabling problems widely studied in the literature. The principle of the electromagnetic-like mechanism was based on Particle Swarm Optimisation (PSO) method first proposed by Birbil and Fang (2003). In this work, the initial solutions were generated by using the GC heuristics (i.e. saturation degree, largest degree and largest enrolment) (Balakrishnan, 1991; Weitz and Lakshminarayanan, 1997; Carter et al., 1996). The aim was to move towards the uncovered area of the search space so as to reach the global optimum where the decay rate of the GDA was frequently decreased. This helped the approach to decrease the probability of being trapped by a local optimum. The approach was competitive compared with other approaches where it obtained the best results on the Toronto and ITC 2007 benchmark datasets.

As aforementioned, the GDA algorithm has been successfully applied to timetabling problems and many optimisation problems. The GDA algorithm has an advantage over other algorithms in that it is simple and needs less effort to implement (Nahas et al., 2008; Jaddi and Abdullah, 2013). In addition, it requires only one input parameter to be tuned (Sin and Kham, 2012). However, the GDA algorithm suffers from the disadvantage of getting trapped quickly in local optimum, most often with low-quality solutions (McMullan, 2007; Ghatei et al., 2012).

2.4.3.6 Genetic Algorithms

Genetic Algorithms (GAs) were first developed by Holland (1992) based on the concept of natural selection. Specifically, they apply the principle of evolution through recombination, selection of the fittest and mutation; in nature this principle leads to the enhancement of species that are better adjusted for survival in a given environment, but Evolutionary Algorithms use this to solve computationally hard problems. GAs are repeated population-based approaches; they first search a set of candidate solutions, then a series of genetic operators (selection, recombination, and mutation) are repeatedly applied (Hoos and Stützle, 2004). In every iteration, these genetic operators generate new candidate solutions, which are replaced with the solution in the current population (partially or completely). Every candidate solution has a cost value, which corresponds to the objective function value (Hoos and Stützle, 2004). The pseudo-code of the basic steps of a genetic algorithm are described in Algorithm 2.6 (Talbi, 2009). The following main steps summarise GAs (Davis, 1991; Mitchell, 1998):

Step 1: The problem variable domain is represented as a fixed-length chromosome, which identifies the number of chromosomes in the population ($Popsize$), the mutation probability is referred to as (pm) and the crossover probability is referred to as (pc).

Step 2: A fitness function is determined to measure the fitness value of each individual chromosome in the problem domain and this then becomes the basis for choosing chromosomes that will be 'mated' during reproduction.

Step 3: The initial population of chromosomes is generated randomly of size $Popsize$: $x_1; x_2; x_3; \ldots; x_{Popsize}$.

Step 4: The fitness value of each chromosome is computed: $f(x_1); f(x_2); f(x_3); \ldots; f(x_{Popsize})$.

Step 5: A pair of chromosomes is chosen for mating from the current population. The selection of parent chromosomes is associated with a probability that is related to their fitness. The probability of a chromosome being selected for mating increases in line with its fitness value.

Step 6: The genetic operators, mutation and crossover, are applied to the selected pair of chromosomes to create a pair of offspring chromosomes.

Step7: The created offspring chromosomes are placed in the new population.

Step 8: Step 5 will be repeated until the size of the new population becomes equal to the size of the initial population ($Popsize$).

Step 9: The new (offspring) population will be replaced with the initial (parent) chromosome population.

As we can see, a GA creates an iterative process in which each iteration is called a generation. A typical number of generations for a simple GA can range from 50 to over 500

(Mitchell, 1998). The entire set of generations is called a run. At the end of a run, we expect to find one or more highly fit chromosomes.

Algorithm 2.6 Pseudo-code for a Genetic Algorithm (GA).

```
1:  Generate the initial population $Pop$ of solutions
2:  $t \leftarrow 0$
3:  Evaluate_Fitness($Pop$)
4:  while stopping criterion is not reached do
5:      $Parents \leftarrow$ Selection($Pop$)
6:      $Offspring \leftarrow$ Reproduction($Parents$)
7:      $Offspring \leftarrow$ Mutation($Offspring$)
8:      Evaluate_Fitness($Offspring$)
9:      $Pop \leftarrow$ Select_Best($Offspring \cup Pop$)
10:     $t \leftarrow t + 1$
11: end while
12: return the best individual in $Pop$.
```

GAs can therefore be characterised by the following features:

1. **The selection phase**: a process for choosing individuals (corresponding to solutions of the optimisation problem) from the population. The selection's primary objective involves emphasising good solutions while eliminating solutions that are bad in the population so that the population's overall fitness becomes better. Also, the selection can copy good chromosomes amongst the current population into the population of the following generation. Roulette Wheel Selection, Rank Selection and Tournament Selection, etc. are among the generally applied selection methods.

2. **The crossover phase**: a process for creating new offspring individuals by combining the information contained in the parent individuals.

3. **The mutation phase**: a mechanism for producing a new solution by random disturbance of the previous solutions.

4. A rule for updating the population (solutions from the current population).

Since the GA generally promised successful implementations, it was utilised in many studies of the timetabling problems. For instance, Erben and Keppler (1995) studied a weekly course timetabling problem in which they used GA to allocate teachers, classes, rooms and course modules to a number of periods within a week. They generated random populations of feasible solutions during the initiation stage. The mutation was executed by allocating new rooms and periods at random and a "cycle crossover operator" was used to create feasible offspring. They tested the algorithm using a large data sample and showed that the algorithm was able to achieve positive results.

Blum et al. (2002a) used GAs to deal with university class timetabling problems, seeking to reduce the complexity of those problems by initialising a solution over a set of consecutive heuristic rules. Myszkowski and Norberciak (2003) applied two GAs to solve theoretical and real-world course timetabling problems. They described a hybrid GA system and briefly debated the architecture of the system with two distinct solution representations.

Their hybridisation system involved hyper-heuristics to set up the operating parameters of the algorithm and TS to speed up the solution finding process. They protected 20% of the population size (10% of most varied solutions and 10% of the best quality solutions) to produce promising solutions. The most varied solutions were selected according to their greatest distance from the rest of the population, and the best quality solutions were selected based on their best fitness cost from the population. The diversity measurement, i.e. the distance between two timetables, was generated through one of the following methods:

- Number of pairs of events organised with identical resources in the same period in both timetables.
- Number of events organised with identical resources in the same period in both timetables.
- Search space convergence: how often the tuple <period, event, resources> is represented in the whole population.

A number of distinct crossover operators were proposed by Lewis and Paechter (2004) (e.g. conflict-based crossover, sector-based, student-based and day-based). In order maintain feasibility after crossover process, a genetic repair function was applied. They tested their algorithm on TTComp2003 datasets. The results showed that the most effective crossover method was the conflict-based crossover, and that the algorithm was able to produce a great number of different feasible timetables in a sensible amount of time.

Massoodian and Esteki (2008) investigated GA-based approaches to solve the curriculum-based course timetabling in track 3 of the ITC 2007. The approach comprises two stages: construction and improvement. The first stage focuses on obtaining a feasible solution, while the second stage minimises violations of the soft constraints. Furthermore, at each stage, Local search is employed to further refind the best chromosome. The approach was able to obtain good quality solutions quickly compared to employing GA alone.

A genetic algorithm was also applied by Sutar and Bichkar (2012) to find a solution to a real university timetabling problem in India. The initial population was generated randomly and the parents were chosen for crossover according to their fitness values. Mutation was applied to all the generated offspring resulting from crossover. The implementation of mutation and crossover operators was not obvious in their research, however. In respect to soft and hard constraints, learning space capacities were not mentioned, and they did not give priority to lecturers' period availabilities, although maximum and minimum working hours per week were allocated.

Colorni et al. (1991) examined GAs applied to the timetable problem, as did Fang (1994). Wong et al. (2002) also solved the examination timetabling problem with a GA. The authors developed an automation tool for exam timetabling using GA, as well as a repair mechanism to tackle infeasible solutions. They tested their system at the École de Technologie Supérieure of the Université du Québec. Pillay and Banzhaf (2010) introduced a two-phase solution to the examination timetabling problem, where the first phase generated feasible timetables, and second phase improved these timetables by reducing the violations of soft constraints. The evolutionary process was guided by domain-specific knowledge as a heuristic approach and a crossover operator is not employed, with only a mutation operator applied.

Burke et al. (1995a) applied GAs to minimise the number of periods required for exam timetabling problems. They employed uniform crossover operators, different selection methods, two graph colouring heuristics (i.e. Largest Degree and Largest Colouring Degree), and a random heuristic. They also developed special heuristics in order to address two constraints (i.e. the number of periods and spreading conflicting exams). These heuristic crossover operators were proposed to prevent infeasible timetables from being created during the recombination process. The experimental results revealed that producing good quality timetables depends on integrating heuristics in crossover operators. Burke et al. (1995b) successfully implemented similar heuristic crossover operators for another set of more difficult timetabling problems.

Colorni et al. (1991) investigated applying the GA to the timetable case. Applying a GA to timetabling, as well as scheduling, has also been investigated by Fang (1994). Wong et al. (2002), meanwhile, applied the GA specifically to the examination timetable problem. They are an exam timetable automation tool, which is based on a genetic algorithm. A mechanism for repair has been developed to repair infeasible solutions, which are the results of applying the variation operators. This system has been tested at the École de Technologie Supérieure of the Université du Québec. Caldeira and Rosa (1997) approached the high school timetabling problem by utilising the genetic algorithm. Based on their algorithm, the researchers adopted a problem specific chromosome representation. They used a repair algorithm after the genetic operators to avoid searching through non-feasible timetables and explored various fitness functions. In addition, Fernandes et al. (1999a,b) employed evolutionary algorithms intending to solve the problem of high school timetabling. Also, Pillay and Banzhaf (2010) applied a two-phase method to the examination timetable problem, where feasible timetables were produced in the first phase, and improvements were made to these in the second phase to reduce the costs of the soft constraints. Domain-specific knowledge in the form of heuristics was utilised to direct the evolutionary process, and only a mutation operator was utilised for the variation operators. Beligiannis et al. (2008) applied an evolutionary algorithm to a related school timetabling problem, and just as in the previous work, no crossover operator was utilised in the algorithm. Through this, the authors revealed that the use of mutation alone was adequate to obtain good new solutions. More research on using GA to solve course timetabling problem and how the diversity control is preserved can be found in Lewis and Paechter (2005), Blum et al. (2002a), Frausto-Solis et al. (2006), and Lewis (2012). In addition, Recent studies applying GAs to examination timetabling problems can be found in Obaid et al. (2012), Jha (2014), Gonsalves and Oishi (2015), Rozaimee et al. (2017), and Mohammed et al. (2017).

There are many advantages of the GA. The GA is robust and well-suited to apply to complex problems such as optimisation problems and in various domains such as Scheduling, Business, and Engineering (Zhang, 2011; Petrovski et al., 2005). Furthermore, the GA is inherently parallel, where it applies to a population of candidate solutions rather than a single solution Cantu-Paz (1999). Another advantage is the random generation of the initial population which means that the GA is able to sample the whole solution space and not just a small region. Variation-inducing tactics, i.e. mutation and crossover, is advantageous to prohibit the GA from being trapped in one part of the solution space Sumathi et al. (2008). Although GA has been extensively used in various problems, it suffers from some disadvantages. The GA has the inherent disadvantages of premature convergence and unpredictable results (Beheshti and Shamsuddin, 2013). It does not offer guarantees with respect to convergence to the global optimum. Additionally, it utilises complex functions in the selection and crossover operators, and the encoding scheme often is complicated and time-consuming (Beheshti and Shamsuddin, 2013). Moreover, in order to guide the searching to find the global optimum, the GA is governed by several parameters (Zhang, 2011). Thus, the effectiveness of GA highly depends on the tweaks that are made to these parameters. Furthermore, The performance of the GA largely depends upon its operators (i.e. crossover and mutation operators). However, the crossover operators often cause violations of the problem requirements (i.e. hard constraint violations) (Raghavjee and Pillay, 2013; Umbarkar and Sheth, 2015). A repair mechanism usually follows the crossover operator in order to repairs the infeasible solutions. Consequently, a longer computational time is required for the GA to employ this mechanism.

2.4.3.7 Ant Colony Optimisation

Ant Colony Optimisation algorithm (ACO) was first proposed by Dorigo et al. (2006). It is inspired by the ability of ants to identify the shortest way to transport their food through their deposit of pheromone. This algorithm simulates the behaviour of ants, and is based on the principle of positive feedback. Every ant relies on pheromone trails that are left by other ants to choose a path in unknown surroundings, adding its own pheromone trails in turn. The probability of future ants choosing that path is increased when more ants pass the same way. This positive feedback in time motivates the ants to choose the shortest path. The algorithm known as the pheromone mode is developed based on a parameterised probabilistic model with different pheromone values. Each trail has a pheromone value, and it is updated during every run-time to get a bias towards high quality solutions (Dorigo and Blum, 2005). Dorigo et al. (1991) employed this principle in a search meta-heuristic. They designed an artificial ant colony, where ants generate solutions. The quality of each solution affects the probability of further solutions being constructed. In choosing solution components, the probability of choosing node j by ant k, which is currently at the node, is calculated using the random proportional rule is determined by the following formula:

$$P_{ij}^{k} = \begin{cases} \frac{\tau_{ij}^{\alpha} \times \eta_{ij}^{\beta}}{\sum\limits_{k \in N_i^k} \tau_{ik}^{\alpha} \times \eta_{ik}^{\beta}} & \forall\ j \in N_i^k \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

Where τ_{ij} is the value of the pheromone path on the edge of connecting node i to node j, the heuristic value of that edge is given by η_{ij} where it indicates an estimation of the partial solution constructed until the current step and which will guide the ants' search with problem specific information. α and β represent the parameters which determine the effect ratio of the pheromone path (i.e. the pheromone information τ_{ij} on edge (i, j)), and the heuristic information η_{ij} on edge (i, j). The feasible neighbourhood of ant k from node i is given by N_i^k (i.e. the set of nodes that have not yet been visited by ant k currently located in node i).

The pheromone trail between node i and event/student node j marks the desirability of scheduling that event/student in that node. In each algorithm iteration, only one of the ants updates the pheromone trails based on the quality of the constructed solution candidate. The heuristic value η_{ij} is controlled by the parameters (i.e. α and β) for each ant. It is important to note that the heuristic value is dynamically modified throughout the algorithm execution (Socha et al., 2003). It is set to zero when it is determined that the constraints of the problem would be violated if the edge (i, j) were included in the tour, and to one otherwise. The pseudo-code of a ACO is illustrated in Algorithm 2.7 (Fidanova et al., 2019). The main pheromone trail update procedure is defined in Equation (2.12) where ρ reduces the value of the pheromone (i.e. the reduction rate of the pheromone), and $\Delta\tau_{ij}$ is a new added pheromone, that is proportional to the quality of the solution. The value of the objective function measures the quality of the solution that is obtained by the ant.

$$\tau_{ij} \leftarrow \rho\tau_{ij} + \Delta\tau_{ij} \tag{2.12}$$

Algorithm 2.7 Pseudo-code of a Ant Colony Optimisation (ACO).

```
1: Initialise number of ants
2: Initialise the ACO parameters: $m, \rho, \alpha, \beta$
3: while stop condition is not reached do
4:     for $k$ =0 to $m$ do
5:         ant $k$ chooses start node
6:         while $Sol$ is not constructed do
7:             ant $k$ selects higher probability ($P_{ij}^k$) node
8:         end while
9:     end for
10:    Update pheromone trails
11: end while
```

The important ACO parameters revealed in Algorithm 2.7 are: total number of ants (m), local pheromone coefficient (i.e. influence of pheromone) (α), heuristic coefficient (i.e. Influence of heuristic information) (β), pheromone evaporation (controls the rate at which historical information is lost)(ρ).

The ACO has been applied to solve complex combinatorial optimisation problems, including timetabling, and surveys on the application of the ACO can be found in Dorigo and Blum (2005). The MAX-MIN ant system and the ant colony system (ACS) system were proposed by Socha et al. (2003) for solving university course timetabling. They aimed to minimise the number of violations of soft constraints in feasible timetables, specifically, minimising the probability of a student having more than one class per day and limiting the number of classes that students have at the end of the day. The two algorithms begin with a population of ants and each ant builds a timetable by assigning all the events to periods. Ants choose periods probabilistically according to a matrix of pheromone values and heuristic information. The difference between the two algorithms lies in the ways in which pheromone values are updated. The ACS uses a special local update rule that is applied to the element which is selected in the pheromone matrix, symmetric to a certain period (p) for an event (e). It aims to decrease the probability of other ants choosing the same period for the same event, and to motivate them to choose other periods. When all events are assigned to periods, rooms are then assigned and a hill-climbing local search heuristic is applied to enhance the solution produced. The numbers of students attending and space capacities were taken into account as hard constraints. In addition to the local update rule, ACS uses a global update rule which sets lower and upper bounds to control the maximum difference between the lowest and highest pheromone levels. On other hand, The MAX-MIN ant system uses only a global update rule.

Dowsland et al. (2002) investigated the advantages of ant colony optimisation in examination timetabling. They attempted to distinguish between exam timetabling problems and random graphs by testing various modifications of the ant colony algorithm, such as assigning candidate exams lists first, as well as the impact of different measures of solution quality on the strength of the pheromone trail, etc. They revealed that these modifications could provide a foundation for extending the ant colony meta-heuristic to incorporate soft constraints.

Costa and Hertz (1997) developed a method called ANTCOL using ACO and a sequential heuristic for addressing graph colouring problems. In successive generations, each ant colours the vertices using dynamic (i.e. recursive largest first, saturation degree) or static (i.e. smallest last, largest first, random) constructive methods. The colour for each vertex is selected based on the probability value of the pheromone. The result of this experiment showed that the dynamic methods performed significantly better than the static methods. The importance of this research is that it highlights the promise of ACO for solving examination timetabling problems.

Dowsland and Thompson (2005) researched the application of ACO to the examination timetabling problem. The objectives of this study were to conduct a comparison between a set of random graphs observed by Costa and Hertz (1997) with the performance of ANTCOL on typical timetabling graphs, and to identify trail calculations, ANTCOL parameter values and promising constructive heuristic combinations. The results showed when ANTCOL was modified for application to the examination timetabling problem it was able to minimise the number of periods required for a feasible timetable comparing to the best published approaches in the literature. More details about ACO applied to exam timetabling problem can be seen in Azimi (2005).

The advantage of the ACO is that it is robust as it is capable of accepting any knowledge transferred by its pheromone trails (Mavrovouniotis, 2013). It is also versatile in dealing with a large range of combinatorial optimisation problems (Selvi and Umarani, 2010). Moreover, It also is inherently parallelised where it can easily search among a population in parallel (Zhang, 2011). Furthermore, it can quickly adapt to dynamic changes such as new distances as it is inspired from nature, so it has adaptation capabilities due to pheromone evaporation (Bonabeau et al., 1999; Mavrovouniotis and Yang, 2013). ACO is strongly based on positive learning on learning from positive feedback (i.e. positive learning). By means of positive feedback, the ACO attempts to determine which solution components are required to assemble high-quality solutions (Nurcahyadi and Blum, 2021). On the other hand, the significant disadvantage of the ACO is that the probability distribution (i.e. probability rule to choose solutions) can change changed for each generation (Guntsch and Middendorf, 2002). Thus, the ACO does not guarantee the discovery of the optimal solution, although it is a stochastic and multi-directional search algorithm (Lutuksin and Pongcharoen, 2010). Blum et al. (2002b) and Dorigo and Blum (2005) also reported that the original ACO has a deficiency in bias deception known as the first order and second order deception which means that some solution components will be updated on average more frequently than others. This is why an optimum solution is not guaranteed.

2.4.3.8 Memetic Algorithms

Memetic algorithms (MAs) are evolutionary-based approaches integrated with local search techniques. MAs are more advanced versions of GAs in that they can improve an individual within a generation. This purpose is often achieved by repairing individuals in the population between generations or using local search methods such as HC. MA begins the search with a random initial population (Hoos and Stützle, 2004). In each iteration, the next generation of candidate solutions is obtained by applying mutation, selection, recombination and perturbative local search, and the search process ends when a number of termination criteria are satisfied (Hoos and Stützle, 2004). Algorithm 2.8 shows the pseudo-code of a simple MA (Hart et al., 2004).

Algorithm 2.8 Pseudo-code for a Memetic Algorithm (MA) .

```
1: Initialise population
2: Evaluate each candidate solution
3: while termination criterion is not satisfied do
4:     Select parents
5:     Recombine to produce offspring
6:     Mutate offspring
7:     Improve offspring via Local Search
8:     Evaluate offspring
9:     Select individuals for next generation
10: end while
```

An MA of two local search methods integrated with the GA for solving Socha's datasets was presented by Jat and Yang (2008). The explorative search ability of GAs is improved by the use of the exploitative search ability of these local search methods. Rossi-Doria et al. (2002) mentioned that MA did not perform well in the experiments with only the first local search, while MA is efficient for solving the problem (Naseem and Shengxiang, 2009). Later, Yang and Jat (2011) widened their work into more sophisticated approaches involving GAs with local search techniques and a guided search strategy, which they called EGSGA and GSGA, respectively.

Generally, their GAs were built based on a steady state GA where one offspring is generated in each iteration. The initial population is generated randomly, and they applied two local search routines to lead the search toward the local optimal solution. Six neighbourhood structures were used in those local search routines to allocate courses into periods, and then a matching algorithm was used to allocate periods and rooms to courses. The good solutions were chosen from the population and these were progressed to the rest of the genetic operators. After mutation, the created offspring were improved by the same local search routines as had been applied in the initialisation. Finally, TS was utilised along with three neighbourhood structures to further enhancement the fitness of the best solution achieved from the preceding local search routines. The results of the two algorithms showed that they were effective in creating good quality results, especially the EGSGA for both the TTComp2003 and Socha datasets. Jat and Yang (2011) also used MA (a GA with TS) to solve the ITC 2007 (Track 2) datasets.

The combination of a GA and HC optimisation was proposed by El-Sherbiny et al. (2015) to construct a university course timetable. The authors' objective was to minimise the violation of any soft constraints. A set of hard constraints regarding professors and teachers were assigned: a class must be assigned to just one teacher in a period, and the room cannot be assigned more than once at any specific period. In addition, a specific number of periods were defined per week and a class should attend a specific number of lectures per week. A cost function representing the utilisation of spaces was considered in the set of soft constraints. The cost function aimed to minimise the difference between the capacity of the learning space and the number of students attending.

In recent work presented by Leite et al. (2016) applied the memetic algorithms proposed by Neri and Cotta (2012) in order to solve the examination timetabling problem. The proposed method is based on the Shuffled Complex Evolution algorithm(SCE) (Duan et al., 1993) and is hybridised with the great deluge (GD) algorithm (Dueck, 1993). Thus, the population is arranged to sets, called complexes, which evolve using recombination independently and then local search operators. The crossover operator, which they developed, maintains the diversity of the population, as does the solution update mechanism. Soria-Alcaraz Jorge et al. (2012) used a different approach for tackling the course timetabling problem, namely a parallel model in a meta-heuristic algorithm. The developed parallel approach uses the methodology of a design model for tackling the problem.

The advantage of MAs is that their ability to examine the search space more efficaciously than local search routines or standard EAs (e.g. GAs) (Burke and Newall, 1999; Blum and Roli, 2003, 2008; Burke and Petrovic, 2002), and it is this advantage that has led to their pervasive application in various fields of optimisation. Moreover, so as to utilise the search experience, a memory is used to supply components of elite solutions for the recombination operator, like using within Tabu Search (as in Burke et al. (2003d), and Hertz (1991)). On the other hand, one of the most typical disadvantages of MA is their tendency to converge prematurely after a number of iterations (Črepinšek et al., 2013). Another disadvantage is that the MAs often possess numerous parameters leading to very time-consuming with an unclear tuning (Nalepa and Blocho, 2016).

2.4.4 Hybrid Meta-heuristics

Two or more algorithms are collectively and cooperatively hybridised in order to solve a predefined problem (Ting et al., 2015). The idea of hybridisation is to enhance performance by merging other algorithms with a meta-heuristics algorithm. Raidl et al. (2010) argue, however, that it is possible to do the hybridisation among the of meta-heuristic algorithms themselves. In addition, there are other methods to do the hybridisation process that include the hybridisation between meta-heuristics with operation research methods, problem specific algorithms or with any other artificial intelligence algorithms. The authors also discussed human interaction hybridisation. The hybrid method aims to combine the best characteristics of one technique with good or better characteristics available in other techniques. Note that hybrid methods have been represented in some of the methods that have already been described (to a lesser or greater extent). Many researchers have proven the effectiveness of the hybridisation method in examination timetabling.

Caramia et al. (2001) implemented hybridisation on various benchmark instances and they got the best known results. They added improvements steps after applying a greedy scheduler which assigns exams that are arranged based on the degree of conflict in turn to the lowest obtainable period with regard to the conflict free requirement. Also, Merlot et al. (2003) obtained the best known results by applying a hybrid method. Their method involved combining CP, HC, and SA for capacitated and uncapacitated examination timetabling. They used a CP method to create feasible initial solutions. SA was used to improve the quality of the timetable, where they employed a Kempe chain neighbourhood (Thompson and Dowsland, 1996b,a, 1998). After that, HC was used to further improve the timetable. Based on their experimental results, the hybrid method was found to be preferable to the other methods currently applied by Melbourne University and also obtained better results when used with other standard benchmark instances.

Burke and Newall (2004) applied the hybridisation process on the method used by Burke and Newall (2003) with GD algorithm used by Burke et al. (2004a). Their method presents the best known results on examination benchmark problems. Côté et al. (2005) presented another good approach to the implementation of hybridisation on the examination timetabling problem.

A three-stage hybrid approach was proposed by Merlot et al. (2003). In the first stage, CP method was utilised in order to generate an initial feasible solution. SA algorithm was employed in the second stage to improve the obtained solution. The final stage was also considered as improvement stage where a further enhancement was applied by implementing a HC algorithm.

Duong and Lam (2004) investigated a hybrid approach that can be composed of constraint programming and simulated annealing. This hybridisation has been used to solve the exam timetabling problem at the HoChiMinh City University of Technology. Simulated annealing was employed on the initial solutions constructed by constraint programming. A Kempe Chain neighbourhood was utilised within the Simulated Annealing, whose cooling schedule was empirically determined by using mechanisms and algorithms. Backtracking and forward checking also were used to minimise the searching effort. The findings revealed that it is crucial to tune the components in hybridisation to solve the specific problems.

Hybridisation offers significant advantages in increasing the diversity in a population and enhancing the search capability of the developed hybrid algorithm (Ting et al., 2015). Hybridisation also provides an efficient behaviour and higher flexibility when suitable meta-heuristics are hybridised (Blum et al., 2011). For instance, the combination of a local search method and an evolutionary algorithm can refine solutions during searching. The evolutionary algorithm has the power of identifying the promising regions of the search space easily, while the local search method is effective to discover the best solutions of high quality in such promising regions, thus promoting efficient behaviour and higher flexibility. However, the disadvantages of these algorithms are the need for proper setting of the algorithm-dependent parameters, and a large number of iterations is required (Pei et al., 2019). Hybridisation increases the complexity of the hybrid algorithm as it is required extra components in the overall architecture of the hybrid algorithm. As a result, there is still some resistance to accept hybrid algorithms by researchers (Ting et al., 2015).

2.4.5 Hyper-heuristics

While meta-heuristics techniques are considered to be among the most efficient methods that can be applied to the benchmark data, their use to solve examination timetabling problems is affected by the reliance on parameter tuning, or the method of embedding domain knowledge (in other words, the hard coding of soft and hard constraints). This means that these techniques have to be tailor-made for one specific problem, and produce poor results when applied to different problems. While parameter tuning is a very effective way of enhancing the techniques, the task of tuning parameters to make the used techniques fit a new problem is equivalent to that of developing an entirely new algorithm. This has prompted many researchers to try to develop new methods that will work at a higher level of generality.

Hyper-heuristics (HH) is a promising approach in this regard, and has attracted substantial scholarly attention. The term hyper-heuristics can be represented as heuristics that select heuristics, where the attention is focused on the heuristics' search space instead of the solutions' search space, as is widely used in meta-heuristic-based implementations (Burke and Erben, 2003; Ross et al., 2003). Hyper-heuristics aims to allow the approaches to be more general, overcoming the fine-tuning, and problem-particular methods that often need great effort for a particular problem.

Meta-heuristics such as SA, TS and GA can be employed as a high-level strategy while constructive heuristics, such as graph heuristics and constraint satisfaction can be low-level strategies. Hyper-heuristic methods have been implemented successfully in order to tackle highly constrained optimisation problems such as timetabling (Burke et al., 2007), course timetabling (Burke and Newall, 2003; Soria-Alcaraz et al., 2014), examination timetabling (Bilgin et al., 2006; Burke et al., 2006), sales summit scheduling (Cowling et al., 2000), vehicle routing (Bai et al., 2007), etc. Ross et al. (2004) proposed a general steady state GA approach in order to construct solutions by search within a simplified search space of problem state descriptions. The GA search was on heuristics instead of actual solutions. Three different fitness functions were tested in order to study the descriptions of the problem state (corresponding to heuristics) experimentally. The study showed that the proposed approach delivered promising results for course and exam timetabling problems and can be applied effectively to a wide range of problems.

The approach proposed by Sabar et al. (2012) involved constructive heuristics selection hyper-heuristics applied to the examination timetabling problem, where the ordering of all exams is dependent on a difficulty index (DI). In the latter, the work difficulty index is calculated using a hierarchical hybridisation of different graph colouring heuristics such as Largest Enrolment, Saturation Degree, Largest Degree and Largest Weighted Degree (as will be discussed in Chapter 3). Also, Roulette Wheel Selection was used to assign exams into periods. The proposed approach was tested on the Toronto and ITC 2007 datasets and showed promising and also competitive results when compared with the previous work in the literature. The most recently-published studies on examination timetabling problems with hyper-heuristics can be found in Soria-Alcaraz et al. (2016), Muklason et al. (2017), and Kusumawardani et al. (2019).

The significant advantage of hyper-heuristics is that they lead to a more general approach that can handle a wide range of problem domains compared to current meta-heuristic algorithms, which tend to be customised to a particular problem (Burke et al., 2013). Another advantage is that hyper-heuristics have the potential of saving a significant amount of effort as they can automate the process of tuning and choosing a heuristic to a given class of problem instances. This process is often the time-consuming and expensive part of the implementation of a heuristic (Burke et al., 2009). However, the major disadvantage of hyper-heuristics is that they are computationally more expensive compared to other heuristics as they often require an iterative process for training (Garza-Santisteban et al., 2020). In addition, designing and implementing hyper-heuristics is excessively expensive and very complicated for users to understand and maintain (Burke et al., 2003c).

2.4.6 Multi-objective Techniques

Multi-objective optimisation involves optimising two or more contrasting objectives simultaneously subject to a number of constraints (Deb and Deb, 2014). Multi-objective evolutionary algorithms (MOEAs) are effective to solve real-world and benchmark Multi-objective optimisation problems (MOPs) due to several reasons, which are: flexible and robust, fewer chances to be trapped in local minima, returning more than one optimal solution, and easy to implement (Guliashki et al., 2009). MOEAs are based on population, where the iterations are performed on a set of solutions (i.e. population), and after each generation, it returns a number of solutions. Multi-objective evolutionary algorithms can be divided into two classifications: elitist and non-elitist algorithms (Rudolph, 1994). Regarding elitist MOEA, there is a mechanism that can preserve good solutions in every iteration, while there is no such mechanism in non-elitist MOEA. Therefore, the performance of the non-elitist algorithm is often worst compared to the elitist algorithm.

In MOEA, a solution a can be said to be better than (i.e. dominate) solution b, if it returns equal or better objective functions on all objectives considered, and is strictly better on at least one. Furthermore, a solution is said to be non-dominated if there are no other solutions considered that dominate it. If no feasible solution dominates a, then solution a is known as a *Pareto optimal* solution (Teich, 2001; Coello et al., 2007).

In general, the examination timetabling problem is often implemented just as a single-objective problem. However, it is inherently a multi-objective optimisation problem (Burke et al., 2001, 2008). Researchers only take into account the number of clashes in an examination timetable. Burke and Newall (2004) declared that when the timetable is allocated a considerable number of periods, it is possible to eliminate the clashes, and Burke et al. (1995a) also declared that a longer timetable is needed to minimise conflicts. But, it is clear that the examination timetabling problem is a two-objective optimisation problem; the two objectives to minimise are the number of periods and the number of clashes. Ideally, such multiple conflicting cost functions should be minimised using multi-objective optimisation (Cheong et al., 2007).

Many studies of MOEAs have been presented recently. Malim et al. (2006), for example, applied three distinct artificial immune systems for both exam and course timetabling problems and claimed that the algorithm is convenient for the two timetabling problems. Later, however, they discovered a mistake in their code, and their research is invalid (Qu et al., 2009b). McCollum et al. (2007) provided a remarkable expansion to the combination of realistic instances of the ITC 2007 competition in the exam timetabling track. Comparing the new model with that used in the examination track with the Toronto one, the new model is used much more extensively since it is a lot of distinct soft constraints that are prepared in order to build timetables that achieve the requirements of the different individual parties. The violations in constraints have penalties, and this can lead to having a multi-objective optimisation problem.

Even though the examination problem has previously been represented as a multi-objective problem, Prida Romero (1982), Burke et al. (2001), McCollum et al. (2007), and McCollum et al. (2012a) have provided a new outstanding formulation that gives rise to new opportunities and obstacles. Each distinct penalty is given a weight in order to appear in the instances as single-objective problems. Although experience informs the weights that express the trade-off between penalties, the consequences of changing them need studying since they are still partially ad hoc.

A lot of previous researches were conducted on multi-objective method for exam timetabling using the Toronto dataset, as an uncapacitated exam timetabling problem. Also, to handle each constraint as a distinct objective function, as in works provided by Paquete and Fonseca (2001), and Petrovic and Bykov (2002), the standard objective function is being minimised by the objectives. For example, decreasing the number of periods and proximity cost. A lot of researches have been conducted for this work as Wong et al. (2004), Mumford (2010), Côté et al. (2005), Cheong et al. (2007), and Cheong et al. (2009).

Paquete and Fonseca (2001) studied bi-objective exams timetabling. In the problem formulation, the objectives are just the soft constraints, specifically between-group and within-group. In between-group constraint, exams in a group have not to be scheduled at the same time with any exam from other groups, whereas in within-group constraint, exams in the same group have not to be scheduled on the same day. In this research, objective evolutionary algorithms are used to solve the problem. In addition, Petrovic and Bykov (2002) studied a nine-criteria and bi-criteria problem. Based on Burke and Newall (1999), in the bi-criteria problem, the objective function includes the number of conflicts; the students have two exams in adjacent periods, and students have to take exams in adjacent overnight periods, whereas the objective functions in the nine-criteria problem are based on Burke et al. (2001) and includes three groups related to room capacities, the proximity of exams, and finally the time and order of exams.

Wong et al. (2004) presented a study that applied a hybrid multi-objective evolutionary algorithm with two objectives. The first one, decreasing the number of students that must take a successive exam, and the second one minimising the timetable length and the number of periods. They represented a solution as a vector, including exams, and the value of each element represents a period. Two local searches that relied on the tabu search algorithm were applied instead of the crossover operators. The function of the first one is to reform the infeasible solution, and the second one is to decrease proximity cost. A random solution is generated without attention to its feasibility when the algorithm starts. Then the first local search algorithm utilises a one-move neighbourhood structure by moving the exams having a conflict to another period. After that, the second local search starts its work, where a VNS is simplified (Hansen and Mladenović, 2001) with two neighbourhood structures; that is one-move and Kempe chain. In addition, based on the Pareto Strength concept, the solutions are ranked (Knowles and Corne, 2000). The mutation operator is applied by allocating a random period to choose exams. The pre-defined mutation rate is identified for each exam.

Extended the work by Burke et al. (1995c), graph colouring heuristics is utilised by the reinsertion method to arrange the eliminated exams. A Pareto fitness ranking scheme, an expansion of the Pareto fitness ranking scheme, is used to rank the solutions (Fonseca, 1995) by penalising the rank when the length of the timetable is not within the required range of the timetable length. It is comparable to the goal sequence domination scheme (Tan et al., 2003). In order to exploit the optimal solution, hill climber and micro-genetic algorithm, which is GA with short evolution and small population (Dozier et al., 1994) are combined with the single objective function. Five problem instances of the Toronto dataset and one problem instance of the Nottingham dataset were used to test the implemented algorithm. The wanted length of the timetable is identified to three below, and three above the particular periods.

Cheong et al. (2009) declare that the implemented method executed well when they compared it with seven other recently applied optimisation approaches, and the best results for four out of the seven datasets were tested. It must be observed, however, that the objective was amended from the original version over the tested techniques that it compared to. There was also doubt about how they compared their results with prior results because the objectives are distinct.

Mumford (2010) applied for the same work where she optimised two objective functions, which were used to optimise the standard objective function (the spread of examinations for individual students), and at the same time, the proposed approach also minimised the length of the timetable. This approach also employed a greedy algorithm with a memetic algorithm to solve the problem. Although these are initial works for solving exam timetable problems, few recent studies have mentioned a multi-objective method for ITC 2007 dataset, despite the description of multi-objective for this problem being specified by Burke et al. (2008).

The most well-known multi-objective algorithms include NSGA-II (Non-dominated Sorting Genetic Algorithm-II) (Deb et al., 2002), SPEA (Strength Pareto Evolutionary Algorithm) (Zitzler and Thiele, 1998), SPEA2 (Strength Pareto Evolutionary Algorithm2) (Zitzler and Thiele, 1998; Zitzler et al., 2001), PAES (Pareto Archived Evolution Strategy) (Knowles and Corne, 2000), PESA (Pareto Enveloped based Selection Algorithm) (Corne et al., 2000), and PESA-II (Pareto Enveloped based Selection Algorithm-II)(Corne et al., 2001). These approaches have been successfully implemented to solve MOPs. Table 2.13 highlights the advantages and disadvantages of these algorithms (Konak et al., 2006).

Algorithm	Advantage
NSGA-II	Single parameter tuned algorithm, and efficient.
SPEA	No need to define any parameter for clustering.
SPEA2	Improved SPEA and extreme solutions are preserved.
PAES	Implementation is simple and computationally efficient.
PESA	Implementation is simple and efficient.
PESA-II	Extended version of PESA, and good solutions diversity.
Algorithm	**Disadvantage**
NSGA-II	Only objective space is considered when the crowding distance is applied.
SPEA	Clustering takes more time and extreme solutions may not get preserved.
SPEA2	Rank assignment and diversity preservation methods are more time consuming.
PAES	Its performance depends on cell size, and does not belong to population-based approaches.
PESA	Its performance depends on size of hyperbox.
PESA-II	Its performance depends on size of hyperbox.

Table 2.13 Advantages and disadvantages of multi-objective evolutionary algorithms (MOEAs).

2.5 State-of-the-Art Meta-heuristics for the Examination Timetabling

The examination timetabling problem has become an interesting research area for a lot of researchers from various fields, such as operational research (OR) and artificial intelligence (AI). In spite of the extensive literature, the exam timetabling problem remains an interesting research topic as many emerging meta-heuristic techniques continue to offer promising results. This section provides a review of seminal works combined with state-of-the-art approaches investigated in the area of research. Three popular examination timetabling problems, namely, The Toronto, ITC 2007, and Yeditepe datasets, are used for evaluating the performance of the proposed approaches in this thesis. As mentioned earlier, examination timetabling problems can be categorised as either capacitated or uncapacitated (Kahar and Kendall, 2010). The Toronto problem is considered a standard uncapacitated examination timetabling problem. Both the ITC 2007 and Yeditepe examination datasets are capacitated problems.

Surveys on state-of-the-art examination timetabling problem algorithms techniques, and formulations have been reported in previous studies, such as those of Carter et al. (1996), Schaerf (1999), McCollum (2007), Lewis (2008), Qu et al. (2009b), and Burke and Petrovic (2002). Furthermore, Tables 2.15 and 2.16 provide a survey of approaches for examination timetabling problems reported between 1967 until recent date. The name of the methods, examination datasets used, and number of examination datasets that have been used are given in these tables. We also extend the literature survey with recently-published literature summarised in Table 2.17.

2.5.1 Approaches Applied to the Uncapacitated Examination Timetabling

Yang and Petrovic (2005) apply a hyper-heuristic approach merged with case-based reasoning in order to select graph heuristics to create a feasible initial solution. A great deluge algorithm is then implemented to enhance the initial solution. The authors use hyper-heuristics to explore the search space of heuristics instead of searching for direct solutions. They applied their approach to the Toronto problem and obtained the best results in the literature for several instances at this time.

As mentioned in Section 2.4.3.7, the ant algorithm has been used to solve hard combinatorial optimisation problems, including exam timetabling (Dorigo and Blum, 2005). Eley (2007) implemented an ant algorithm and incorporated the ant systems and the max-min ant systems with two randomised strategies in order to find the pheromone trail and the constructive heuristic. Several parameters need to be considered during the implementation, such as the evaporation rate, the weighting factors, and the number of cycles, in order to make sure that the proposed algorithm worked efficiently. The proposed approach was tested on the Toronto benchmark sets, and obtained results that were competitive with the best published approaches.

Caramia et al. (2008) implemented a hybrid approach to tackle capacitated and uncapacitated examination timetabling problems. This study attempts to generate a high-quality exam timetable with small length (i.e. a minimum number of periods). The scheduling process starts with a greedy scheduler by attempting to assign examinations into the least number of periods as well as conflict-free slots. The approach also allocates exams by placing those with the highest conflicts first in order to identify the number of periods and ensure that all exams can be scheduled. Once the process has been completed, HC is employed as a 'penalty decreaser' to minimise the number of periods and maximise the quality of the timetable. The process continues until there is no more improvement.

At this stage, HC has applied again as a ‘penalty trader’. This approach was tested on the Toronto and Nottingham benchmark sets and demonstrated an ability to produce high-quality solutions superior to many best-known approaches in the literature.

A late acceptance strategy was proposed by Burke and Bykov (2008), being a new variant of hill-climbing. They investigated the performance of the proposed method by applying it to uncapacitated exam timetabling problem (i.e. the Toronto benchmark sets). Although it is categorised as an iterative search method, it depends on a more sophisticated move acceptance mechanism. A new candidate solution is compared to solutions obtained from prior iterations. A list C of length L is created to store the cost function values of accepted solutions from each iteration. During the search iterations, a neighbouring solution with a better or equal cost value is accepted in the list.

In addition, Burke et al. (2010b) proposed a variable neighbourhood search (VNS) incorporated within a GA. In this work, different neighbourhood structures such as descent-ascent, biased VNS, and problem-specific neighbourhoods were investigated. Also, various initialisation strategies (i.e. greedy and a random construction technique) were applied. Although the proposed approach was capable of obtaining a high quality solution for one instance of the Toronto problem, it requires a relatively large amount of computation time.

Pillay and Banzhaf (2010) used a two-phase GA approach to solve the examination timetabling problem. In the first phase, GAs are used to generate feasible solutions for constructing the initial population. During the second phase, GAs are further utilised to generate feasible solutions from the previous phase, while attempting to minimise the cost of the soft constraint. They use domain-specific knowledge in the form of heuristics to guide the evolutionary process. The effectiveness of this approach has been verified on the Toronto benchmark sets.

Abdullah and Alzaqebah (2013) presented a hybridisation approach combining a modified bees algorithms with local search algorithms (i.e. simulated annealing, late acceptance hill-climbing). In order to exploit and fully explore the entire search space, they use three selection strategies (i.e. disruptive, tournament, and raking) and a self-adaptive technique. The aim of the selection strategies is to improve the diversity of the population, while the self-adaptive method is used to monitor the neighbourhood search and prevent the algorithm from getting stuck in a local optimum. The approach has been tested on thirteen instances of the Toronto and eight instances of the ITC 2007 benchmark sets.

In Leite et al. (2016), the proposed approach implements a memetic algorithm so-called “Shuffled Complex Evolution Algorithm” in which the population is organised into complexes (sets) that are evolved individually by using local search operators and recombination. Diversity of population is maintained by using various recombination operators and applying a special mechanism to update the solution. Fong et al. (2015) proposed a hybrid swarm-based algorithm to academic timetabling. The algorithm can be used in different university timetabling problems, namely both examination and course timetabling, and has been tested on the Toronto benchmark dataset and the Socha benchmark set, respectively.

Alzaqebah and Abdullah (2015) conducted two hybridisations using the bee colony optimisation algorithm (BCO) in both cases. The first hybrid couples the BCO algorithm with the HC algorithm by using late acceptance strategies, while the second hybrid couples BCO with SA. In comparison with others, the first hybrid achieves the best results for both datasets (Toronto and ITC 2007 problems).

A cellular memetic algorithm is proposed in Leite et al. (2018). It incorporates a cellular evolutionary algorithm with threshold acceptance local search to tackle the examination timetabling problem and is evaluated on the Toronto and ITC 2007 problems. Experimental results showed that the approach is able to improve on four out of thirteen instances of the Toronto set and three out of twelve of the ITC 2007 set.

Khair et al. (2018) developed an approach implementing an ACO, where the ant system (AS) has been used with the aim of obtaining good feasible solutions for university exam timetabling problems. The main purpose of this system is to optimise and discover paths ways based on index updating of the pheromone. The proposed approach was empirically tested over 33 instances of uncapacited examination timetabling problem from the Faculty of Informatics and Computing (FIC), University of Sultan Zainal Abidin (UniSZA). The experimental results showed that the performance of ACO was capable of solving the exam timetable scheduling problems generated from UniSZA and expanding between two or more adjacent exams for each student and obtaining good quality solutions. However, further improvements are needed for the proposed approach in order to solve public benchmark problem datasets and allow for the production of high-quality solutions.

2.5.2 Approaches Applied to the Capacitated Examination Timetabling

Müller (2008) sought to solve the three problems that were established by the ITC 2007 competition, winning two of them and being a finalist on the third. To solve the problems, he preferred a hybrid approach, which is organised in a two-phase algorithm. The Iterative Forward Search (IFS) algorithm is used in the first phase (Müller, 2005) so that feasible solutions, as well as Conflict-based Statistics, are obtained (Müller et al., 2004) in order that the IFS is prevented from looping. The second phase involves using multiple optimisation algorithms applied in the following order: HC (Russell and Norvig, 2010), GD (Dueck, 1993), and, optionally, SA (Skiundefinedcim and Golden, 1983).

Gogos et al. (2008) achieved second place in the Examination Timetabling track. Like Müller's approach, this involves two phases. The first phase commences with a preprocessing stage. In this stage, hidden dependencies between the exams are checked in order to accelerate the optimisation phase. The second stage is a construction stage using a meta-heuristic called Greedy Randomised Adaptive Search Procedure (GRASP). In this phase, optimisation methods can be applied in the following order: HC, SA, IP (the Branch and Bound procedure), finishing with the so-called 'Shaking Stage', which can only be applied according to certain conditions. This Shaking Stage 'shakes' the current solution by passing back to the SA phase in order to create a similarly good solution. This stage aims at forcing SA to restart with solutions that are more promising so as to generate better results.

Atsuta et al. (2008) won third place in the ITC 2007 Examination Timetabling track, as well as second place on other tracks, applying a similar approach in each case. The approach involves applying a constraint satisfaction problem-solver that adopts a hybridisation of TS, as well as Iterated Local Search (ILS).

De Smet (2008) differs from other approaches because he did not use a problem-specific heuristic, which is known to find a feasible solution. Instead, he used the Drool's rule engine called the drools-solver (Drools, 2020). The drools-solver involves a combination of optimisation heuristics, in addition to meta-heuristics with a very efficient score calculation. The score of the solution is the sum of the constraints' weight, which is being broken. After a feasible solution is found, he used TS to enhance the solutions obtained from the drools-solver.

A two-phase algorithm variant was proposed by Pillay (2008) using a developmental approach based on cell biology. The goal involves forming a well-developed organism through creating cells and proceeding with the cell division, cell interaction and cell migration processes. In this approach, each of the cells signifies a period. The first phase involves the process of creating the first cell, cell division and cell interaction, while the second phase involves cell migration.

McCollum et al. (2009) applied an adaptive ordering heuristic to construct solutions followed by an extended meta-heuristic version of GD. The approach was tested on the exam timetabling problems from ITC 2007. It was confirmed as an effective approach, obtaining the best results in five out of the eight instances.

Demeester et al. (2012) employed a hyper-heuristic based approach to resolve three timetabling problems: Toronto, ITC 2007, and the KAHO Sint–Lieven (Ghent, Belgium) timetabling problem. The authors applied a construction, as well as an improvement approach. In the event that no feasible solution is obtained in the construction phase, the algorithm continues with the improvement phase. Also, extra correcting actions can be performed to eliminate infeasibilities.

Gogos et al. (2012) proposed an enhanced algorithm version. The authors claimed that the enhanced behaviour is due to more sophisticated process flow, early detection of plateaus, added heuristics, as well as optimised data structures, which achieve the exploration of a much larger number of Kempe Chain moves.

An adaptive BCO was proposed by Alzaqebah and Abdullah (2014) combined with a late-acceptance HC algorithm. This proposed method was applied to the Toronto and ITC 2007 benchmark sets. Also, in a study that was recently conducted by Alzaqebah and Abdullah (2015), the researchers proposed a hybrid BCO to solve the examination timetable problem.

A single-stage procedure was proposed by Battistutta et al. (2017) based on the SA approach for the ITC 2007's examination timetable problem. Based on this method, non-feasible solutions are included in the search space, dealing with appropriate penalties. A statistically-principled experimental analysis was conducted to investigate the parameter selection effect. Then, a feature-based parameter tuning is performed. A memetic algorithm, called the cellular memetic algorithm, was proposed by Leite et al. (2018), involving a hybrid-based approach.

Muklason et al. (2017) proposed a multi-phase approach in order to solve the Toronto, ITC 2007, and Yeditepe problems and find good solutions that cloud match student preferences and meet their satisfaction. This approach consists of three phases. In phase 1, an initial feasible solution is produced by an adaptive heuristic ordering approach. Phases 2 and 3, respectively, use a selection of hyper-heuristics to improve the quality of the initial solution, and attempt to achieve fairness in the optimised solutions.

More recently, the structure-based partial solution search (SBPSS) by Rajah and Pillay (2019) improved on the best results of ITC 2007 Examination track. The SBPSS is a multi-point search approach that aims to solve this problem incrementally. Solutions are initialised partially, and at each generation, one solution component is selected randomly and added to each partial solution continuously. This process stops when all solution components are completely added to the solutions. Leite et al. (2019a) introduced the latest version of the Simulated Annealing algorithm, which is called "FastSA". In the approach provided, each chosen exam would only be changed if there were any permitted changes in the immediately preceding temperature bin the stated exam. Ten temperature bins have been developed, and FastSA has verified that an equal number of evaluations are carried out in each bin. If an exam does not have any accepted movement in the preceding temperature bin, the exam possibly would have few or zero accepted future movement. The proposed FastSA and the basic SA have been applied over the ITC 2007 exanimation scheduling problem. The experimental results showed that the proposed approach outperformed the basic SA for four out of twelve problem instances of the ITC 2007 and obtained result beating the best-known result over one problem instance of the ITC 2007. However, the proposed approach needs a large amount of computational time to produce final solutions compared to the with state-of-the-art approaches.

Various hyper-heuristic strategies were proposed by Muklason et al. (2017). They consist of combining several hyper-heuristics strategies, i.e. great deluge (GD), extended great deluge (XGD), and modified extended great deluge (MXGD) and different low-level heuristic selection strategies, including simple random (SR), reinforcement learning (RL), and self-adaptive (SA) learning. Therefore, nine hyper-heuristic strategies, as shown in Table 2.14, are introduced in this work and applied to the Yeditepe Benchmark Set. In addition, the author proposed three methods to construct feasible initial solutions involving adaptive ordering heuristics with hierarchical hybridisation of basic graph colouring heuristics, sequential construction with the maximal clique and saturation degree heuristic, and an adaptive linear combination ordering heuristic with a heuristic modifier based on squeaky wheel optimisation (SWO) (Joslin and Clements, 1999).

No	Hyper-heuristics	Heuristics Selection Method	Move Acceptance Method
1	SR-GD-HH	Simple Random	Great Deluge
2	RL-GD-HH	Self-adaptive	Great Deluge
3	SA-GD-HH	Reinforcement Learning	Great Deluge
4	SR-XGD-HH	Simple Random	Extended Great Deluge
5	RL-XGD-HH	Self-adaptive	Extended Great Deluge
6	SA-XGD-HH	Reinforcement Learning	Extended Great Deluge
7	SR-MXGD-HH	Simple Random	Modified Extended Great Deluge
8	RL-MXGD-HH	Self-adaptive	Modified Extended Great Deluge
9	SA-MXGD-HH	Reinforcement Learning	Modified Extended Great Deluge

Table 2.14 The strategy within Hyper-heuristics (HH) framework (Muklason, 2017).

Approach	Method	Year	Examination Dataset			No. Examination dataset used	Reference
			Toronto	ITC 2007	Other		
Graph based sequential techniques	Reduction to Graph Colouring	1967	x	x	x	0	(Welsh and Powell, 1967)
		1981	x	x	✓	1	(Mehta, 1981)
		1982	x	x	✓	1	(Mehta, 1982)
	Graph based ordering: saturation degree.	1979	x	x	✓	1	(Brélaz, 1979)
	Largest weighted degree	1996	✓	x	x	1	(Carter et al., 1996)
	Largest enrolment	1968	x	x	✓	1	(Wood, 1968)
		2004	x	x	x	0	(Burke et al., 2004b)
	Extended clique	1998	✓	x	x	1	(Burke et al., 1998b)
		2001	✓	x	x	1	(Carter and Johnson, 2001)
		2004	✓	x	x	1	(Burke and Newall, 2004)
	Fuzzy system	2005	✓	x	x	1	(Asmuni et al., 2005a)
		2007	✓	x	x	1	(Asmuni et al., 2006)
	Neural network	2006	✓	x	x	1	(Corr et al., 2006a)
	Backtracking	1996	✓	x	x	1	(Carter et al., 1996)
	Look ahead	2007	✓	x	x	1	(Burke and Newall, 1999)
Constraint based techniques		1997	x	x	✓	1	(David, 1998)
		1999	x	x	x	0	(Brailsford et al., 1999)
		1999	x	x	✓	1	(Reis and Oliveira, 1999)
		2003	✓	x	✓	2	(Merlot et al., 2003)
		2004	x	x	✓	1	(Duong and Lam, 2004)
		2006	✓	x	x	1	(Le Huédé et al., 2006)
Meta-heuristics: 1) Local search based algorithms	Tabu search	2000	✓	x	✓	2	(Di Gaspero and Schaerf, 2000)
		2001	✓	x	✓	2	(White and Xie, 2000)
		2002	x	x	✓	2	(Paquete and Stützle, 2002)
		2002	✓	x	x	1	(Di Gaspero, 2002)
		2004	✓	x	x	1	(Di Gaspero, 2002)
	Simulated annealing	1996	x	x	✓	5	(Thompson and Dowsland, 1996b)
		1998	x	x	✓	8	(Thompson and Dowsland, 1998)
		1998	x	x	✓	1	(Bullnheimer, 1997)
		2002	✓	x	✓	2	(Merlot et al., 2003)
	Great deluge algorithm	2003	✓	x	x	1	(Burke and Newall, 2003)
		2004	✓	x	✓	2	(Burke et al., 2004a)
		2004	✓	x	✓	2	(Burke and Newall, 2004)
	Large neighbourhood search	2007	✓	x	✓	2	(Abdullah et al., 2007b)
		2007	✓	x	x	1	(Abdullah et al., 2007a)
	Variable neighbourhood search (VNS)	2006	✓	x	x	1	(Burke et al., 2010a)
	Iterated local search	2000	✓	x	x	1	(Caramia et al., 2001)
		2002	x	x	✓	2	(Paquete and Stützle, 2002)
		2008	✓	x	✓	2	(Caramia et al., 2008)
	Greedy randomised adaptive search procedure (GRASP)	2002	✓	x	x	1	(Casey and Thompson, 2002)
2) Population based algorithms	Genetic algorithm	1994	x	x	✓	2	(Corne et al., 1994b)
		1994	x	x	x	0	(Corne et al., 1994a)
		1995	x	x	✓	1	(Ross et al., 1995)
		1997	✓	x	x	1	(Ross et al., 1997)
		1999	✓	x	x	1	(Terashima-Marín et al., 1999b)
		1999	x	x	x	0	(Terashima-Marín et al., 1999a)
		2000	x	x	✓	1	(Erben, 2000)
		2002	x	x	✓	1	(Sheibani, 2002)
		2002	x	x	✓	1	(Wong et al., 2002)
		2003	✓	x	✓	2	(Ross et al., 2003)
		2004	✓	x	x	1	(Côté et al., 2005)
		2006	✓	x	✓	2	(Ülker et al., 2006)
	Memetic algorithm	1995	✓	x	x	1	(Burke et al., 1995c)
		1998	✓	x	x	1	(Burke et al., 1998a)
		1998	✓	x	x	1	(Burke et al., 1998b)
		1999	✓	x	x	1	(Burke and Newall, 1999)
	Ant algorithm	2005	✓	x	x	1	(Azimi, 2005)
		2005	✓	x	x	1	(Dowsland and Thompson, 2005)
		2007	✓	x	x	1	(Eley, 2007)
	Artificial immune algorithms	2006	✓	x	x	1	(Malim et al., 2006)
Multi-criteria techniques		1995	x	x	✓	1	(Colijn and Layfield, 1995)
		2000	x	x	✓	1	(Burke et al., 2001)
		2002	✓	x	✓	2	(Petrovic and Bykov, 2002)
		2004	x	x	x	0	(Silva et al., 2004)
		2005	x	x	✓	3	(Burke et al., 2005b)
		2006	✓	x	✓	2	(Le Huédé et al., 2006)
		2007	✓	x	✓	2	(Cheong et al., 2007)
Hyper-heuristics	Tabu-search based hyper-heuristic	2005	✓	x	x	1	(Kendall and Hussin, 2005a)
	Graph-Based hyper-heuristic	2007	✓	x	x	1	(Burke et al., 2007)

Table 2.15 Survey of approaches for examination timetabling problems by 2008.

Approach	Method	Year	Examination Dataset			No.dataset used	Reference
			Toronto	ITC 2007	Other		
Graph based sequential techniques	Standard constructive graph colouring base heuristics	2009	x	x	✓	2	(Redl, 2009)
		2010	x	x	✓	1	(Kahar and Kendall, 2010)
		2014	x	x	✓	1	(Abdul-Rahman et al., 2014)
	Adaptive hybrid heuristics	2009	✓	x	x	1	(Qu et al., 2009a)
	Fuzzy multiple heuristic ordering	2009	✓	x	x	1	(Asmuni et al., 2009)
	Roulette wheel graph colouring	2009	✓	x	x	1	(Sabar et al., 2009b)
	Construction with grid resources	2009	x	✓	x	1	(Gogos et al., 2009)
	Clustering and graph colouring heuristics	2012	x	x	✓	2	(Othman and Mashhod, 2012)
	Rough set approach	2011	✓	x	x	1	(Thomas et al., 2011)
	Adaptive selection of heuristics	2014	✓	✓	x	2	(Burke et al., 2014)
	Adaptive decomposition and ordering strategy based on GC heuristics	2014	✓	x	x	1	(Abdul-Rahman et al., 2014b)
Meta-heuristics: 1) Local search based algorithms	Integrated problem solving steering framework	2012	✓	x	x	1	(Thomas et al., 2012)
	Rough sets	2011	✓	x	x	1	(Khader et al., 2011)
	HC combined with neighbourhood structure	2011	✓	x	x	1	(Ahandani and Vakil-Baghmisheh, 2011)
	Hybrid multi neighbourhood Tabu search	2011	✓	x	x	1	(Ab Malik et al., 2011)
	Scatter search	2009	✓	x	x	1	(Sabar and Ayob, 2009)
		2010	x	✓	x	1	(Gogos et al., 2010)
	Tabu Search (TS)	2009	✓	x	x	1	(Sabar et al., 2009a)
		2012	✓	x	x	1	(Pais and Amaral, 2012)
	Great Deluge algorithm	2009	✓	x	x	1	(Abdullah et al., 2009)
		2015	x	x	✓	2	(Mohmad Kahar and Kendall, 2015)
	Integrated hybrid approach	2011	✓	✓	x	2	(Turabieh and Abdullah, 2011b)
	Iterated two-stage multi-neighbourhood TS	2009	✓	x	x	1	(Ab Malik et al., 2009)
	Variable Neighbourhood Search	2009	✓	x	x	1	(Qu and Burke, 2009)
		2010	✓	x	x	1	(Burke et al., 2010b)
	GRASP with hybrid meta-heuristic local search	2012	x	✓	x	1	(Gogos et al., 2012)
	Intelligent Water Drop algorithm	2014	✓	x	x	1	(Aldeeb et al., 2014)
2) Population based algorithms	Genetic algorithm	2010	x	✓	x	1	(Pillay and Banzhaf, 2010)
		2013	x	x	✓	1	(Innet, 2013)
	Memetic algorithm	2012	x	✓	x	1	(Abdullah and Turabieh, 2012)
		2012	x	✓	x	1	(Özcan et al., 2012)
	Evolutionary ruin and stochastic rebuild	2015	✓	x	x	1	(Li et al., 2015)
	Hybrid fish swarm	2011	x	✓	x	1	(Turabieh and Abdullah, 2011a)
		2015	✓	x	x	1	(Fong et al., 2015)
	Hybrid artificial bee colony	2011	x	✓	x	1	(Alzaqebah and Abdullah, 2011)
		2013	✓	✓	x	2	(Abdullah and Alzaqebah, 2013)
		2015	✓	✓	x	2	(Alzaqebah and Abdullah, 2015)
	An adaptive artificial bee colony and late-acceptance HC algorithm	2014	✓	✓	x	2	(Alzaqebah and Abdullah, 2014)
	Memetic algorithm with harmony search algorithm	2014	✓	x	x	1	(Al-Betar et al., 2014)
	Hybrid intelligent Water Drops algorithm(HIWD)	2015	x	x	✓	1	(Aldeeb et al., 2015)
Multi-Objective		2009	✓	x	x	1	(Cheong et al., 2009)
		2010	✓	x	x	1	(Mumford, 2010)
Hyper-heuristics	Graph-colouring constructive hyper-heuristics	2009	✓	x	x	1	(Qu et al., 2009a)
		2012	✓	x	x	1	(Pillay, 2012)
		2012	✓	✓	x	2	(Sabar et al., 2012)
	Monte-carlo hyper-heuristics	2012	✓	x	x	1	(Burke et al., 2012a)
	Linear combination heuristics hyper-heuristic	2012	✓	x	x	1	(Burke et al., 2012b)
	Adaptive selection of heuristics	2014	✓	✓	x	2	(Burke et al., 2014)
	Hyper-heuristics with late acceptance strategy	2012	✓	✓	✓	3	(Demeester et al., 2012)
	Novel Hyper-heuristic Approaches in Exam Timetabling	2012	✓	✓	x	2	(Soghier, 2012)
	Scatter search hyper-heuristic	2009	✓	x	x	1	(Sabar and Ayob, 2009)

Table 2.16 Survey of approaches for examination timetabling problems reported between 2009-2015.

Approach	Methods	Year	Examination Dataset			No. Examination datasets used	Reference
			Toronto	ITC 2007	Other		
Graph based sequential techniques	Graph colouring heuristic with Simulated annealing	2016	✓	x	x	1	(Cheraitia and Haddadi, 2016)
	Graph colouring based optimised algorithm	2016	x	x	✓	1	(Saharan and Kumar, 2016)
	Nonlinear heuristic modifier of Graph Colouring Heuristics	2017	✓	x	x	1	(Abdul-Rahman et al., 2017a)
	Web-based examination timetabling system	2017	x	x	✓	1	Abdul-Rahman et al. (2017b)
Meta-heuristics Local search-based algorithms	Simulated annealing	2016	✓	x	x	1	(Cheraitia and Haddadi, 2016)
		2017	x	✓	x	1	(Battistutta et al., 2017)
		2019	x	✓	x	1	(Leite et al., 2019b)
		2019	x	x	✓	2	(June et al., 2019b)
	Hill-Climbing	2016	x	✓	x	1	(Bykov and Petrovic, 2016)
		2017	x	✓	x	1	(Burke and Bykov, 2017)
	Great deluge algorithm	2016	✓	✓	x	2	(Burke and Bykov, 2016)
		2020	✓	✓	x	2	(Mandal et al., 2020)
Meta-heuristics population based algorithms	Genetic algorithm	2016	✓	x	x	1	(Ishak et al., 2016)
	Hybrid particle swarm Optimisation	2017	x	x	✓	1	(Marie-Sainte, 2017)
	Ant colony algorithm	2018	x	x	✓	1	(Khair et al., 2018)
		2019	x	x	✓	1	(Khair et al., 2019)
	Memetic algorithm	2016	✓	✓	x	2	(Leite et al., 2018)
		2018	✓	✓	x	2	(Leite et al., 2018)
	Plant Propagation with local search	2017	✓	x	x	1	(Cheraitia et al., 2017)
Multi-objective Technique	Tabu Search multi-criteria approach	2016	✓	x	x	1	(Amaral and Pais, 2016)
	Multi-objective approach based on weighted Tchebyceff scalarisation	2017	✓	✓	x	2	(Muklason et al., 2017)
	Memetic multi-objective optimisation algorithm based on NNIA	2017	✓	x	x	1	(Lei and Shi, 2017)
	Multi-Objective Based E-constraint method	2017	✓	x	x	1	(Btissam and Abounacer, 2017)
	Fuzzy Grouping Genetic Algorithm for Multi-Criterion examination timetabling	2017	✓	x	x	1	(Mutingi and Mbohwa, 2017)
	Memetic algorithm based on MOEA/D for the examination timetabling problem	2018	✓	x	x	1	(Mutingi and Mbohwa, 2017)
Hyper-heuristics	Hyper-heuristics and fairness in examination timetabling problems	2017	✓	✓	✓	3	(Muklason et al., 2017)
	Great Deluge Based Hyper-heuristics	2019	x	x	✓	1	(Muklason et al., 2017)

Table 2.17 Survey of approaches for examination timetabling problems reported in state-of-the-art.

2.6 Significant Challenges in the Examination Timetabling Literature

The literature study revealed that the most critical challenge encountered in this field is that the problem was often derived from an individual definition, and the quality of solutions obtained was measured differently. Consequently, many formulations of the examination timetabling problem were introduced, attempting to capture most aspects of the problem that most universities have to encounter every academic term. Hence, researchers have proposed several academic benchmarks to provide the examination timetabling community with a standard test base on which approaches can be tested.

From our review also, we have found that many approaches proposed in the literature involve multiple phases, which usually consist of an initialisation phase followed by optimisation phases. Graph colouring heuristics are among the most widely used methods in the initialisation phase. This is due to their significant strength, where they can construct good examination timetables within a short computational time and are also very easy to implement. Notwithstanding the advantages and capabilities of the graph colouring heuristics, many concerns found in the literature related to them:

- They can lead to early assignments, in which no feasible periods will be available for exams later in the construction process.
- High-quality initial solutions are missed during the construction phase as attempting to satisfy soft constraints are neglected.
- Current research trends in this field include using an adaptive method or hybridisation graph heuristics with other methods for ensuring the feasibility of the initial solution.
- They cannot appropriately address the complex examination timetabling problems and sometimes fail to produce feasible solutions.

These issues have potential ramifications for the optimisation process as a whole. Good initial solutions can facilitate the improvement technique to locate the optima (i.e. global optima or good local optima) (Rahnamayan et al., 2007; Clerc, 2008). On the other hand, starting from bad areas may prevent the approach proposed from obtaining the optima (Maaranen et al., 2004). Therefore, the initialisation phase should, we argue, be better considered a significant driver for obtaining better final solutions increases. I.e. starting the improvement phase with a good initial solution is necessarily result in a good final solution. Furthermore, a common disadvantage of all existing approaches is that they use an iterative procedure of improving an initial solution, where the search for the result takes place around this solution, and this means that the result directly depends on the initial solution, and there is a problem of its choice.

In our review of the literature, we have devoted the majority of our discussion towards the application of meta-heuristics to these problems. There are still other problems related to the improvement phase, which are particularly interesting to the research carried out in this thesis. To date, there is no optimal solution to this problem as numerous parameters and constraints have to be taken into account. Furthermore, effort is required to adjust a number of parameters and enhance operators for particular problems to achieve high-quality solutions to those problems.

Another challenge encountered is choosing an appropriate algorithm to improve initial solutions in the optimisation phase. This is because numerous algorithms that have been used to solve the examination timetabling problems. After reviewing the advantages and disadvantages of each algorithm applied to solve this problem, the genetic algorithm is chosen in this work due to its advantages compared to other algorithms. However, some issues related to the crossover operator have been revealed in the literature. Some GA operators also need to be enhanced.

As we have also seen, it is also quite common, particularly in examination timetabling applications, for authors to state that their approaches were able to obtain better results than previously produced results by published works in the literature. There are a number of potential pitfalls in comparison between these approaches. Some necessary information is not provided to conduct a fair comparison. Thus, there is a real need to develop a standardised analysis approach in order to analyse and fairly compare various examination timetabling algorithms.

Therefore, all of the above challenges discussed offer motivations for further research. In addition, the literature review and background study have provided us with some ideas for our further investigations.

2.7 Summary

A lot of work seen in the literature has focused on developing and tuning optimisation heuristics for examination timetabling problems, and for 'healing' mechanisms to fix putative solutions which break various hard constraints. In addition, much work has also been directed toward developing effective search heuristics in this domain in order to obtain high-quality exam timetables. In this chapter, constructive heuristic approaches such as graph colouring heuristics, fuzzy-based techniques, and neural networks have been introduced. Exact approaches, including integer programming and constraint programming, have also been successfully implemented to solve this problem.

This chapter has also presented various approximation algorithms, including heuristics and meta-heuristics, have been employed when the nature of the combinatorial optimisation problem makes an optimal solution intractable. These approaches have demonstrated great success in the area of examination timetabling problem. It can be observed that the most successful and effective approaches in the literature are typically the combination of graph colouring based ordering methods and meta-heuristics. Graph colouring-based ordering methods are often applied in the first phase to construct initial exam timetables. The quality of the obtained timetables are then improved by a meta-heuristic approach. Other recent approaches (i.e. hyper-heuristic, multi-objective and multi-criteria) were also presented.

The approaches discussed in this chapter offer many ideas for the following chapters that aim to propose methods to construct and tackle the examination timetabling problem. Furthermore, the Toronto, ITC 2007 examination timetabling track and Yeditepe benchmark sets are used to evaluate the performance of approaches proposed in this thesis. The next chapter discusses the implementation of a new initialisation strategy for constructing high-quality examination timetables.

Chapter 3

A New Initialisation Method for Examination Timetabling Heuristics

The work presented in this chapter was published in 2019 IEEE Symposium Series on Computational Intelligence (SSCI) (Alsuwaylimi and Fieldsend, 2019).

3.1 Introduction

Before tackling the examination timetabling pipeline, this chapter presents a new initialisation strategy for constructing initial feasible solutions for examination timetabling. The aim is to construct high-quality solutions for the benchmark examination timetabling problems widely studied in the literature with their standard single objective function.

As mentioned earlier, exam timetabling is classified in the class of NP-complete optimisation problem. The problem requires the scheduling of the exams corresponding to a set of courses, while satisfying a range of constraints determined by exam timetabling staff. There are commonly two types of constraints: hard constraints which must be satisfied for a solution to be viable (e.g. a student should not sit two exams at the same time), and soft constraints which we would prefer to satisfy, but are not absolutely necessary to meet (e.g. a student should not sit two exams in quick succession), see for example (Sigl et al., 2003). Concisely: satisfying all the hard constraints produces a feasible timetable; whereas soft constraints can be violated.

Many papers have proposed methods to construct examination timetables as discussed in Chapter 2. Often graph-colouring forms the basis, e.g. Carter and Laporte (1996) and Burke et al. (2007). Computational intelligence approaches such as Fuzzy Logic (Asmuni et al., 2005b) and Neural Networks (Corr et al., 2006a) have been employed, however the bulk of work in this area has employed methods from the broad area of evolutionary computation. For instance, Di Gaspero and Schaerf (2000), Kendall and Hussin (2005b), and White and Xie (2000) applied tabu search. Simulated Annealing was used in Burke and Newall (2003), Thompson and Dowsland (1998), and Memetic Algorithms in Burke et al. (1995c), Burke and Silva (2005). Genetic Algorithms were used in e.g. (Burke et al., 1995b), Ant Colony optimisation in e.g. Dowsland and Thompson (2005) and Eley (2007), Particle Swarm Optimisation in e.g. Chu et al. (2006). The Great Deluge Algorithm in Burke et al. (2004a), and hybridisations of distinct heuristic methods in e.g. (Caramia et al., 2001) and Merlot et al. (2003). More recently Hyper-heuristic approaches have gained popularity (Muklason, 2017; Pillay and Özcan, 2019). The efficacy of an optimisation heuristic is often dependent on the initial population from which it starts. High quality and diverse initial solutions are generally thought to improve the subsequent performance of a timetabling algorithm (Burke et al., 1998a).

The initial population can be generated by a number of methods. Most commonly these attempt to generate 'high quality' solutions (i.e. ones that have low soft constraint violations, and preferably no hard constraint violations) so that the search is initialised in 'good' areas of design space. This minimises the subsequent run time required of the algorithm to find an acceptable solution. It is this area of algorithm design we are particularly concerned with in this chapter. Therefore, we propose a new initialisation approach, whose main novelty stems from the way it manages three interacting lists of examinations to allocate. Thus, these lists are arranged and processed in a step-wise fashion in order to provide a good satisfaction of hard and soft constraints.

The rest of the chapter is as follows. Section 3.2 describes popular initialisation approaches from the literature. Our proposed approach is outlined in Section 3.3. Experimental results and comparison are discussed in Section 3.4. Section 3.5 presents statistical analysis and discusses the results. Finally, a summary of the work conducted in this chapter is presented in Section 3.6.

3.2 Initialisation for Examination Timetabling Problems

Many researchers have studied initialisation methods for evolutionary algorithms (EAs) to help solve timetabling problems. Corne and Ross (1995a) proposed an initialisation method where a random operator was used to ensure diversity to support either a higher quality solution or a more random solution (hard constraints could be violated). They concluded that the strictness of the achievement of the hard constraints could result in an inferior ultimate outcome in regard to the soft constraints. However, it is hard to apply this method to problems that enforce a high plenty weight to the violation of the hard constraints. Corne and Ross (1995b) used this approach to initialise EAs, and presented a comparison between genetic algorithms (GAs), simulated annealing (SA), and multi-start stochastic hill-climbing (MSSH) algorithms in regard to the performance when they started from both seeded solutions and random solutions. Their results show that the GA's performance was the worst among the studied algorithms when it started from random solutions. However, when the algorithms started from seeded solutions, the GA's performance increased to the extent that it performed the best, whilst the SA and MSSH were much less affected. Burke et al. (1995b) proposed a hybrid approach between heuristic measures and a roulette wheel-style method to enhance the solutions' quality while keeping some diversity between them. Work presented in Burke and Newall (1997) studied the effect of random sequential initialisation followed by hill-climbing — each solution resulted in a better group of solutions than the former approach.

As discussed in Section 2.6.1 of Chapter 2, graph colouring heuristic algorithms are widely used to construct initial feasible exam timetables, see for example Carter and Laporte (1996), de Werra (1985), Qu et al. (2009b), Burke and Petrovic (2002), and Ayob et al. (2007b). To exploit the link between the two problems, some authors employed two-phase algorithms. In this algorithm, the GC heuristics can be used in the first phase so that an initial feasible solution is obtained.

The GC problem involves assigning colours to an element graph's type following certain constraints. The simplest sub-type is vertex colouring, which mainly aims to give several vertices, as well as edges unique colours so that no adjacent vertices are coloured the same. The goal in this case involves finding a solution with the lowest number of colours possible.

Considering the mapping between the GC problem and the examination timetable problem (see Chapter 2), GC heuristics like the Saturation Degree Ordering are very common for obtaining initial solutions. A number of graph colouring heuristics have been introduced in the literature and have been applied to the examination timetable problem (Carter, 1986). Among the different GC heuristics, Brélaz (1979) has investigated the application of the saturation degree heuristic (Cheong et al., 2009) to the examination timetable problem. The findings concluded that saturation degree is capable of obtaining lower-conflict timetables for all the datasets in comparison to the other graph colouring heuristics (largest degree, colour degree, extended saturation degree, and random). These heuristics will be discussed in more detail later in this chapter.

In the initialisation process, the exams are represented as vertices, while the edges are the conflicts between them. Immediately neighbouring vertices are assigned distinct colours. The colour therefore denotes the time period(s) in the timetable. These heuristic methods prioritise scheduling according to the level of the difficulty in scheduling (number or weight of edges) in order to schedule the most difficult exams first. It is widely recognised in the literature that one of the challenges in the process of assigning examinations is deciding which examination should be scheduled to which resource (i.e. room and period) first, and different ordering strategies have been utilised in this domain (for example, Broder (1964); Cole (1964); Peck and Williams (1966); Welsh and Powell (1967); Laporte and Desroches (1984); Burke et al. (1994b); Carter et al. (1994); Joslin and Clements (1999); Burke et al. (2004a); Rahman et al. (2009), and Kahar and Kendall (2010)).

In this work, four graph colouring heuristics have been applied on the basis of them being the most commonly-used such heuristics that have demonstrated the ability to obtain high-quality examination timetables (Asmuni et al., 2005b; Burke et al., 2007; Kendall and Hussin, 2005a). These are: Largest Degree (LD), Largest Weighted Degree (LWD), Largest Enrolment (LE), and saturation Degree (SD). These heuristics can be divided into two categories: dynamic and static heuristics (Qu and Burke, 2009). The LD, LE and LWD are considered static methods in which the value for each examination does not change during the iteration. SD, meanwhile, can be categorised as a dynamic method since it conducts dynamic changes when any consecutive exam has been assigned. The sub-sections below describe each of these methods.

3.2.1 Largest Degree

The LD method was first utilised for tackling examination timetabling problems by (Broder, 1964). In this method, the degree represents the number of exams in conflict (i.e. exams are ordered in decreasing number of conflicts). This method gives the priority in scheduling to the exams that have the most conflicts with other exams.

Cole (1964) has employed this method to construct examination timetables. In that work, an examination conflict matrix was generated, which is a square matrix of dimension equal to the number of examinations. This matrix is used to determine any pair of examinations that are conflicting, as well as the number of students enrolled in these examinations.

In Peck and Williams (1966), the largest degree method of graph colouring was again presented as a means of solving the examination timetabling problem. Here, however, the ordering was modified by rearranging the allocation of exams to non-conflicting periods.

Welsh and Powell (1967) investigated the graph colouring method to identify the fewest number of colours (chromatic number) applied to the vertices of a graph. The aim was to construct a conflict-free graph in which two adjacent vertices did not have matching colour. The LD heuristic was capable of finding the colour of the vertices.

3.2.2 Largest Weighted Degree

LWD is the same as the LD, but it applies a weight to each conflict by counting the number of students participating in that conflict. As such, the priority in scheduling is given to core exams (Carter et al., 1996; ARANI and Lotfi, 1989) (i.e. those with largest conflicting cohorts).

3.2.3 Largest Enrollment First

Wood (1968) developed a university timetabling system for the University of Manchester. The LE was presented to determine exams that should be scheduled into timetables. This method depends on arranging exams so that those with the highest number of enrolled students are scheduled first.

3.2.4 Saturation Degree

As previously mentioned, SD is considered to be a dynamic ordering heuristic since it has the ability to colour vertices dynamically. This heuristic was first introduced by Brélaz (1979). The SD method was successfully applied to solve the examination timetabling problem (Rahman et al., 2009; Burke and Newall, 2004; Carter et al., 1996). In this method, the priority in assignment is given to those exams that have the fewest number of free periods for scheduling without violating of any hard constraints. In order to settle ties between exams, the LD approach is used.

3.2.5 Random Schedule Allocation

In addition to the graph-colouring approaches listed above, random schedule allocation (RD) is adapted widely to generate initial solutions in exam timetabling (Jha, 2014). RD orders the examinations list randomly and schedules exams into the first valid period satisfying the hard constraints. Exams are taken sequentially starting from the top of a randomly ordered list. If the scheduling process fails to obtain non-conflicting examination timetable (i.e. satisfy all hard constraints), the process is repeated with a new random ordering.

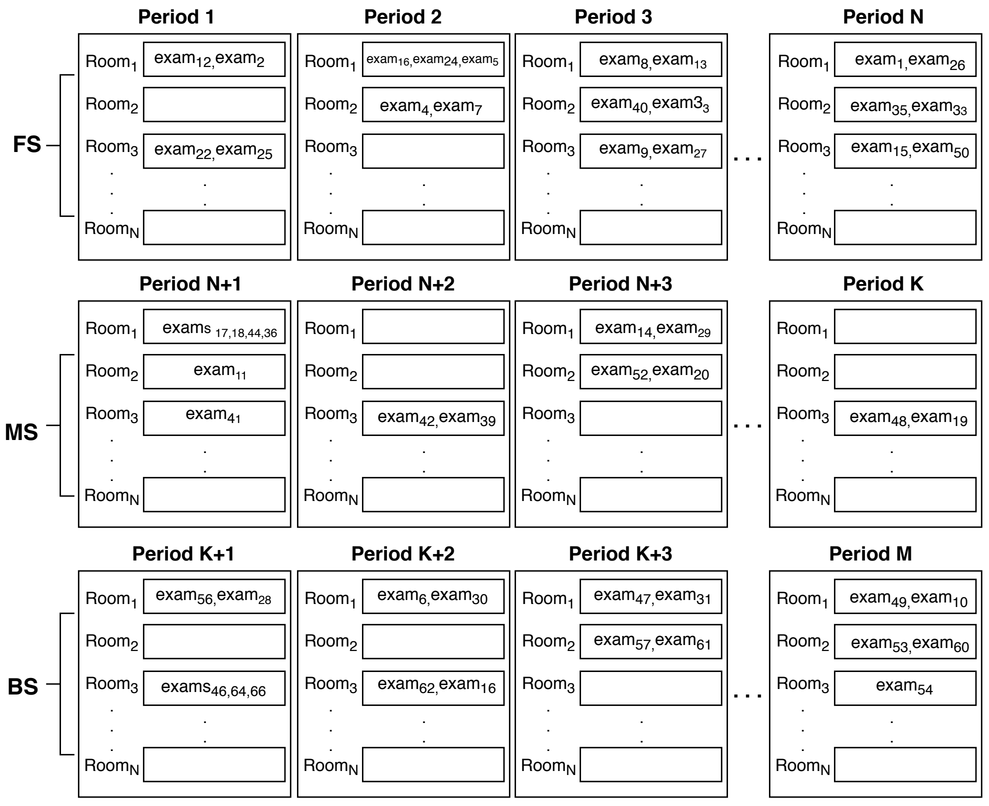

Figure 3.1 The Front Section (FS), the Middle Section (MS), and the Back Section (BS).

3.3 Proposed Initialisation Approach

We now propose a new algorithm to construct initial solutions for examination timetabling problems, that can be used as seed solutions for meta-heuristic algorithms. The proposed algorithm attempts to schedule the largest number of conflicting exams as possible in the first periods of a timetable, which we name the **Front Section (FS)**, and in the last periods of the timetable named **Back Section (BS)** while also attempting to satisfy all hard and soft constraints. The number of periods in the FS and BS are equal to a given number of spread periods determined by the **Period Spread (PS)** constraint, which aims to spread each student's exams over a given number of periods. The overarching idea behind the proposed algorithm is to satisfy the PS constraint for each student of any exam scheduled into periods of the **Middle Section (MS)** (i.e. the periods after FS periods and before BS periods). Figure 3.1 shows the FS, MS, and BS sections, where the FS starts from the first period of the exam timetable until period N. whereas the MS starts directly after period N and the last period of the MS is period K. The last section (i.e. BS) starts immediately after period K until the last period of the timetable. Furthermore, the solution is well constructed by creating a list of period objects composited of room objects that can contain exam objects. Each object type has its own properties used to check hard and soft constraints before scheduling exams in any room for a particular period.

In order to avoid violation of the PS constraint for an exam timetabled in any period, T, of MS, with period spread range X, then the X periods before and after period T must not contain any conflicting exam (i.e. an exam with students in common). On the other hand, if an exam is timetabled in the first period of the FS, then the number of periods N that must not contain any conflicting exam is just the X periods which fall after this period, so $N = X$. Also the same period range $N = X$ is used if an exam is timetabled in the last period of the BS, where just the N periods that fall before the last period must be free of conflicting exams. In case an exam is timetabled in the second period of FS, then the number of periods N that must have no conflicting exams in order to satisfy the PS constraint is $N = 1 + X$ (one period falling before and X periods falling after that period T). Also, scheduling an exam in the period before the last period of BS requires the same period range, $N = X + 1$ (i.e. X periods falling before and one period falling after period T).

Timetabling an exam in the third period of FS or in the third period before the last period of BS requires $N = X + 2$ and so on until the period number becomes equal to PS (i.e. the last period in the FS or the first period in the BS). Then $N = (X - 1) + X$, which is still lower than timetabling an exam in any period of MS, which will require $2X$ periods $N = X + X$ having no conflict exam(s) to satisfy the PS constraint. More clarification on the method described above is shown in Figure 3.2.

According to the proposed method, for each period the available rooms of a timetable are ordered increasingly by their capacities, also the exams are ordered in three lists which are: **Front List (FL)**, **Back List (BL)**, and **Middle List (ML)**. Each list initially contains all exams to be scheduled.

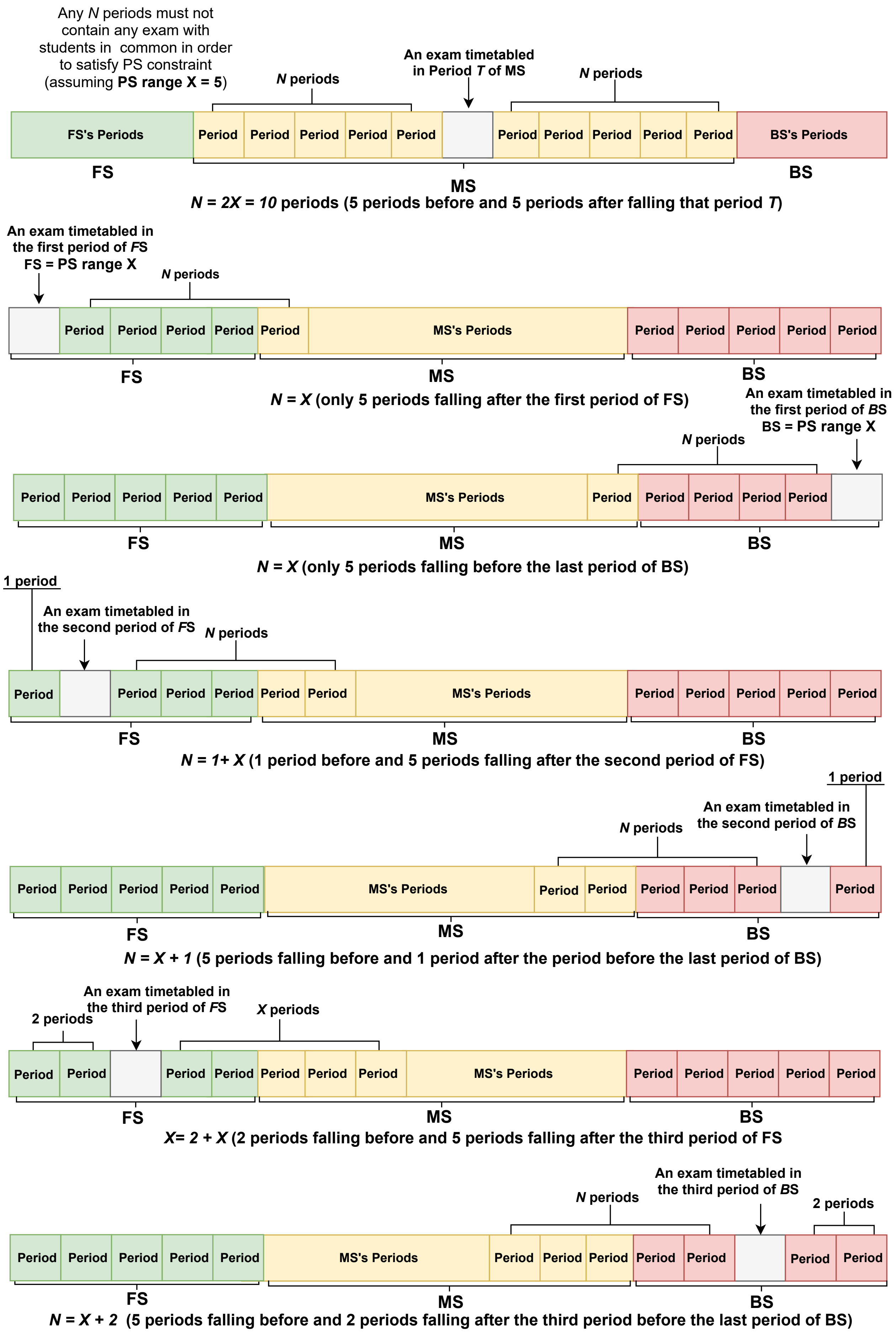
Any N periods must not contain any exam with students in common in order to satisfy PS constraint (assuming PS range X = 5)
N periods
An exam timetabled in Period T of MS
N periods
FS's Periods
Period
BS's Periods
FS
MS
BS
N = 2X = 10 periods (5 periods before and 5 periods after falling that period T)
An exam timetabled in the first period of FS
FS = PS range X
N periods
MS's Periods
N = X (only 5 periods falling after the first period of FS)
An exam timetabled in the first period of BS
BS = PS range X
N = X (only 5 periods falling before the last period of BS)
1 period
An exam timetabled in the second period of FS
N = 1+ X (1 period before and 5 periods falling after the second period of FS)
An exam timetabled in the second period of BS
N = X + 1 (5 periods falling before and 1 period after the period before the last period of BS)
An exam timetabled in the third period of FS
2 periods
X periods
X= 2 + X (2 periods falling before and 5 periods falling after the third period of FS
An exam timetabled in the third period of BS
N = X + 2 (5 periods falling before and 2 periods falling after the third period before the last period of BS)

Figure 3.2 An example of satisfying Period Spread (PS) constraint using the proposed initialisation approach.

3.3.1 Front List

In the Front List (FL), as detailed in Algorithm 3.1, the exams are sorted by decreasing number of conflicts with other exams (line 4), subject to 'after' constraints being observed. This list is then processed from the front.

Once any exam in this list is scheduled in any period in the FS, it is removed from the other lists (ML and BL — lines 19–20). On the other hand, if any exam cannot be scheduled in any period in the FS, then it is removed only from the FL and remains in the other lists to be scheduled via one of them later (line 29). Note that, the variables and methods used in Algorithm 3.1 are briefly explained in Tables 3.1 and 3.2.

3.3.2 Back List

In the Back List (BL), the exams are sorted by decreasing number of conflicts with other exams, subject to 'before' constraints being observed (line 5 of Algorithm 3.1). This constraint is derived from the 'after' constraint that stipulates which exams must occur after others.

The scheduling process starts by attempting to schedule BL exams as much as possible in the BS, starting from the last period of BS until to the first period of BS while satisfying all hard and soft constraints. Any exam scheduled in these last periods will be removed from the BL and the ML (Algorithm 3.2 lines 12–13), otherwise, in case an exam cannot be scheduled at this stage, it is removed only from BL (line 21). The variables and methods used in Algorithm 3.2 are explained in Tables 3.1 and 3.2.

3.3.3 Middle List

The Middle List (ML) contains all remaining exams that have not been scheduled by Algorithms 3.1 and 3.2. The ML exam list is ordered depending on the number of the remaining periods available, where the exams having the fewest number of available periods in the timetable have the highest priority to be scheduled first in any period while satisfying only the hard constraints.

The algorithm selects the periods randomly (Algorithm 3.3, line 2), giving the priority to the previous selected periods in order to maximise the periods' utilisation and keep the diversity. The process stops when all the exams in the middle list are completely scheduled to a feasible period, and a feasible solution is returned. However, if there is any exam in this list which has no available period, i.e. it could not be assigned to any feasible period, an infeasible solution is returned instead (Algorithm 3.3, line 34). Tables 3.1 and 3.2 give an explanation of variables and methods used in Algorithm 3.3.

Algorithm 3.1 Pseudo-code of Front List (FL) heuristic.

```
Require: $PS$                                         ▷ The period spread number
Require: $E$                                          ▷ Exams to be scheduled
Require: $R$                                          ▷ Set of room lists, available each period
    • Initialisation step
 1: $\alpha \leftarrow \emptyset$                     ▷ Schedule of exams to rooms at periods
 2: $FS \leftarrow$ init_FS($PS$)                     ▷ Front section length determined by $PS$
 3: $BS \leftarrow$ init_BS($PS$)                     ▷ Back section length determined by $PS$
 4: $FL \leftarrow$ sort_by_conflict($E$)             ▷ Front list contains exams ordered from front by
    largest to smallest conflict, subject to meeting ‘after’ constraints.
 5: $BL \leftarrow$ sort_by_conflict($E$)             ▷ Back list contains exams ordered, from back, by
    largest to smallest conflict, subject to meeting ‘before’ constraints.
 6: $ML \leftarrow$ random_sorting($E$)               ▷ Middle list exams ordered randomly.
 7: for all periods $p$ do                            ▷ For each period
 8:     $R_p \leftarrow$ ascend_sort($R_p$)           ▷ Order the available rooms by their capacities.
 9: end for
    • Schedule $FL$ exams into Front Section periods ($FS$)
10: while $FL$ is not empty do
11:     isScheduled $\leftarrow$ false
12:     $p \leftarrow 1$    ▷ Attempt to schedule from the first period ($p$) of $FS$ until reaching the
    last period of $FS$
13:     while $p \leq |FS|$ do
14:         $R_p^i \leftarrow$ first_suitable_room($R_p, \alpha, FL_1$)     ▷ get first available room
15:         if $R_p^i \neq \emptyset$ then                                  ▷ If it can be scheduled
16:             if satisfies($FL_1$) = true then    ▷ Exam in first element of $FL$ satisfies
    all hard and soft constraints
17:                 $\alpha \leftarrow$ schedule($\alpha, FL_1, R_p^i$)    ▷ Schedule exam in $FL_1$ at time period $p$
    into room $R^i$
18:                 $FL \leftarrow FL \setminus \{FL_1\}$            ▷ Delete exam $FL_1$ from $FL$
19:                 $BL \leftarrow BL \setminus \{FL_1\}$            ▷ Delete exam $FL_1$ from $BL$
20:                 $ML \leftarrow ML \setminus \{FL_1\}$            ▷ Delete exam $FL_1$ from $ML$
21:                 isScheduled $\leftarrow$ true
22:                 $p \leftarrow 1$
23:                 Break                                           ▷ The current exam is scheduled
24:             end if
25:         end if
26:         $p \leftarrow p + 1$
27:     end while
28:     if isScheduled $\leftarrow$ false then
29:         $FL \leftarrow FL \setminus \{FL_1\}$                    ▷ Del. exam $FL_1$ from $FL$
30:     end if
31: end while
32: return $\alpha$
```

Algorithm 3.2 Pseudo-code of Back List (BL) heuristic.

```
Require: $BL$                                          ▷ Back list
Require: $ML$                                          ▷ Middle list
Require: $\alpha$                                      ▷ Current schedule state
Require: $R$                                           ▷ Set of room lists, available each period
 1: for all periods $p$ do                             ▷ For each period
 2:     $R_p \leftarrow$ ascend_sort($R_p$)            ▷ Order the available rooms by their capacities.
 3: end for
    • Schedule $BL$ exams into Back Section periods ($BS$)
 4: while $BL$ is not empty do
 5:     isScheduled $\leftarrow$ false
 6:     $p \leftarrow |BS|$ ▷ Attempt to schedule from the last period ($p$) of $BS$ until reaching the first period of $BS$
 7:     while $p \geq 1$ do
 8:         $R_p^i \leftarrow$ first_suitable_room($R_p, \alpha, BL_1$)    ▷ get first available room
 9:         if $R_p^i \neq \emptyset$ then             ▷ If it can be scheduled, given the current state of $S$
10:             if satisfies($BL_1$) = true then       ▷ Exam in first element of $BL$ satisfies all hard and soft constraints
11:                 $\alpha \leftarrow$ schedule($\alpha, BL_1, R_p^i$)    ▷ Schedule exam in $BL_1$ at time period $p$ into room $R^i$
12:                 $BL \leftarrow BL \setminus \{BL_1\}$             ▷ Del. exam $BL_1$ from $BL$
13:                 $ML \leftarrow BL \setminus \{BL_1\}$             ▷ Del. exam $BL_1$ from $ML$
14:                 isScheduled $\leftarrow$ true
15:                 Break                              ▷ Exam is now scheduled
16:             end if
17:         end if
18:         $p \leftarrow p - 1$                       ▷ inverse selecting
19:     end while
20:     if isScheduled $\leftarrow$ false then
21:         $BL \leftarrow BL \setminus \{BL_1\}$                     ▷ Del. exam $BL_1$ from $BL$
22:     end if
23: end while
24: return $\{\alpha, ML\}$
```

Algorithm 3.3 Pseudo-code of Middle List (ML) heuristic.

```
Require: $ML$                                  ▷ Middle list
Require: $\alpha$                              ▷ Current schedule state
Require: $R$                                   ▷ Set of room lists, available each period
Require: $PL$ list of all periods in the timetable.
    • Schedule remaining exams
 1: $SPL \leftarrow \emptyset$                  ▷ Initial empty selected periods list
 2: $rp \leftarrow$ random_period($PL$)         ▷ Get a random period
 3: $SPL \leftarrow SPL \cup \{rp\}$            ▷ Add $rp$ to $SPL$
 4: for all periods $p$ do                      ▷ For each period
 5:     $R_p \leftarrow$ ascend_sort($R_p$)     ▷ Order rooms by size
 6: end for
 7: $PL \leftarrow PL \setminus \{rp\}$         ▷ Remove $rp$ from $PL$
 8: $sp \leftarrow 1$                           ▷ Index into SPL
 9: while $ML \neq \emptyset$ do
10:     isScheduled $\leftarrow$ false
11:     while $sp \leq |SPL|$ do
12:         if $ML_1$ can be scheduled into $SPL_{sp}$ satisfying all the hard constraints into
    the $i$th smallest room then $\alpha \leftarrow$ schedule($\alpha, ML_1, R^i_{SPL_{sp}}$)
13:             $ML \leftarrow ML \setminus ML_1$       ▷ Remove exam from list
14:             isScheduled $\leftarrow$ true
15:             goto line 19                    ▷ Exit loop as exam scheduled
16:         end if
17:         $sp \leftarrow sp + 1$              ▷ select the next period of $SPL$
18:     end while                               ▷ End scheduling $ML_1$ in the $SPL$
19:     if isScheduled = false then
20:         $temp \leftarrow PL$                ▷ Temporary list
21:         while $temp \neq \emptyset$ do
22:             $sp \leftarrow$ random_element($temp$)
23:             if $ML_1$ can be scheduled into $sp$ satisfying all the hard constraints into the
    $i$th smallest room then $\alpha \leftarrow$ schedule($\alpha, ML_1, R^i_{temp_{sp}}$)
24:                 $ML \leftarrow ML \setminus ML_1$   ▷ Remove from list
25:                 isScheduled $\leftarrow$ true
26:                 $SPL \leftarrow SPL \cup \{temp_{sp}\}$
27:                 $PL \leftarrow PL \setminus \{temp_{sp}\}$
28:                 goto line 9                 ▷ Exit from loop
29:             else
30:                 $temp \leftarrow temp \setminus \{temp_{sp}\}$
31:             end if
32:         end while
33:     end if
34:     return infeasible solution
35: end while
36: return $\alpha$
```

Variable	Description
PS	The period Spread number.
E	Exam to be scheduled.
R	Set of room lists, available each period.
α	Schedule of exams to rooms at periods
FS	Front section.
BS	Back section.
FL	A list contains exams ordered from front by largest to smallest conflict, subject to meeting 'after' constraints.
BL	A list contains exams ordered, from back, by largest to smallest conflict, subject to meeting 'before' constraints.
ML	Middle list exams ordered randomly.
p	Period.
R_p	Available room in period.
PL	List of all periods in the timetable.
SPL	Initial empty selected periods list.
rp	Get a random period.
$temp$	Temporary list.

Table 3.1 List of variables used in Algorithm 3.1, Algorithm 3.2, and Algorithm 3.3.

Method	Description
`init_FS`(PS)	Identify Front section length by PS.
`init_BS`(PS)	Identify Back section length by PS.
`sort_by_conflict`(E)	Ordering exams from largest to smallest conflict.
`random_sorting`(E)	Ordering exams randomly.
`ascend_sort`(R_p)	Order the available rooms in period by their capacities.
`first_suitable_room`	Get first available room.
`satisfies`(FL_1)	Exam in first element of FL satisfies all hard and soft constraints.
`schedule`(α, FL_1, R_p^i)	Schedule exam in FL_1 at time period p into room R^i.
`satisfies`(BL_1)	Exam in first element of BL satisfies all hard and soft constraints.
`schedule`(α, BL_1, R_p^i)	Schedule exam in BL_1 at time period p into room R^i.

Table 3.2 List of methods used in Algorithm 3.1, Algorithm 3.2, and Algorithm 3.3.

3.4 Experimental Results and Comparison

The experiments were conducted over two examination timetabling problems, namely, ITC 2007, and Yeditepe. The characteristic of each these datasets has been discussed in Section 2.3 of Chapter 2. Our experiment has two phases. In the first phase, a set of initial solutions is constructed by the proposed algorithm (devoted OBSI - Ordering-Based Scheduling Initialisation) and the comparison methods are (LD - Largest Degree, LE - Largest Enrollment, LWD - Largest Weighted Degree, SD - Saturation Degree, and RD - Random Allocation). In the second phase, a basic evolutionary algorithm (EA) uses the initial population provided by each of the considered methods to start its (total) time-limited search. These are implemented in Java, with experiments run on an Intel Core (TN) i5-6200U (CPU @ 2.30 GHz with 16 GB RAM) PC running Windows 7 Enterprise with 64-bit operating system. Source code is available at `https://github.com/alsuwaylimi/Initialisation-Method-Project`.

3.4.1 Phase 1: Initialisation

The median results (out of 30 runs for each instance) on both datasets (ITC 2007 examination track and Yeditepe) are shown in Tables 3.3 and 3.4, respectively, including statistical significance. We make appropriate p-value adjustment using the Holm-Bonferronio method (Ludbrook, 1998), to adjust for increased family-wise error when multiple hypothesis are tested on the same data.

The ITC 2007 results indicate that using our proposed OBSI strategy generally outperforms graph heuristics and random initialisation for initial solution quality. Additionally, the graph heuristics and random initialisation fail to obtain feasible solutions for some of the more difficult instances. Furthermore, OBSI solutions tend to be more diverse in performance, with total soft constraint costs often spread in a wider range than the comparison methods (as shown in Figures 3.3 and 3.4). Generating an initial population with graph heuristics, on the other hand, tends to occur in a shorter time-frame compared with the OBSI, albeit delivering lower quality solutions.

The results on the Yeditepe problems are provided in Table 3.4. This shows that the OBSI strategy is able to generate feasible solutions for all instances in contrast to graph heuristics and RD, which again cannot generate a feasible solution for some of the more difficult problem instances. Its performance is less striking than on the ITC 2007 datasets, outperforming some graph heuristics in term of quality on some problems, but on others it performs less well (see Figures 3.5 and 3.6).

Figures 3.7, 3.8, 3.9, 3.10, and 3.11 show the distribution of execution time in milliseconds for the different initialisation approaches on both datasets. Generally speaking the OBSI tends to be relatively faster on the Yeditepe problems (due to the fewer constraints), but often slower on the ITC 2007 problems. Apart from the SD method, the other graph heuristics fail for some instances of ITC 2007 and Yeditepe. Some particular problems instances, such as Exam4 and Exam11 of ITC 2007 and YUE20012, YUE20021, and YUE20032 of Yeditepe, have a large number of exams with high conflict density and few available periods. This can lead to saturation cases where there are no valid periods to schedule the next exam. The main reason for saturation cases occurring is where subroutines schedule exams into periods while giving scheduling priority to exams having largest degree (LD), largest weighted degree (LWD), or largest enrollment degree (LE). SD prevents saturation cases arising by dynamically ordering exams depending on a number of available periods and giving scheduling priority to exams which have the fewest number of periods available. In addition, from Figures 3.12, 3.13, 3.14, and 3.15, we can identify a positive relationship between execution time and cost value for the OBSI initialisation.

	OBSI		LD		LWD		LE		SD		RD	
Problem Instance	Median cost	Median t (ms)	Median cost	Median t (ms)	Median cost	Median t (ms)	Median cost	Median t (ms)	Median cost	Median t (ms)	Median cost	Median t (ms)
Exam1	**27987.5**	6130	31805.5	3408	30666.5	3463	30224.5	**2269.5**	31872.5	7714	39630	130463
Exam2	**26661**	10022.5	38974	**6973**	40838.5	7121	40791	7020	40848	11418.5	150514.5	7817
Exam3	**72713**	23095	102158	12620	101874.5	18704	104357.5	19344	100657	32736.5	249617.5	**9393.5**
Exam4	**50139**	**1263**	-	-	-	-	-	-	51346.5	2542.5	-	-
Exam5	**74394.5**	11829.5	133080.5	7163	130333	7106	133910	**7103.5**	135884.5	18150.5	319573.5	45975.5
Exam6	**50190**	**624**	53115	686	52190	32845.5	51285	16411	51000	904	63957.5	41347.5
Exam7	**49253**	25232.5	80718.5	11011	77993	10834.5	77386	**10829**	83277	31225.5	64212.5	31107.5
Exam8	**114559**	7230	136475.5	3104	145734.5	**3096**	138280.5	3198	134043.5	8197.5	-	-
Exam9	**7705.5**	148	8454.5	**62**	8550	**62**	8727.5	**62**	8998.5	156	11646	874.5
Exam10	**66741**	452	67936.5	165.5	71808	206	69649	**158**	67881	872.5	127559.5	8344
Exam11	**218227.5**	**21067.5**	219267.5	70019.5	-	-	-	-	223552.5	36347	334398.5	86444
Exam12	**10995.5**	452	12327	5187	12748.5	3338	12323	11840	13000.5	**218**	13277.5	7873.5

Table 3.3 Median soft constraint cost results, and timings on the ITC 2007 dataset. The hyphen symbol '-' means the method cannot produce feasible solutions for the corresponding instance. Bold is lowest median soft constraint cost result. Bold and underlined results are significantly better than all others according to the Mann-Whitney U test with the Holm-Bonferroni correction at the critical level ($\alpha = 0.05$).

	OBSI		LD		LWD		LE		SD		RD	
Problem Instance	Median cost	Median t (ms)	Median cost	Median t (ms)	Median cost	Median t (ms)	Median cost	Median t (ms)	Median cost	Median t (ms)	Median cost	Median t (ms)
YUE20011	831	**62**	774.5	148	764	163.5	**706.5**	537.5	713	**62**	912	62.5
YUE20012	885	**62**	**779**	2372.5	-	-	-	-	799.5	475.5	886	1057.5
YUE20013	64	15	65	15	**63.5**	15	64	15	66	**0**	69	11
YUE20021	865	116.5	833.5	2381.5	-	-	-	-	**790.5**	473	1499.5	**116**
YUE20022	1093.5	**156**	1140.5	2098	1112.5	19975.5	**1057**	51768.5	1110.5	171	1235.5	5938.5
YUE20023	115	**15**	110.5	50.5	108	56	**102.5**	46.5	117.5	39	116.5	142.5
YUE20031	11285.5	**124**	3306	797.5	4337.5	554	3474	534.5	**2582**	137	7658	688.5
YUE20032	11378	**156**	-	-	-	-	-	-	**2360.5**	197	-	-

Table 3.4 Yeditepe dataset results. Notation as in Table 3.3.

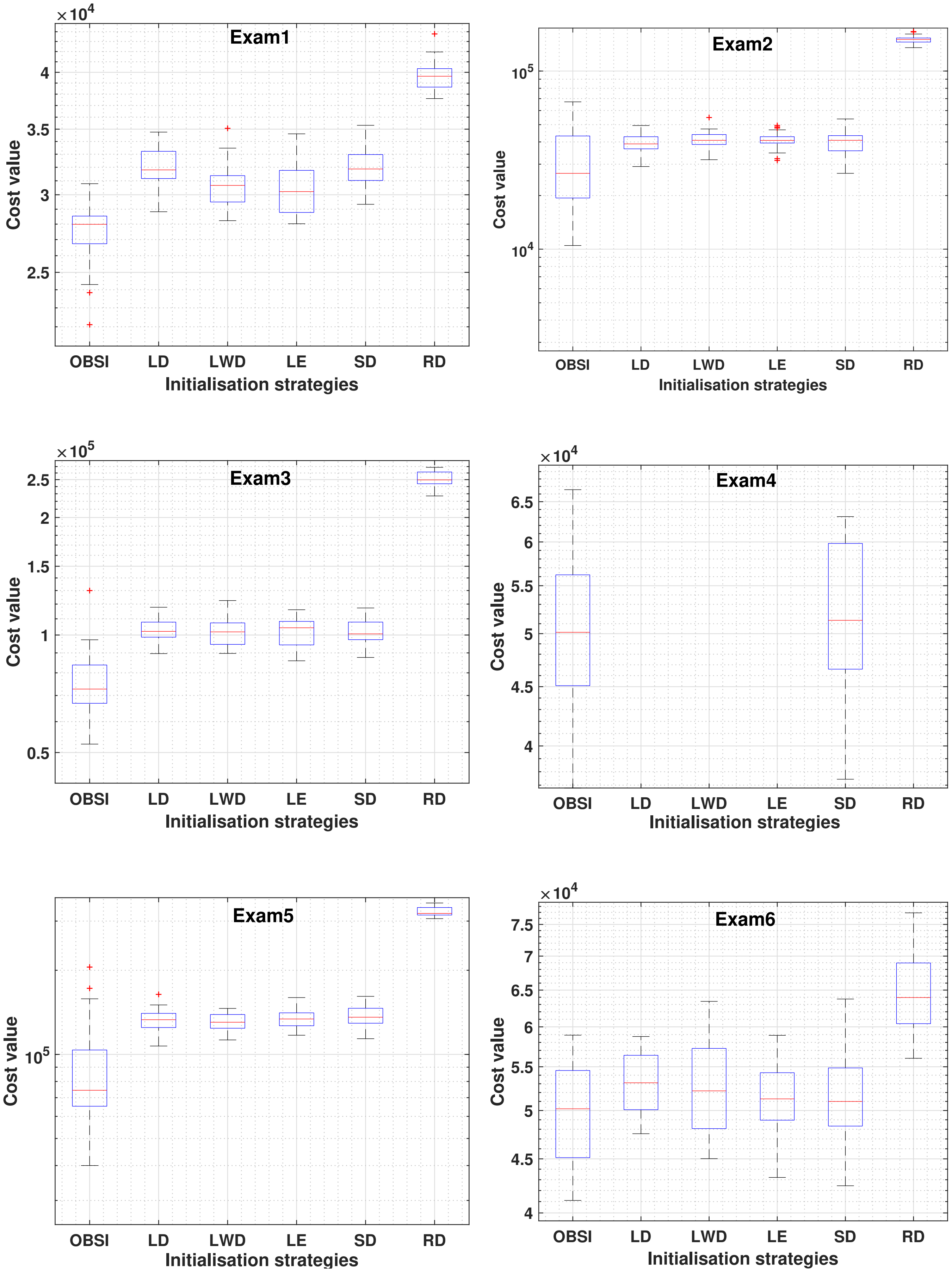

Figure 3.3 Boxplots of the soft constraint costs for all initialisation strategies performed on the ITC 2007 problem instances 1–6, where OBSI: ordering-based scheduling initialisation, LD: largest degree, LWD: largest weighted degree, LE: largest enrollment, SD: saturation degree and RD: random. Y-axis on log scale.

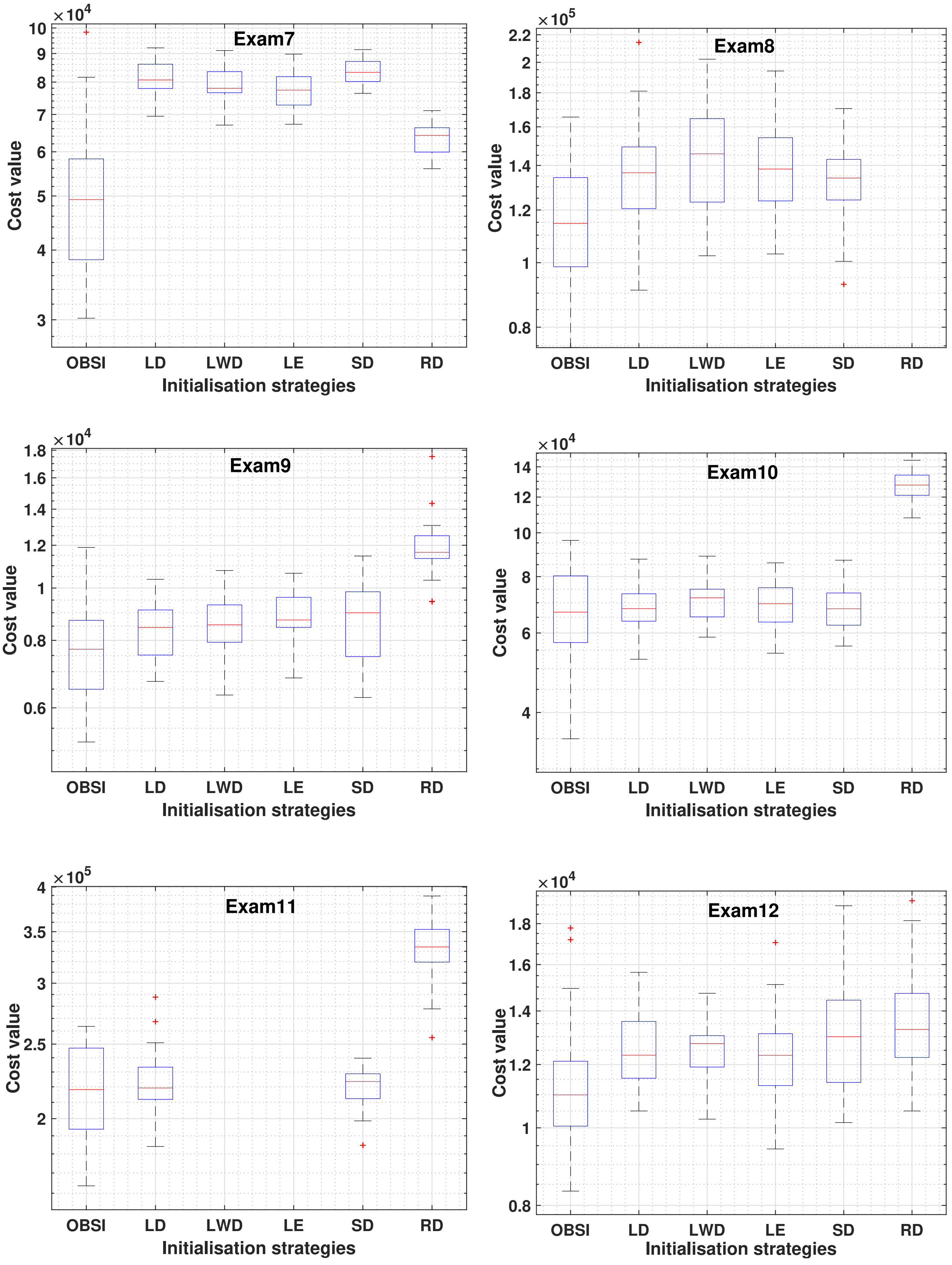

Figure 3.4 Boxplots of the soft constraint costs for all initialisation strategies performed on the ITC 2007 problem instances 7–12. Notation as in Figure 3.3

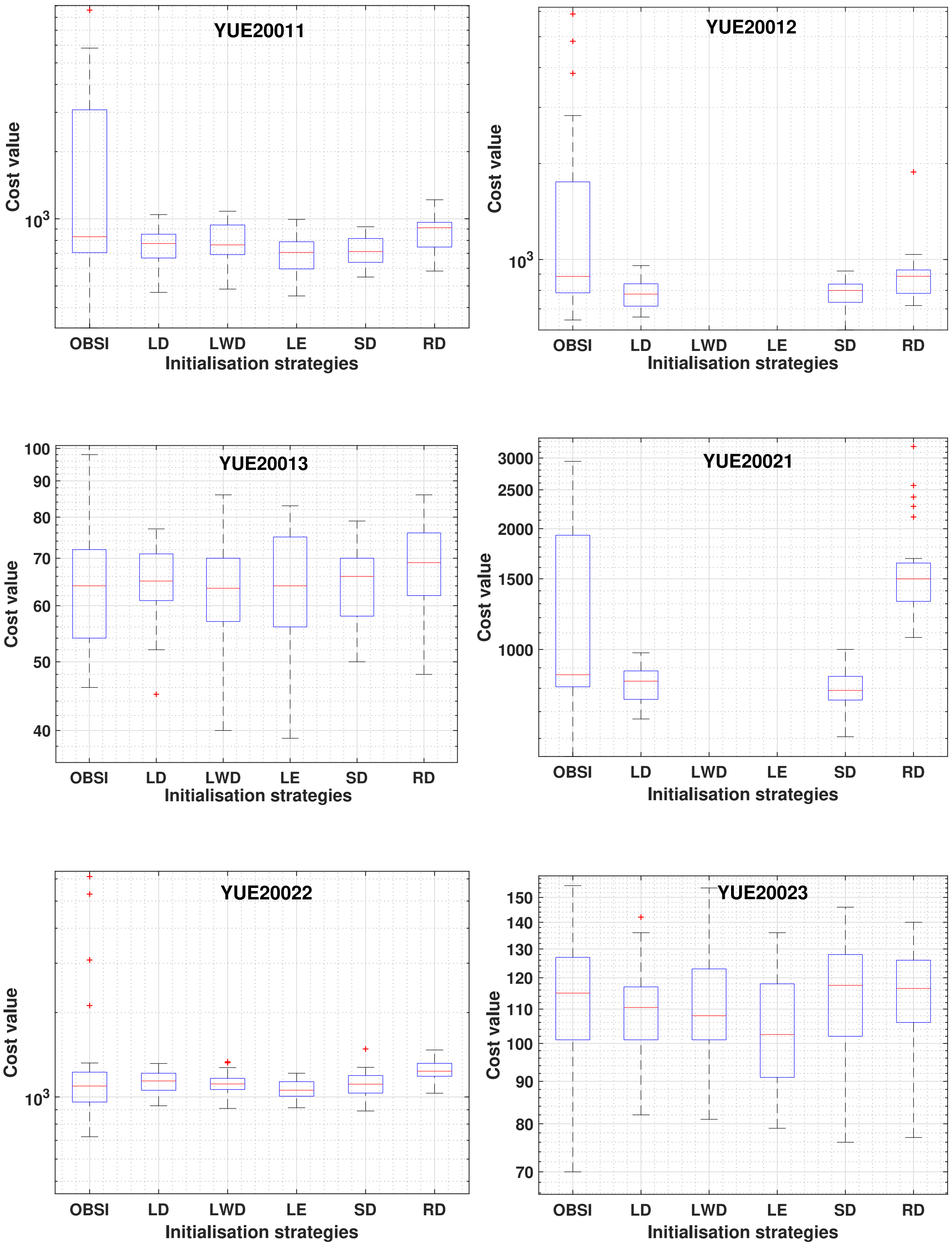

Figure 3.5 Boxplots of the soft constraint costs for all initialisation strategies performed on the Yeditepe problem instances 1–6. Notation as in Figure 3.3. Y-axis on log scale.

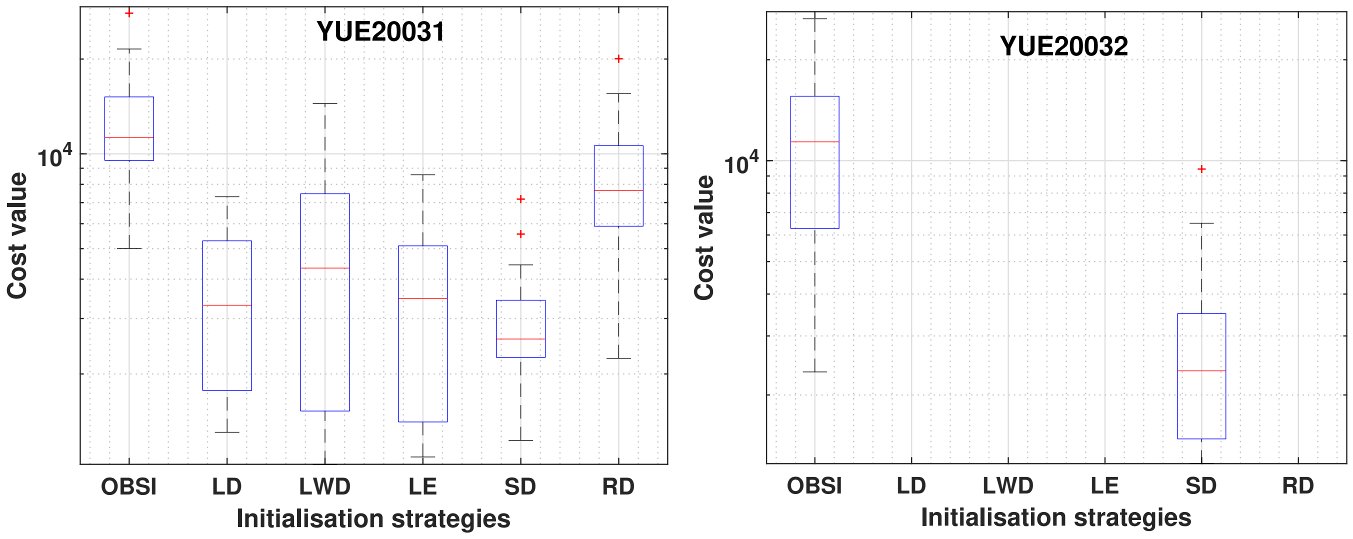

Figure 3.6 Boxplots of the soft constraint costs for all initialisation strategies performed on the Yeditepe problem instances 7–8. Notation as in Figure 3.3. Y-axis on log scale.

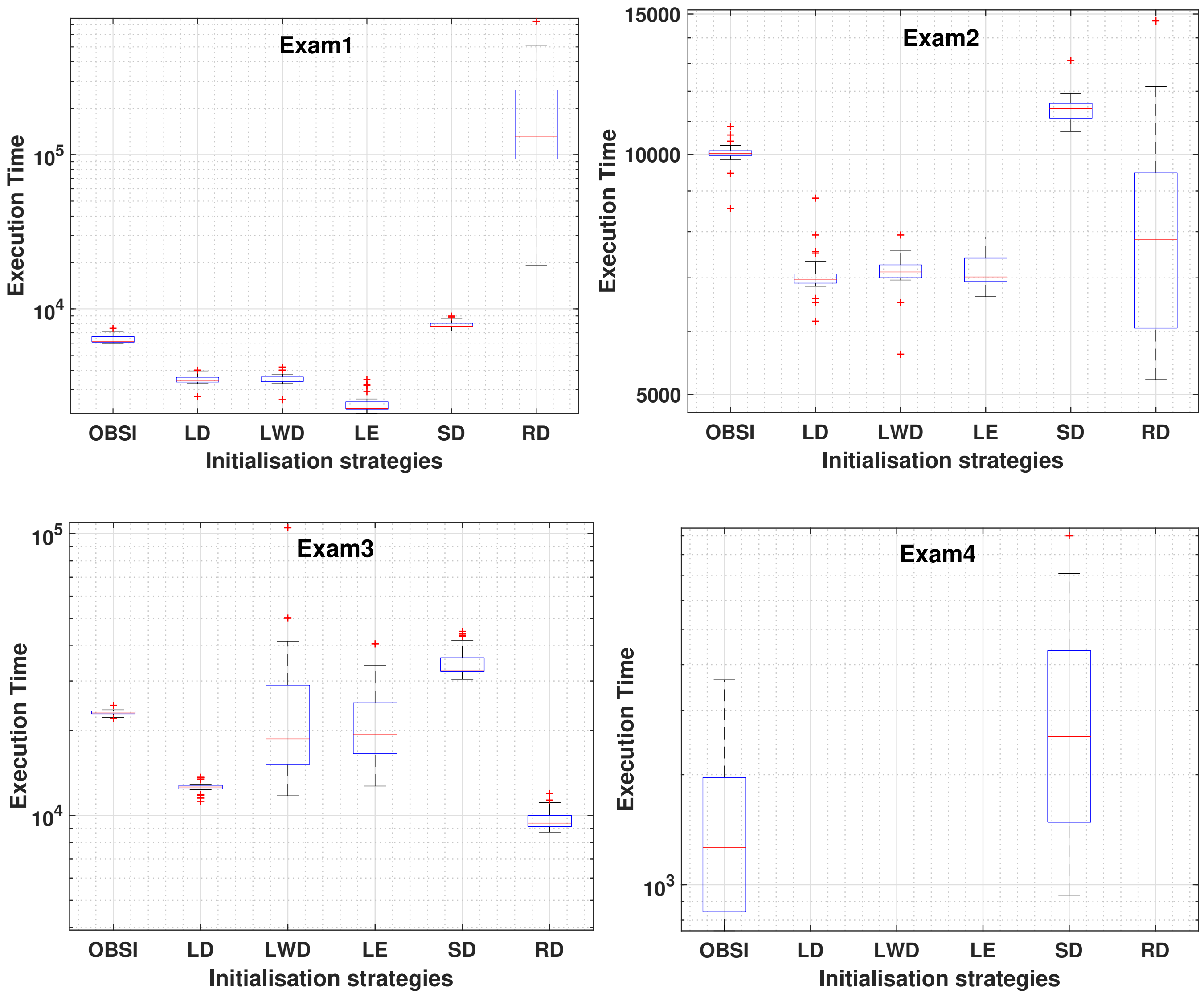

Figure 3.7 Boxplots of the log execution time in milliseconds for all initialisation strategies on the ITC 2007 problem instances 1–4. Notation as in Figure 3.3.

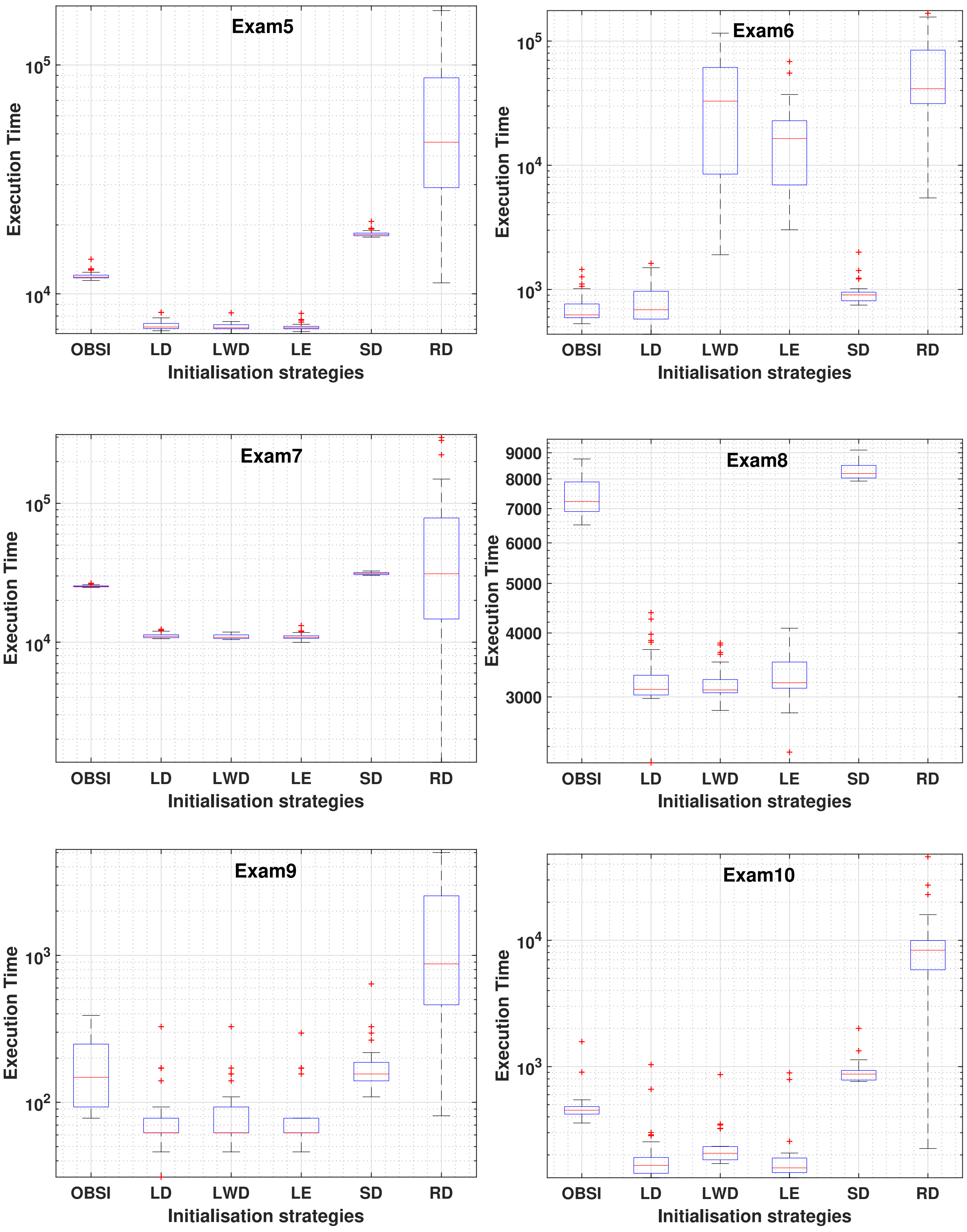

Figure 3.8 Boxplots of the log execution time in milliseconds for all initialisation strategies on the ITC 2007 problem instances 5–10. Notation as in Figure 3.3.

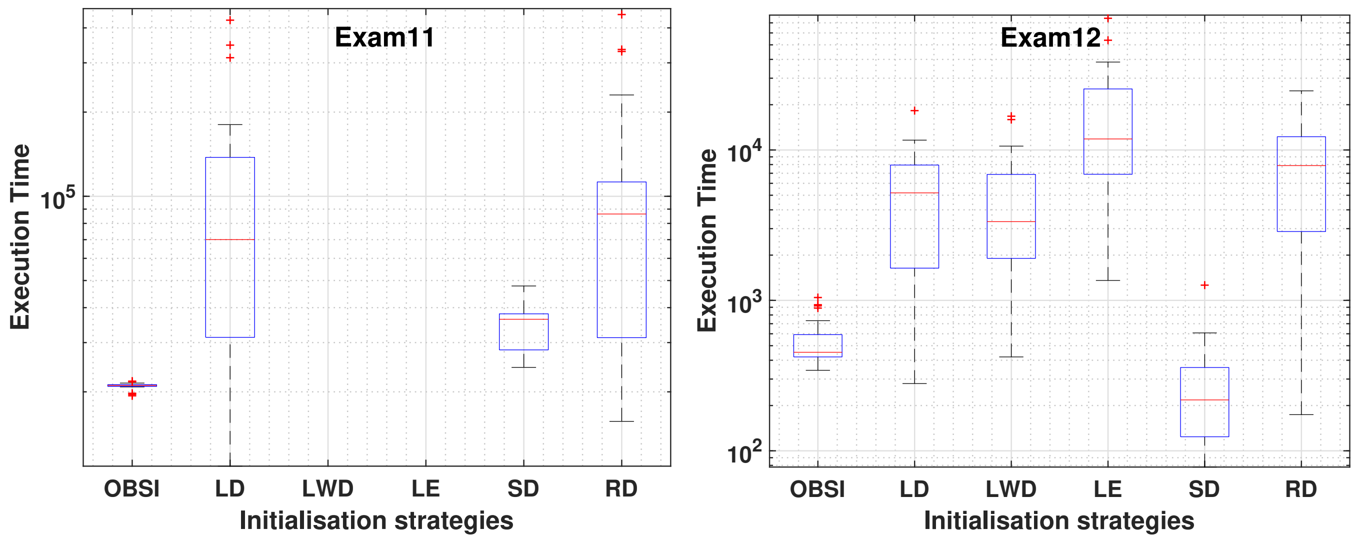

Figure 3.9 Boxplots of the log execution time in milliseconds for all initialisation strategies on the ITC 2007 problem instances 11–12. Notation as in Figure 3.3.

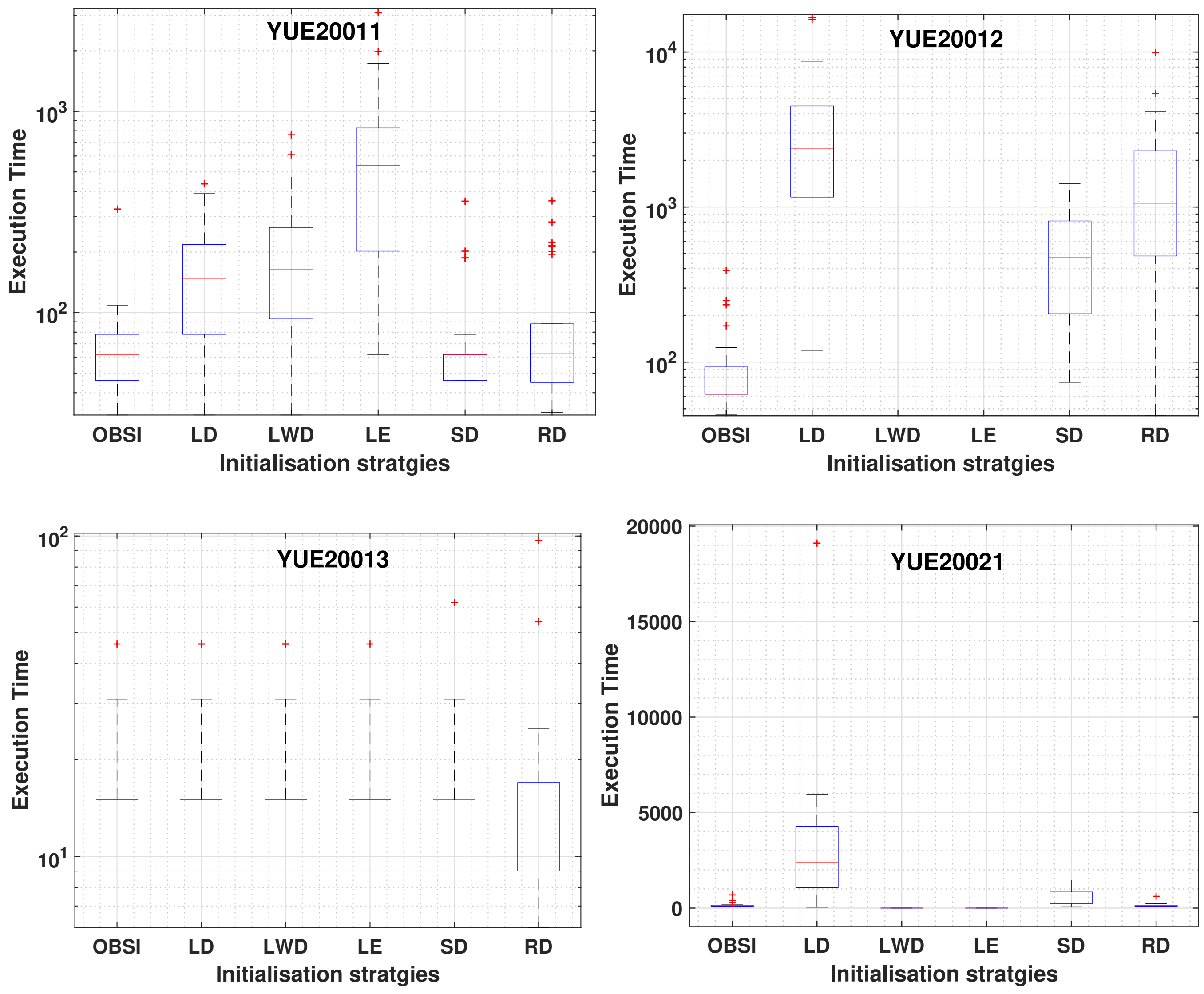

Figure 3.10 Boxplots of the log execution time of all initialisation strategies on the Yeditepe problem instances 1–4. Notation as in Figure 3.7.

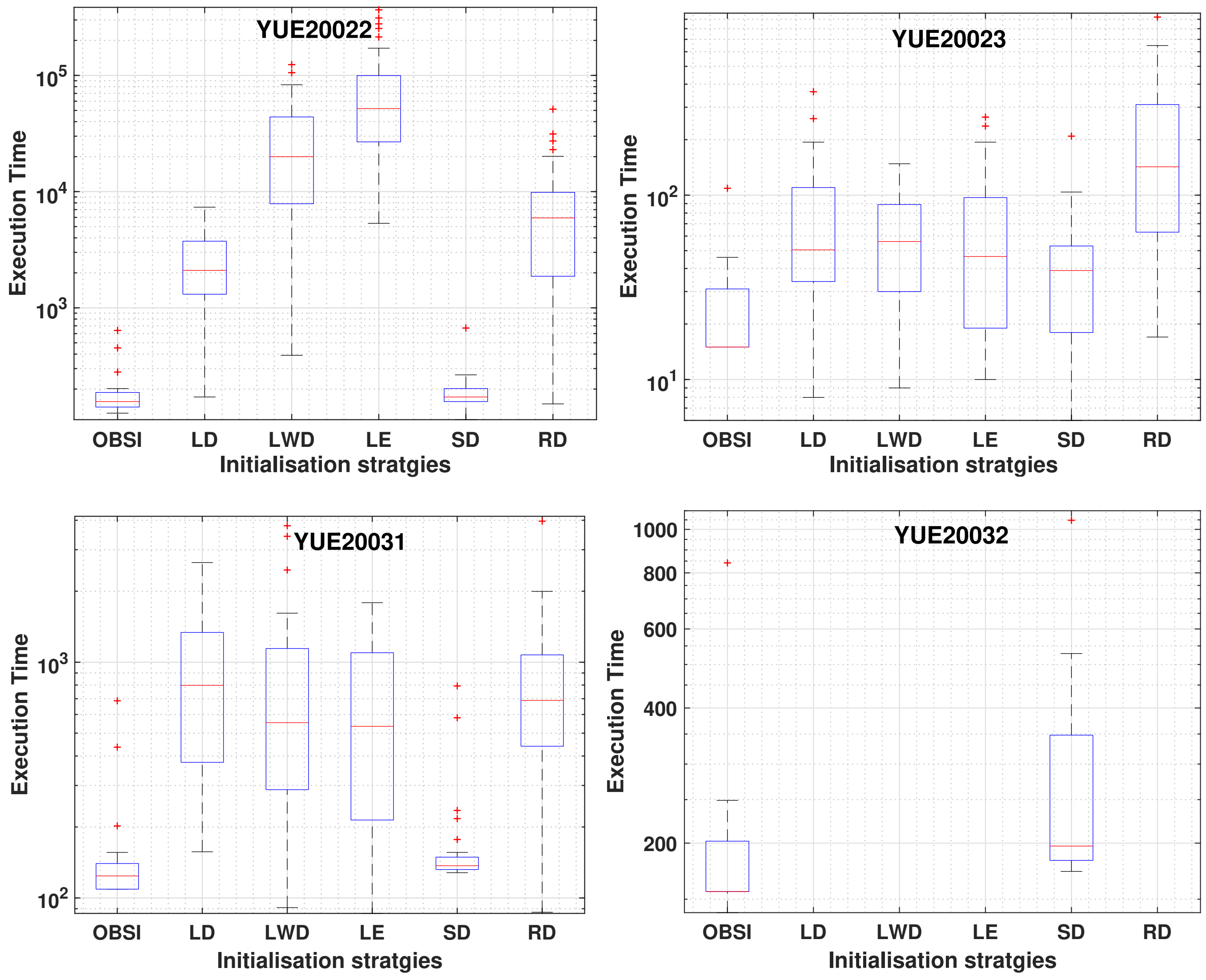

Figure 3.11 Boxplots of the log execution time of all initialisation strategies on the Yeditepe problem instances 5–8. Notation as in Figure 3.7.

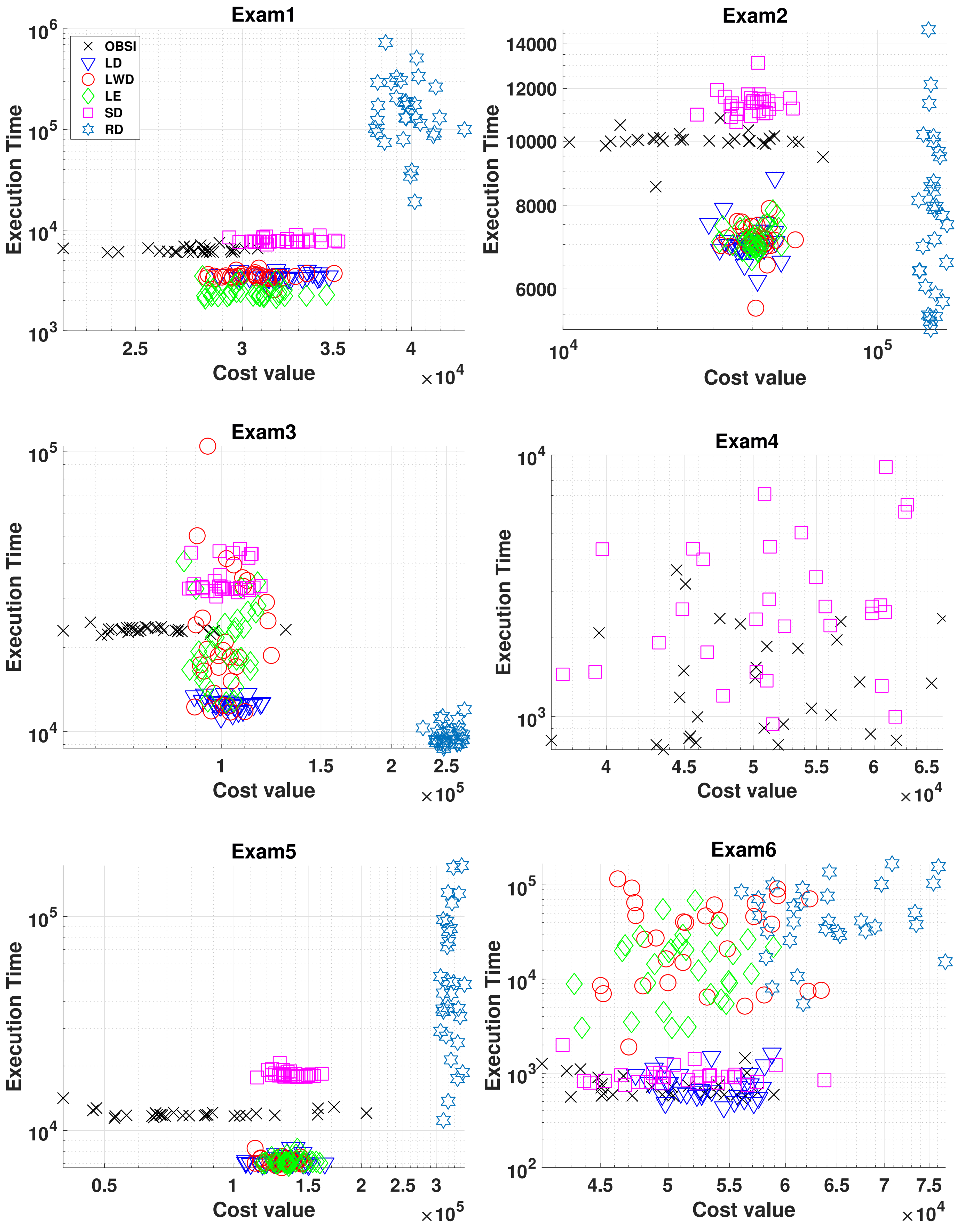

Figure 3.12 The scatter plot with cost values and execution time amounts for all initialisation strategies performed on ITC 2007 instances 1–6.

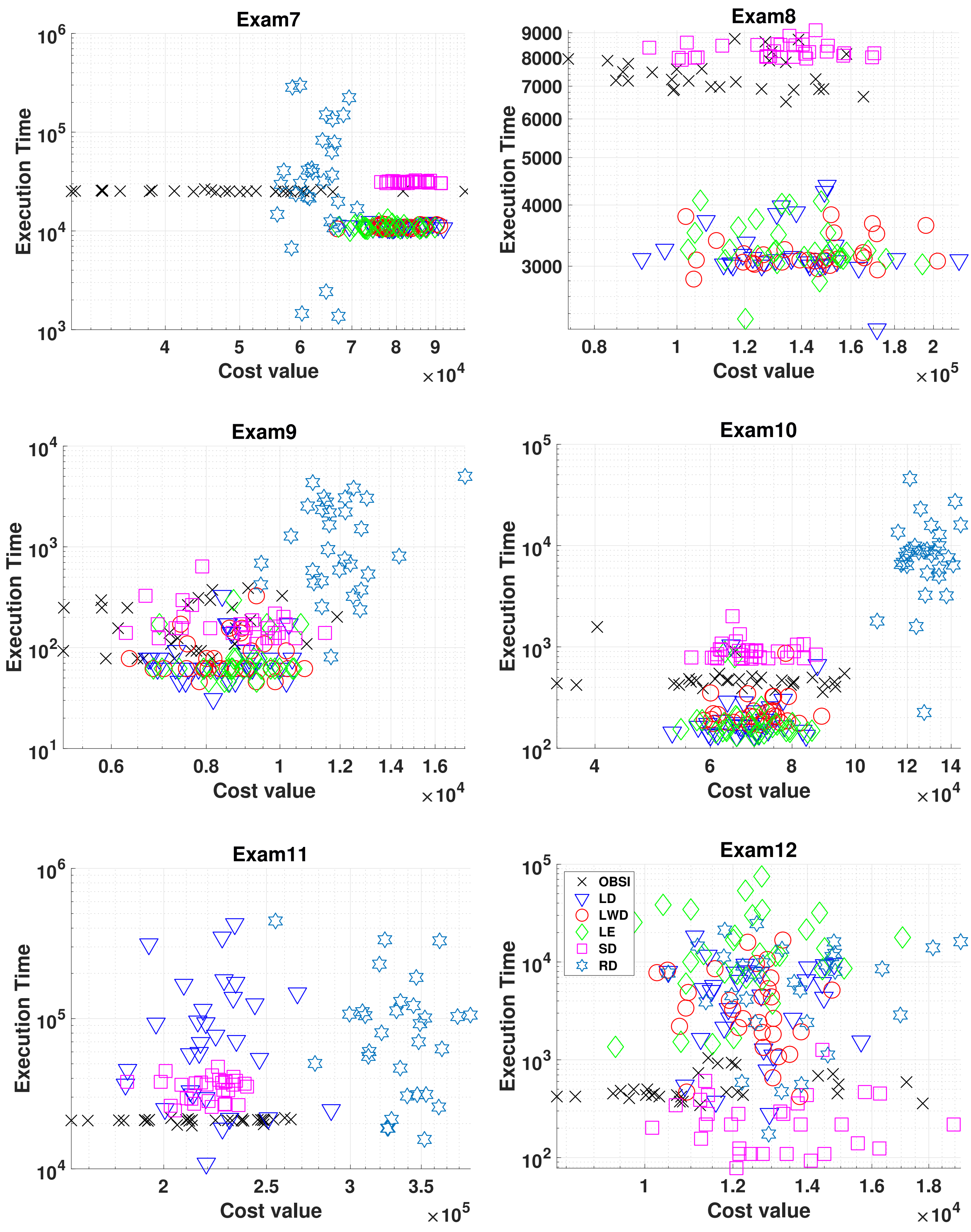

Figure 3.13 The scatter plot with cost values and execution time amounts for all initialisation strategies performed on the ITC 2007 problem instances 7–12.

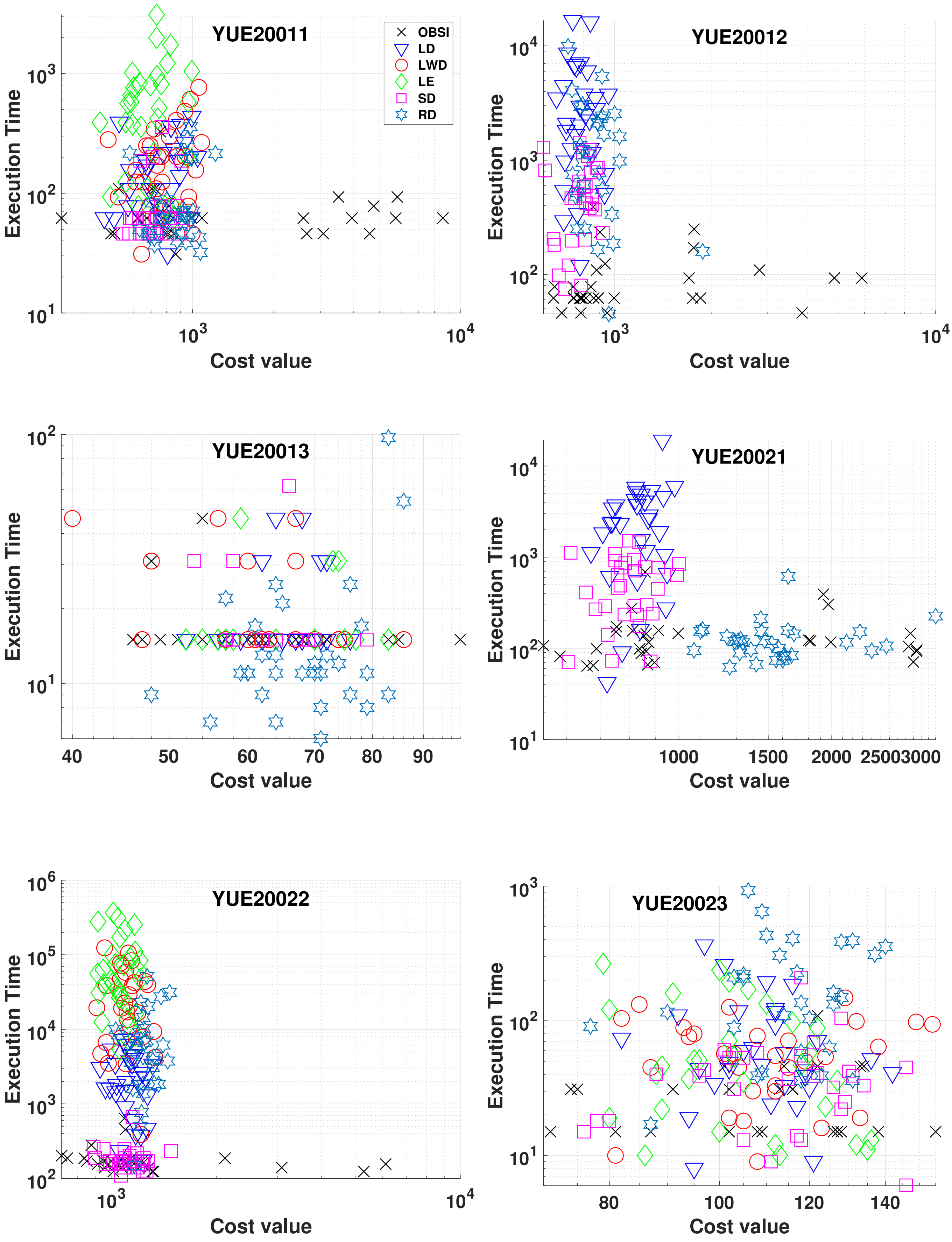

Figure 3.14 The scatter plot with cost values and execution time amounts for all initialisation strategies performed on the Yeditepe problem instances 1–6.

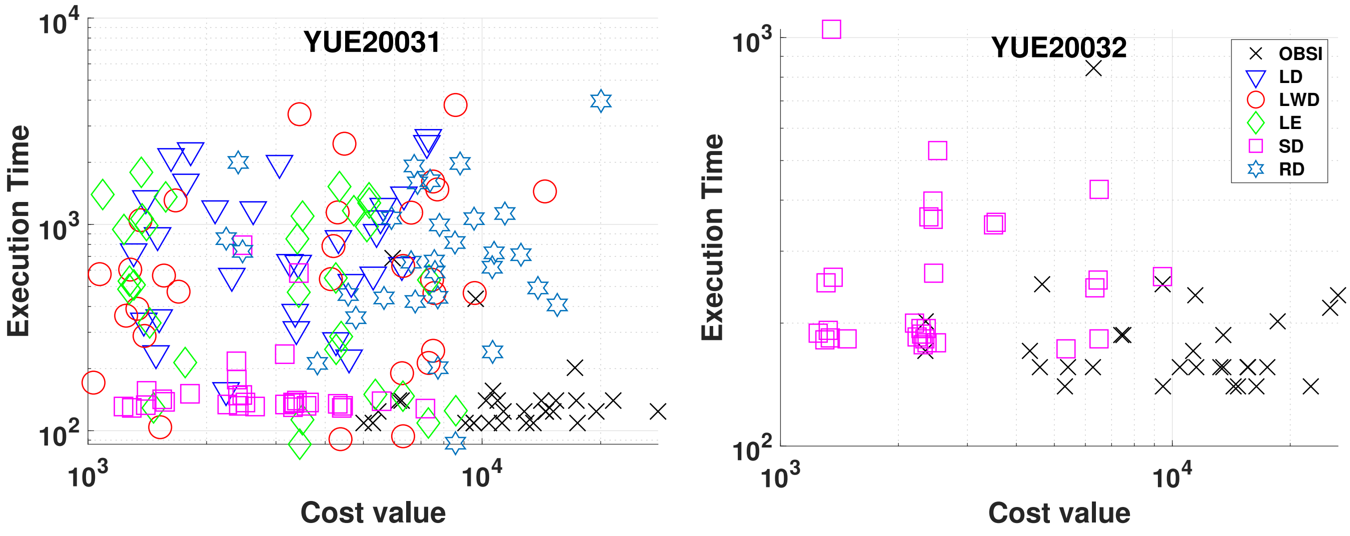

Figure 3.15 The scatter plot with cost values and execution time amounts for all initialisation strategies performed on the Yeditepe problem instances 7–8.

3.4.2 Phase 2: Optimisation

An EA was implemented following the approach of Beligiannis et al. (2008) (which eschews crossover) to quantify the effectiveness of the initialisation on subsequent meta-heuristic performance. We utilise the initial solutions that have been constructed in phase 1 by the OBSI, as well as other initialisation methods, and use these to determine the start populations of the EA. The EA population size is 40. Total run time (initialisation plus optimisation) is capped at 8 minutes.

Each child solution in the population for every generation (iteration) is generated by applying a combination of both light and heavy mutation (Burke et al., 1995c). Light mutation chooses a number of exams randomly from any period in the timetable and attempts to reschedule them at any other period while satisfying all hard constraints. The heavy mutation is performed by disturbing all the exams in one or more periods in the timetable. The periods to be disturbed are determined by the procedure from Burke et al. (1995c). After applying the two mutation operators at any time to every parent population member, the subsequent generation's parent population is populated by performing roulette wheel selection on the combined parent and child set.

Results indicate that the OBSI strategy gives not only relatively high quality solutions compared to other initialisation approaches, but that these solutions are effective for seeding an efficient population-based search. The results for most problem instances using OBSI result in a final solution surpassing the quality of runs using other initialisation strategies (see Tables 3.5 and 3.6 for the results of the ITC 2007 and Yeditepe problems respectively).

Problem	OBSI		LD		LWD		LE		SD		RD	
Instance	Median	MAD	Median	MAD	Median	MAD	Median	MAD	Median	MAD	Median	MAD
Exam1	**12955**	509.5	16802	580.5	15920.5	189	15739	136	15508.5	350.5	15731	178.5
Exam2	**2228.5**	273.5	5450.5	532	3961	204.5	3942.5	63.5	3904	112	3823	132.5
Exam3	**17442.5**	82.5	27457.5	100	27364.5	57.5	28166	410.5	27963	108.5	32174.5	1459.5
Exam4	**24345.5**	510	-	-	-	-	-	-	32664.5	957	-	-
Exam5	**7113.5**	240.5	24574.5	1867.5	18245	326.5	16831	81.5	16794.5	610.5	17743.5	124.5
Exam6	**29310**	12	32142.5	47.5	32520	200	31700	120	31567.5	102.5	31812.5	105
Exam7	**18460.5**	12	39860.5	1190.5	35726	129.5	35893.5	554	34587.5	326	32941	230
Exam8	**15837.5**	190	20077	965.5	18097.5	463.5	17490	91.5	17358.5	259	-	-
Exam9	**1530.5**	34.5	2095.5	57	2039.5	18	2020.5	43.5	1973.5	22.5	2006.5	37.5
Exam10	**15219**	446	17948.5	165	17669	55	17509.5	146	17620	148.5	17502.5	133
Exam11	**54594.5**	593.5	87940	4322	-	-	-	-	79346	472.5	89335.5	3855
Exam12	**6340.5**	79.5	7543	95.5	7587	133	7502	82	7458.5	107	7650	182

Table 3.5 Median best solution cost EA results over 30 runs and Median Absolute Deviation (MAD) after 8 minutes (per run) — ITC2007 problems. Notation as in Table 3.3.

Problem	OBSI		LD		LWD		LE		SD		RD	
Instance	Median	MAD	Median	MAD	Median	MAD	Median	MAD	Median	MAD	Median	MAD
YUE20011	**112**	7.5	257	15	240	13	241	9.5	231.5	14	246.5	5.5
YUE20012	**219**	18	355.5	10	-	-	-	-	375	7	364.5	10.5
YUE20013	41	7.5	55.5	5	56	6	54	10	56	8	59	7
YUE20021	**181**	14.5	382.5	11	-	-	-	-	374.5	12	364	13
YUE20022	**341.5**	34.5	568.5	21.5	542.5	5.5	551.5	11.5	561.5	11.5	563	15
YUE20023	**71.5**	6	100.5	8.5	98	12	92.5	13.5	107.5	12	107	10.5
YUE20031	**327.5**	18	587.5	11.5	569	21.5	564	12	541.5	24	570	9.5
YUE20032	**647**	21.5	-	-	-	-	-	-	888	13.5	-	-

Table 3.6 Median best solution cost EA results over 30 runs and Median Absolute Deviation (MAD) after 8 minutes (per run) — Yeditepe problems. Notation as in Table 3.3.

In addition, I now further expand the discussion on why and how much the proposed OBSI contributes to better scheduling results. One of the critical factors in the development of meta-heuristic approaches is the connectivity of the search space. This is largely defined by the initialisation method used and can significantly impact the quality of solutions obtained. As shown by several studies, a good initial solution can help to obtain better final solutions (Burke and Newall, 2003, 2004; Gogos et al., 2012). If the search technique starts from location(s) in the search space that is highly connected within good regions, it helps the search technique to yield final high-quality results. However, the global optimum will be difficult to reach if the technique starts searching from an area (i.e. a bad region) that is isolated from the global optimum. Therefore, the reliance on the initial solution becomes significant for the success of the improvement approach (Burke et al., 2010a).

In the case of the examination timetabling problem, this problem has a large, complex, and constrained search space. The main contribution of the proposed initialisation method is to guide the search technique to start optimisations from promising areas (i.e. those having fewer soft constraint violations) in the search space. In contrast, there are many initialisation methods commonly used in this problem, such as random initialisation. The possibility of constructing an infeasible solution is high due to hard constraints violations. Moreover, the algorithm often starts searching in a bad region of the search space as all soft constraints are neglected during the construction phase. Consequently, the quality of obtaining a feasible solution is often low.

The OBSI initialises with the Front List, including all the examinations with a specific ordering. Unscheduled examinations that cannot be assigned to any period in the first section still have a higher possibility of getting valid periods in other sections to be scheduled. This typically avoids getting stuck during the construction process. Thus, the OBSI can guarantee to generate high-quality solutions across all problem instances of considered datasets as shown in Tables 3.3 and 3.4. On the other hand, from Table 3.7, we observe that using the random initialisation may produce a feasible solution in some problem instances. However, infeasible (i.e. incomplete) examination timetables can be obtained as the case in Exam 4 and Exam 8 since it gives priority of adding examinations to periods sequentially, and that can lead to the methods to be stuck in some steps during the process of building the exam timetable due to hard constraints violations. Furthermore, the random initialisation is unable to return a solution that is equivalently good to the proposed initialisation method.

Table 3.7 shows the comparison of the initialisation and improvement approach (i.e. EA) with the OBSI as well as RD. We extract the results obtained by the OBSI and RD from Tables 3.3 and 3.5 in order to easily quantify the effectiveness of the OBSI compared to RD on the final scheduling results. The results clearly show that the OBSI have performed significantly better than the RD. Before proceeding to the improvement phase, for instance, the median cost for the OBSI in Exam 1 of the ITC 2007 was 27987.5, and after applying the improvement approach, the result shows a remarkable increase in the solution quality where the median cost became 12955. While the median cost for the RD in the same problem instance was 39630, and then the median cost was decreased to a value more than the OBSI, which is 15731. This revealed that the EA performed badly in comparison, as shown in Table 3.7 when starting from solutions constructed by the RD. However, the situation changed the performance of the EA when using the OBSI. Therefore, the greatest benefit of this method (i.e. OBSI) is the initialising of solutions in 'good' areas of design space where it greatly increased the performance of the EA and final solutions quality in most problem instances.

Problem	OBSI		RD	
Instance	Initialisation	Optimisation	Initialisation	Optimisation
Exam1	27987.5	12955	39630	15731
Exam2	26661	2228.5	150514.5	3823
Exam3	72713	17442.5	249617.5	32174.5
Exam4	50139	24345.5	-	-
Exam5	74394.5	7113.5	319573.5	17743.5
Exam6	50190	29310	63957.5	31812.5
Exam7	49253	18460.5	64212.5	32941
Exam8	114559	15837.5	-	-
Exam9	7705.5	1530.5	11646	2006.5
Exam10	66741	15219	127559.5	17502.5
Exam11	218227.5	54594.5	334398.5	98335.5
Exam12	10995.5	6340.5	13277.5	7650

Table 3.7 Comparison between the results obtained from OBSI and RD in initialisation and optimisation phases. The hyphen symbol '-' means the method cannot produce feasible solutions for the corresponding problem instance.

With regards to the computing time, the one notable disadvantage to the proposed OBSI described is that it takes a considerable larger computational time on large problem instances of the ITC 2007, which can be as much as graph colouring heuristics (as shown in Tables 3.3 and 3.4). However, the OBSI is capable of saving a considerable amount of time to reach high-quality solutions compared to the RD. As such, the time to generate the feasible examination timetable is not a significant issue for real-world timetabling problems, as the examination timetable is often only taken once or twice a year where it is usually carried out one to two months before the examinations take place.

3.5 Analysis of Results

3.5.1 Statistical Test Method

We analyse our results by conducting statistical tests, as shown in Tables 3.3, 3.4, 3.5, and 3.6 using the Mann-Whitney U test (Mann and Whitney, 1947) (also referred to as Mann–Whitney Wilcoxon test (Wilcoxon, 1950)) followed by the Holm-Bonferroni method to compensate for multiple hypothesis testing.

3.5.2 Statistical Analysis

For the statistical analysis, the Mann-Whitney U test was performed first, followed by the Holm-Bonferroni method as a post hoc method for obtaining the adjusted p-value for each comparison between a control method (i.e. the OBSI method is considered the best performing method) in the initialisation phase and the rest (Holm, 1979; Gaetano, 2018). The aim of the post-hoc method is to verify if significant differences were detected and to show results are significantly better than all others. Tables 3.8 and 3.9 present the adjusted (Mann-Whitney U test) p-value and the further results of the post-hoc methods (Holm-Bonferroni test) on the ITC 2007 and Yeditepe datasets, respectively.

For most instances of the ITC 2007 dataset (as shown in Table 3.8), it can be observed that the p-value and Holm's p-value correction for any results between two approaches (i.e. the control approach (OBSI approach) and another approach) are significantly different (SIG) which indicates that the corrected p-value is below the critical level ($\alpha = 0.05$). Whereas for instances from Yeditepe dataset, the p-value for the observed results in Table 3.9 is more than 0.05, indicating that the difference between the results is not statistically significant ($p \geq \alpha$).

The Mann-Whitney U-test was also used to identify differences between results when EA was incorporated with the OBSI and other methods (i.e. LD, LWD, LE, SD, and RD) in phase 2. A post-hoc test (Holm–Bonferroni method) were also performed for multiple comparison correction and comparing differences among these observed results. Tables 3.10 and 3.11 present the adjusted (Mann-Whitney U test) p-value and the further results of the post-hoc methods (Holm-Bonferroni test) on the ITC 2007 and Yeditepe datasets, respectively. According to these results, statistically significant differences were found in this phase.

3.5.2.1 Phase 1: Initialisation

Exam1				Exam2			
OBSI vs.	p-value	P.Holm	Outcome	OBSI vs.	p-value	P.Holm	Outcome
LD	1.03E-02	0.0106	*SIG*	LD	1.46E-10	4.3929E-10	*SIG*
LWD	2.20E-03	0.0084	*SIG*	LWD	2.60E-08	5.203E-08	*SIG*
LE	2.10E-03	0.0084	*SIG*	LE	3.52E-07	3.5201E-07	*SIG*
SD	8.99E-11	3.59736E-10	*SIG*	SD	5.30E-03	0.0106	*SIG*
RD	3.02E-11	1.50995E-10	*SIG*	RD	3.02E-11	1.50995E-10	*SIG*
Exam3				Exam4			
OBSI vs.	p-value	P.Holm	Outcome	OBSI vs.	p-value	P.Holm	Outcome
LD	2.44E-09	9.7544E-09	*SIG*	LD	-	-	-
LWD	1.01E-08	2.021E-08	*SIG*	LWD	-	-	-
LE	31.56E-08	2.021E-08	*SIG*	LE	-	-	-
SD	6.52E-09	1.95549E-08	*SIG*	SD	2.28E-01	0.2282	*NON SIG*
RD	3.02E-11	1.50995E-10	*SIG*	RD	-	-	-
Exam5				Exam6			
OBSI vs.	p-value	P.Holm	Outcome	OBSI vs.	p-value	P.Holm	Outcome
LD	1.46E-10	4.3929E-10	*SIG*	LD	1.56E-02	0.0624	*NON SIG*
LWD	2.60E-08	5.203E-08	*SIG*	LWD	5.75E-02	0.1725	*NON SIG*
LE	3.52E-07	3.5201E-07	*SIG*	LE	3.56E-01	0.711	*NON SIG*
SD	5.30E-03	0.0106	*SIG*	SD	4.64E-01	0.711	*NON SIG*
RD	3.02E-11	1.50995E-10	*SIG*	RD	9.92E-11	4.9593E-10	*SIG*
Exam7				Exam8			
OBSI vs.	p-value	P.Holm	Outcome	OBSI vs.	p-value	P.Holm	Outcome
LD	2.92E-09	1.1686E-08	*SIG*	LD	2.60E-03	0.0052	*SIG*
LWD	3.20E-09	1.1686E-08	*SIG*	LWD	2.01E-04	0.00080232	*SIG*
LE	4.18E-09	1.1686E-08	*SIG*	LE	6.91E-04	0.00207375	*SIG*
SD	4.64E-01	0.711	*SIG*	SD	8.30E-03	0.0083	*SIG*
RD	2.88E-06	0.000002879	*SIG*	RD	-	-	-
Exam9				Exam10			
OBSI vs.	p-value	P.Holm	Outcome	OBSI vs.	p-value	P.Holm	Outcome
LD	3.26E-02	0.0326	*SIG*	LD	8.42E-01	1	*NON SIG*
LWD	8.70E-03	0.0174	*SIG*	LWD	4.73E-01	1.000	*NON SIG*
LE	1.20E-03	0.0048	*SIG*	LE	7.28E-01	1.000	*NON SIG*
SD	5.10E-03	0.0153	*SIG*	SD	6.95E-01	1.000	*NON SIG*
RD	2.37E-10	1.18575E-09	*SIG*	RD	3.02E-11	1.50995E-10	*SIG*
Exam11				Exam12			
OBSI vs.	p-value	P.Holm	Outcome	OBSI vs.	p-value	P.Holm	Outcome
LD	7.73E-01	1.000	*NON SIG*	LD	4.60E-03	0.0138	*SIG*
LWD	-	-	-	LWD	5.00E-03	0.0138	*SIG*
LE	-	-	-	LE	1.70E-02	0.017	*SIG*
SD	8.30E-01	1	*NON SIG*	SD	1.20E-03	0.0048	*SIG*
RD	3.69E-11	1.10691E-10	*SIG*	RD	1.78E-04	0.0008918	*SIG*

SIG: the corrected p-value of rank j is significant, $p < \alpha$

NON SIG: the corrected p-value is not significant, $p \geq \alpha$

Table 3.8 Adjusted (Mann-Whitney U Test) p-values on the ITC 2007 datasets. The hyphen symbol '-' means the method cannot produce feasible solutions for the corresponding problem instance.

YUE20011				YUE20012			
LE vs.	p-value	P.Holm	Outcome	LD vs.	p-value	P.Holm	Outcome
OBSI	3.00E-11	1.49765E-10	*SIG*	OBSI	2.98E-11	2.98E-11	*SIG*
LD	3.20E-03	0.0128	*SIG*	SD	8.29E-05	8.29E-05	*SIG*
SD	1.83E-02	0.0549	*NON SIG*	RD	4.70E-03	4.70E-03	*SIG*
RD	5.06E-01	1.000	*NON SIG*	LE	-	-	-
LWD	5.39E-01	1	*NON SIG*	LWD	-	-	-
YUE20013				YUE20021			
LWD vs.	p-value	P.Holm	Outcome	SD vs.	p-value	P.Holm	Outcome
OBSI	1.13E-06	5.6345E-06	*SIG*	OBSI	3.00E-11	9.003E-11	*SIG*
RD	2.51E-01	1.000	*NON SIG*	RD	1.62E-01	0.3244	*NON SIG*
SD	6.20E-01	1.000	*NON SIG*	LD	0.169	0.3244	*NON SIG*
LD	9.65E-01	1.000	*NON SIG*	LE	-	-	-
LE	9.76E-01	1	*NON SIG*	SD	-	-	-
YUE20022				YUE20023			
LE vs.	p-value	P.Holm	Outcome	LE vs.	p-value	P.Holm	Outcome
OBSI	2.99E-11	1.49675E-10	*SIG*	OBSI	3.15E-06	0.000015755	*SIG*
LWD	4.50E-03	0.018	*SIG*	RD	5.10E-03	0.0204	*SIG*
LD	3.70E-02	0.111	*NON SIG*	SD	5.27E-02	0.1581	*NON SIG*
RD	6.33E-02	0.1266	*NON SIG*	LD	2.40E-01	0.4792	*NON SIG*
SD	1.49E-01	0.149	*NON SIG*	LWD	2.49E-01	0.4792	*NON SIG*
YUE20031				YUE20032			
SD vs.	p-value	P.Holm	Outcome	SD vs.	p-value	P.Holm	Outcome
OBSI	2.96E-11	1.479E-10	*SIG*	OBSI	3.00E-11	3.0047E-11	*SIG*
LD	7.58E-08	3.03364E-07	*SIG*	LD	-	-	-
RD	1.20E-05	0.000036096	*SIG*	RD	-	-	-
LWD	1.30E-03	0.0026	*SIG*	LWD	-	-	-
LE	1.18E-02	0.0118	*SIG*	LE	-	-	-

Table 3.9 Adjusted (Mann-Whitney U Test) p-values on the Yeditepe datasets. Notation as in Table 3.8

3.5.2.2 Phase 2: Optimisation

	Exam1			Exam2		
OBSI + EA vs.	p-value	P.Holm	Outcome	p-value	P.Holm	Outcome
LD + EA	3.01E-11	1.50425E-10	*SIG*	3.01E-11	1.50425E-10	*SIG*
LWD + EA	3.01E-11	1.50425E-10	*SIG*	8.12E-11	1.50425E-10	*SIG*
LE + EA	3.01E-11	1.50425E-10	*SIG*	3.01E-11	1.50425E-10	*SIG*
SD + EA	3.01E-11	1.50425E-10	*SIG*	3.01E-11	1.50425E-10	*SIG*
RD + EA	3.01E-11	1.50425E-10	*SIG*	3.01E-11	1.50425E-10	*SIG*
	Exam3			Exam4		
OBSI vs.	p-value	P.Holm	Outcome	p-value	P.Holm	Outcome
LD + EA	3.01E-11	1.4827E-10	*SIG*	-	-	-
LWD + EA	3.00E-11	1.4827E-10	*SIG*	-	-	-
LE + EA	3.00E-11	1.4827E-10	*SIG*	-	-	-
SD + EA	2.97E-11	1.4827E-10	*SIG*	3.01E-11	3.0123E-11	*SIG*
RD + EA	2.98E-11	1.4827E-10	*SIG*	-	-	-
	Exam5			Exam6		
OBSI vs.	p-value	P.Holm	Outcome	p-value	P.Holm	Outcome
LD + EA	3.00E-11	1.4958E-10	*SIG*	2.98E-11	1.4911E-10	*SIG*
LWD + EA	3.00E-11	1.4958E-10	*SIG*	3.00E-11	1.4911E-10	*SIG*
LE + EA	2.99E-11	1.4958E-10	*SIG*	2.99E-11	1.4911E-10	*SIG*
SD + EA	3.00E-11	1.4958E-10	*SIG*	2.99E-11	1.4911E-10	*SIG*
RD + EA	3.00E-11	1.4958E-10	*SIG*	2.99E-11	1.4911E-10	*SIG*
	Exam7			Exam8		
OBSI vs.	p-value	P.Holm	Outcome	p-value	P.Holm	Outcome
LD + EA	3.00E-11	1.49955E-10	*SIG*	3.00E-11	1.19964E-10	*SIG*
LWD + EA	3.00E-11	1.49955E-10	*SIG*	3.00E-11	1.19964E-10	*SIG*
LE + EA	3.00E-11	1.49955E-10	*SIG*	8.93E-11	1.7869E-10	*SIG*
SD + EA	3.00E-11	1.49955E-10	*SIG*	1.60E-10	1.7869E-10	*SIG*
RD + EA	3.00E-11	1.49955E-10	*SIG*	-	-	-
	Exam9			Exam10		
OBSI vs.	p-value	P.Holm	Outcome	p-value	P.Holm	Outcome
LD + EA	3.01E-11	1.5052E-10	*SIG*	3.02E-11	1.5071E-10	*SIG*
LWD + EA	3.01E-11	1.5052E-10	*SIG*	3.02E-11	1.5071E-10	*SIG*
LE + EA	3.02E-11	1.5052E-10	*SIG*	3.02E-11	1.5071E-10	*SIG*
SD + EA	3.02E-11	1.5052E-10	*SIG*	3.01E-11	1.5071E-10	*SIG*
RD + EA	3.02E-11	1.5052E-10	*SIG*	3.02E-11	1.5071E-10	*SIG*
	Exam11			Exam12		
OBSI vs.	p-value	P.Holm	Outcome	p-value	P.Holm	Outcome
LD + EA	3.02E-11	9.0597E-11	*SIG*	3.01E-11	1.50425E-10	*SIG*
LWD + EA	-	-	-	3.01E-11	1.50425E-10	*SIG*
LE + EA	-	-	-	3.01E-11	1.50425E-10	*SIG*
SD + EA	3.02E-11	9.0597E-11	*SIG*	3.01E-11	1.50425E-10	*SIG*
RD + EA	3.02E-11	9.0597E-11	*SIG*	3.01E-11	1.50425E-10	*SIG*

Table 3.10 Adjusted (Mann-Whitney U Test) p-values on the ITC 2007 datasets. Notation as in Table 3.8

	YUE20011			YUE20012		
OBSI + EA vs.	p-value	P.Holm	Outcome	p-value	P.Holm	Outcome
LD + EA	3.00E-11	1.49485E-10	*SIG*	2.10E-03	0.0054	*SIG*
LWD + EA	1.71E-01	0.3428	*NON SIG*	-	-	-
LE + EA	5.70E-03	0.0285	*SIG*	-	-	-
SD + EA	2.24E-02	0.0896	*NON SIG*	1.80E-03	0.0054	*SIG*
RD + EA	8.94E-01	0.8941	*SIG*	4.92E-01	0.4917	*NON SIG*
	YUE20013			YUE20021		
OBSI vs.	p-value	P.Holm	Outcome	p-value	P.Holm	Outcome
LD + EA	3.01E-11	1.4827E-10	*SIG*	3.01E-11	9.003E-11	*SIG*
LWD + EA	3.00E-11	1.4827E-10	*SIG*	-	-	-
LE + EA	3.00E-11	1.4827E-10 *SIG*	-	-	-	
SD + EA	2.97E-11	1.4827E-10	*SIG*	3.00E-11	9.003E-11	*SIG*
RD + EA	2.98E-11	1.4827E-10	*SIG*	3.00E-11	9.003E-11	*SIG*
	YUE20022			YUE20023		
OBSI vs.	p-value	P.Holm	Outcome	p-value	P.Holm	Outcome
LD + EA	3.00E-11	1.4939E-10	*SIG*	4.70E-09	1.87976E-08	*SIG*
LWD + EA	3.00E-11	1.4939E-10	*SIG*	4.79E-08	1.32705E-07	*SIG*
LE + EA	2.99E-11	1.4939E-10	*SIG*	3.15E-06	0.000003151	*SIG*
SD + EA	2.99E-11	1.4939E-10	*SIG*	4.42E-08	1.32705E-07	*SIG*
RD + EA	3.00E-11	1.4939E-10	*SIG*	6.02E-10	3.0092E-09	*SIG*
	YUE20031			YUE20032		
OBSI + EA vs.	p-value	P.Holm	Outcome	p-value	P.Holm	Outcome
LD + EA	3.00E-11	1.49955E-10	*SIG*	-	-	-
LWD + EA	3.00E-11	1.49955E-10	*SIG*	-	-	-
LE + EA	3.00E-11	1.49955E-10	*SIG*	-	-	-
SD + EA	3.00E-11	1.49955E-10	*SIG*	3.00E-11	3.0047E-11	*SIG*
RD + EA	3.00E-11	1.49955E-10	*SIG*	-	-	-

Table 3.11 Adjusted (Mann-Whitney U Test) p-values on the Yeditepe datasets. Notation as in Table 3.8

3.6 Summary

This chapter has presented a detailed description of a new strategy for constructing initial examination timetables. A number of initialisation approaches (i.e. different graph colouring heuristics and random initialisation) have been presented in this chapter. Comparing with other popular initialisation strategies, our results demonstrate that this strategy statistically outperforms graph heuristic and random approaches on nearly all of the ITC 2007 benchmark instances for both quality and diversity and also on some of Yeditepe benchmark instances for quality. Furthermore, incorporation within a simple EA has demonstrated the advantage that using OBSI to generate an initial population of high quality and diverse solutions provides in the final timetables returned post optimisation. This is for the same total time cost (initialisation time plus optimisation time).

In the next section, we present an new approach to improve solutions where the proposed initialisation method is used in the construction phase.

Chapter 4

An Exam Specialised Genetic Algorithm for Examination Timetabling Problem

4.1 Introduction

This chapter proposes a novel Genetic Algorithm (GA) variant. The new variant is called the Exam Specialised Genetic Algorithm (ESGA) and aims to solve the examination timetabling problem. A variety of novel methods and operators are applied to the GA to tackle particular problems of timetabling. These operators attempt to prevent various types of violation, including directed, as well as undirected mutations, which are applied to examinations or periods.

The chapter describes several mutation types that are applied to certain individuals under specific circumstances. These include new methods that aim to avoid early convergence and to control the size of mutations and the probability of a mutation being applied. This can allow an appropriate balance to be achieved between exploration and exploitation.

To ensure a fair comparison, both the proposed algorithm and the Basic GA use the OBSI method to generate the initial population. Three popular benchmark sets are used for the testing, and we also compare to an extensive range of published results.

The chapter is structured as follows. Section 4.2 discusses applying GAs to examination timetabling problems. Examination solution representation is given in Section 4.3. Proposed modifications to the basic GA are presented in Section 4.4. Section 4.5 describes the enhanced roulette wheel selection strategy. Section 4.6 presents the proposed algorithm, and Section 4.7 presents and discusses the experimental results. The chapter ends with the summary in Section 4.8.

4.2 Genetic Algorithm for Examination Timetabling

A recent survey of GAs for solving examination timetabling problems was conducted by Adnan et al. (2018). This survey showed GAs' ability to solve optimisation problems and that they perform well in examination timetabling problem. In the work presented in this chapter, GAs are chosen to solve examination timetabling problems, since they are known for their validity (Pelikan, 2010; Sani and Yabo, 2016) in solving complex combinatorial problems. Moreover, GAs have the ability and flexibility to search in complicated, large spaces (Montana et al., 1998). Ross et al. (1994), Dhande et al. (2012), and Adnan et al. (2018) considered GAs as one of the most forceful tools in solving exam timetabling problems. In addition to that, GAs are characterised by being adaptive search algorithms (Juang et al., 2007). Corne et al. (1993), Corne et al. (1994b), Paechter et al. (1994), and Burke et al. (1994a) have also used GAs to solve the problems of examination timetabling. However, they revealed that conventional GAs need to be enhanced if they are to generate good solutions to the examination timetabling problem. Furthermore, the effectiveness of a GA is highly dependent on the changes made to its parameters. Therefore, a GA needs often intelligent settings to be provided to its parameters such as population size, crossover rate, mutation rate and type, the number of generations, and selection operators.

Figure 4.1 shows an example of a GA for solving examination timetabling problems. At the initialisation of the algorithm, a conflict matrix (Burke and Newall, 1999) is created. This N by N matrix is utilised with the aim of enabling competent conflict checking. For generating the initial population, a number of graph colouring heuristics are used. Then, the fitness function, which sums up all violations of all constraints, evaluates the overall quality of each individual in the initial population. In order for the algorithm to function effectively, the selection operator must randomly select parents from the population, but in such a way that better parents have a higher probability of being chosen (this is called evolutionary pressure). In this approach, it is anticipated that, on average, a child will have a higher quality, and that, after entering the population and replacing a less desirable solution, the population's overall quality will improve. Subsequently, the crossover and mutation operators are applied to the selected solution (chromosome). Crossover occurs when genetic information is exchanged between two randomly chosen partners, resulting in one or more offspring with chromosomes that are unique from their parents. The mutation is the process of changing allele values in a chromosome at random in order to produce genetic variety. Thereafter, if an offspring is infeasible, a repair procedure is used on the offspring to remove duplicate examinations and assign missing examinations at random to restore the offspring's feasibility. Finally, the best offspring are selected after revaluating each one. The process is repeated until a termination condition is satisfied or a certain number of generations has been attained.

Based on this, an exam specialised genetic algorithm is proposed in this chapter to solve the examination timetabling problem. As such solutions are often modified using a repair function, which might influence the algorithm's performance (Osaba et al., 2014; Norgren and Jonasson, 2016). Various sets of mutations are introduced by the proposed method (violation directed, blind, heavy, light), as well as new operators and strategies to explore

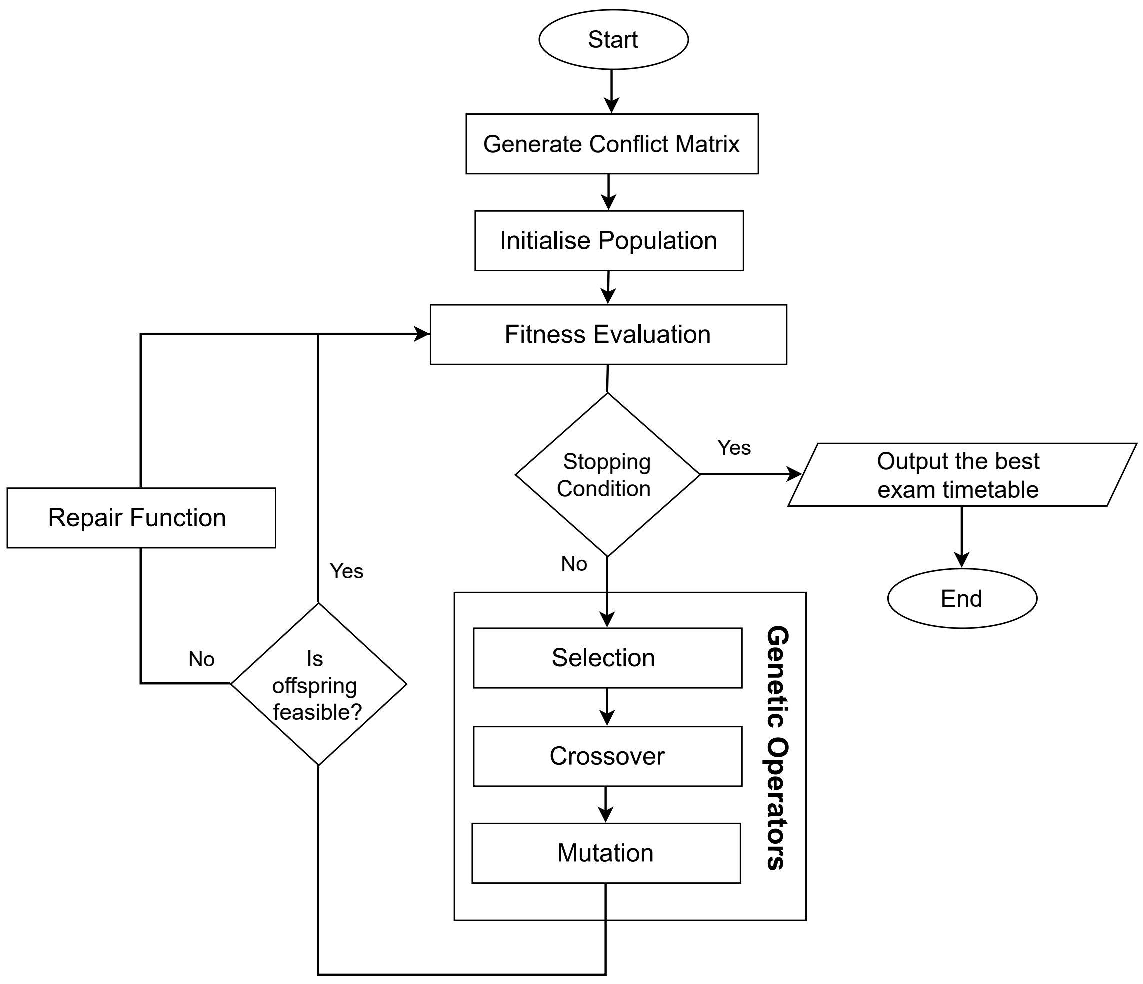

Figure 4.1 An example of GA for solving examination timetabling problem.

the solution space effectively and exploit it efficiently so that high quality solutions are produced. The crossover operator is eliminated since it caused premature convergence, which is unavoidable in this domain, i.e. individuals are inclined to be similar. Also, infeasible solutions are often generated in a highly constrained combinatorial optimisation problem like the examination timetabling problem. Therefore, a repair function is required to fix the infeasible solutions, which in turn increases the execution time significantly. On the other hand, a mutation is a tiny change to an individual that takes much less time than a crossover operator (De Giovanni et al., 2013). Beligiannis et al. (2008) proposed an algorithm to create feasible and efficient timetables for high schools in Greece and to show that mutation is sufficient to provide new good solutions for population evolution and that the crossover operator does not provide a satisfactory contribution, and it adds an excessive amount of complexity and time delay. Their experimental results were better if a crossover was not used since it led to very slow convergence (and sometimes even not to convergence) and also changed the location where the search took place because it applied very large steps in the search space due to the large number of changes in each chromosome. Ross et al. (1994) suggested just employing the mutation operator to create offspring solutions for addressing timetabling problems in their institutions. Their method outperformed a genetic algorithm that used a uniform crossover operator, according to their test data. According to Reeves and Wright (1995), modest steps in the search space should be taken. Using a crossover operator, clearly, does not do this. As a result, it was decided that crossover would not be used. Finally, an evolutionary algorithm (based solely on mutation and survivor selection functions) is more effective than a standard genetic algorithm in solving combinatorial optimisation problems, according to the results of the experiments of Osaba et al. (2013).

4.3 Examination Solution Representation

Each examination timetable is represented as a chromosome, and its exams are the genes. According to the literature, there are two methods to represent exam timetables in a GA: direct and indirect representation. In a "direct" representation (Adamidis and Arapakis, 1999), all event attributes (day, period, room, etc.) are explicitly encoded as a vector of positive integer numbers. As a result, the GA must decide on all timetable parameters and offer a complete and constraint-free schedule in these circumstances. This approach leads to a huge search space in which solutions that satisfy all hard constraints appear to be like finding needles in a haystack. Therefore, by this representation, it could produce infeasible solutions (Rothlauf, 2002; Algethami et al., 2016) that require a repair function, which is often computationally costly in examination timetable problems and could have an impact on the algorithm's performance (Osaba et al., 2014; Norgren and Jonasson, 2016).

To simplify the direct representation even further, a list of the numbers is created of a certain length. This represents the exams to be arranged or scheduled, where each of the elements is between 1 and p (representing the available number of periods). Interpreting this chromosome is as follows: when the nth number on the list is p, exam e is then scheduled or planned to take place at period p. The chromosome [8,11,6,1,2,5,1,2,3,6], for instance, represents a specific candidate solution, where $exam_1$ occurs at $period_8$, $exam_2$ occurs at $period_{11}$, and so on.

[3,7,9,2,1,10,4,5,8,11] represents a different chromosome. A certain point (i.e. crossover point) is chosen randomly, let's say 3, and a crossover is performed to produce two children, as illustrated in Figure 4.2. This evolutionary cycle is repeated many times until an ideal timetable can be located or until a specific maximum number of generations is reached. The mutation operator alters the features of copy of a current "parent" as illustrated in Figure 4.3. After this mutation, $exam_8$, which occurs at $period_2$ will be at $period_5$. After the genetic operators (crossover in addition to mutation), a repair algorithm is utilised with the aim of repairing an infeasible solution, which aims at fixing any non-feasible timetables that are produced in this process.

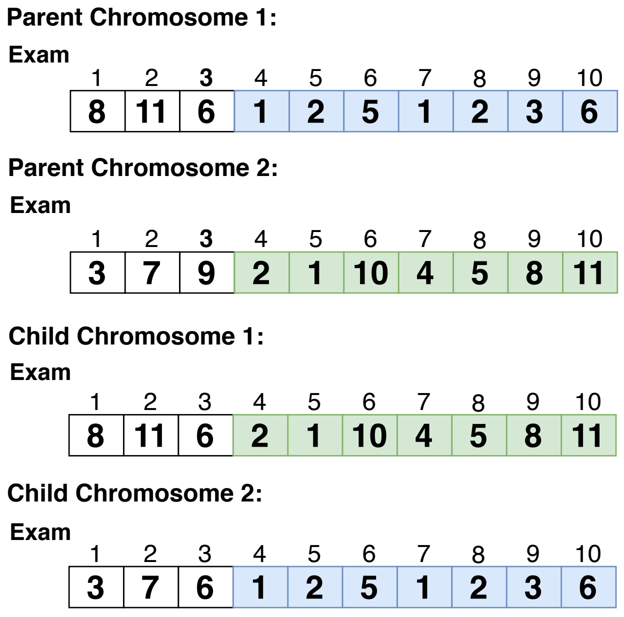

Figure 4.2 Representation and Crossover Parent.

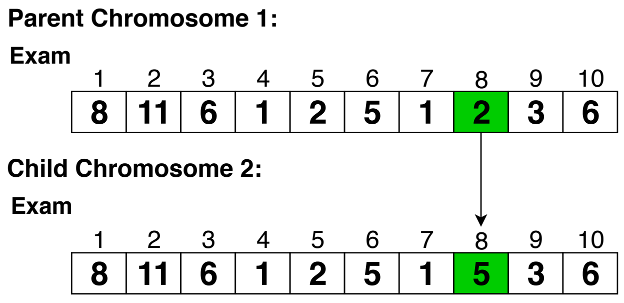

Figure 4.3 The Mutation Parent.

In “indirect” representations (Paechter et al., 1996), the encoded solution (chromosome) often comprises an ordered list of events that are inserted into the timetable using a predetermined technique (or “timetable builder”). While monitoring the problem’s restrictions, the timetable builder can utilize any mix of heuristics and local search to insert events into the timetable. Indirect representation for the ESGA implementation of this study is employed via optimised data structures that preserve all the essential input information in every solution in such a way that the algorithm verifies that all of the hard restrictions have been met before performing any type of mutation. This eliminates the need for the ESGA to do repairs. In general, compared to basic direct methods, using an indirect representation of the timetable considerably increases the efficiency of timetable generation (Paechter et al., 1994).

There are numerous encodings for the same problem, such as directly encoding a candidate solution or indirectly encoding parameters/features for a constructive algorithm that generates a candidate solution. In addition to the features listed above, it is simple and easy to estimate mutation effects in the direct representation. Moreover, chromosomal interpretation is quick (resulting in faster fitness evaluation). In contrast, the indirect representation frequently involves the use of a constructive algorithm to easily exploit domain knowledge — (e.g., in the constructive heuristic), it is also capable of significantly reducing the size of the search space but with slow interpretation as well as the possibility of encoding away undesirable features, and finally, its neighbourhoods are extremely rugged (Brabazon and O'Neill, 2006). However, according to the findings of Brabazon and O'Neill (2006), problem-specific knowledge should be included in the representation of solutions to the scheduling problem, and the chromosomal representation should be natural. It should contain all relevant details and be directly connected to the original problem.

Figure 4.4 shows the implemented indirect representation in this thesis. Regarding the software implementation, the solution can be constructed through the creation of a list of periods as 'objects' that comprises further room objects, which in turn contain exam objects. Each of the object types possesses its individual properties, which are utilised for checking hard, as well as soft constraints before the rescheduling of the examinations in any of the examination rooms for a specific period. For instance, when the algorithm aims to reschedule a specific exam in a specific period, the exam's duration property has to be compared against the period's duration property to fulfil the period duration's constraint. Another example is that when rescheduling an exam that has an 'after' constraint like $exam_2$ after $exam_1$ in a new period, the proposed algorithm checks whether $exam_1$ exists in the exam list property of any period before that period. This representation enables us to implement the ESGA and prevents us from using a repair algorithm that needs a high computing cost to regenerate feasible solutions, particularly when more constrained ITC 2007 datasets are used.

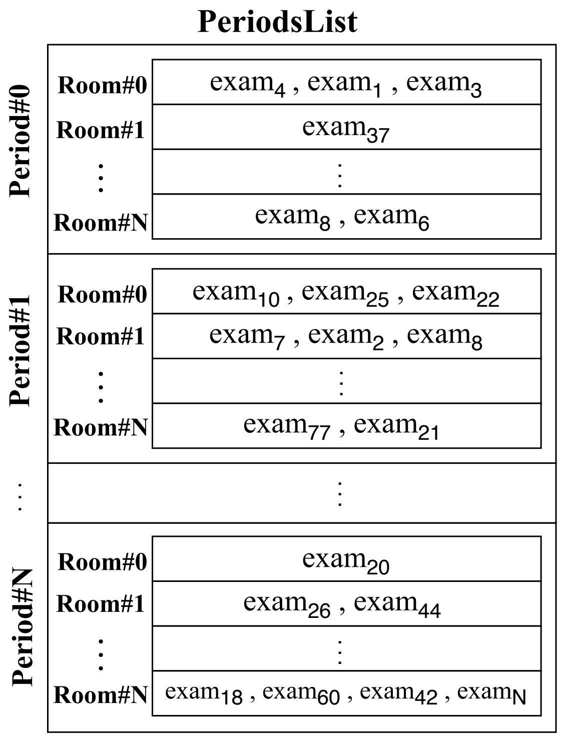

Figure 4.4 Chromosome representing a solution.

4.4 Proposed Modifications to the Basic Genetic Algorithm

This work mainly involves tackling the traditional genetic algorithm's drawbacks when trying to solve the examination timetable problems by proposing specific modifications to the basic GA. In that regard, this work includes the following objectives:

1. To test a GA-based optimiser on the ITC 2007 and Yeditepe benchmark sets for the first time (as far as the author is aware). See Tables 2.15, 2.16, and 2.17 in Chapter 2.

2. To design new mutation operators so that an effective balance is achieved between exploration and exploitation in this application domain.

3. To adapt several parameters such as mutation probability and mutation size, which are relative to the number of periods and/or the number of exams in a given timetable to make all the mutation operators suitable and applicable to any timetable. In this work, the selection of the parameter values, mutation probability values and mutation size values were carried out based on experimental tuning studies, whereby a reasonable balance is considered between the solution quality and the computation time.

4. To propose a probabilistic selection technique in order to prevent the solutions from being too similar after many generations (premature convergence).

5. To eliminate the crossover phase, which substantially increases the execution time of the genetic algorithm that is applied to the examination timetable problem, without improving the results (following the work of Ross et al. (1994), Burke et al. (1995c), Reeves and Wright (1995), Beligiannis et al. (2008), and Osaba et al. (2013)).

6. To introduce optimised data structures to represent the solution in such a way that the algorithm is able to check that all the hard constraints are satisfied before applying any type of mutation. This enables the ESGA to obviate need for a repair function, which is often computationally expensive in examination timetable problems.

7. To utilise room resources effectively and efficiently by regularly reordering all the rooms of any period according to the available seats at any time an examination is scheduled in a specific period. In this way, the process of rescheduling aims at timetabling the examinations in the room with the least number of available seats first, then the room with the highest number of the available seats, and so forth.

There are two phases in this overall approach. In the first phase, the focus is placed on the evolving timetables. These timetables meet hard constraints. In the second phase, meanwhile, the focus is on improving the timetables generated in the first phase by minimising the soft constraint values. Also, each timetable can be characterised as a list, which comprises the available periods in a specific exam session. Each period comprises the predefined room list, which involves as many exams as possible based on fulfilling the constraints that are stipulated, as well as the rooms' capacities. The following subsections describe the two phases of the proposed algorithm in detail.

4.4.1 Phase 1: Initialisation

The Ordering-Based Scheduling Initialisation (OBSI) is implemented in this phase to generate a feasible construction of the initial population (Alsuwaylimi and Fieldsend, 2019). This phase mainly focuses on producing timetables that fulfil hard constraints, while attempting also to minimise the cost of the soft constraints. The ESGA invokes the OBSI method n times with the aim of creating an initial population of size n for the second phase. The results (out of 30 runs for each instance) on both datasets (ITC 2007 examination track and Yeditepe) are shown in Section 3.4 in Chapter 3, respectively. Section 5.2.1 in Chapter 5 provides the experimental results of the proposed OBSI of the Toronto datasets.

4.4.2 Phase 2: Optimisation

As previously mentioned, the optimisation phase aims mainly to minimise the violation cost of the feasible timetables' soft constraints, which are constructed in the first phase. During this phase, the modified genetic algorithm (i.e. the proposed ESGA) works on refining the initial population over a set number of generations by means of the newly introduced mutation operators. The key advantage of mutation is that it opens up new spaces to be explored. It also involves jumping in the solution space, which can be beneficial. The key disadvantage, however, is that the randomness of the variation it creates may decrease the fitness of a specific solution rather than increase it.

The focus of the process of refinement is to reduce the timetables' soft constraint costs. In this work, the proposed mutation operators are different from each other in terms of:

- The solutions to which they will be applied. In other words, some mutation operators are applied to specific solutions, whereas other operators are applied to all the solutions, and each solution in the population has its own mutation probability value for each type of mutation operator. The mutation probability value of each solution is determined by its cost/fitness value.
- Whether they are classified as exploration components or exploitation components (in order to establish a balance between exploration and exploitation).
- The level at which they are applied (some mutations are performed on exams, and others are performed on periods). At the exam level, the mutation operators attempt to change the period and room of one or more selected exams taken from one or more periods. At the period level, they attempt to change the period and room of all the exams (i.e. change for all possible) of one or more selected periods.
- Guided and unguided. In other words, mutation operators explore and exploit the search space using unguided strategies and guided strategies, i.e. violation directed and blind (random) strategies. These mutation strategies can be applied to the exams' level and periods' level.

- Mutation size (heavy mutation and light mutation). In other words, how many exams or periods will be mutated in a single mutation? From our earlier preliminary experiments, we derived considerably good formulas for each proposed mutation, where it has been determined the 4^{th} and 5^{th} roots in some formulas (as shown in the following subsections) in order to provide appropriate values for the probability of selecting exams or periods to be mutated and the maximum size of all different mutations proposed in this chapter. Thus, these values attempt to produce an appropriate balance to be achieved between exploration and exploitation. The following subsections present the formulas and the details of the sizes of the proposed mutations.

- Repeated or single application. Some mutation operators are repeatedly applied to selected solution(s) many times to generate new and various solutions, whereas others are applied to selected solution(s) just once.

The aim of all the proposed mutation operators is to change the period, and the room for the selected exam(s), and all of them check that all the hard constraints are met before rescheduling the selected exam(s) in the new selected period and room. This means that there is no need to use the repair function. All the mutation operators create a copy for each selected solution before applying mutation. When applying any type of mutation to a solution copy, all the chosen exams to be rescheduled are added to a list called the *MutationList*. If the selected exam has a coincidence hard constraint, the set of exams that coincide with this selected exam are also added to the same *MutationList* by using *Check Coincidence* algorithm. The pseudo-code of the *Check Coincidence* algorithm is given in Algorithm 4.1.

Algorithm 4.1 Pseudo-code of Check Coincidence.

```
Require: $Sol$                                  ▷ Selected solution (timetable) to be mutated.
Require: $eList$                                ▷ A list containing the exam to be mutated.
1: if coin_constraint($eList$) = $true$ then    ▷ If $eList$ has coincidence hard constraint.
2:     $eList^{coin\_exams} \leftarrow$ get_coin_exams($eList$)  ▷ Get all exams coincidence with $eList$.
3:     $eList \leftarrow eList \cup eList^{coin\_exams}$         ▷ Add all exams that coincide with $eList$.
4: end if
5: return $eList$
```

When the *MutationList* becomes empty, this means that the mutation operation has succeeded and the mutated solution copy (child) will be added to the offspring pool. Otherwise, if there are exams remaining in the *MutationList*, and there are no periods that can be used to schedule these exams, the solution copy is discarded. Further, that mutation will not be reapplied to that solution so that an infinite loop is avoided, and it will also not attempt to replace that solution with a new solution, i.e. it will not recompense the current solution with a new solution. All the mutation operators employ a function named *Rollback*, which is called immediately after adding each exam to the *MutationList*. This function removes any chosen exam from the Exam List property of its original period, removes it from its original room, recalculates the capacity of the room, and checks if the removed exam has the room exclusive constraint in order to change the value of the exclusive property of its original room to be non-exclusive. Its original room, therefore, becomes available for any other exam(s) later. The pseudo-code of the *Rollback* function is detailed in Algorithm 4.2.

Algorithm 4.2 Pseudo-code of Rollback.

```
Require: $Sol$                  ▷ A selected solution (timetable) to be mutated.
Require: $MutL$                 ▷ Mutation List consists of a list of exams to be mutated.
 1: $PL \leftarrow$ getAllExams_in_period($Sol$)      ▷ A list of exams taken from each period.
 2: $RL \leftarrow$ getAllExams_in_room($Sol$)        ▷ A list of exams taken from each room.
 3: for each room in $RL$ do
 4:     $R_{seats} \leftarrow$ capacity($RL_i$)        ▷ Get capacity of each room in RL.
 5: end for
 6: $MutL_{Size} \leftarrow$ size($MutL$)             ▷ Number of exams in $MutL$.
 7: $Size \leftarrow 0$                               ▷ Counter is set to zero.
 8: while $Size \leq MutL_{Size}$ do
 9:     $PL \leftarrow PL \setminus \{MutL_i\}$  ▷ Delete the exam from the exam list of its current period.
10:     $RL \leftarrow RL \setminus \{MutL_i\}$  ▷ Delete the exam from the exam list of its current room.
11:     $R_{seats} \leftarrow R_{seats}+$ capacity($MutL_i$)   ▷ Recalculate the available seats of its current room.
12:     if is_exclusive($MutL_i$) $= true$ then   ▷ If the exam has room exclusive constraint.
13:         $R_{exclusive} \leftarrow$ false      ▷ To remake the room available other exam(s).
14:     end if
15:     $Size \leftarrow Size - 1$                ▷ Move to the next exam at $MutL$.
16: end while
17: return $Sol$
```

All the proposed mutation operators invoke a method named *Rescheduling*, which is responsible for making random changes to the period and room for all the exams in a given *MutationList*. The method, in turn, calls the *Assignment* algorithm that is in charge of assigning a period and a room for each exam in the *MutationList*. If all the exams in the given *MutationList* are rescheduled, the Rescheduling algorithm returns/becomes true; otherwise, it returns false. The pseudo-codes of *Assignment* and *Rescheduling* is illustrated in Algorithms 4.3 and 4.4, respectively.

Algorithm 4.3 Pseudo-code of Assignment.

```
Require: $Sol$                                   ▷ A selected solution (timetable) to be mutated.
Require: $temp$                                  ▷ A temporary list with all periods.
Require: $eList$                                 ▷ A list containing the exam to be mutated.
 1: while $temp \neq \emptyset$ do
 2:     $p \leftarrow$ random_element($temp$)              ▷ Select a period from $temp$ randomly.
 3:     $R_p \leftarrow$ getAllRooms_in_period($p$)        ▷ A list of rooms available in period $p$.
 4:     $eList_{size} \leftarrow$ size($eList$)            ▷ Number of exams in $eList$.
 5:     $Size \leftarrow 0$                               ▷ Counter is set to zero.
 6:     while $Size \leq eList_{size}$ do
 7:         $R_p \leftarrow$ ascend_sort($R_p$) ▷ Sort the rooms of period $p$ based on recent capacity.
 8:         isScheduled $\leftarrow$ false
 9:         for all rooms $R_p$ do
10:             if $eList_i$ can be scheduled into $R_{pj}$ of $p$ satisfying the hard constraints then
11:                 $Sol \leftarrow$ schedule($Sol, eList_i, p, R_{pj}$) ▷ Schedule $eList_i$ at $R_j$ of period $p$.
12:                 isScheduled $\leftarrow$ true
13:                 $Size \leftarrow Size - 1$            ▷ Move to the next exam at $eList$.
14:                 Break                              ▷ The current exam is now scheduled.
15:             end if
16:         end for
17:         if isScheduled $\leftarrow$ false then        ▷ Unsuccessful scheduling of $elist$.
18:             $temp \leftarrow temp \setminus \{p\}$         ▷ Remove the period from $temp$ list.
19:         end if
20:     end while
21:     if $temp \neq \emptyset$ then
22:         return false                               ▷ Unsuccessful scheduling of $MutL$.
23:     end if
24: end while
25: return true                                        ▷ Successful scheduling of $MutL$.
```

Algorithm 4.4 Pseudo-code of the Rescheduling.

```
Require: $Sol$                          ▷ A selected solution (timetable) to be mutated.
Require: $MutL$                         ▷ A list of exams to be mutated.
Require: $Offs$                         ▷ A list containing new solutions by applying this mutation.
 1: $PL \leftarrow$ get_AllPeriods($Sol$)        ▷ A list of all periods in the solution (timetable).
 2: while $MutL \neq \emptyset$ do
 3:     $eList \leftarrow \emptyset$                          ▷ Initial empty selected exams list.
 4:     $eList \leftarrow$ random_exam($MutL$)       ▷ Add a random exam from MutL to $eList$.
 5:     $eList \leftarrow$ check_coincidence($eList$)       ▷ See Algorithm 4.1
 6:     $temp \leftarrow \emptyset$                          ▷ Empty Temporary Periods list.
 7:     $temp \leftarrow PL$                         ▷ Add all periods to Temporary Periods list.
 8:     isScheduled $\leftarrow$ false
 9:     if assignment($Sol, temp, eList$) $\leftarrow$ ture then       ▷ See Algorithm 4.3
10:         $MutL \leftarrow MutL \setminus \{eList\}$               ▷ Remove exam from List.
11:         isScheduled $\leftarrow$ true
12:     else
13:         Break                              ▷ The current exam is now scheduled.
14:     end if
15: end while
16: if isScheduled $\leftarrow$ true then                ▷ Unsuccessful scheduling of $elist$.
17:     $Offs \leftarrow Offs \cup Sol$      ▷ Add the successful mutated solution to the offspring list.
18: end if
19: return $Offs$                            ▷ Successful scheduling of $MutL$.
```

4.4.2.1 Moderate Mutation

The moderate mutation (MM) operator is applied to half of the solutions in the population, with each solution being selected using the roulette wheel selection operator based on its cost value. The moderate mutation size is determined by Equation (4.2), i.e. the number of genes examinations (genes) that will be mutated, is a random number in the space [1, *Max*], with *Max* given from Equation (4.1):

$$Max = \lceil \sqrt[5]{np * ne} \rceil \tag{4.1}$$

$$Size_{moderate} = random(Max) + 1 \tag{4.2}$$

Where *np* is the number of periods in the timetable, and *ne* is the number of exams in the timetable. When the number of periods and exams in a given schedule increases, the size of the solution space expands exponentially. As a result, these numbers are taken into account in order to make the mutation size relevant to the domain size. Therefore, moderate mutations might blend recombinative evolutionary algorithms' explorative search ability with local search methods' exploitive search ability, and the final solution can be located in fewer generations. The mutation size tuning experiments were carried out on a number of instances with diverse characteristics from all datasets. The 5th root was chosen in order to keep the maximum number of potential mutations to a manageable level. In addition, we noticed through experiments that determining the 5th root can produce a reasonable maximum size of the MM on small and large problem instances. For instance, pur83 instance is the largest problem instance of the Toronto benchmark, where it has 2419 students and 42 periods. By applying the 5th root after multiplying 2419 (i.e. the number of students) by 42 (i.e.the number of periods), the maximum mutation size of MM is 11 in this problem, which is considered the largest size would be obtained for the MM among the considered datasets in this chapter. However, this size is reasonably appropriate to concentrate the search in a localised region and seek to discover new space. Therefore, the mutation is advantageous as intensification and diversification are balanced, new spaces are discovered, and a moderate degree of genetic variation is produced, i.e. a reasonable diversity is generated in the population. The pseudo-code of the proposed moderate mutation is given in Algorithm 4.5.

Algorithm 4.5 Pseudo-code of the proposed MM mutation.

```
Require: $Pop$                                                  ▷ Current population.
 1: $Offs \leftarrow \emptyset$          ▷ A list consists of new solutions by applying this mutation.
 2: $Size \leftarrow$ size($Pop$)/2                   ▷ Number of solutions to be mutated.
 3: while $Size \neq \emptyset$ do
 4:     $T \leftarrow$ roulette_wheel_selection($Pop$)     ▷ Select an solution (timetable) to be mutated.
 5:     $Sol \leftarrow$ copy($T$)                 ▷ Create a solution copy under consideration.
 6:     $np \leftarrow$ getNumber_periods($Sol$)                    ▷ Number of periods of $Sol$.
 7:     $ne \leftarrow$ getNumber_exams($Sol$)                        ▷ Number of exams of $Sol$.
 8:     $MutL \leftarrow \emptyset$                          ▷ Empty the list exams of mutation.
 9:     $Max \leftarrow \lfloor \sqrt[5]{np * ne} \rfloor$                ▷ Maximum number reached by random.
10:     $Mut_{size} \leftarrow$ random($Max$) + 1                   ▷ Determine the size of mutation.
11:     while $Mut_{size} \neq 0$ do
12:         $eList \leftarrow$ random_exam($Sol$)           ▷ Get a random exam from the solution.
13:         $MutL \leftarrow MutL \cup$ check_coincidence($eList$)             ▷ See Algorithm 4.1
14:         $Mut_{size} \leftarrow Mut_{size} - 1$
15:     end while                              ▷ Terminate adding exams to be mutated.
16:     $Sol \leftarrow$ rollback($Sol, MutL$)                               ▷ See Algorithm 4.2
17:     $Offs \leftarrow Offs \cup$ rescheduling($Sol, MutL, Offs$)               ▷ See Algorithm 4.4
18:     $Size \leftarrow Size - 1$
19: end while
20: return $Offs$
```

4.4.2.2 Period Based Mutation

The existence of a kind of mutation that makes sudden and large jumps in the solution space becomes necessary. This goal can be achieved with Period-Based Mutation (PBM). It can also help with a broad search for suitable locations. The search can be directed towards areas of the solution space with a feasible timetable using this mutation operator. It randomly selects several periods of any selected solution and attempts to reschedule each exam in any selected period into another period to satisfy all the hard constraints. The PBM aims to mutate all the solutions in the current population with a probability that is inversely equivalent to their cost value. In order to calculate the selection probability of each solution, the PBM first ranks all the solutions in the current population ascendingly based on their cost value from best to worst by setting an index of sr for each solution, where the best cost value (i.e. the lowest cost value) is $sr = 1$, and the worst is $sr =$ population size. Next, it calculates the selection probability value sp using the formula in Equation (4.3).

$$sp = \frac{sr}{Popsize} \tag{4.3}$$

Where sr is the solution rank in the population and $Popsize$ is the population size. The pseudo-code of the proposed PBM is given in Algorithm 4.6. After assigning the probability of selection for each solution, the PBM generates a random number between 0 and 1 for each solution. If its selection probability is larger than the generated number, this solution is selected to be mutated; otherwise, it is ignored. According to Equation (4.3), the lower the rank is, the more likely it is mutated. Since this mutation creates a high degree of genetic variation, it is preferable for the best solutions in the current population to have a lower probability of being mutated by this mutation operator. The number of periods np to be disturbed is determined by Equation (4.4).

$$mp = random(\lceil \sqrt[4]{np} \rceil) + 1 \tag{4.4}$$

The above equation was determined empirically, where np is the number of all the periods in the given timetable (solution), and mp is the number of periods to be mutated. As previously stated, the more periods in a timetable, the larger the search space. So, the mutation size must be suitable and applicable to any timetable, in which an acceptable balance between solution quality and computing time is taken into account. The 4^{th} root was chosen in order to limit the number of possible mutations to a reasonable quantity and make more sudden jumps in the solution space while maintaining an appropriate balance between solution quality and processing time.

Algorithm 4.6 Pseudo-code of the proposed PBM mutation.

```
Require: $Pop$                                                  ▷ Current population.
 1: $Offs \leftarrow \emptyset$          ▷ A list consists of new solutions by applying this mutation.
 2: $Size \leftarrow$ size($Pop$)                               ▷ Number of solutions in the population.
 3: $SL \leftarrow$ ascend_rank($Pop$)                          ▷ Solutions List ordered by their cost values.
 4: $rank \leftarrow 1$
 5: while $rank \leq Size$ do
 6:     $sp \leftarrow rank/Size$                              ▷ Calculate the selection probability of $SL$.
 7:     if $sp >$ random(1) then                               ▷ random(1) returns a value between 0 and 1.
 8:         $Sol \leftarrow$ copy($SL_{rank}$)                 ▷ Create a solution copy under consideration.
 9:         $np \leftarrow$ getNumber_periods($Sol$)           ▷ Number of periods of $Sol$.
10:         $MutL \leftarrow \emptyset$                        ▷ Empty the list exams of mutation.
11:         $Mut_{size} \leftarrow$ random($|\sqrt[4]{np}| + 1$)  ▷ Determine the size of mutation.
12:         while $Mut_{size} \neq 0$ do
13:             $p \leftarrow$ random_period($Sol$)            ▷ Get a random period from $Sol$.
14:             $MutL \leftarrow MutL \cup \{p_{exams}\}$      ▷ Add all exams in the period to the MutationList.
15:             $Sol \leftarrow$ rollback($Sol, MutL$)         ▷ See Algorithm 4.2
16:             $Mut_{size} \leftarrow Mut_{size} - 1$
17:         end while                                          ▷ Terminate adding exams to be mutated.
18:         $Offs \leftarrow Offs \cup$ rescheduling($Sol, MutL, Offs$)   ▷ See Algorithm 4.4
19:     end if
20:     $rank \leftarrow rank + 1$
21: end while
22: return $Offs$
```

4.4.2.3 Deep Mutation

Deep Mutation (DM) is considered to be a violation directed strategy (guided mutation), which aims to change the period and room of most of the exams that cause violations to the soft constraints. In other words, the DM calculates soft constraint violations (penalty values) using Equation (4.5) for each exam in a given timetable (solution) depending on the period and room of the exam is currently scheduled in. Next, the DM ranks all the exams in the solution based on their penalty value with most of the exams that cause violations having a rank value of 1. The DM is, therefore, biased toward working on the exams with higher ranks (the greatest violations). The aim of this ranking is to define a range (starting from the first exam in the rank until a predefined rank value, which is determined experimentally by Equation (4.6)). Therefore, the number of exams ne that can be selected and be mutated is determined randomly between lower limit (i.e. can be 1 as the least number of exams to be mutated) and upper limit (i.e. the largest number of exams to be mutated is $ne = random(Upper\_limit) + 1$) of this range. The aim of this ranking and range is to force the mutation to take place with this range of exams, in order to rearrange these exams into better periods to produce a good-quality timetable (solution).

$$c = \sum_{i=1}^{k} w_i * pe_i \tag{4.5}$$

Where c is the soft constraints cost of a given exam, pe_i is a penalty value imposed to the violation of a specific constraint, and w_i an attached weight.

$$Upper\_limit = \lceil \sqrt[5]{ne} \rceil + random(\lceil \sqrt[4]{ne} \rceil) \tag{4.6}$$

Where ne is the number of exams in a given timetable. Lastly, the selection probability sp of each solution can be calculated by means of Equation (4.7). The high fitness solutions possess a high probability of being selected.

$$sp = \frac{Popszie - sr}{Popsize} \tag{4.7}$$

DM is also biased toward working on the solutions of the highest (i.e. worst) fitness value in the population. This mutation aims to equip the ESGA with an additional exploitative search. This means that better solutions are more likely to be obtained. The pseudo-code of the proposed DM mutation is given in Algorithm 4.7.

Algorithm 4.7 Pseudo-code of the proposed DM mutation.

```
Require: $Pop$                                                    ▷ Current population.
1: $Offs \leftarrow \emptyset$                  ▷ A list consists of new solutions by applying this mutation.
2: $Size \leftarrow$ size($Pop$)                                  ▷ Number of solutions in the population.
3: $SL \leftarrow$ ascend_rank($Pop$)                             ▷ Solution List ordered by their cost values.
4: $rank \leftarrow 1$
5: while $rank \leq Size$ do
6:     $sp \leftarrow (Size - rank)/Size$                         ▷ Calculate the selection probability of $SL$.
7:     if $sp >$ random(1) then                                   ▷ random(1) returns a value between 0 and 1.
8:         $Sol \leftarrow$ copy($SL_{rank}$)                     ▷ Create a solution copy under consideration.
9:         $Sol \leftarrow$ descend_rank_exams($Sol$)             ▷ Rank all exams descending by their penalties.
10:        $np \leftarrow$ getNumber_periods($Sol$)               ▷ Number of periods of $Sol$.
11:        $ne \leftarrow$ getNumber_exams($Sol$)                 ▷ Number of exams of $Sol$.
12:        $Range \leftarrow |\sqrt[5]{ne}| +$ random($|\sqrt[4]{ne}|$)   ▷ Range is determined by the exams no.
13:        $MutL \leftarrow \emptyset$                            ▷ Empty the list exams of mutation.
14:        $Mut_{size} \leftarrow |\sqrt[5]{np}| +$ random($|\sqrt[4]{np}|$)   ▷ Determine the size of mutation.
15:        while $Mut_{size} \neq 0$ do
16:            $eList \leftarrow$ random_exam_in_range($Sol$)     ▷ A random exam in the provided range.
17:            $MutL \leftarrow MutL \cup$ check_coincidence($eList$)   ▷ See Algorithm 4.1
18:            $Mut_{size} \leftarrow Mut_{size} - 1$
19:        end while                                              ▷ Terminate adding exams to be mutated.
20:        $Sol \leftarrow$ rollback($Sol, MutL$)                 ▷ See Algorithm 4.2
21:        $Offs \leftarrow Offs \cup$ rescheduling($Sol, MutL, Offs$)   ▷ See Algorithm 4.4
22:    end if
23:    $rank \leftarrow rank + 1$
24: end while
25: return $Offs$
```

4.4.2.4 Period Based Deep Mutation

Period Based Deep Mutation (PBDM) operates in the same way as DM in terms of ranking, selecting the range, and the solution selection probability with a minor difference. The PBDM mutation is forced to take place on whole periods. The periods are ranked based on the extent to which these periods violate the soft constraints. For any period, the PBDM calculates the penalty value of each exam that is scheduled in that period. Next, it sums all these exams' penalty values and assigns them to this period. This process is repeated for all the periods in the solution. A selecting rank is defined using Equation (4.8) and this ranking aims to define a range (beginning with the first period in the rank and ending with a predetermined rank value, as established experimentally by Equation (4.8)). As a result, the number of periods *np* that can be chosen and mutated is determined randomly between the lower limit (i.e. 1, the smallest number of periods to be mutated) and the upper limit (i.e. the greatest number of periods to be mutated).

$$Upper\_limit = \lceil \sqrt[5]{np} \rceil + random(\lceil \sqrt[4]{np} \rceil) \tag{4.8}$$

Where np is the number of periods in a specific timetable. All the exams in the selected periods are removed and added to the MutationList.

Algorithm 4.8 Pseudo-code of the proposed PBDM mutation.

```
Require: $Pop$                                          ▷ Current population.
 1: $Offs \leftarrow \emptyset$          ▷ A list consists of new solutions by applying this mutation.
 2: $Size \leftarrow$ size($Pop$)                      ▷ Number of solutions in the population.
 3: $SL \leftarrow$ ascend_rank($Pop$)             ▷ Solution List ordered by their cost values.
 4: $rank \leftarrow 1$
 5: while $rank \leq Size$ do
 6:     $sp \leftarrow (Size - rank)/Size$          ▷ Calculate the selection probability of $SL$.
 7:     if $sp >$ random(1) then              ▷ random(1) returns a value between 0 and 1.
 8:         $Sol \leftarrow$ copy($SL_{rank}$)          ▷ Create a solution copy under consideration.
 9:         $Sol \leftarrow$ descend_rank_periods($Sol$)          ▷ Rank all periods in the solution
    descending by their penalties.
10:         $np \leftarrow$ getNumber_periods($Sol$)                 ▷ Number of periods of $Sol$.
11:         $Range \leftarrow |\sqrt[5]{np}| +$ random($|\sqrt[4]{np}|$)   ▷ Range is determined by the periods no.
12:         $MutL \leftarrow \emptyset$                          ▷ Empty the list exams of mutation.
13:         $Mut_{size} \leftarrow |\sqrt[5]{np}| +$ random($|\sqrt[4]{np}|$)       ▷ Determine the size of mutation.
14:         while $Mut_{size} \neq 0$ do
15:             $p \leftarrow$ random_period_in_range($Sol$)     ▷ A random period in the provided
    range.
16:             $MutL \leftarrow MutL \cup \{p_{exams}\}$▷ Add all exams in the period to MutationList.
17:             $Mut_{size} \leftarrow Mut_{size} - 1$
18:         end while                      ▷ Terminate adding periods to be mutated.
19:         $Sol \leftarrow$ rollback($Sol, MutL$)                          ▷ See Algorithm 4.2
20:         $Offs \leftarrow Offs \cup$ rescheduling($Sol, MutL, Offs$)            ▷ See Algorithm 4.4
21:     end if
22:     $rank \leftarrow rank + 1$
23: end while
24: return $Offs$
```

4.4.2.5 Heavy Mutation

Heavy mutation (HM) is similar to the MM mutation in most aspects except for the mutation size, which is larger, based on Equation (4.9); and the selection probability *sp* of the solutions, based on Equation (4.10), which we empirically found that the proper size of this mutation and the selection probability that can increase the likelihood of selecting worse solutions as opposed to better solutions. By improving the solutions with the worse fitness, the HM endeavours to maintain the population diversity. The pseudo-code of the proposed HM mutation is given in Algorithm 4.9.

$$Size_{heavy} = \lceil \sqrt[5]{np * ne} \rceil + 1 \tag{4.9}$$

$$sp = \frac{sr}{Popsize} \tag{4.10}$$

Algorithm 4.9 Pseudo-code of the proposed HM mutation.

```
Require: $Pop$                                              ▷ Current population.
 1: $Offs \leftarrow \emptyset$      ▷ A list consists of new solutions by applying this mutation.
 2: $Size \leftarrow$ size($Pop$)                      ▷ Number of solutions in the population.
 3: $SL \leftarrow$ ascend_rank($Pop$)            ▷ Solution List ordered by their cost values.
 4: $rank \leftarrow 1$
 5: while $rank \leq Size$ do
 6:     $sp \leftarrow (rank/Size)$                  ▷ Calculate the selection probability of $SL$.
 7:     if $sp >$ random(1) then ▷ if $P$ is larger than the generated number($\leq 0$ and $\leq 1$),
        $SL_{rank}$ will be mutated.
 8:         $Sol \leftarrow$ copy($SL_{rank}$)          ▷ Create a solution copy under consideration.
 9:         $MutL \leftarrow \emptyset$                        ▷ Empty the list exams of mutation.
10:         $np \leftarrow$ getNumber_periods($Sol$)                ▷ Number of periods of $Sol$.
11:         $ne \leftarrow$ getNumber_exams($Sol$)                    ▷ Number of exams of $Sol$.
12:         $Mut_{size} \leftarrow |\sqrt[5]{np * ne}| +$ random($|\sqrt[5]{np * ne}|$)    ▷ Select the size of mutation
        randomly based on numbers of periods and exams.
13:         while $Mut_{size} \neq 0$ do
14:             $eList \leftarrow$ random_exam($Sol$)               ▷ Get a random exam from $Sol$.
15:             $MutL \leftarrow MutL \cup$ check_coincidence($eList$)          ▷ See Algorithm 4.1
16:             $Mut_{size} \leftarrow Mut_{size} - 1$
17:         end while      ▷ Terminate adding exams to be mutated in the MutationList.
18:         $Sol \leftarrow$ rollback($Sol, MutL$)                                  ▷ See Algorithm 4.2
19:         $Offs \leftarrow Offs \cup$ rescheduling($Sol, MutL, Offs$)                ▷ See Algorithm 4.4
20:     end if
21:     $rank \leftarrow rank + 1$
22: end while
23: return $Offs$
```

The mutation operators applied in this work complement each other. In other words, each one has a distinct purpose and is applied to specific solutions to enable effective and efficient exploration and exploitation, particularly in the absence of a crossover operator. The proposed mutation operators are applied to each generation in the following order (Moderate, Period Based, Deep, Period Based Deep, Heavy, Light, and Elite Light). All offspring yielded by applying each one is added to a pool, as depicted in Equation (4.6). Therefore, some solutions may be selected many times to be mutated by different mutation operators and each time, a different one can be produced.

4.4.2.6 Light Mutation

In Light Mutation (LM), a small number of exams N are randomly removed from the timetable, as empirically given in Equation (4.11). Next, the removed exams are rescheduled to legal periods. In contrast to HM, LM aims to exploit better quality solutions. Thus, LM increases the likelihood of selection solutions with better fitness values based on Equation (4.7). It seeks to enhance the neighbourhood search utilising one or more local changes in the current solution. It also aims to equip the ESGA with reasonable exploitation by means of creating at least slightly better-quality solutions. The pseudo-code of the proposed LM mutation is given in Algorithm 4.10.

$$Size_{light} = random(3) + 1 \tag{4.11}$$

Algorithm 4.10 Pseudo-code of the proposed LM mutation.

```
Require: $Pop$                                              ▷ Current population.
 1: $Offs \leftarrow \emptyset$                 ▷ A list consists of new solutions by applying this mutation.
 2: $Size \leftarrow$ size($Pop$)                     ▷ Number of solutions in the population.
 3: $SL \leftarrow$ ascend_rank($Pop$)                ▷ Solution List ordered by their cost values.
 4: $rank \leftarrow 1$
 5: while $rank \leq Size$ do
 6:     $sp \leftarrow (Size - rank/Size)$             ▷ Calculate the selection probability of $SL$.
 7:     if $sp >$ random(1) then                   ▷ Random(1) returns a value between 0 and 1.
 8:         $Sol \leftarrow$ copy($SL_{rank}$)             ▷ Create a solution copy under consideration.
 9:         $MutL \leftarrow \emptyset$                          ▷ Empty the list exams of mutation.
10:         $Mut_{size} \leftarrow$ (random(3) + 1)       ▷ The mutation size will be within 1 and 4.
11:         while $Mut_{size} \neq 0$ do
12:             $eList \leftarrow$ random_exam($Sol$)              ▷ Get a random exam from $Sol$.
13:             $MutL \leftarrow MutL \cup$ check_coincidence($eList$)          ▷ See Algorithm 4.1
14:             $Mut_{size} \leftarrow Mut_{size} - 1$
15:         end while                            ▷ Terminate adding exams to be mutated.
16:         $Sol \leftarrow$ rollback($Sol, MutL$)                         ▷ See Algorithm 4.2
17:         $Offs \leftarrow Offs \cup$ rescheduling($Sol, MutL, Offs$)       ▷ See Algorithm 4.4
18:     end if
19:     $rank \leftarrow rank + 1$
20: end while
21: return $Offs$
```

4.4.2.7 Elite Light Mutation

Elite Light Mutation (ELM) is similar to the LM Mutation, but attempts to improve higher quality solutions iteratively. Consequently, the ELM is performed on the best solutions in the current population, with the amount of elitism being equal to half the population size. Thus, with sixteen solutions in total the amount of elitism is eight solutions (highly fit) to deal with the absence of a crossover operator. The selection probability sp of the elite solutions is given in Equation (4.7). For the diversity to be reasonable in the population, the ELM will be run many times for each of the solutions in the elite selection. rn is given in Equation (4.12). The pseudo-code of the proposed ELM mutation is given in Algorithm 4.11.

$$rn = \lceil Popsize/4 \rceil \tag{4.12}$$

Algorithm 4.11 Pseudo-code of the proposed ELM mutation.

```
Require: $Pop$                                          ▷ Current population.
 1: $Offs \leftarrow \emptyset$          ▷ A list consists of new solutions by applying this mutation.
 2: $Elite \leftarrow$ select_best_solution($Pop$) ▷ Select the best 8 solutions in this population.
 3: $Size \leftarrow$ size($Pop$)                 ▷ Number of solutions in the population.
 4: $SL \leftarrow$ ascend_rank($Elite$)           ▷ Solution List ordered by their cost values.
 5: $Elite_{size} \leftarrow$ size($Elite$)                ▷ Number of solutions in the $Elite$.
 6: $rank \leftarrow 1$
 7: $recurring \leftarrow (Size/4)$                         ▷ Determine recurring mutation.
 8: while $rank \leq Elite_{size}$ do
 9:     $sp \leftarrow (Size - rank)/Size$          ▷ Calculate the selection probability of $SL$.
10:     while $recurring \neq 0$ do
11:         if $sp >$ random(1) then          ▷ Random(1) returns a value between 0 and 1.
12:             $Sol \leftarrow$ copy($SL_{rank}$)      ▷ Create a solution copy under consideration.
13:             $MutL \leftarrow \emptyset$                       ▷ Empty the list exams of mutation.
14:             $Mut_{size} \leftarrow$ (random(3) + 1)    ▷ The mutation size will be within 1 and 4.
15:             while $Mut_{size} \neq 0$ do
16:                 $eList \leftarrow$ random_exam($Sol$)          ▷ Get a random exam from $Sol$.
17:                 $MutL \leftarrow MutL \cup$ check_coincidence($eList$)      ▷ See Algorithm 4.1
18:                 $Mut_{size} \leftarrow Mut_{size} - 1$
19:             end while                     ▷ Terminate adding exams to be mutated.
20:             $Sol \leftarrow$ rollback($Sol, MutL$)                       ▷ See Algorithm 4.2
21:             $Offs \leftarrow Offs \cup$ rescheduling($Sol, MutL, Offs$)      ▷ See Algorithm 4.4
22:         end if
23:     end while
24:     $rank \leftarrow rank + 1$
25: end while
26: return $Offs$
```

4.5 Enhanced Roulette Wheel Selection Strategy

Many works found in the literature have been employed conventional and modified roulette wheel selection methods within GA (e.g. Al Jadaan et al. (2008) and Kumar et al. (2012)). Al Jadaan et al. (2008) proposed a ranked based roulette wheel selection. Each solution (i.e. individual) was assigned a rank based on its fitness value in which the highest rank has a high chance to be selected. However, this selection mechanism cannot ensure diversity (i.e. lack of solution diversity) and prevent premature convergence. Thus, in this thesis, the proposed selection strategy, which we name **Enhanced Roulette Wheel (ERW)** selection strategy, uses the "id" for each solution assigned in the first phase by the proposed ESGA to disallow similar individuals from joining the population.

Once the ESGA has applied all the mutation operators, the offspring pool will contain all the new solutions. The cost value of these might be better or might be worse than their parents' cost values, and the total of solutions can exceed the original size of the population. The proposed selection strategy employs the roulette wheel to build the next generation but if the roulette wheel selection operator is used without these ids, one exact copy of each solution will be produced in the next generation. In the proposed solution, therefore, the roulette wheel is invoked many times based on the population size; when any solution is selected to be added to the pool of the next generation, it is added to the pool of the next generation and its id retrieved in order to remove all the remaining solutions in the offspring pool that have the same id to stop duplicate solutions and thus accelerate the selection process for the existing solutions.

4.6 Proposed Exam Specialised Genetic Algorithm

All the steps and processes that have been described above can be put together in an algorithm. The proposed algorithm first generates 50 initial solutions using the OBSI method, then it selects the best sixteen of these solutions based on their cost values. A fairly small population is selected because of the relatively high computing cost to generate feasible solutions, particularly when the more constrained ITC 2007 datasets are used. Next, the proposed method assigns different "ids" for each solution, to be used later by the enhanced roulette wheel selection. Then, all the mutation operators, as well as the selection of the population are performed for several generations until a termination criterion is fulfilled.

After all the mutation operators have been applied, the offspring pool will contain all the new solutions. Next, all the solutions in the offspring pool are ranked by the proposed algorithm based on cost values. Then, the best 50% are selected and added to the next population (i.e. half of the next population contains the best individuals from the previous population). The remaining solutions in the new population are selected using the improved roulette wheel selection approach. The algorithmic flow of the proposed ESGA is shown in Figure 4.5, while the pseudo-code is given in Algorithm 4.12.

Algorithm 4.12 Pseudo-code of the Exam Specialised Genetic Algorithm (ESGA).

```
1: $Pop \leftarrow$ initialise_50_solutions_by_OBSI        ▷ See Algorithms 3.1, 3.2, and 3.3
2: evaluate($Pop$)                                          ▷ Evaluate population.
3: $Pop \leftarrow$ select_16_solutions($Pop$)              ▷ Population list.
4: $Pop \leftarrow$ assign_IDs($Pop$)       ▷ Assign unique ID to each solution (0 until 15).
5: $Offs \leftarrow \emptyset$        ▷ A list contains the solutions of the next population.
6: $Elitism \leftarrow \emptyset$   ▷ A list contains the best 8 solution of the current population.
7: $t \leftarrow 0$                                          ▷ Iteration number.
8: while (Stopping condition is not achieved) do
9:     $Offs \leftarrow Offs \cup$ MM($Pop$)           ▷ Add MM solutions, See Algorithm 4.5
10:    $Offs \leftarrow Offs \cup$ PM($Pop$)           ▷ Add PM solutions, See Algorithm 4.6
11:    $Offs \leftarrow Offs \cup$DM($Pop$)            ▷ Add DM solutions, See Algorithm 4.7
12:    $Offs \leftarrow Offs \cup$PBDM($Pop$)          ▷ Add PBDM solutions, See Algorithm 4.8
13:    $Offs \leftarrow Offs \cup$ HM($Pop$)           ▷ Add HM solutions, See Algorithm 4.9
14:    $Offs \leftarrow Offs \cup$ LM($Pop$)           ▷ Add LM solutions, See Algorithm 4.10
15:    $Offs \leftarrow Offs \cup$ ELM($Pop$)          ▷ Add ELM solutions, See Algorithm 4.11
16:    evaluate($Offs$)                                ▷ Evaluate all Offspring.
17:    $Offs \leftarrow Offs \cup Pop$          ▷ Add the current population to Offspring list.
18:    $Elitism \leftarrow$ select_best_8($Offs$)      ▷ Select the best 8 solutions.
19:    if (Stopping condition is achieved) then
20:        Output the best exam timetables
21:        Break                   ▷ Terminate the process if the condition is reached.
22:    end if
23:    $Pop \leftarrow \emptyset$                      ▷ Empty the Population list.
24:    $Pop \leftarrow Elitism \cup$ solutions_by_ERW($Offs$)    ▷ Next population.
25:    $Elitism \leftarrow \emptyset$                  ▷ Empty the Elitism list.
26:    $Offs \leftarrow \emptyset$                     ▷ Empty the Offspring list.
27:    $t \leftarrow t + 1$                            ▷ Next iteration
28: end while
```

Crossover operator is not utilised in this work, despite being regarded as a key aspect of GAs. This is because it can change the parents of a high fitness function to the extent that they no longer spatially fit in the more constrained problem (Osaba et al., 2013). Regarding the runtime and based on the experimentation conducted, using blind crossover operators in GAs significantly increases the technique's execution time without any improvements in the results (Osaba et al., 2013). Meanwhile, a function that makes small leaps in the solution space is required for a more thorough search. This goal can be accomplished with the mutation function, and it can also aid in the search for potential regions (Eiben and Schippers, 1998; Wong et al., 2003). Since each generation generates a vast number of solutions by applying numerous mutations, the best approaches in terms of efficiency and efficacy, such as Binary Search for searching elements (Knuth, 1973) and Quicksort for sorting elements (Cormen et al., 2009) are included in the evolution function to minimise the execution time to estimate the fitness of each offspring.

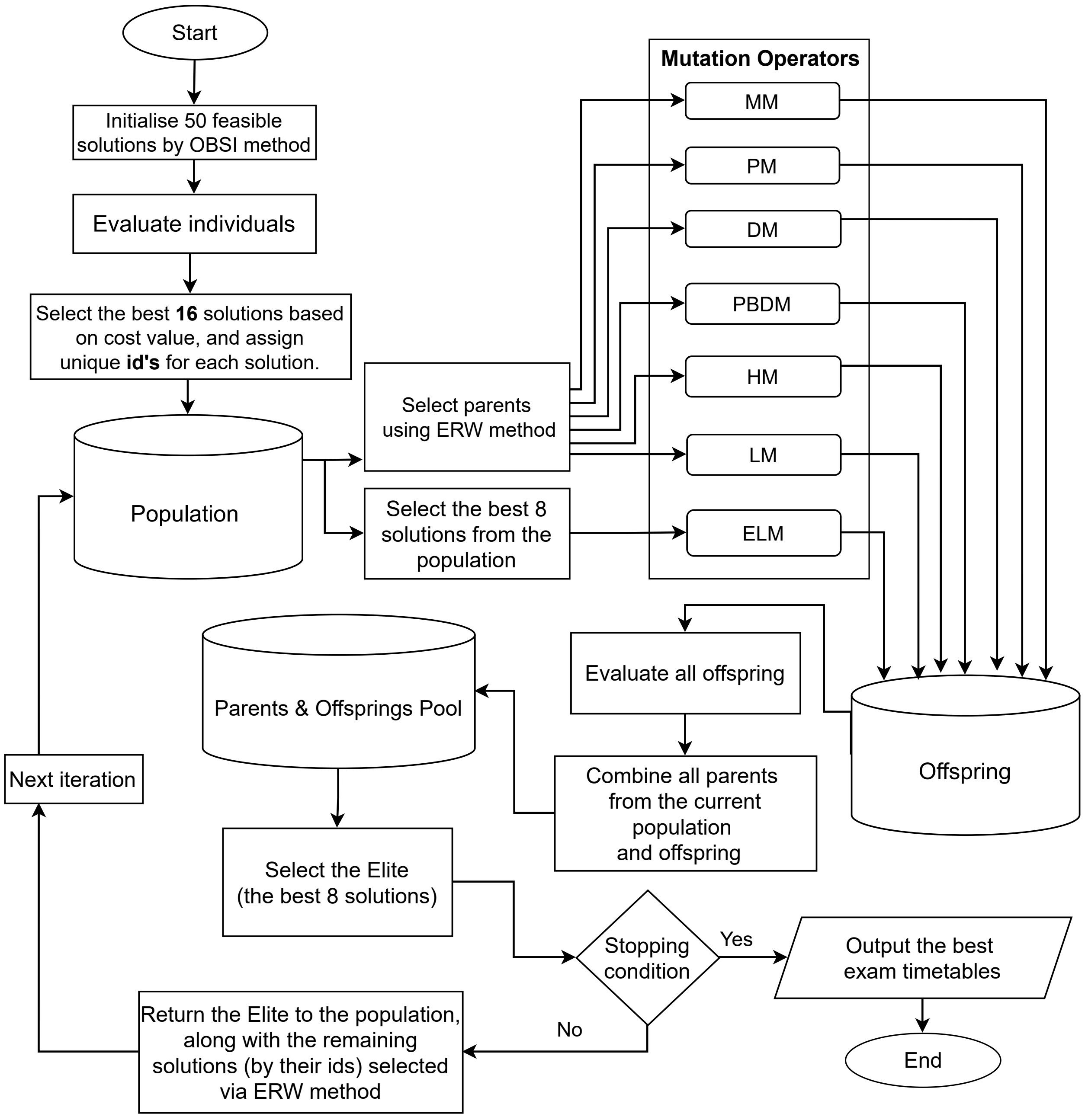
Start
Initialise 50 feasible solutions by OBSI method
Evaluate individuals
Select the best 16 solutions based on cost value, and assign unique id's for each solution.
Population
Select parents using ERW method
Select the best 8 solutions from the population
Mutation Operators
MM
PM
DM
PBDM
HM
LM
ELM
Offspring
Evaluate all offspring
Combine all parents from the current population and offspring
Parents & Offsprings Pool
Select the Elite (the best 8 solutions)
Stopping condition
Yes
No
Output the best exam timetables
End
Return the Elite to the population, along with the remaining solutions (by their ids) selected via ERW method
Next iteration

Figure 4.5 Flowchart of the proposed ESGA.

4.7 Experimental Results and Discussion

This section illustrates the experimental results of the proposed approach on the Toronto, ITC 2007, and Yeditepe benchmark datasets. Experimental environment and parameter settings are given in Sections 4.7.1 and 4.7.2, respectively. Section 4.7.3 presents the best-known approaches reported in the literature for solving the Toronto, the ITC 2007, and Yeditepe datasets. The results are presented in Sections 4.7.4 and 4.7.5 below. In section 4.7.4, a comparison is also conducted between different versions of the GA using the same three datasets. In Section 4.7.5, meanwhile, the results are compared with those achieved by the top five performing algorithms in the ITC 2007 competition, based on the ITC 2007 datasets. Also, in Section 4.7.5 a comparison is conducted with the results achieved by a selection of state-of-the-art methods on the Toronto, the ITC 2007 and Yeditepe. The characteristics of each of these datasets have been discussed in Section 2.3 of Chapter 2.

4.7.1 Experimental Environment

The algorithm proposed in this study was applied by utilising the Java programming language. The simulations were completed a PC running Windows 7 Enterprise with a 64-bit operating system and an Intel Core (TN) i5-6200U (CPU@2.30 GHz with 16 GB RAM).

4.7.2 Parameter Settings

Several preliminary experiments were conducted so that the most appropriate values could be obtained for the number of iterations, the size of the population, the size of the elite pool, the probability of selecting solutions for each mutation operator, and the extent of mutation for each mutation operator. The experiments were based on the average result from a total of thirty runs on ITC 2007 and Yeditepe where the algorithm running time was limited to 392 seconds per run (i.e. in accordance with the rules of the ITC 2007 competition).

For the Toronto problem instances, a comparison was conducted between a basic GA and the ESGA where each algorithm was run thirty times, with a set number of generations acting as a termination criterion for each run. The algorithm performed best with 1,000 iterations, the population size of sixteen and an elite pool of eight. The values of the remaining parameters varied based on the solution cost value and the type of mutation operator (see Section 4.4.2). In the ESGA comparison with the state-of-the-art approaches, the algorithm was capped at 10800 seconds and run ten times on each problem instance as the execution time of all approaches considered was not enforced by a fix time limit (Leite et al., 2018). This is because there is no such benchmark timing (as this case for the ITC 2007 problem) to attempt on the Toronto problems.

4.7.3 Reported ‘best’ Results of the Benchmark Sets

This section analyses and exhibits the results of applying the approach to Toronto benchmark datasets. The best results obtained from the ESGA approach are taken from each problem instance in order to compare with other results reported in the literature. Authors’ name concatenated with publishing year will be used for the proposed approaches name.

4.7.3.1 Reported ‘best’ Results of the Toronto Benchmarks

CarterEtAl96 -(Carter et al., 1996): the authors introduced the Toronto Benchmark (as discussed in Section 2.5.1 in Chapter 2) and investigated two different objectives. The first objective aims to generate a feasible examination timetable in the minimum number of periods—the second objective aims to spacing out the exams for the same student. In this work, it has been incorporated sequential heuristics such as Largest Degree (LD), Largest Weighted Degree (LWD), Saturation Degree (SD), and Colour Degree (CD) with clique initialisation and a backtracking procedure to construct the solutions. The idea is that the backtracking procedure is applied when it is difficult to accommodate any exam into the periods due to earlier assigned exams (i.e. exams that have conflicts with the current examination). In addition, the size of the clique plays an effective role to determine the number of periods needed for the problem (i.e. the size of the largest clique defines the least number of periods required). The proposed approach was tested on all instances of real problems (i.e. the Toronto Benchmark sets) and random problems. The experimental results showed that sequential heuristics with backtracking were capable of reducing the number of periods required in the final examination timetable compared to these heuristics without backtracking. Furthermore, the combination of a backtracking procedure with SD was able to generate reasonably good quality examination timetables in a short computational time.

Yang&Petrovic04 -(Yang and Petrovic, 2005): a hyper-heuristic approach merged with case-based reasoning was applied in order to select graph heuristics for creating a feasible initial solution. Then, a great deluge algorithm was utilised to enhance the solution. The authors use a hyper-heuristic to explore the search space of heuristics in replacement of searching for direct solutions. Their approach was applied to the Toronto problems, and obtained the best results in the literature for several problem instances at this time.

Eley07 -(Eley, 2007): the author implemented an ant algorithm and incorporated the ant systems and the max-min ant systems with two randomised strategies in order to find the pheromone trail and the constructive heuristic. Several parameters needed to be considered during the implementation such the evaporation rate, the weighting factors and the number of cycles, in order to make sure that the proposed algorithm worked efficiently. The proposed approach was tested on the Toronto benchmark sets, and obtained results were competitive with the best published approaches.

CaramiaEtAl08 -(Caramia et al., 2008): the authors implemented a hybrid approach to tackle capacitated and uncapacitated examination timetabling problems. This study attempted to generate high-quality exam timetable with small length (i.e. the minimum number of periods). The scheduling process started with a greedy scheduler by attempting to assigning examinations into the least number of periods as well as conflict-free slots. The approach also allocates exams by placing those with the highest conflicts first in order to identify the number of periods (i.e. the length of the timetable) and ensure that all exam to be scheduled. Once the process has been completed, hill climbing employed as penalty-decreaser. It was used to minimise the number of periods and maximise the quality of timetable. The process continues until there is no more improvement. At this stage, hill climbing was applied as a penalty trader. This approach was tested on the Toronto and Nottingham benchmark sets. The results showed that this approach could easily produce high-quality solutions and were superior to many best-known published approaches in the literature.

Burke&Bykov08 -(Burke and Bykov, 2008): A late acceptance strategy was proposed by Burke and Bykov (2008), which was a new variant of hill climbing. They investigated the performance of the proposed method by applying it to uncapacitated exam timetabling problem (i.e. the Toronto benchmark sets). Although it is categorised as an iterative search method, it depends on more sophisticated move acceptance mechanism. A new candidate solution is compared to some solutions that obtain from prior iterations. A list is created and stores the cost function values of accepted solution from each iteration. During the searching iterations, a neighbouring solution with a better or equal cost value is accepted in the list.

BurkeEtAl10 -(Burke et al., 2010b): the authors proposed a variable neighbourhood search (VNS) incorporated within a Genetic Algorithm. In this work, different neighbourhood structures such as descent-ascent, biased VNS, and problem-specific neighbourhoods were investigated. Also, various initialisation strategies (i.e. greedy and a random construction technique) were applied. Although the proposed approach was capable of obtaining a high quality solution for one instance of the Toronto problem, it requires a relatively large amount of computation time.

Pillay&Banzhaf10 -(Pillay and Banzhaf, 2010): the use of genetic algorithms (GAs) is presented in Pillay and Banzhaf (2010) for solving the examination timetabling problem. The algorithm was considered as a two-phased approach. In the first phase, generating feasible solutions for constructing the initial population was conducted by GAs. During the second phase, GAs also utilised the generated feasible solutions from the previous phase and attempted to minimise the cost of the soft constraint. They used domain-specific knowledge in the form of heuristics to guide the evolutionary process. The effectiveness of this approach was verified on the Toronto benchmark sets.

DemeesterEtAl12 -(Demeester et al., 2012): the authors employed a hyper-heuristic based approach to resolve three timetabling problems, including the Toronto and the ITC 2007 problems, in addition to the KAHO Sint–Lieven (Ghent, Belgium) timetabling problem. The authors applied a construction, as well as an improvement approach. However, the construction algorithm, which is used for the ITC 2007 benchmark set, cannot guarantee feasible solutions. If no feasible solution is obtained, the algorithm continues with the improvement phase. Also, extra correcting actions were performed to eliminate infeasibilities.

Abdullah&Alzaqebah13 -(Abdullah and Alzaqebah, 2013): the authors presented a hybridisation approach which combined a modified bees algorithms with local search algorithms (i.e. simulated annealing, late acceptance hill-climbing). In order to exploit and fully explore the entire search space, they used three selection strategies (i.e. disruptive, tournament, and raking) and a self-adaptive technique. The aim of the selection strategies is to improve the diversity of the population, while the self-adaptive method was used for monitoring the neighbourhood search and preventing the algorithm from getting stuck in a local optimum. The approach was tested on thirteen problem instances of the Toronto and eight problem instances of the ITC 2007 benchmark sets.

Alzaqebah&Abdullah14 -(Alzaqebah and Abdullah, 2014): An adaptive artificial bee colony was proposed and combined with a late-acceptance hill-climbing algorithm to solve the examination timetabling problem. This proposed method was applied to Toronto, as well as the ITC 2007 benchmark sets.

Alzaqebah&Abdullah15 -(Alzaqebah and Abdullah, 2015): in this research, two hybridisations were conducted using bee colony optimisation algorithm in both of them. The first hybrid couples bee colony optimisation (BCO) algorithm with a hill-climbing algorithm by using the late acceptance strategies and the second hybrid couples BCO with simulated annealing. In comparison with others, the first hybrid achieves the best results for both datasets (the Toronto and the ITC 2007).

FongEtAl15 -(Fong et al., 2015): a hybrid swarm-based algorithm to academic timetabling is proposed. The algorithm used in different university timetabling, namely the examination timetabling and the course timetabling and tested on the Toronto and the Socha benchmark sets respectively.

LeiteEtAl16 -(Leite et al., 2016): the proposed approach implemented a memetic algorithm so-called “Shuffled Complex Evolution Algorithm” in which the population is organised into complexes (sets) that are evolved individually by using local search operators and recombination. Diversity of population was kept by using various recombination operators and applying a special mechanism to update solution.

MuklasonEtAl17 -(Muklason et al., 2017): the authors proposed a multi-phase approach in order to solve the Toronto, the ITC 2007, and the Yeditepe problems and find good solutions that could match student preferences. This approach consists of three phases. In phase 1, an initial feasible solution was produced by an adaptive heuristic ordering approach. Improving the quality of the initial solution and attempting to achieve fairness in the optimised solutions were conducted by a selection hyper-heuristic in phase 2 and phase 3, respectively.

LeiteEtAl18 -(Leite et al., 2018): in this study, a cellular memetic algorithm was proposed which incorporated a cellular evolutionary algorithm with threshold acceptance local search to tackle the examination timetabling problem. The approach was evaluated on the Toronto and ITC 2007 problems. Experimental results showed that the approach was able to improve on four out thirteen problem instances of the Toronto set and improve on three out twelve of the ITC 2007 problem.

MandalEtAl20 -(Mandal et al., 2020): the authors proposed partial graph heuristic orderings with a modified great deluge algorithm (PGH-mGD) for tackling capacitated and uncapacitated examination timetabling problems. The proposed approach used different graph heuristic ordering in the constructive phase in order to generate feasible exam timetables, while modified great deluge algorithm was utilised in phase 2 (i.e. improvement phase) to improve the obtained timetables. The proposed PGH-mGD partially timetables a set of exams that are selected based on graph heuristic orderings as well as a parameter which called “exam assignment value v”. Then, a modified GD algorithm was employed to improves these selected exams. The process of scheduling continues to allocate the next set of selected exams and repeats until all exams have been scheduled. Moreover, the proposed PGH-mGD was tested on the Toronto and the ITC 2007 benchmark sets and compared with traditional graph heuristic orderings with a modified great deluge algorithm (TGH-mGD) as well as related state-of-the-art approaches. Experimental results illustrated that the proposed PGH-mGD was capable of producing high-quality exam timetables which were competitive with those of the approaches published in the literature.

4.7.3.2 Reported 'best' Results of the ITC 2007 Benchmarks

For each track in the 2nd International Timetabling Competition (ITC 2007), a set of five finalists was chosen based on the quality of the submitted solutions (Mccollum et al., 2010; Leite et al., 2018). The five finalists of the examination track of the competition are listed below, along with more recently developed optimisers that have been applied to this problem suite.

- 1st Place - Tomáš Müller (Müller09).
- 2nd Place - Christos Gogos, Panayiotis Alefragis, and Efthymios Housos (GogosEtAl08).
- 3rd Place - Mitsunori Atsuta, Koji Nonobe, and Toshihide Ibaraki (AtsutaEtAl08).
- 4th Place - Geoffrey De Smet (De Smet08).
- 5th Place - Nelishia Pillay (Pillay08).

Müller09 -(Müller, 2008): Müller's approach was applied with the aim of solving the three problems that were established by the ITC 2007 competition. To solve the problems, he preferred a hybrid approach, which is organised in a two-phase algorithm. In the first phase, the Iterative Forward Search (IFS) algorithm (Müller, 2005) was employed, so that feasible solutions, as well as Conflict-based Statistics, are obtained (Müller et al., 2004) in order that the IFS is prevented from looping. The second phase involves using multiple optimisation algorithms. The algorithms were applied in the following order: Hill Climbing (HC)(Russell and Norvig, 2010), Great Deluge (GD) (Dueck, 1993), and optionally Simulated Annealing (SA) (Skiundefinedcim and Golden, 1983).

GogosEtAl08 -(Gogos et al., 2008): Like Müller's approach, Gogos et al.'s approach is an approach that involves two phases. The first phase commences with a pre-processing stage. In this stage, hidden dependencies between the exams are checked to accelerate the optimisation phase. A construction stage occurs after the first stage. The second stage uses a meta-heuristic called Greedy Randomised Adaptive Search Procedure (GRASP). In this phase, optimisation methods can be applied in the following order: HC, SA, Integer Programming (the Branch and Bound procedure), finishing with the so-called Shaking Stage, which can only be applied according to certain conditions. This Shaking Stage shakes the current solution to create a similarly good solution that is given to SA. This stage aims at forcing SA to restart with solutions that are more promising and generate better results.

AtsutaEtAl08 -(Atsuta et al., 2008): Atsuta et al. won third place on the Examination Timetabling track, as well as second places on other tracks, with a similar approach for all of them. The used approach involves applying a constraint satisfaction problem-solver, which adopts a hybridisation of TS, as well as Iterated Local Search.

De Smet08 -(De Smet, 2008): De Smet's approach differs from other approaches because he did not use a problem-specific heuristic, which is known, to find a feasible solution. Instead, he used the Drool's rule engine called the drools-solver (Drools, 2020). The drools-solver involves a combination of optimisation heuristics, in addition to meta-heuristics with a score calculation that is very efficient. The score of the solution is the sum of the constraints' weight, which is being broken. After a feasible solution is found, De Smet used a local search algorithm, i.e. Tabu Search (TS) to enhance the obtained solutions utilising the drools-solver.

Pillay08 -(Pillay, 2008): A two-phase algorithm variant was proposed by Pillay by using a Developmental Approach based on Cell Biology. The goal involves forming a well-developed organism by a creation process of a cell and proceeding with the cell division and cell interaction, as well as cell migration. Based on this approach, each of the cells signifies a period. The first phase involves the process of creating the first cell and cell division, as well as cell interaction, while the second phase involves cell migration.

In addition, there are further approaches that have been applied to the ITC 2007 benchmark set, which were proposed after the 2007 competition and reported the best-known results. We now briefly describe each approach in turn of the competitive methods we have identified in the literature.

McCollumEtAl09 -(McCollum et al., 2009): the authors applied an adaptive ordering heuristic to construct solutions, followed by applying the Great Deluge meta-heuristic (extended version). The approach was tested on the exam timetabling problems from ITC 2007. It was confirmed as an effective approach where the best results for 5 out of the 8 problem instances were obtained.

DemeesterEtAl12 -(Demeester et al., 2012): see Section 4.7.3.1.

GogosEtAl12 -(Gogos et al., 2012): the authors proposed an enhanced algorithm version. The authors claimed that the enhanced behaviour is due to more sophisticated process flow, early detection of plateaus, added heuristics, as well as optimised data structures, which achieve the exploration of a much larger number of Kempe Chain moves.

Alzaqebah&Abdullah14 -(Alzaqebah and Abdullah, 2014): see Section 4.7.3.1.

Alzaqebah&Abdullah15 -(Alzaqebah and Abdullah, 2015): see Section 4.7.3.1.

BattistuttaEtAl17 -(Battistutta et al., 2017): a single-stage procedure was proposed by Battistutta et al. based on the SA approach for the ITC 2007's ETP. Based on this method, non-feasible solutions are included in the search space, dealing with appropriate penalties. A statistically-principled experimental analysis is conducted to investigate the parameter selection effect. Then, a feature-based parameter tuning is performed.

MuklasonEtAl17 -(Muklason et al., 2017): see Section 4.7.3.1.

LeiteEtAl18 -(Leite et al., 2018): see Section 4.7.3.1.

Rajah&Pillay19 - More recently, the Structure-Based Partial Solution Search (SBPSS) was proposed by Rajah and Pillay (2019) and improved on the best results of ITC 2007 Examination track. The SBPSS is a multi-point search approach that aims to solve this problem incrementally. Solutions are initialised partially, and at each generation, one solution component is selected randomly and added to each partial solution continuously. This process stops when all solution components are completely added to the solutions.

MandalEtAl20 - see Section 4.7.3.1

4.7.3.3 Reported 'best' Results of the Yeditepe Benchmarks

All approaches (i.e. Muklason17, Muller's approach, and several hyper-heuristic strategies from (Muklason, 2017)) applied in the Yeditepe problem and used in comparison with the proposed ESGA have been discussed in Section 2.5.2 in Chapter 2.

4.7.4 Comparison with the Basic Genetic Algorithm

This section provides the results obtained by the ESGA described in Section 4.6. The results of this work were compared with those of the basic GA explained in Section 2.8.6 in Chapter 2. To obtain the results in this work, each algorithm was run thirty times on each dataset (Toronto, ITC 2007 and Yeditepe).

Table 4.1 illustrates the results obtained when the basic and exam specialised GA were applied to the Toronto, ITC 2007 and Yeditepe datasets, respectively. The Table shows the lowest values, the highest values, the average soft constraint costs, and their standard deviation. In this study, the worst algorithm gets the highest value, whereas the best solution on a given instance gets the lowest value (i.e. the algorithm with the overall lowest value can be considered as the best performing algorithm). f_{min} denotes the best solution value (minimum penalty) over thirty executions, f_{avg} is the average and f_{max} the worst solution value, while σ denotes the standard deviation.

In order to make a fair comparison between the exam specialised and the basic GA, the former also used the OBSI method to create the initial population. Based on Table 4.2, it can be noticed that, as expected, the exam specialised GA outperformed the basic one on each problem instance of the Toronto, the ITC 2007 and the Yeditepe datasets.

Dataset	Problem Instance	GA					ESGA				
		f_{min}	f_{max}	f_{avg}	σ	$Max_{time(sec)}$	f_{min}	f_{max}	f_{avg}	σ	$Max_{time(sec)}$
	car91	5.90	7.15	6.50	0.42	2268	**5.21**	**6.10**	**5.57**	0.27	2480
	car92	5.05	6.21	5.73	0.32	2485	**4.10**	**5.04**	**4.50**	0.27	2540
	ear83	35.10	37.99	36.47	0.74	2548	**33.47**	**35.59**	**34.25**	0.64	2577
	hec92	10.76	11.43	11.05	0.20	1375	**10.34**	**11.10**	**10.67**	0.22	1497
	kfu93	13.52	14.16	13.88	0.20	2288	**13.11**	**14.07**	**13.40**	0.28	3197
TORONTO	lse91	11.48	13.24	12.32	0.51	2063	**10.11**	**11.99**	**10.88**	0.58	2048
	pur93	5.08	5.95	5.60	0.23	2826	**4.55**	**5.48**	**5.00**	0.28	2876
	rye92	10.01	11.04	10.58	0.30	1995	**9.30**	**10.56**	**10.11**	0.38	2010
	sta83	158.22	159.54	158.81	0.41	486	**157.80**	**158.52**	**158.18**	0.17	512
	tre92	9.16	10.55	9.78	0.43	1642	**8.67**	**9.60**	**9.09**	0.26	1676
	uta92	3.52	4.19	3.86	0.21	4118	**3.39**	**3.97**	**3.62**	0.18	4171
	ute92	25.10	26.28	25.75	0.33	1215	**24.75**	**25.70**	**25.08**	0.21	1234
	yor83	37.47	39.91	38.50	0.83	2016	**36.24**	**37.35**	**36.96**	0.35	2038
	Exam1	10213	16491	12001.83	1557.17		**8068**	**12533**	**9552.03**	977.85	
	Exam2	704	1754	1031.06	251.31		**528**	**1385**	**772.48**	195.62	
	Exam3	17420	23905	19585.83	1791.65		**11845**	**17689**	**13743.70**	1234.30	
	Exam4	28576	52204	35470.64	5073.10		**18574**	**31844**	**23281.16**	2821.74	
	Exam5	6361	14904	8736.90	1948.70		**4325**	**9687**	**6032.22**	1271.92	
ITC 2007	Exam6	35441	47728	39238.74	2900.45	392	**26226**	**34841**	**28926.58**	1747.23	392
	Exam7	18491	31424	22296.70	3014.16		**11649**	**20425**	**14259.41**	2153.04	
	Exam8	20702	39514	26870.29	4323.04		**10351**	**20547**	**13169.41**	2352.78	
	Exam9	1731	4675	2665.83	890.21		**1142**	**3132**	**1787.41**	442.02	
	Exam10	21706	29119	24280.16	1669.23		**12589**	**15724**	**13581.45**	774.70	
	Exam11	49062	73260	56785.06	5144.58		**35815**	**52014**	**41081.51**	4266.67	
	Exam12	7812	11250	8705.96	805.31		**5468**	**7762**	**6128.03**	640.56	
	YUE20011	82	167	125.46	20.15		**64**	**102**	**82.56**	11.60	
	YUE20012	180	286	250.73	21.34		**129**	**190**	**153.96**	18.84	
	YUE20013	38	47	41.16	2.01		**29**	**34**	**31.03**	1.70	
YEDITEPE	YUE20021	165	208	178.76	10.82	392	**83**	**153**	**151.23**	18.65	392
	YUE20022	236	380	315.7	42.75		**167**	**250**	**215.36**	25.54	
	YUE20023	99	105	101.23	1.20		**55**	**62**	**58.16**	1.93	
	YUE20031	266	386	330	32.05		**192**	**264**	**230.33**	22.98	
	YUE20032	627	810	720.5	57.40		**438**	**518**	**490.06**	23.31	

Table 4.1 Results of a basic GA and ESGA on the Toronto, the ITC 2007, and the Yeditepe problem instances are obtained after 30 runs. The best values are shown in bold.

Table 4.2 shows the improvement factor between the basic and exam specialised GA. This was calculated by working out the difference between the average values obtained for each of the basic and exam specialised GA (across all instances), and this was then divided by the decrease in the average value of the basic GA and multiplied by 100. Obviously, these results can be considered as better results based on Table 4.2.

A Mann-Whitney U-test was carried out between the basic GA with the ESGA, for which the p-value for each problem instance in the Toronto, the ITC 2007, and the Yeditepe datasets is provided in Table 4.2. For all datasets, the p-value for the results of the two approaches is less than 0.05, indicating a statistically significant difference between the results. All problem results show statistically significant difference.

Dataset	Problem Instance	GA f_{avg}	ESGA f_{avg}	Improvement Factor (%)	p-value
TORONTO	car91	6.50	5.57	14.31	6.72E-10
	car92	5.73	4.50	21.47	3.02E-11
	ear83	36.47	34.25	6.09	1.21E-10
	hec92	11.05	10.67	3.56	3.81E-07
	kfu93	13.88	13.40	3.46	3.01E-11
	lse91	12.32	10.88	11.04	3.47E-10
	pur93	5.60	5.00	10.71	1.86E-09
	rye92	10.58	10.11	4.44	1.75E-05
	sta83	158.81	158.18	0.40	7.77E-09
	tre92	9.78	9.09	7.06	5.09E-08
	uta92	3.86	3.62	6.22	1.04E-04
	ute92	25.75	25.08	2.6	1.41E-09
	yor83	38.50	36.96	4.17	3.02E-11
ITC 2007	Exam1	12001.83	9552.03	20.41	4.34E-09
	Exam2	1031.06	772.48	25.08	4.19E-05
	Exam3	19585.83	13743.70	29.83	1.70E-11
	Exam4	35470.64	23281.16	34.36	3.67E-11
	Exam5	8736.90	6032.22	30.95	4.33E-08
	Exam6	39238.74	28926.58	26.28	1.40E-11
	Exam7	22296.70	14259.41	36.04	4.88E-11
	Exam8	26870.29	13169.41	50.98	1.40E-11
	Exam9	2665.83	1787.41	32.95	2.26E-05
	Exam10	24280.16	13581.45	44.06	1.40E-11
	Exam11	56785.06	41081.51	27.65	3.03E-11
	Exam12	8705.96	6128.03	29.61	1.40E-11
YEDITEPE	YUE20011	125.46	82.56	34.19	3.93E-10
	YUE20012	250.73	153.96	38.59	4.03E-11
	YUE20013	41.16	31.03	24.61	2.31E-11
	YUE20021	178.76	115.23	35.53	2.98E-11
	YUE20022	315.70	215.36	31.78	1.94E-10
	YUE20023	101.23	42.55	57.97	2.18E-11
	YUE20031	330	230.33	30.20	3.00E-11
	YUE20032	720.50	490.06	31.98	3.01E-11

Table 4.2 Improvement factor (percentage) between the results on the Toronto, the ITC 2007, and Yeditepe benchmark sets of ESGA with the basic GA.

Problem	ESGA			
Instance	f_{min}	f_{max}	f_{avg}	σ
car91	4.64	5.34	5.0	0.26
car92	3.84	4.66	4.2	0.27
ear83	32.24	32.89	32.5	0.22
hec92	10.12	11.03	10.56	0.33
kfu93	12.92	13.55	13.17	0.23
lse91	9.90	11.91	10.57	0.75
pur93	4.33	4.87	4.57	0.20
rye92	7.67	9.24	8.5	0.60
sta83	157.26	158.16	157.62	0.31
tre92	7.72	9.15	8.33	0.51
uta92	3.11	3.56	3.39	0.36
ute92	25.26	28.1	26.83	1.16
yor83	35.70	36.62	36.18	0.31

Table 4.3 Results of the ESGA on the Toronto benchmark set are obtained after ten runs with a fix time limit (10800 seconds).

4.7.5 Comparison with the State-of-the-Art Approaches

As has been previously mentioned in section 4.7.2, two experiments were conducted with the proposed ESGA on the Toronto benchmark set. The ESGA was implemented after utilising the initial solutions that were constructed using the OBSI method in phase 1. Table 4.3 shows the minimum, maximum, average and standard deviation over ten runs with a fixed time limit (Max time) on each problem instance, where total run time (optimisation) was capped at three hours. Table 4.5 shows the running times for the proposed approach and all approaches considered. Also in this section, a selection of the best algorithms from the literature as listed above (in Subsection 4.7.3.1) are compared against our proposed ESGA approach. For each problem instance of the Toronto, the ITC 2007, the Yeditepe benchmark sets, all of the considered approaches are ranked according to their cost values (as shown in Table 4.4, Table 4.6, and Table 4.7).

To calculate the ranking (RK) and the average ranking (AVG RK), let us consider total pr approaches are compared for g given problem instances with respect to their lowest 'best' results. For a given problem instance, each approach of g is given ordinal value V_o in which $1 \leq V_o \leq pr$ and $1 \leq V \leq g$. This value denotes the ranking of the corresponding problem instance. For the AVG RK, we sum all V_o values for each approach for g problem instances and calculates the AVG value. All of the approaches are ranked based on their AVG RK values.

4.7.5.1 Toronto Dataset

Table 4.4 presents a comparison between the proposed ESGA and these algorithms on the Toronto datasets, with the columns showing the lowest cost value (i.e. best cost) for each considered algorithm and RK and AVG RK.

From Table 4.4, it can be observed that we have been fully capable of obtaining conflict-free solutions for all of the thirteen problem instances of the Toronto problem. Our proposed approach outperforms Pillay&Banzhaf10 (Pillay and Banzhaf, 2010) and CarterEtAl96 (Carter et al., 1996), and Eley07 (Eley, 2007) in all problem instances. Also, the proposed ESGA produces better results than the approach by FongEtAl15 (Fong et al., 2015) except for sta83 and uta92. For one problem instance (i.e. ear83), our approach has achieved best results comparing to other approaches except the approach proposed by CaramiaEtAl08 (Caramia et al., 2008), and the third-best results for rye92, the fourth-best results for uta92, and the fifth-best results for instances lse91, pur93, and tre92. Therefore, It can also be verified that the proposed ESGA attains competitive costs on the Toronto benchmark set, an overall comparison shows that we are not able to beat any of the best results in the literature. Nevertheless, we are still able to produce sufficiently good solutions for all problem instances of the Toronto, and the comparison also shows that our results are competitive, thus proving the efficiency of the ESGA.

Approach	M	Problem Instance													AVG
		car91	car92	ear83	hec92	kfu93	lse91	pur93	rye92	sta83	tre92	uta92	ute92	yor83	RK
CarterEtAl96 (Carter et al., 1996)	f_{min}	7.10	6.20	36.40	10.80	14.00	10.50	3.90	7.30	161.50	9.60	3.50	25.80	41.70	14
	RK	17	17	15	13	14	11	2	2	17	17	14	15	17	
Yang&Petrovic04 (Yang and Petrovic, 2005)	f_{min}	4.50	3.93	33.71	10.83	13.82	10.35	-	8.53	158.35	7.92	3.14	25.39	36.53	13
	RK	4	9	13	14	13	9	-	8	16	9	8	14	13	
Eley07 (Eley, 2007)	f_{min}	5.20	4.30	36.80	11.10	14.50	11.30	4.60	9.80	157.30	8.60	3.50	26.40	39.40	17
	RK	14	14	17	16	17	17	7	15	13	14	14	16	16	
CaramiaEtAl08 (Caramia et al., 2008)	f_{min}	6.60	6.00	**29.30**	**9.20**	13.80	**9.60**	**3.70**	**6.80**	158.20	9.40	3.50	**24.40**	36.20	10
	RK	16	16	1	1	12	1	1	1	15	16	14	1	11	
Burke&Bykov08 (Burke and Bykov, 2008)	f_{min}	4.58	3.81	32.65	10.06	**12.81**	9.86	4.53	7.93	157.03	7.72	3.16	24.79	34.78	4
	RK	6	4	7	5	1	4	6	5	2	3	9	4	4	
BurkeEtAl10 (Burke et al., 2010b)	f_{min}	4.90	4.10	33.20	10.30	13.20	10.40	-	-	**156.90**	8.30	3.30	24.90	36.30	8
	RK	12	12	9	7	6	10	-	-	1	12	12	6	12	
Pillay&Banzhaf10 (Pillay and Banzhaf, 2010)	f_{min}	4.92	4.22	35.87	11.50	14.37	10.89	4.65	9.30	157.81	8.38	3.35	27.24	39.33	16
	RK	13	13	14	17	16	15	8	13	14	13	13	17	15	
DemeesterEtAl12 (Demeester et al., 2012)	f_{min}	4.52	3.78	32.49	10.03	12.90	10.04	5.67	8.05	157.03	7.69	3.13	24.77	34.64	3
	RK	5	3	5	2	4	6	10	7	2	2	5	2	3	
Abdullah&Alzaqebah13 (Abdullah and Alzaqebah, 2013)	f_{min}	4.76	3.94	33.61	10.56	13.44	10.87	-	8.81	157.09	7.94	3.27	25.36	35.74	12
	RK	10	10	12	12	9	14	-	9	10	10	10	13	9	
Alzaqebah&Abdullah14 (Alzaqebah and Abdullah, 2014)	f_{min}	4.62	4.00	33.14	10.43	13.59	10.75	-	9.17	157.06	8.00	3.27	25.16	35.58	11
	RK	8	11	8	10	10	13	-	12	7	11	10	8	7	
Alzaqebah&Abdullah15 (Alzaqebah and Abdullah, 2015)	f_{min}	4.38	3.88	33.34	10.39	13.23	10.52	-	8.92	157.06	7.89	3.13	25.12	35.49	7
	RK	2	7	10	9	7	12	-	10	7	8	5	7	6	
FongEtAl15 (Fong et al., 2015)	f_{min}	4.79	3.89	33.43	10.49	13.72	10.29	-	-	157.07	7.86	3.10	25.33	36.12	9
	RK	11	8	11	11	11	8	-	-	9	7	3	12	10	
LeiteEtAl16 (Leite et al., 2016)	f_{min}	4.41	3.75	32.62	10.03	12.88	9.85	4.10	7.98	157.03	7.75	3.08	24.78	**34.44**	2
	RK	3	2	6	2	3	3	3	6	2	6	2	3	1	
MuklasonEtAl17 (Muklason et al., 2017)	f_{min}	5.30	4.51	36.73	10.91	14.36	11.02	5.03	9.01	157.12	8.75	3.60	25.20	38.03	15
	RK	15	15	16	15	15	16	9	11	11	15	17	9	14	
LeiteEtAl18 (Leite et al., 2018)	f_{min}	**4.31**	**3.68**	32.48	10.03	**12.81**	9.78	4.14	7.89	157.03	**7.66**	**3.01**	24.80	34.45	1
	RK	1	1	3	2	1	2	4	4	2	1	1	5	2	
MandalEtAl20 (Mandal et al., 2020)	f_{min}	4.58	3.82	32.48	10.32	13.34	10.24	-	9.79	157.03	7.72	3.13	25.28	35.46	5
	RK	6	5	3	8	8	7	-	14	2	3	5	11	5	
Proposed ESGA	f_{min}	4.64	3.84	32.24	10.12	12.92	9.90	4.33	7.67	157.26	7.72	3.11	25.26	35.70	6
	RK	9	6	2	6	5	5	5	3	12	3	4	10	8	

Table 4.4 Comparison 'best' results of the proposed approach with reported 'best' results of various state-of-the-art approaches on the Toronto benchmark set. In the measure column 'M', (f_{min}) presents the best solution value (minimum penalties and the row 'R' is rank. The comparison is made between the best soft constraints cost of each approach. The best solutions are in boldface. (-) indicates that a feasible solution could not be obtained, or the following datasets were not tested.

Running times (s) for Toronto benchmark sets.													
Approach	car91	car92	ear83	hec92	kfu93	lse91	pur93	rye92	sta83	tre92	uta92	ute92	yor83
CarterEtAl96	20.7	47	24.7	7.4	120	48	**21729.4**	507.2	5.7	107.4	664.3	9.1	174.5
Yang&Petrovic04	**2773.2**	1518	1135.2	1093.8	2152.8	1729.2	-	1381.8	739.8	1816.2	1890	745.8	1591.2
Eley07	**16200**	**16200**	**16200**	18	**16200**	**16200**	**16200**	**16200**	**16200**	**16200**	**16200**	**16200**	**16200**
CaramiaEtAl08	172	1120	300	48	320	446	**12042**	452	30	430	2046	20	452
Burke&Bykov08	664.8	604.2	450	589.8	882	640.8	747	901.2	586.8	607.8	805.2	528	528
BurkeEtAl10	5400	5400	120	120	120	120	-	-	120	120	5400	120	120
Pillay&Banzhaf10	5880	4260	751	451	3168	2863	**113760**	4320	469	1121	3639	663	552
DemeesterEtAl12	**43200**	**43200**	**43200**	**43200**	**43200**	**43200**	**43200**	**43200**	3600	**43200**	**43200**	**43200**	**43200**
Abdullah&Alzaqebah13	**	**	**	**	**	**	-	**	**	**	**	**	**
Alzaqebah&Abdullah14	9600	9600	300	300	300	300	-	300	300	300	9600	300	300
Alzaqebah&Abdullah15	9600	9600	600	600	600	600	-	600	600	600	9600	600	600
FongEtAl15	**33120**	**31260**	**23460**	**13860**	**21900**	**15240**	-	-	10680	**24840**	**35280**	**12600**	**25200**
LeiteEtAl16	**86400**	**86400**	**86400**	**86400**	**86400**	**86400**	**172800**	**86400**	**86400**	**86400**	**86400**	**86400**	**86400**
MuklasonEtAl17	360	360	360	360	360	360	360	360	360	360	360	360	360
LeiteEtAl18	**172800**	**172800**	**86400**	**86400**	**86400**	**86400**	**172800**	**86400**	**86400**	**86400**	**172800**	**86400**	**86400**
ManadalEtAl20	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
Proposed ESGA	10800	10800	10800	10800	10800	10800	10800	10800	10800	10800	10800	10800	10800

Table 4.5 Running times (in seconds $t(s)$) for the Toronto benchmark set. (**) indicates that the authors did not report the algorithm execution time, and (-) means that the algorithm was not able to obtain a feasible solution.

4.7.5.2 ITC 2007 Dataset

According to Table 4.6 and in relation to the ITC 2007, the experimental results from our proposed algorithm are compared with those of the top five performing algorithms in the ITC 2007 contest, as well as some other more up-to-date approaches. These results, however, cannot be regarded as competitive with Müller09's approach, which was the winner of the ITC 2007 competition. On the other hand, Müller09's approach did not perform well for the hidden instances such as Exam10, Exam11, and Exam12. Based on Table 4.6, most of the best results were beaten by the BattistuttaEtAl17, LeiteEtAl18, and Rajah&Pillay19 algorithms, with some of these results exceeding those of Müller09 (Müller, 2008). Furthermore, in a number of datasets, the developed method could reach, and even beat most contestants. It can be noticed that the highest scoring datasets, i.e. those with the worst results, are those with additional $exam_{coincident}$ hard constraints.

While our proposed approach does not outperform other state-of-the-art approaches, it is comparable to them. It can be observed from Table 4.6, however, that with one particular problem instance (i.e. Exam 10) our proposed algorithm is able to return a solution beating Müller's approach as well as other best-known solutions. It is, therefore, anticipated that further improvements in the algorithm will address its limitations to make it still more competitive the best of the current approaches.

4.7.5.3 Yeditepe Dataset

As for the Yeditepe dataset, the comparison between our results and recently-reported results from the literature is given in Table 4.7 since there is a lack of prior experimental results for this dataset. We compared our results with the results obtained from optimising the standard objective function of Muklason17's approach, and with results generated from Müller09's solver (Müller, 2008). Although MuklasonEtAl17's approach outperformed Müller09's approach, our proposed ESGA outperforms both those approaches for two out of the eight problem instances in this dataset. The results also indicate that our proposed approach is able to obtain better results compared to the other state-of-the-art approaches.

Further, from Table 4.7, it can be observed that the SA-GD-HH algorithm outperforms the SR-GD-HH and RL-GD-HH algorithms. SR-XGD-HH is comparable with SA-XGD-HH when combining the strategies of low-level heuristic selection with the extended great deluge algorithm rather than the standard great deluge. In addition to, when combining the strategies of low-level heuristic selection strategies with the modified extended great deluge algorithm, it can be noticed that SR-MXGD-HH outperforms SA-MXGD-HH and RL-MXGD-HH. An overall comparison with the best-known results, however, showed that the approach in this work is an efficient approach in comparison with other approaches. In several datasets, the introduced method reached the level set by most contestants and achieved another best result in the YUE20023 instance.

Approach	M	Problem Instance												AVG
		Exam1	Exam2	Exam3	Exam4	Exam5	Exam6	Exam7	Exam8	Exam9	Exam10	Exam11	Exam12	RK
Müller09 (Müller, 2008)	f_{min}	4370	400	10049	18141	2988	26585	4213	7742	1030	16682	34129	5535	9
	RK	3	2	6	11	5	10	5	5	4	11	5	8	
GogosEtAl08 (Gogos et al., 2008)	f_{min}	5905	1008	13771	18674	4139	27640	6572	10521	1159	-	43888	-	15
	RK	10	15	13	13	12	14	14	14	8	-	10	-	
AtsutaEtAl08 (Mccollum et al., 2010)	f_{min}	8006	3470	17669	22559	4638	29155	10473	14317	1737	15085	-	5264	16
	RK	15	17	15	14	15	16	15	15	12	9	-	4	
De Smet08 (De Smet, 2008)	f_{min}	6670	623	-	-	3847	27815	5420	-	1288	14778	-	-	10
	RK	13	13	-	-	11	15	9	-	11	8	-	-	
Pillay08 (Pillay, 2008)	f_{min}	12035	2886	15917	23582	6860	32250	17666	15592	2055	17724	40535	6310	17
	RK	17	16	14	15	17	17	17	16	13	12	9	11	
McCollumEtAl09 (McCollum et al., 2009)	f_{min}	4633	405	9064	15663	3042	**25880**	4037	7461	1071	14374	29180	5693	5
	RK	4	4	4	5	6	3	3	2	6	6	4	10	
DemEtAl12(Demeester et al., 2012)	f_{min}	6060	515	23580	-	4855	27605	6065	9038	1184	15561	-	5483	12
	RK	11	9	16	-	16	13	13	11	9	10	-	7	
GogEtAl12 (Gogos et al., 2012)	f_{min}	4775	**385**	8996	16204	2929	25740	4087	7777	-	-	-	-	2
	RK	5	1	3	7	4	2	4	5	-	-	-	-	
Bykov&Petrovic13 (Bykov and Petrovic, 2013)	f_{min}	4008	404	**8012**	**13132**	**2582**	25448	3893	**6944**	**949**	12985	**25194**	5181	1
	RK	2	3	1	1	1	1	2	1	1	2	1	3	
Alzaqebah&Abdullah14 (Alzaqebah and Abdullah, 2014)	f_{min}	5328.00	512	10178	16465	3624	26240	4562	8098	-	-	-	-	7.5
	RK	8	8	7	10	8	9	7	7	-	-	-	-	
Alzaqebah&Abdullah15 (Alzaqebah and Abdullah, 2015)	f_{min}	5154	420	10182	15716	3350	26160	4271	7922	-	-	-	-	4
	RK	6	7	8	6	7	7	6	6	-	-	-	-	
BattistuttaEtAl17(Battistutta et al., 2017)	f_{min}	**3926.96**	407.72	8849.46	15617.82	2849	26081.35	**3661.64**	7729.46	991.57	13999.56	27781.5	5550.2	3
	RK	1	6	2	4	3	6	1	3	2	5	2	9	
MuklasonEtAl17(Muklason et al., 2017)	f_{min}	6856	632	11659	16325	3837	27370	5528	9798	1246	14556	36810	5300	14
	RK	14	14	9	9	10	12	12	12	10	7	7	5	
LeiteEtAl18 (Leite et al., 2018)	f_{min}	6207	535.00	13022	14302	3829	26710	5508	8716	1030	13894	39783	5142	11
	RK	12	11	12	3	9	11	11	10	3	4	8	2	
Rajah&Pillay19 (Rajah and Pillay, 2019)	f_{min}	5192	546	9109	14073	2825	26030	4697	8570	1044	13352	28181	**5138**	6
	RK	7	12	5	2	2	5	8	9	5	3	3	1	
MandalEtAl20 (Mandal et al., 2020)	f_{min}	5328	407	11692	16204	4300	25880	5507	8238	-	-	-	-	7.5
	RK	8	5	10	7	13	3	10	8	-	-	-	-	
Proposed ESGA	f_{min}	8068	528	11845	18574	4325	26226	11649	10351	1142	**12589**	35815	5468	13
	RK	16	10	11	12	14	8	16	13	7	1	6	6	

Table 4.6 Comparison ‘best’ results of the proposed approach with reported ‘best’ results of the ITC 2007 finalists and more recent approaches applied to the ITC 2007 benchmark set. Notation as in Table 4.4

Approach	M	Problem Instance								AVG
		YUE20011	YUE20012	YUE20013	YUE20021	YUE20022	YUE20023	YUE20031	YUE20032	RK
Müller09 (Müller, 2008)	f_{min}	62	125	**29**	70	170	70	223	440	9
	RK	7	7	1	6	8	12	9	9	
MuklasonEtAl17 (Muklason et al., 2017)	f_{min}	56	122	**29**	76	162	56	143	434	6
	RK	6	6	1	7	5	2	3	7	
SR-GD-HH (Muklason, 2017)	f_{min}	88	126	**29**	97	150	56	186	**362**	7
	RK	11	8	1	9	3	2	7	1	
RL-GD-HH (Muklason, 2017)	f_{min}	69	152	45	143	281	59	274	650	11
	RK	10	10	12	12	11	9	10	11	
SA-GD-HH (Muklason, 2017)	f_{min}	49	**98**	**29**	**46**	133	56	**121**	370	1
	RK	2	1	1	1	2	2	1	3	
SR-XGD-HH (Muklason, 2017)	f_{min}	50	109	**29**	64	170	56	124	405	3.5
	RK	3	2	1	5	8	2	2	6	
RL-XGD-HH (Muklason, 2017)	f_{min}	63	161	33	101	233	61	279	648	10
	RK	8	11	11	10	10	10	11	10	
SA-XGD-HH (Muklason, 2017)	f_{min}	55	113	**29**	51	163	56	146	377	3.5
	RK	5	4	1	3	6	2	4	4	
SR-MXGD-HH (Muklason, 2017)	f_{min}	**48**	107	**29**	49	**125**	56	148	**362**	2
	RK	1	3	1	2	1	2	5	1	
RL-MXGD-HH (Muklason, 2017)	f_{min}	110	213	30	126	311	65	345	685	12
	RK	12	12	10	11	12	11	12	12	
SA-MXGD-HH (Muklason, 2017)	f_{min}	50	117	**29**	55	156	56	161	400	5
	RK	3	5	1	4	4	2	6	5	
Proposed ESGA	f_{min}	64	129	**29**	83	167	**55**	192	438	8
	RK	9	9	1	8	7	1	8	8	

Table 4.7 Comparison 'best' results of the proposed ESGA approach with reported 'best' results of state-of-the-art approaches for Yeditepe dataset. Notation as in Table 4.4.

4.8 Summary

Various real-world timetabling problems have been investigated by many researchers. Genetic algorithms are very commonly used to tackle these problems. In this chapter, various variators, including directed and random mutations, varied sizes of mutations, solution-oriented mutations, as well as selection operators were introduced, while the crossover phase was omitted. In the experiments conducted for this study, the proposed GA yielded better results than the basic GA across all problem instances on the Toronto, ITC 2007, and Yeditepe datasets. The proposed mutation operators efficiently and effectively exploited and explored the solutions to create a high-quality solution. The proposed solution selection keeps the population diversified by preventing similar individuals from joining the population during the replacement process. Furthermore, the crossover phase was omitted in this work since it was found to increase the execution time of the GA significantly, without any improvements in the results of the examination timetabling problem. It is, therefore, crucial to assert that the operators in this work can also be utilised in GAs to solve other types of timetabling problems and/or other types of problems of constraint satisfaction like the timetabling of a high school course, nurse rostering or the classroom assignments.

In the following chapter, we will use the approach proposed in Chapter 3 (i.e. OBSI plus OBSI combined with the EA) to solve an additional examination timetabling problem, namely, the Toronto benchmark set and employ a multi-objective comparison scheme based on (uncertain) Pareto dominance in order to analyse published exam timetabling approaches on the Toronto benchmark sets and identify the (probabilistic) Pareto set of optimisers.

Chapter 5

A Novel Multi-objective Framework to Analyse 25 years of Exam Timetable Optimisation

Work presented in this chapter is currently under review at *the Journal of the Information Sciences.*

5.1 Introduction

In this chapter we identify a number of issues with the common comparison between published works on popular exam timetabling benchmarks in the literature. An uncertain Pareto analysis approach is proposed and developed to compare published work, which compensates for uncertainties in their effective computational budget, and represents the trade-off between optimisation time and solution quality. The aim is to compare and contrast results spanning the last 25 years of exam timetable optimisation in order to identify the Pareto set of optimisers – i.e. those that can reasonably be considered the 'best' for each problem, given the data available.

The popular Toronto and ITC 2007 benchmarks have been briefly described in Section 2.5 in Chapter 2. The following section analyses and highlights the best-known approaches reported in the literature for solving the Toronto and the ITC 2007 benchmark sets. Section 5.3 outlines the fundamental problems when comparing published algorithm performance on exam timetabling benchmarks from the literature. Section 5.4 describes a multi-objective analysis of the results that can address some of the issues identified in Section 5.3, and presents a comparison of published results of 16 different algorithms using it. The chapter ends with the summary in Section 5.5.

5.2 Reported 'best' Results from the Literature over Time

The examination timetabling problem is regarded as a minimisation problem where the objective formulation for this problem seeks to satisfy all hard constraints and minimise violations with regard to soft constraints (which can, therefore, be cast as a *cost* or *loss* minimisation task). As such, the algorithm which returns a solution with the overall lowest soft constraint violation value, subject to satisfying the hard constraints, can be considered as the 'best' performing algorithm.

Various optimisers have been developed over the last 25 years for the exam timetabling task, and between them have obtained several best-known results on the Toronto and the ITC 2007 benchmark instances. These are listed in Section 5.2.1 and Section 5.2.2 respectively and described in Section 4.7.3 in Chapter 4. The best-published results (i.e. the minimum cost solution values found) and the mean results for each considered optimiser on the Toronto benchmark set are presented in Table 5.1. Moreover, Table 5.2 presents the ITC 2007 results of the five finalists of the examination track of the along with more recently developed optimisers that have been applied to this problem suite and fulfilled the ITC 2007 rules. Authors' name concatenated with publishing year are used to label the various approaches. Additionally, the running times (in seconds) of theses optimisers for the Toronto and ITC 2007 benchmark sets are reported in Table 5.3 and Table 5.4.

5.2.1 Reported 'best' Results of the Toronto Benchmarks

- **CarterEtAl96** (Carter et al., 1996).
- **Yang&Petrovic04** (Yang and Petrovic, 2005).
- **Eley07** (Eley, 2007).
- **CaramiaEtAl08** (Caramia et al., 2008).
- **Burke&Bykov08** (Burke and Bykov, 2008).
- **BurkeEtAl10** (Burke et al., 2010b).
- **Pillay&Banzhaf10** (Pillay and Banzhaf, 2010).
- **DemeesterEtAl12** (Demeester et al., 2012).
- **Abdullah&Alzaqebah13** (Abdullah and Alzaqebah, 2013).
- **Alzaqebah&Abdullah14** (Alzaqebah and Abdullah, 2014).
- **Alzaqebah&Abdullah15** (Alzaqebah and Abdullah, 2015).
- **FongEtAl15** (Fong et al., 2015).
- **LeiteEtAl16** (Leite et al., 2016).
- **MuklasonEtAl17** (Muklason et al., 2017).

- **LeiteEtAl18** (Leite et al., 2018).
- **Alsuwaylimi&Fieldsend19** (Alsuwaylimi and Fieldsend, 2019): We also use OBSI[1], see Chapter 3.

5.2.2 Reported 'best' Results of the ITC 2007 Benchmarks

- **Müller09** (Müller, 2008).
- **GogosEtAl08** (Gogos et al., 2008).
- **AtsutaEtAl08** (Atsuta et al., 2008).
- **De Smet08** (De Smet, 2008).
- **Pillay08** (Pillay, 2008).
- **McCollumEtAl09** (McCollum et al., 2009).
- **DemeesterEtAl12** (Demeester et al., 2012).
- **GogosEtAl12** (Gogos et al., 2012).
- **Alzaqebah&Abdullah14** (Alzaqebah and Abdullah, 2014).
- **Alzaqebah&Abdullah15** (Alzaqebah and Abdullah, 2015).
- **BattistuttaEtAl17** (Battistutta et al., 2017).
- **MuklasonEtAl17** (Muklason et al., 2017).
- **LeiteEtAl18** (Leite et al., 2018).
- **Alsuwaylimi&Fieldsend19** (Alsuwaylimi and Fieldsend, 2019).

5.3 Problems with Comparing Results from the Literature

There are a number of issues when comparing algorithm performances published in the literature for these tasks:

1. algorithms have been run for different lengths of time, often with performance only reported for the solution returned at the end of the time allocated;
2. algorithms in different works have been run on different machines (with different processor speeds, memory speeds, cache sizes, operating systems, languages, compilation optimisations, etc.), so even where the wall-clock time reported is the same/similar, the effective CPU time could still vary greatly;
3. algorithms have been run for a different number of *repeated* runs, from which the best found across these repeats is reported.

[1]Raw data available from `https://github.com/alsuwaylimi/OBSI_method_toronto`.

Approach	M	car91	car92	ear83	hec92	kfu93	lse91	pur93	rye92	sta83	tre92	uta92	ute92	yor83
CarterEtAl96	(b)	7.10	6.20	36.40	10.80	14.00	10.50	3.90	7.30	161.50	9.60	3.50	25.80	41.70
	(m)	***	***	***	***	***	***	***	***	***	***	***	***	***
Yang&Petrovic04	(b)	4.50	3.93	33.71	10.83	13.82	10.35	-	8.53	158.35	7.92	3.14	25.39	36.53
	(m)	4.53	3.99	34.87	11.36	14.35	10.87	-	8.79	158.02	8.10	3.20	26.10	36.88
Eley07	(b)	5.20	4.30	36.80	11.10	14.50	11.30	4.60	9.80	157.30	8.60	3.50	26.40	39.40
	(m)	5.10	4.40	38.30	11.40	14.90	11.70	4.60	10.00	157.50	8.70	3.50	27.00	40.40
CaramiaEtAl08	(b)	6.60	6.00	29.30	9.20	13.80	9.60	3.70	6.80	158.20	9.40	3.50	24.40	36.20
	(m)	***	***	***	***	***	***	***	***	***	***	***	***	***
Burke&Bykov08	(b)	4.58	3.81	32.65	10.06	12.81	9.86	4.53	7.93	157.03	7.72	3.16	24.79	34.78
	(m)	4.68	3.92	32.91	10.22	13.02	10.14	4.71	8.06	157.05	7.89	3.26	24.82	35.16
BurkeEtAl10	(b)	4.90	4.10	33.20	10.30	13.20	10.40	-	-	156.90	8.30	3.30	24.90	36.30
	(m)	***	***	***	***	***	***	-	-	***	***	***	***	***
Pillay&Banzhaf10	(b)	4.92	4.22	35.87	11.50	14.37	10.89	4.65	9.30	157.81	8.38	3.35	27.24	39.33
	(m)	5.05	4.28	36.49	11.69	14.42	10.95	4.70	9.41	158.07	8.45	3.39	27.45	39.74
DemeesterEtAl12	(b)	4.52	3.78	32.49	10.03	12.90	10.04	5.67	8.05	157.03	7.69	3.13	24.77	34.64
	(m)	4.64	3.86	32.69	10.06	13.24	10.21	5.75	8.20	157.05	7.79	3.17	24.88	34.83
Abdullah&Alzaqebah13	(b)	4.76	3.94	33.61	10.56	13.44	10.87	-	8.81	157.09	7.94	3.27	25.36	35.74
	(m)	4.96	4.16	34.44	10.76	13.93	11.34	-	9.20	157.22	8.30	3.45	25.75	36.71
Alzaqebah&Abdullah14	(b)	4.62	4.00	33.14	10.43	13.59	10.75	-	9.17	157.06	8.00	3.27	25.16	35.58
	(m)	4.74	4.08	33.72	10.59	13.86	11.00	-	9.54	157.16	8.14	3.33	25.37	36.32
Alzaqebah&Abdullah15	(b)	4.38	3.88	33.34	10.39	13.23	10.52	-	8.92	157.06	7.89	3.13	25.12	35.49
	(m)	4.52	4.09	33.66	10.90	13.46	10.82	-	9.26	157.16	8.09	3.23	25.33	35.69
FongEtAl15	(b)	4.79	3.89	33.43	10.49	13.72	10.29	-	-	157.07	7.86	3.10	25.33	36.12
	(m)	4.85	4.27	34.48	10.61	13.76	10.39	-	-	157.37	8.04	3.31	26.04	36.67
LeiteEtAl16	(b)	4.41	3.75	32.62	10.03	12.88	9.85	4.10	7.98	157.03	7.75	3.08	24.78	34.44
	(m)	4.45	3.77	32.69	10.06	13.00	9.93	4.17	8.06	157.03	7.80	3.15	24.81	34.73
MuklasonEtAl17	(b)	5.30	4.51	36.73	10.91	14.36	11.02	5.03	9.01	157.12	8.75	3.60	25.20	38.03
	(m)	5.44	4.66	38.27	11.43	15.08	12.00	5.20	9.53	157.39	9.19	3.72	26.39	39.56
LeiteEtAl18	(b)	4.31	3.68	32.48	10.03	12.81	9.78	4.14	7.89	157.03	7.66	3.01	24.80	34.45
	(m)	4.39	3.72	32.61	10.05	12.83	9.81	4.18	7.93	157.03	7.70	3.04	24.83	34.63
Alsuwaylimi&Fieldsend19:	(b)	7.31	6.75	41.54	12.11	14.38	13.78	6.75	11.25	160.40	11.99	4.27	26.75	45.19
OBSI	(m)	8.38	7.13	44.65	14.46	16.72	16.30	8.63	13.88	163.62	12.78	4.69	28.97	49.52
OBSI + EA	(b)	5.24	4.42	36.31	11.14	13.17	11.51	4.45	9.66	158.16	9.58	3.29	25.89	37.71
	(m)	5.55	4.63	38.18	11.63	13.39	12.02	4.72	10.13	158.76	10.02	3.57	26.11	38.49

Table 5.1 Reported results of various state-of-the-art approaches on the Toronto benchmark set. In the measure column 'M', (b) presents the best solution values reported over the multiple runs for an approach (minimum penalties) and the row (m) is the mean results. (-) indicates that a feasible solution could not be obtained, or the following datasets were not tested. In the row marked with (***), the authors did not report the mean results.

Issues 1 and 2 mean there is no common yardstick in order to fairly compare algorithm performance. Indeed, as we can see from Tables 5.3 and 5.4, some algorithms are run for an order-of-magnitude (or more) run time than others that they subsequently compare themselves to. For ITC 2007, we are fortunate, as there is an application provided by the

Approach	M	Exam1	Exam2	Exam3	Exam4	Exam5	Exam6	Exam7	Exam8	Exam9	Exam10	Exam11	Exam12
Müller09	(b)	4370	400	10049	18141	2988	26585	4213	7742	1030	16682	34129	5535
	(m)	4574	414	10789.16	21639	3320.7	27808.5	4396.3	7950.3	1085.2	18580.9	34129	6403.2
GogosEtAl08	(b)	5905	1008	13771	18674	4139	27640	6572	10521	1159	-	43888	-
	(m)	6064	1048.6	14133.5	20666.6	4229.1	28077.5	6759.5	10809	1203.875	-	49861.3	-
AtsutaEtAl08	(b)	8006	3470	17669	22559	4638	29155	10473	14317	1737	15085	-	5264
	(m)	9083.90	3669.4	19367.4	26346.8	4920.3	29935	11004.33	14869.9	1935.6	15579.8	-	5541.9
De Smet08	(b)	6670	623	-	-	3847	27815	5420	-	1288	14778	-	-
	(m)	6670.80	623	-	-	3858.4	28155	5432.3	-	1288	14778	-	-
Pillay08	(b)	12035	2886	15917	23582	6860	32250	17666	15592	2055	17724	40535	6310
	(m)	12819.20	3925.8	19812.1	25728.8	11176	34028.88	19669.3	16720.7	2277	20332.6	44277.1	7179
McCollumEtAl09	(b)	4633	405	9064	15663	3042	25880	4037	7461	1071	14374	29180	5693
	(m)	***	***	***	***	***	***	***	***	***	***	***	***
DemeesterEtAl12	(b)	6060	515	23580	-	4855	27605	6065	9038	1184	15561	-	5483
	(m)	6330.20	612.5	23580	-	5323	28578.13	6250	9260.9	1255.9	16113.33	-	5829.14
GogosEtAl12	(b)	4775	385	8996	16204	2929	25740	4087	7777	-	-	-	-
	(m)	5032	404	9484	19607	3158	26310	4352	8098	-	-	-	-
Alzaqebah&	(b)	5328	512	10178	16465	3624	26240	4562	8098	-	-	-	-
Abdullah14	(m)	5517.30	537.9	10324.9	16589.1	3631.9	26275	4592.4	8328.8	-	-	-	-
Alzaqebah&	(b)	5154	420	10182	15716	3350	26160	4271	7922	-	-	-	-
Abdullah15	(m)	5227.81	457.55	10421.64	16108.27	3443.72	26247.27	4415	8225.81	-	-	-	-
BattistuttaEtAl17	(b)	3926.96	407.72	8849.46	15617.82	2849	26081.35	3661.64	7729.46	991.57	13999.56	27781.5	5550.2
	(m)	3900.30	400.5	8145.4	13868.8	2804.1	25929.5	3643.3	7680.3	997.6	13353.3	25204	5745.6
MuklasonEtAl17	(b)	6856	632	11659	16325	3837	27370	5528	9798	1246	14556	36810	5300
	(m)	7146	720	12574	18874	4157	28440	6006	10272	1320	15218	40752	5577
LeiteEtAl18	(b)	6207	535	13022	14302	3829	26710	5508	8716	1030	13894	39783	5142
	(m)	6478.20	572.90	13680.50	15493.70	4155.60	26873	5844.40	8942.30	1080.30	14208.70	43585.80	5249
Alsuwaylimi&	(b)	22109	22485	64089	43599	40012	41105	43014	92887	5187	50308	163590	8336
Fieldsend19: OBSI	(m)	27558.60	28859.26	75950	51200.4	68418.1	50069.33	53606.6	118309.23	7711	69737.33	219894.93	11620.8
OBSI + EA	(b)	12055	1705	17122	23154	6732	26420	17755	15022	1361	13658	45212	6186
	(m)	13092.66	2276.66	17482.7	24492.86	7095	29239.5	18491.03	15823.2	1528.5	14957.73	52998.81	6357

Table 5.2 Reported results of the ITC 2007 finalists and more recent approaches applied to the ITC 2007 benchmark set. Notation as in Table 5.1

competition organisers that participants can run, in order to stipulate the time for their *particular* machine to attempt to mitigate points 1 and 2. However, this is not the case for other benchmark sets like Toronto. Point 3 means that where algorithms are compared on the 'best' solution found over multiple repetitions (as opposed to the average) then simply increasing the number of repetitions of a stochastic optimiser, will improve the *expected* value of the observed best performance (see Table 5.5 for the various number of repetitions employed by different works). At the limit of repeated runs, any algorithm with an initialisation or variation capability that spans the space will return the global optima under the best found over multiple runs criteria.

Given points 1–3 above, comparisons in the literature relying on values from other published works are either unfair (due to allocated compute time differences), or are subject to bias effects. It is difficult to reasonably conclude that algorithm A is better than algorithm B, if the comparison is based on markedly different compute budgets and/or a substantially different number of chances to acquire the better cost score.

The question then arises, can we usefully extract *any* information from prior published results to apply a putative ranking on published performance, and contrast new results to these? Point 3 is difficult to mitigate for the best-returned value, as we do not have ready access to the distribution of costs for particular problems, and therefore the expected improvement due to repeated runs is difficult to model, bar via estimation approaches. However, we can retrospectively attempt to mitigate for points 1 and 2. This is possible through the additional information obtained from results published on the ITC 2007 benchmark. Furthermore, where additional moments are provided in papers (e.g. standard deviation and variance), the standard error in the approximation of the expectation from the mean can be derived.

Due to the aforementioned timing application distributed with ITC 2007 problems (Mccollum et al., 2010), we have access to the range of times employed by researchers on this problem set (see Tables 5.3 and 5.4). The times used have been allocated based on the competition tool's evaluation of the computation power of the device used to run the evaluated optimiser. This time budget is allocated by the tool so that all algorithms are effectively judged using the same *computational* budget. In the published works considered here, the fastest machine was allocated 276 seconds and the slowest machine 546 seconds. For some of the publications reporting ITC 2007 results and times, we *also* have access to the times the author(s) ran their approaches for on *other* problems (e.g. Toronto). We can, therefore, use the time range on ITC 2007, to generate an *uncertainty bound* on the *effective* (normalised) time for other problems. A diagrammatic representation including temporal information is shown in Figure 5.1. We emphasis that the second fastest machine and the slowest machine both were employed by different research teams in 2012, so regressing to the publication year alone is not sufficient to accurately determine the likely computing capability used for those works that do not have corresponding ITC 2007 results.

As mentioned above, the second issue in comparison in this field is that comparisons are conducted even if there is a variation of the computational abilities between approaches

	Running times (s) for Toronto benchmark sets.												
Approach	car91	car92	ear83	hec92	kfu93	lse91	pur93	rye92	sta83	tre92	uta92	ute92	yor83
CarterEtAl96	20.7	47	24.7	7.4	120	48	21729.4	507.2	5.7	107.4	664.3	9.1	174.5
Yang&Petrovic04	2773.2	1518	1135.2	1093.8	2152.8	1729.2	-	1381.8	739.8	1816.2	1890	745.8	1591.2
Eley07	16200	16200	16200	18	16200	16200	16200	16200	16200	16200	16200	16200	16200
CaramiaEtAl08	172	1120	300	48	320	446	12042	452	30	430	2046	20	452
Burke&Bykov08	664.8	604.2	450	589.8	882	640.8	747	901.2	586.8	607.8	805.2	528	528
BurkeEtAl10	5400	5400	120	120	120	120	-	-	120	120	5400	120	120
Pillay&Banzhaf10	5880	4260	751	451	3168	2863	113760	4320	469	1121	3639	663	552
DemeesterEtAl12	43200	43200	43200	43200	43200	43200	43200	43200	3600	43200	43200	43200	43200
Abdullah&Alzaqebah13	**	**	**	**	**	**	-	**	**	**	**	**	**
Alzaqebah&Abdullah14	9600	9600	300	300	300	300	-	300	300	300	9600	300	300
Alzaqebah&Abdullah15	9600	9600	600	600	600	600	-	600	600	600	9600	600	600
FongEtAl15	33120	31260	23460	13860	21900	15240	-	-	10680	24840	35280	12600	25200
LeiteEtAl16	86400	86400	86400	86400	86400	86400	172800	86400	86400	86400	86400	86400	86400
MuklasonEtAl17	360	360	360	360	360	360	360	360	360	360	360	360	360
LeiteEtAl18	172800	172800	86400	86400	86400	86400	172800	86400	86400	86400	172800	86400	86400
Alsuwaylimi&Fieldsend19:													
OBSI	11.02	5.41	12.05	8.16	42.47	53.56	167.97	74.24	0.12	60.52	147.2	4.21	70.32
OBSI + EA	2302	2328	2483	1347	778	186	2824	978	468	1618	2113	1203	457

Table 5.3 Running times (in seconds $t(s)$) for the Toronto benchmark set. (**) indicates that the authors did not report the algorithm execution time, and (-) means that the algorithm was not able to obtain a feasible solution.

considered. Thus, we noticed that most recent studies of solving the Toronto problem still consider any approach reported in the literature that obtained several best-known results even though its computational ability is quite out of date. This is because the best-known results reported for each problem instance in this dataset are still very competitive results to currently reported approaches. Hence, there is broadly a concern on the fairness of the comparison taking out in the literature right now. However, we actually address this

Running times (s) for ITC 2007 benchmark sets.												
Approach	Exam1	Exam2	Exam3	Exam4	Exam5	Exam6	Exam7	Exam8	Exam9	Exam10	Exam11	Exam12
Müller09	**	**	**	**	**	**	**	**	**	**	**	**
GogosEtAl08	422	422	422	422	422	422	422	422	422	-	422	-
AtsutaEtAl08	**	**	**	**	**	**	**	**	**	**	-	**
De Smet08	**	**	-	-	**	**	**	-	**	**	-	-
Pillay08	495	495	495	495	495	495	495	495	495	495	495	495
McCollumEtAl09	**	**	**	**	**	**	**	**	**	**	**	**
DemeesterEtAl12	300	300	300	-	300	300	300	300	300	300	-	300
GogosEtAl12	546	546	546	546	546	546	546	546	546	546	546	546
Alzaqebah&Abdullah14	460	460	460	460	460	460	460	460	-	-	-	-
Alzaqebah&Abdullah15	460	460	460	460	460	460	460	460	-	-	-	-
BattistuttaEtAl17	324	324	324	324	324	324	324	324	324	324	324	324
MuklasonEtAl17	360	360	360	360	360	360	360	360	360	360	360	360
LeiteEtAl18	276	276	276	276	276	276	276	276	276	276	276	276
Alsuwaylimi&Fieldsend19:												
OBSI	6.3	10.02	23.13	1.48	11.99	0.73	25.32	7.45	0.18	0.50	20.98	0.53
OBSI + EA	392	392	392	392	392	392	392	392	392	392	392	392

Table 5.4 Running times (in seconds $t(s)$) for the ITC 2007 benchmark set. Notation as detailed in caption of Table 5.3.

explicitly compensating for the variation in timing that has as identified being experienced in the historical papers. We found that they are competitive with many of the state-of-the-art approaches, which should not be precluded from consideration.

	Repetitions		
Algorithm	Toronto	ITC 2007	Reference
CarterEtAl96	10	-	(Carter et al., 1996)
Yang&Petrovic04	5	-	(Yang and Petrovic, 2005)
Eley07	20	-	(Eley, 2007)
Müller09	-	10	(Müller, 2008)
GogosEtAl08	-	10	(Gogos et al., 2008)
AtsutaEtAl08	-	10	(Atsuta et al., 2008)
De Smet08	-	10	(De Smet, 2008)
Pillay08	-	10	(Pillay, 2008)
CaramiaEtAl08	12	-	(Caramia et al., 2008)
Burke&Bykov08	20	-	(Burke and Bykov, 2008)
McCollumEtAl09	-	51	(McCollum et al., 2009)
BurkeEtAl10	100	-	(Burke et al., 2010b)
Pillay&Banzhaf10	10	-	(Pillay and Banzhaf, 2010)
DemeesterEtAl12	10	10	(Demeester et al., 2012)
GogosEtAl12	-	100	(Gogos et al., 2012)
Abdullah&Alzaqebah13	10	-	(Abdullah and Alzaqebah, 2013)
Alzaqebah&Abdullah14	11	11	(Alzaqebah and Abdullah, 2014)
Alzaqebah&Abdullah15	10	10	(Alzaqebah and Abdullah, 2015)
FongEtAl15	30	-	(Fong et al., 2015)
LeiteEtAl16	5	-	(Leite et al., 2016)
BattistuttaEtAl17	-	10	(Battistutta et al., 2017)
MuklasonEtAl17	21	21	(Muklason et al., 2017)
LeiteEtAl18	10	10	(Leite et al., 2018)
Alsuwaylimi&Fieldsend19	30	30	(Alsuwaylimi and Fieldsend, 2019)

Table 5.5 Number of repetitions for approaches that have been applied to the Toronto and the ITC 2007 benchmark sets. (-) indicates that algorithm was not applied to the corresponding benchmark set.

5.4 Uncertain Multi-Objective Analysis of Published Results

To facilitate putting an order (rank) on the published algorithm performances. We will employ a bi-objective comparison scheme based on (uncertain) Pareto dominance. If we consider the deterministic case of Pareto dominance first, algorithm x can be said to be better than (i.e. dominate) algorithm y if it returns an equal or better solution in an equal or faster time than algorithm y, but the performance on both criteria are not simultaneously equal. I.e. $x \prec y$ iff $f_c(x) \leq f_c(y) \wedge t(x) < t(y)$, or $f_c(x) < f_c(y) \wedge t(x) \leq t(y)$, where $t(x)$ returns the time taken by x to achieve the cost being compared under $f_c()$. If we consider the illustration in the left-hand panel of Figure 5.2, we can see that $b \prec c$, and $b \prec d$, but $b \not\prec a$ and $a \not\prec b$. In this situation, a and b are said non-dominated and given the same (highest) rank: b finds a solution with a lower cost than a, but takes twice as long to return its solution.

The right-hand panel of Figure 5.2 illustrates the situation we find ourselves in this study, with bounded uncertainty on the time due to the different relative performance of the machines used. As the uncertainty is of a bounded form, rather than with infinite tails (e.g. a Gaussian), we can still ascribe a dominance relation *with certainty* when we consider the bound extremes (Teich, 2001). Let us denote by $t_l(x)$ the lower bound of the time taken by algorithm x and $t_u(x)$ the upper bound of the time taken. As there is no

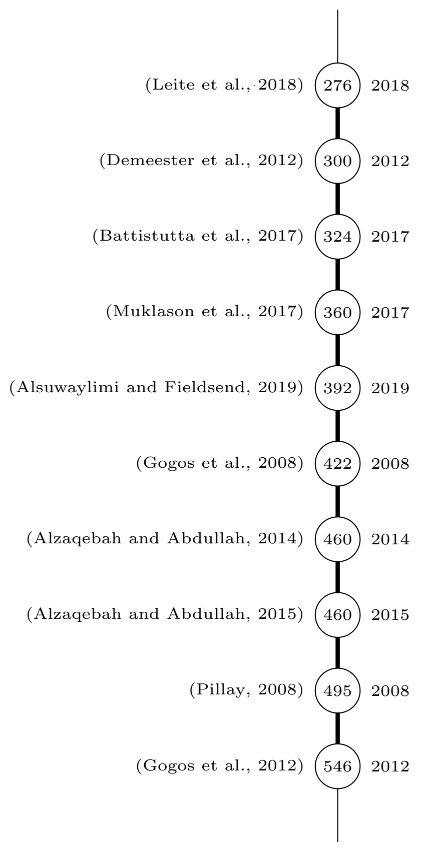
(Leite et al., 2018) 276 2018
(Demeester et al., 2012) 300 2012
(Battistutta et al., 2017) 324 2017
(Muklason et al., 2017) 360 2017
(Alsuwaylimi and Fieldsend, 2019) 392 2019
(Gogos et al., 2008) 422 2008
(Alzaqebah and Abdullah, 2014) 460 2014
(Alzaqebah and Abdullah, 2015) 460 2015
(Pillay, 2008) 495 2008
(Gogos et al., 2012) 546 2012

Figure 5.1 Times (s) employed on ITC 2007 benchmark datasets (as assigned by competition software tool) are shown inside the nodes. These are sorted from fastest (top of diagram) to slowest (bottom of diagram) and labelled by reference number and year of publication. Note for (Alsuwaylimi and Fieldsend, 2019) this is the time for OBSI+EA, as the initialisation by itself (OBSI) takes significantly under the budget assigned (which is the same budget as OBSI + EA, as experiments were conducted on the same machine).

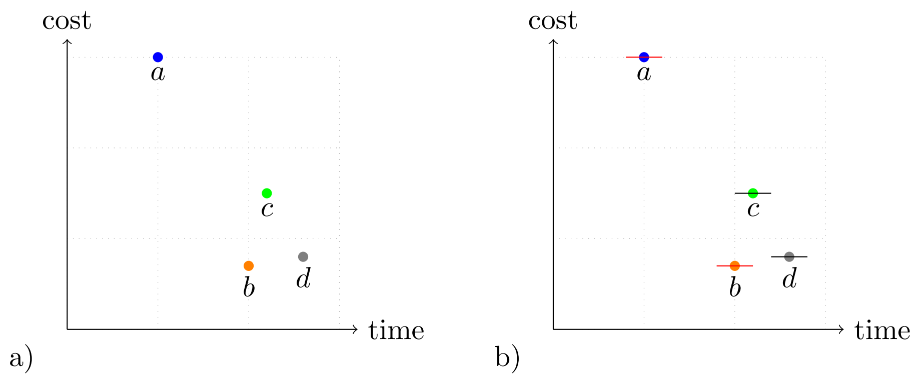

Figure 5.2 Illustration of uncertain multi-objective algorithm comparison. (a) Deterministic comparison. Algorithm b dominates algorithms c and d (it is faster and gives better solutions), and is mutually non-dominated with a (which returns a worse solution, but is much quicker). The uncertain Pareto set in this case is $\{a, b\}$. (b) Comparison using uncertainty bound on timings. Although b is still seen to dominate d, we cannot be certain that it dominates c due to the uncertainty over the relative effective timings (indicated by the width of the horizontal bars going through the respective operating points). The uncertain Pareto set in this case is $\{a, b, c\}$.

uncertainty on the solution cost returned, we can say with confidence that x is better than y iff $f_c(x) \leq f_c(y) \wedge t_u(x) < t_l(y)$, or $f_c(x) < f_c(y) \wedge t_u(x) \leq t_l(y)$. The next question is obvious: how can we determine the functions t_u and t_l?

Let us first identify the lowest and highest recorded times on ITC 2007 from the literature, which we label l and u. Now, for an algorithm x applied to e.g. a Toronto problem, with a reported time of $t(x)$, we can calculate its lowest *effective* time as $t_l(x) = t(x)l/u$ (which corresponds to the assumption that the results came from running the optimiser on a machine with a capability equivalent to the fastest machine used by those publishing ITC 2007 results) and its highest effective time as $t_u(x) = t(x)u/l$ (which corresponds to the assumption that the reported works were run on a machine with a capability equivalent to the slowest machine used in ITC 2007 works). Where a research group has applied their algorithm to both ITC 2007 *and*, e.g. Toronto in the same paper, (and they have reported their ITC 2007 allocated time), r, we can narrow the uncertainty bands further for the reported Toronto problems. Specifically, $t_l(x) = t(x)r/u$ and $t_u(x) = t(x)u/r$. These calculations allow us to identify a reasonable approximation to the *uncertain Pareto set* of algorithms from the literature, with respect to each exam scheduling problem which there are reported results on. This representation explicitly compensates for the uncertainty in the effective (machine-normalised) timings.

There is a chance that some works have used machines outside of the envelope of capabilities of works applied to the ITC 2007 suite, but we do not have any additional information to guide a further widening of the tolerances applied. Furthermore, by applying a hard boundary at the limits, we are treating the uncertainty estimate that we *do* have in the most conservative way possible. An alternative approach would be to use the different times recorded in Figure 5.1 to approximate a distribution on times, and exploit that to assign a soft domination probability rather than rely solely on the bounds, but we have insufficient data to do this robustly.

Using the approach detailed above, we can plot in the 2D solution cost/time space the performance (and corresponding uncertainty) of all the algorithms considered, for the Toronto problem instances on best performance, and (for those that report it) mean performance. Using this, we may identify the uncertain Pareto set of algorithm solutions — denoted $\mathcal{P}$, which efficiently trade-offs solution performance and time to return solution. Note that in all these past works, we have only a single point estimate of performance, but can infer the uncertainty band around this point.

Figures 5.3–5.4 shows the uncertain Pareto front, and dominated algorithms, for the 15 published approaches we consider that were evaluated on the Toronto benchmark on the best results reported. Figures 5.5–5.6 shows the uncertain Pareto front, and dominated algorithms, for the same algorithms on the mean results reported (note this is a subset of the algorithms shown in Figures 5.3–5.4, due to a number not reporting the mean).[2] Uncertain Pareto set members are highlighted in red on their uncertain timing bars. A number of interesting properties are immediately identifiable. Firstly, the relative position of algorithms in these scatter plots is not consistent across problems — even on the timings axis (for some problems an algorithm will be run for a longer period of time than another, and on other problems this is reversed). Likewise, the relative cost positions change

[2]Most works reporting the mean also reported the standard deviation, so we could represent the uncertainty in the expected cost performance via vertical bars in these plots — but we do not know if the run data are actually symmetrically distributed, or Normal.

between problems, and the relative positioning of algorithms on the *same* problem under best performance analysis is also often at variation with the relative mean performance positioning. Even when taking into consideration the uncertainty over timings it is clear that some algorithms are better than others on particular problems. However, there are between 3 and 10 approaches on each problem which are *incomparable* under an uncertain Pareto analysis for best performance, and between 6 and 9 for the corresponding mean performance analysis.

	Best cost		Mean cost	
Approach	# Times in $\mathcal{P}$	% Times in $\mathcal{P}$	# Times in $\mathcal{P}$	% Times in $\mathcal{P}$
CarterEtAl96	12	92	-	-
Yang&Petrovic04	5	38	6	46
Eley07	1	8	2	15
Burke&Bykov08	10	77	13	100
CaramiaEtAl08	12	92	-	-
BurkeEtAl10	7	54	-	-
Pillay&Banzhaf10	1	8	4	31
DemeesterEtAl12	3	23	6	46
Alzaqebah&Abdullah14	4	31	9	69
Alzaqebah&Abdullah15	5	38	10	77
FongEtAl15	1	8	1	8
LeiteEtAl16	3	23	11	85
MuklasonEtAl17	5	38	12	92
LeiteEtAl18	5	38	12	92
Alsuwaylimi&Fieldsend19 (OBSI)	13	100	13	100
Alsuwaylimi&Fieldsend19 (OBSI+EA)	9	69	9	69

Table 5.6 Number and percentage of times based on the best and the mean costs for each above algorithm appears in uncertain Pareto set of optimal algorithm. A '-' indicates the authors did not report the mean costs.

If we consider the number of times an algorithm appears in the top-ranked group of non-dominated approaches (i.e. $\mathcal{P}$) across the Toronto problem instances, we can identify two different global rankings of approaches for best and mean performance, which is shown in Table 5.6. There is some degree of correlation between the two rankings, but they are not identical. At the extreme, if we consider the approach of Leite et al. (2016), it is in the best cost based uncertain Pareto front for Toronto 3 out of 13 problems, but the mean cost based uncertain Pareto front for Toronto 11 out of 13 problems.

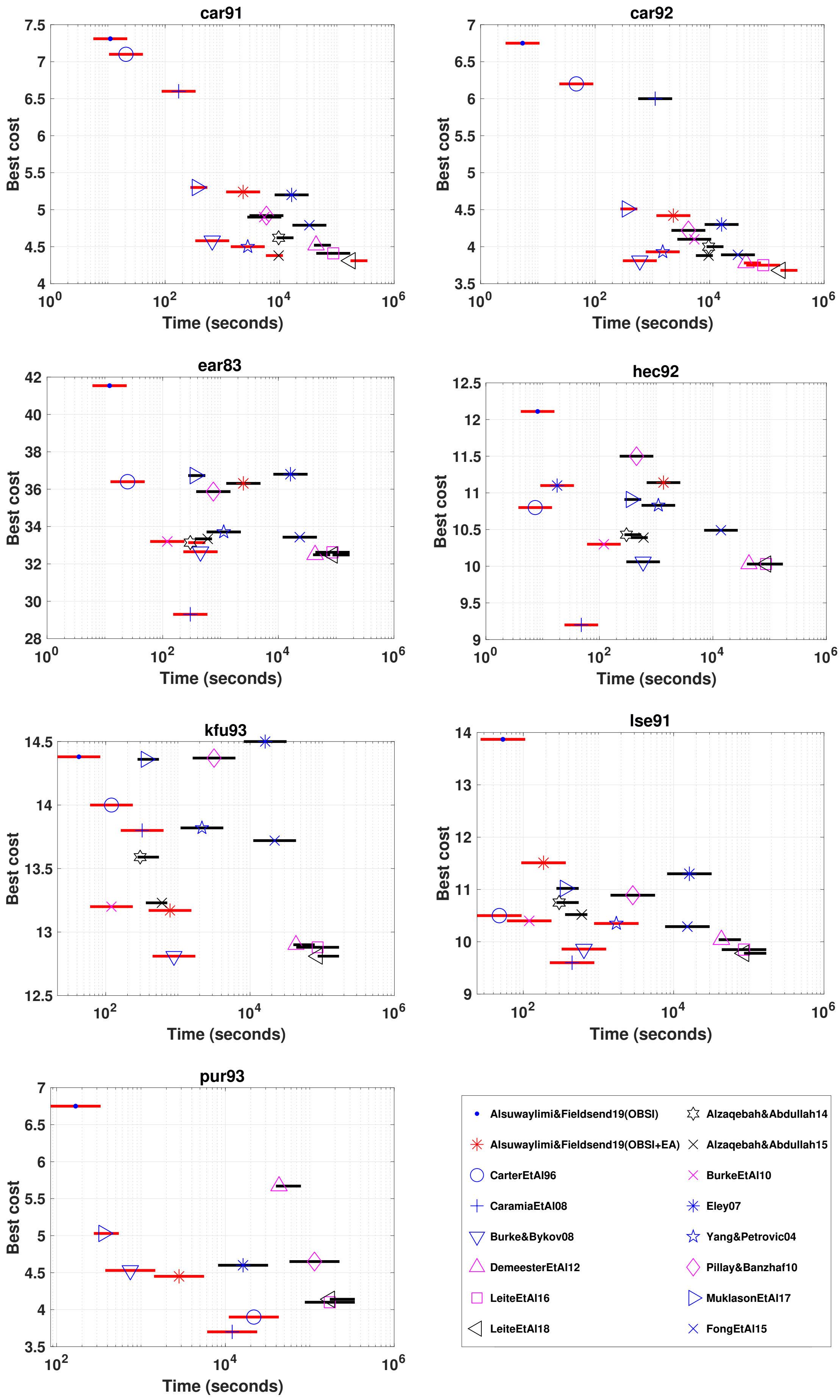

Figure 5.3 Uncertain Pareto fronts of published algorithm results (**the best cost over multiple runs**) on the Toronto problem instances 1–7. Horizontal uncertain timing bar coloured red for uncertain Pareto front members (i.e. $\mathcal{P}$).

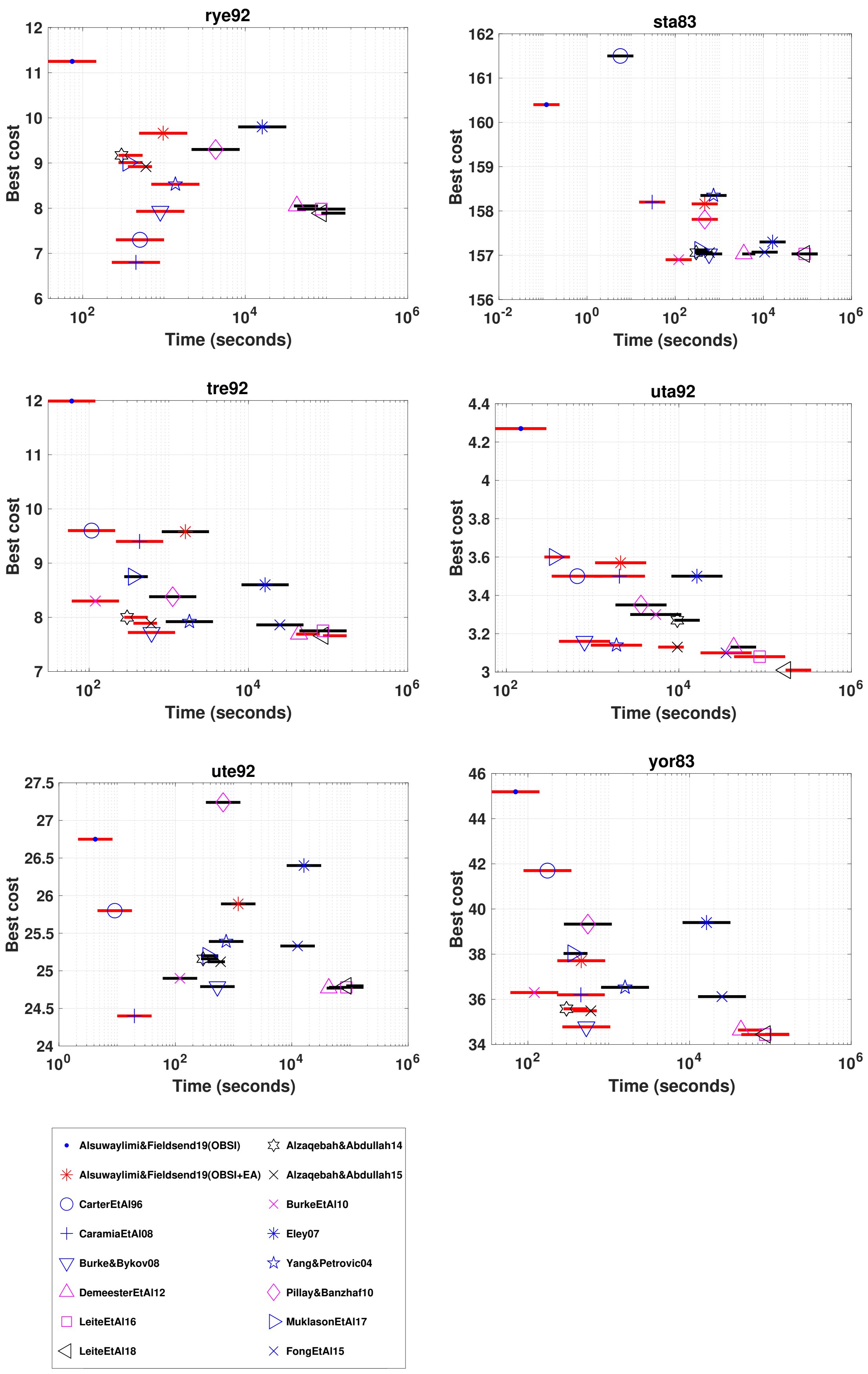

Figure 5.4 Uncertain Pareto fronts of published algorithm results (**the best cost over multiple runs**) on the Toronto problem instances 8–13. Horizontal uncertain timing bar coloured red for uncertain Pareto front members (i.e. $\mathcal{P}$).

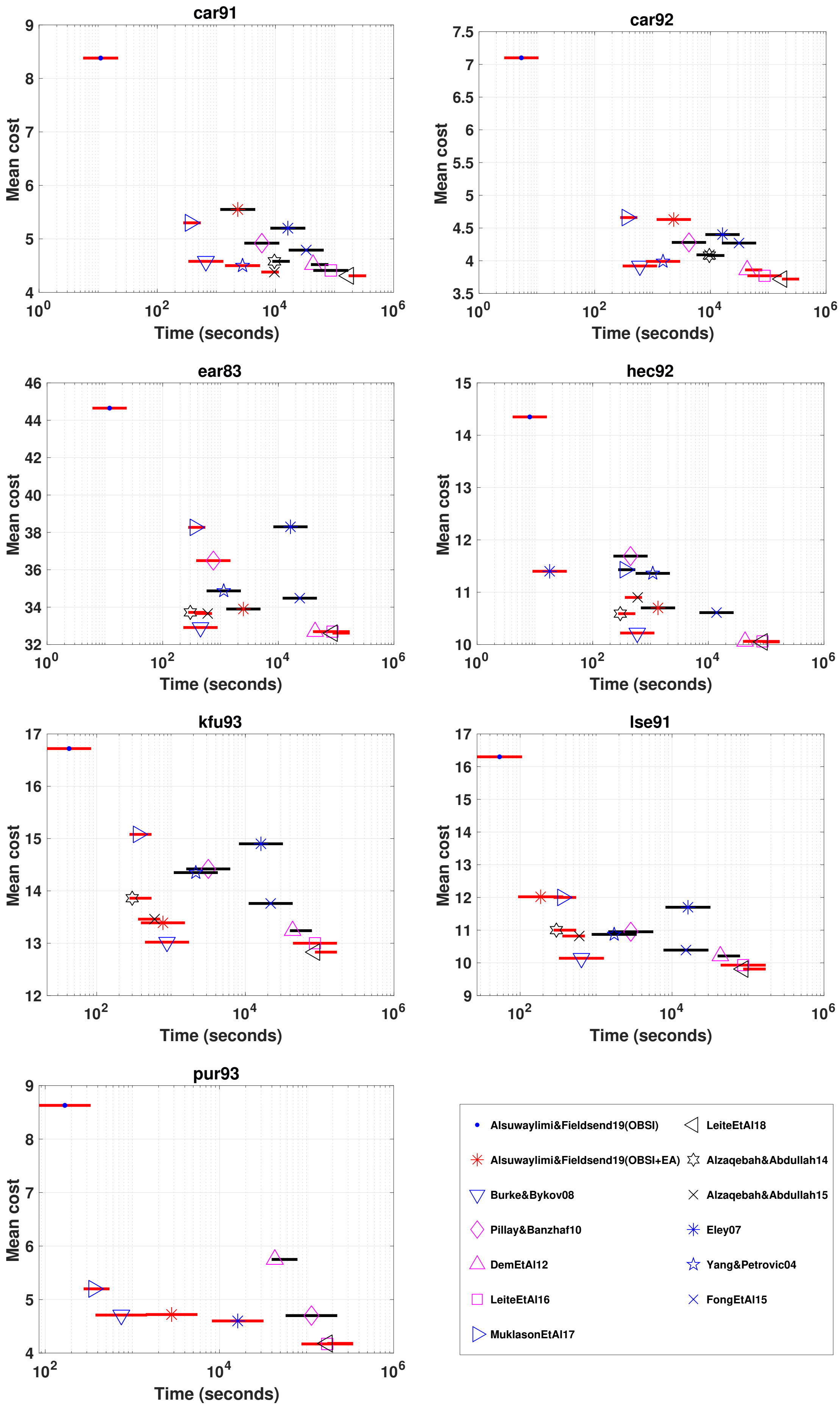

Figure 5.5 Uncertain Pareto fronts of published algorithm results (**the mean cost over multiple runs**) on the Toronto problem instances 1–7. Horizontal uncertain timing bar coloured red for uncertain Pareto front members (i.e. $\mathcal{P}$).

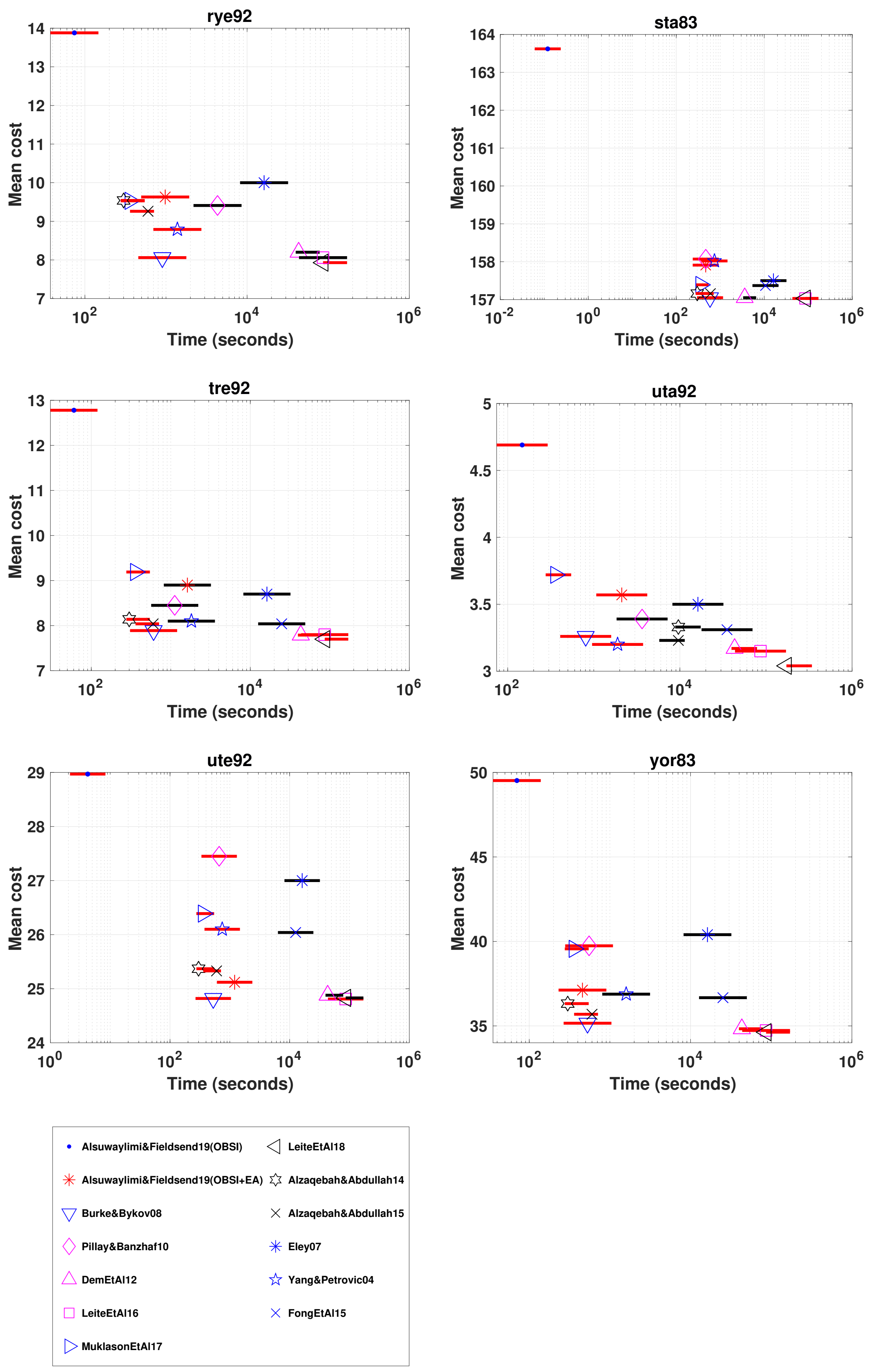

Figure 5.6 Uncertain Pareto fronts of published algorithm results (**the mean cost over multiple runs**) on the Toronto problem instances 8–13. Horizontal uncertain timing bar coloured red for uncertain Pareto front members (i.e. $\mathcal{P}$).

It is also interesting to note that there is no algorithm from this collection — which spans more than two decades of research and development — which is not in the uncertain optimum set for at least one problem instance, under both best and mean performance criteria. That is, all algorithms considered are competitive (or incomparable) on at least one of the problems — and a few of the approaches are competitive/incomparable on many. This could suggest that even in this narrow application domain, there may still be *no free lunch* (Wolpert and Macready, 1997).

The analysis above does not preclude that a particular algorithm x which has found a better solution than y, albeit in twice the time, might not find a better solution than y if it was run for half the time — and therefore, in reality, dominate it. However, we only have access to single performance combinations for these published works. This leads us on to one of the key points leading from this analysis. Ideally an anytime performance approach should be employed (Grass and Zilberstein, 1996; Zilberstein, 1996; Dean and Boddy, 1988), as used in other optimisation comparison areas (Shih and Liu, 1995; Zhang et al., 2003; Dubois-Lacoste et al., 2015; Nguyen et al., 2016). Using such an approach would result in performance *curves.* Corresponding uncertainty bounds on timings could also be applied if algorithms are not run on the same machine when can still compare them. However, as of yet, this approach appears not to have gained traction in the exam timetabling community.

As well as problem-level comparison and benchmark-wide Pareto set memberships ranking, I now also provide a benchmark-wide pair-wise uncertain dominance comparison between algorithms tested on thirteen problem instances of the Toronto dataset in Table 5.7 with the discussion:

The first row cells in Table 5.7 includes algorithms, and the first column cells also consist of the same algorithms. The other cells consist of two values: the first value is the count of the number of problem instances in benchmark sets in which the algorithm in the row certainly dominated another algorithm in the column. The second value represents the count of the number of problem instances where the algorithm in the column certainly dominated the algorithm in the row. Therefore, for example, the value between algorithms a and b is 3/5 means the algorithm a dominated the algorithm b in three problem instances. The algorithm a was also dominated by the algorithm b in five problem instances. For a given value of 0/0, it indicates that none of them dominated the other.

Benchmark-wide pair-wise comparisons were conducted on the same sixteen algorithms that were used to apply pair-wise instance-level comparisons. The comparisons showed that Alzaqebah&Abdullah14 dominated Yang&Petrovic04 four times (i.e. four problem instances), while Yang&Petrovic04 dominated Alzaqebah&Abdullah14 in three different problem instances. Moreover, Alsuwaylimi&Fieldsend19 (OBSI+EA) dominated Pillay&Banzhaf10 in two problem instances, whereas Pillay&Banzhaf10 could not dominate Alsuwaylimi&Fieldsend19 (OBSI+EA) in any problem instances. Elay07 also was dominated in twelve problem instances by Burke&Bykov08. In addition, LeiteEtAl16 and LeiteEtAl18 were not able to dominate any other algorithms. Unlike LeiteEtAl16 and LeiteEtAl18, Alsuwaylimi&Fieldsend19 (OBSI) overwhelmed all other algorithms. Furthermore, CaramiaEtAl08 was capable of beating all algorithms for *all* problem instances except Alsuwaylimi&Fieldsend19 (OBSI) and CarterEtAl96. Conversely, Eley07 was dominated by others except for LeiteEtAl16 and LeiteEtAl18.

The last column cells in Table 5.7 contains two values: the first one indicates the total number of times the algorithm in a row defeats the other algorithms, and the second one refers to the total number of times the algorithm in a row is defeated by others. As shown in the last column, CaramiaEtAl08 and BurkeEtAl10 have the highest figure in terms of the number of times they dominated the other algorithms. However, CaramiaEtAl08 was defeated only once (i.e. by MuklasonEtAl17), whilst BurkeEtAl10 was prevailed eight times (i.e. algorithms) across different instances. Therefore, the comparisons showed that CaramiaEtAl08 is far superior (in terms of the results' quality and the time it takes to compute them) to the other algorithms. Furthermore, we found significant variances in the overall performance comparisons regarding pair-wise instance-level and benchmark-wide pair-wise comparisons. In other words, it is not necessary for the top algorithm in the benchmark-wide pair-wise comparison to being among the top algorithms in the pair-wise instance-level comparison.

5.5 Summary

Exam scheduling is a common problem encountered in educational institutions and is an active area of optimisation research. Over four decades, much research in this area has been undertaken. This has covered both formulating and modeling examination timetabling problems, and developing domain specific optimisation approaches. Based on the nature of the exam timetabling problem, the exact constraints to satisfy my vary from problem instance to another, affecting what is desired in an exam timetable solution. This chapter has presented a detailed description of two widely used benchmark datasets — the Toronto and ITC 2007 datasets. We have also presented the most successful approaches which have been applied to both these popular benchmarks in the last two decades. We compare these works with respect to their published results, however these data have been directly obtained from corresponding articles (as implementations were often not available). As such, the running times refer to various platforms, different implementation languages, compilation optimisations, where the effect of these factors is uncertain. Such issues are widespread when comparing algorithms performance solely in terms of performance obtained at a certain run-time. Other statistical issues also arise in the field, where comparison between approaches is often conducted depending on only on the extreme value obtained over a non-standard number of repetitions. In the ITC 2007, the time limit is determined by a benchmarking tool provided from the ITC 2007 competition organisers, however there is no such benchmark timing attempt for the works published on the Toronto problems. As such, the results of published works are often compared even when the studies have significantly different generation circumstances, such as wall-clock time and employed compute capability, and different repetitions used to generate the extreme performance value. We have derived an analysis that mitigates for timing and capability effects based on (bounded) uncertain Pareto dominance, and have shown that for all published optimiser works considered there is at least one problem instance

Algorithm	CarterEtAl96	Yang&Petrovic04	Eley07	Burke&Bykov08	CaramiaEtAl08	BurkeEtAl10	Pillay&Banzhaf10	DemeesterEtAl12	Alzaqebah&Abdullah14	Alzaqebah&Abdullah15	FongEtAl15	LeiteEtAl16	MuklasonEtAl17	LeiteEtAl18	OBSI	OBSI+EA	**Total**
CarterEtAl96	—	1/0	6/0	0/0	0/0	0/0	6/0	1/0	1/0	1/0	0/0	2/0	4/0	2/0	0/1	2/0	26/1
Yang&Petrovic04	0/1	—	10/0	0/0	0/4	0/6	0/0	1/0	3/4	0/3	1/0	0/0	0/0	0/0	0/0	0/0	15/18
Eley07	0/6	0/10	—	0/12	0/7	0/7	2/6	1/1	0/8	0/8	0/1	0/0	0/7	0/0	0/1	1/6	4/80
Burke&Bykov08	0/0	0/0	12/0	—	0/2	3/1	6/0	5/0	3/0	1/0	10/0	4/0	0/0	3/0	0/0	1/0	48/3
CaramiaEtAl08	0/0	4/0	7/0	2/0	—	1/0	6/0	6/0	2/0	2/0	4/0	6/0	2/1	6/0	0/0	3/0	51/1
BurkeEtAl10	0/0	6/0	7/0	1/3	0/1	—	6/0	1/0	5/0	6/0	5/0	1/0	8/0	1/0	0/0	4/0	51/8
Pillay&Banzhaf10	0/6	0/0	6/2	0/6	0/6	0/6	—	0/0	0/4	0/3	0/0	0/0	0/2	0/0	0/1	0/2	6/38
DemeesterEtAl12	0/1	0/1	1/1	0/5	0/6	0/1	0/0	—	0/0	0/2	0/0	1/0	0/1	3/0	0/0	0/1	5/19
Alzaqebah&Abdullah14	0/1	4/3	8/0	0/3	0/2	0/5	4/0	0/0	—	0/0	6/0	0/0	0/0	0/0	0/0	4/0	26/14
Alzaqebah&Abdullah15	0/1	3/0	8/0	0/1	0/2	0/6	3/0	2/0	0/0	—	8/0	1/0	0/0	0/0	0/0	2/0	27/10
FongEtAl15	0/0	0/1	1/0	0/10	0/4	0/5	0/0	0/0	0/6	0/8	—	0/0	0/1	0/0	0/0	0/1	1/36
LeiteEtAl16	0/2	0/0	0/0	0/4	0/6	0/1	0/0	0/1	0/0	0/1	0/0	—	0/0	0/0	0/0	0/0	0/15
MuklasonEtAl17	0/4	0/0	7/0	0/0	1/2	0/8	2/0	1/0	0/0	0/0	1/0	0/0	—	0/0	0/0	3/0	15/14
LeiteEtAl18	0/2	0/0	0/0	0/3	0/6	0/1	0/0	0/3	0/0	0/0	0/0	0/0	0/0	—	0/0	0/0	0/15
OBSI	1/0	0/0	1/0	0/0	0/0	0/0	1/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	—	0/0	3/0
OBSI+EA	0/2	0/0	6/1	0/1	0/3	0/4	2/0	1/0	0/4	0/2	1/0	0/0	0/3	0/0	0/0	—	10/20

Table 5.7 Benchmark-wide pair-wise uncertain dominance comparison between the algorithms on the Toronto benchmark sets.

where we cannot be certain it is not among the optimal set of algorithms, i.e. those which most effectively trade-off solution performance for speed of solution return. The recent OBSI approach is in the uncertain Pareto set for *all* problem instances, however as this is a smart *initialisation* rather than optimisation approach this membership is primarily derived by being much faster than all of the competitors. Nevertheless, it is also seen to directly dominate some optimisers on a few problems also, and the OBSI+EA optimiser variant is in the set of optimal trade-off solutions for over two-thirds of the problems considered. The next chapter provides a summary of all the work reported in this thesis and some suggestions for future development.

Chapter 6

Conclusions and Future Work

6.1 Introduction

This chapter presents a summary of the research work that has been conducted in this thesis. The proposed approaches (i.e. the OBSI method and the ESGA) for constructing and tackling uncapacitated and capacitated examination timetabling problems are briefly presented in Section 6.2. In this same section, the proposed multi-objective approach to analyses published results is summarised, along with the comparison with previously-published algorithms. Potential avenues for future work are highlighted in Section 6.3.

6.2 Research Summary

The main contribution of Chapter 3 was to present a new initialisation method (i.e. the Ordering-Based Scheduling Initialisation (OBSI)) to construct solutions for examination timetabling problems. We tested the proposed initialisation approach on capacitated examination timetabling problem instances (ITC 2007 examination track (McCollum et al., 2007) and Yeditepe (Parkes and Ozcan, 2010)) and compared the obtained results with popular initialisation approaches from this domain. This showed that the OBSI method is competitive with other initialisation approaches reported in the literature. The proposed method manages three interacting lists of exams (i.e. Front List, Middle List, and Back List) to allocate exams into three corresponding sections of the timetable (i.e. Front section, Middle Section, and Back Section). These lists are arranged and processed in a step-wise fashion in order to construct a good satisfaction of both hard and soft constraints, particularly the expand constraints, which involve the "two exams in a row" and "two exams in a day" constraints. The goal of this research is to develop an initialisation method in which all solutions are feasible, and also the most challenging and highly-constrained exams are scheduled in the most appropriate periods. However, constructing an initial exam timetable while taking into account its quality is expensive and arduous. Thus, most approaches exclusively dedicated to examination timetabling problems have employed other strategies such as squeaky wheel optimisation (Joslin and Clements, 1999), graph colouring heuristics (Burke and Newall, 2004), and decomposition techniques

(Qu and Burke, 2007) to produce only feasible solutions in the construction phase. This motivated the attempt in this work to create a method that would focus on producing timetables that fulfil hard constraints and minimise the cost of the soft constraints.

The proposed method was compared to the most commonly-used graph colouring heuristics in exam timetabling (namely, largest degree, largest weighted degree, largest enrolment and saturation degree) as well as to random schedule allocation. Most studies on examination timetabling problems have applied graph colouring heuristics, depending on ordering strategies in the first phase in order to construct initial feasible timetables. However, these studies have not considered the quality of the initial timetables, rather than attempting to produce feasible solutions initially and then improving the quality of these in the second phase. The experimental results revealed that the proposed method effectively constructed a feasible solution for all problem instances from the two different datasets. More specifically, it outperformed graph colouring heuristics and random approaches on nearly all of the ITC 2007 benchmark instances for both quality and diversity, as well as some of the Yeditepe benchmark instances for quality. In order to improve the initial feasible solutions and compare the effectiveness of the proposed method with other methods, all of these methods were incorporated within a simple evolutionary algorithm (EA) using the same parameter settings and objective function. This comparison indicated that the incorporation of the proposed method within the EA not only resulted in relatively high-quality solutions compared to other initialisation approaches but that these solutions were effective for seeding an efficient population-based search. Furthermore, most problem instances using the initial OBSI outcome to generate initial solutions surpassed the quality of runs using other initialisation strategies, and this was achieved for the same total time cost (initialisation time plus optimisation time).

Chapter 4 proposed a novel Genetic Algorithm, with a range of novel parameters and operators to tackle particular examination timetabling problems. This new variant is called the Exam Specialised Genetic Algorithm (ESGA). The operators involve varied types of operation, which are applied to examinations or periods. Several mutation types (i.e. MM, PBM, DM, PBDM, HM, LM, and ELM) are applied to certain individuals under certain circumstances. These include new parameters that aim to achieve a successful balance between exploration and exploitation by preventing premature convergence and controlling the size of the mutation and the probability that the mutation will be applied. This proposed algorithm, along with a basic GA, uses the OBSI method with the aim of generating the first population so that fair comparisons are conducted. The Toronto, ITC 2007, and Yeditepe benchmark sets were used for comparison with the Basic GA. The findings revealed that the ESGA achieved the best average fitness for all the problem instances of the datasets and that it required more execution time. A comparison was also conducted between the ESGA results for the three datasets and those of other state-of-the-art approaches. Furthermore, the results of this work were also compared with the top five performing algorithms in the ITC 2007 competition. It was found that the improved approach reached and beat most contestants. Regarding the Yeditepe dataset, the proposed algorithm results were compared with the results of state-of-the-art methods. This showed that the proposed approach is as efficient as other approaches, reaching the level of most of those alternative algorithms in most problem instances and obtaining a new best result for the YUE20023 instance.

Chapter 5 highlights several issues that arise when comparing the performance of published algorithms on common examination timetabling benchmarks. These are often compared even when they have significantly different generation circumstances, such as wall-clock time and the computation capability employed. We have derived an uncertain Pareto analysis approach that mitigates for timing and capability effects based on (bounded) uncertain Pareto dominance, and show that for all the published optimisation works considered, there is at least one problem instance where we cannot be certain that is not among the optimal set of algorithms trading-off solution cost for speed of solution return.

We have observed that under the uncertain Pareto analysis all approaches considered provide a competitive (non-dominated) performance on at least one problem instance from the Toronto benchmark: even in this narrow application domain, there appears to be "no free lunch". However, there is a great variation between optimisers in terms of the number of problems in which their performance is non-dominated.

6.3 Future Work

This thesis has presented several new approaches for tackling the problem of university examination timetabling, generating competitive results compared with other examples in the literature for the main benchmark problem instances. It is challenging to generalise a methodology for solving the examination timetabling problem, however, because restrictions (i.e. constraints) vary significantly from one institution to another. In order to enhance the efficiency of the proposed approaches, several suggestions can be considered in future work. Several of these are identified below.

6.3.1 Improving the Investigated Approaches

As seen in Chapter 3, this work has focused on applying a new initialisation strategy (OBSI) to generate initial solutions for the ITC 2007 and Yeditepe benchmark sets. The proposed OBSI was compared against other initialisation strategies (i.e. several graph colouring heuristics and the random schedule allocation method), and additionally, the effectiveness of the proposed approach, and other approaches, were also compared by incorporating them in turn in a simple evolutionary algorithm. Each solution constructed by the proposed OBSI has been represented in a way that simulates reality (indirect representation); therefore the solution consists of a list of periods available in exam session, where each period contains a predefined list of the rooms that will include as many exams as possible depending on the satisfaction of the stipulated constraints and their capacities.

Also in Chapter 3, it can be noted that generating an initial solution with the four graph colouring heuristics can be achieved more quickly than with the OBSI, albeit delivering lower quality solutions. The drawback of the proposed OBSI, therefore, is that it requires significantly higher computational time to construct an initial population (particularly for the ITC 2007 datasets). This is because when the proposed OBSI attempts to schedule an exam into a period, the duration property of the exam is compared with the duration property of the period in order to satisfy the period duration constraint, and this process adds computational time. While in real-world examination timetabling problems, the time taken to generate the examination timetables is often not critical, in the timetabling literature, the methods are required to produce good results with fast execution times. It would be interesting to see if this proposed algorithm obtains results very quickly, taking into account the quality and diversity. Efficiency time could be realised by identifying a pre-determined mechanism that aims to satisfy the period duration constraint in which it determines an appropriate list (i.e. FL, BL, and ML) for each exam before assigning the exam into the list. With this mechanism, each exam would consist of a list of valid periods from the corresponding section (i.e. FS or BS or MS) that must be greater than or equal to the duration of the exam. The revised OBSI would then attempt to schedule an exam in a period from that valid list.

6.3.2 Anytime Analysis

An anytime algorithm is defined as an algorithm that has the capability to return solutions to a specific problem whenever it is interrupted (Dean and Boddy, 1988). This is in contrast with traditional algorithms which do not guarantee solutions until a certain stopping criterion is met. Moreover, this concept always offers a trade-off between solution quality and execution time.

The work in Chapter 5 underpins the need for the use of *anytime* analysis in the area of exam timetable optimisation going forward (which has already gained traction in other optimisation areas) (Shih and Liu, 1995; Zhang et al., 2003; Dubois-Lacoste et al., 2015; Nguyen et al., 2016) – as this would allow a performance curve to be traced out for each algorithm, and therefore may identify a much smaller set of high performance algorithms. Although, even in this situation, an *uncertain* performance curve analysis would need to be employed to compensate for uncertainties in compute capabilities, alongside the error in the estimation of expected performance. We also assert that using the best result found over multiple runs rather than the average is a poor choice — particularly when there is no fixed number of repetitions mandated. Compounding this is the fact that the distribution form of a heuristic algorithm's performance is commonly not known, and indeed for many distributions there is no known unbiased estimator for the maximum or minimum in any respect (D'Eramo et al., 2016).

6.3.3 Hybridisation

The findings in Chapter 3 showed the advantage of using OBSI to generate an initial population of high quality and diverse solutions lies in the quality of the final timetables returned. Accordingly, the quality of final examination timetables is determined by the quality of the initial examination timetables. The limitation of this work is that the performance of the proposed OBSI is only evaluated by incorporating it within a simple EA. It would be very interesting to integrate this new approach into some recently proposed state-of-the-art timetabling meta-heuristics. We expect that such an integration would produce outcomes that would be even more competitive with other results published in the literature. Furthermore, the examination timetables generated by the proposed OBSI (as shown in chapter 3) could be further improved by implementing many meta-heuristic algorithms. It may be worth investigating this when the algorithms start from the same seeded solutions. The objective would be to select the meta-heuristic that would obtain the best improvement.

The proposed approach investigated in Chapter 4 is open for hybridisation with some local search techniques. Hybridisations of GA with local search techniques could lead to further improvements. The objective would be to find a local search algorithm that can work and increase the speed of the proposed GA. Future work could also study hill-climbing methods (Qu et al., 2009b; Rahim et al., 2013).

6.3.4 Parameter Tuning

It is known that parameter tuning is a very effective way to enhance many of the techniques discussed in this thesis. Chapter 4 offered an intensive empirical investigation of parameter tuning. Several preliminary experiments were conducted in order to determine appropriate values for parameters, such as the number of iterations, the size of the population, the size of the elite pool, the probability of a solution being selected for mutation and the size of the mutation supplied by each mutation operator. The limitation of this work is that a tailor-made approach for getting good parameter settings was needed for the proposed GA in order to produce high-quality and competitive solutions for all problem instances tested in the experiments. Future research work could investigate an adaptation approach and automated parameter tuning strategy.

6.3.5 Many-objective Optimisation Problem Formulation and Applying Many-objective Optimisation Approaches

As discussed in Section 2.11 in Chapter 2, many meta-heuristic methods that have utilised to tackle the examination timetabling problem as a single objective optimisation problem have also been extended for use with multi-objective problems (Muklason, 2017; Coello et al., 2007). Although some previous studies have sought to employ multi-objective meta-heuristics in examination timetabling problems, these have mainly been conducted on uncapacitated examination timetabling problems (i.e. the Toronto dataset) (Côté et al., 2005), with only a very few investigating multi-objective approaches to the capacitated ITC 2007 dataset (e.g. (Muklason et al., 2017)). It would be worthwhile to investigate how multi-objectivisation and the use of many-objective optimisation approaches can improve the search performance and the quality of the solutions found for the ITC 2007 problem while minimising violations of soft constraints. This could be done by casting the ITC 2007 as many-objective optimisation problem and considering all its soft constraints (i.e seven soft constraints) as objectives in this problem (i.e. C^{2R}, C^{2D}, C^{PS}, C^{NMD}, C^{FL}, C^{R}, and C^{P}) and attempting to minimise the violations in parallel.

Bibliography

Aarts, E. and Korst, J. (1989). Simulated annealing and boltzmann machines: a stochastic approach to combinatorial optimization and neural computing, 1989. *Great Britain: John Wiley & Sons.*

Ab Malik, A. M., Ayob, M., and Hamdan, A. R. (2009). Iterated two-stage multi-neighbourhood tabu search approach for examination timetabling problem. In *Data Mining and Optimization, 2009. DMO'09. 2nd Conference on*, pages 141–148. IEEE.

Ab Malik, A. M., Othman, A. K., Ayob, M., and Hamdan, A. R. (2011). Hybrid integrated two-stage multi-neighbourhood tabu search-emcq technique for examination timetabling problem. In *Data Mining and Optimization (DMO), 2011 3rd Conference on*, pages 232–236. IEEE.

Abdul-Rahman, S., Abdullah, S. S. S., and Benjamin, A. M. (2017a). A nonlinear heuristic modifier for constructing examination timetable. *Journal of Theoretical and Applied Information Technology*, 95:5642–5653.

Abdul-Rahman, S., Benjamin, A. M., Faizal, O. M., Ramli, R., u Mahamud, K.-R., and Abdul Jabbar, W. K. (2017b). Designing and implementation a web-based architecture for an examination timetabling system. *Journal of Engineering and Applied Sciences*, 12:7299–7305.

Abdul-Rahman, S., Burke, E. K., Bargiela, A., McCollum, B., and Özcan, E. (2014a). A constructive approach to examination timetabling based on adaptive decomposition and ordering. *Annals of Operations Research*, 218(1):3–21.

Abdul-Rahman, S., Burke, E. K., Bargiela, A., McCollum, B., and Özcan, E. (2014b). A constructive approach to examination timetabling based on adaptive decomposition and ordering. *Annals of Operations Research*, 218(1):3–21.

Abdul-Rahman, S., Sobri, N. S., Omar, M. F., Benjamin, A. M., and Ramli, R. (2014). Graph coloring heuristics for solving examination timetabling problem at Universiti Utara Malaysia. In *American Institute of Physics Conference Series*, volume 1635 of *American Institute of Physics Conference Series*, pages 491–496.

Abdullah, S., Ahmadi, S., Burke, E. K., and Dror, M. (2007a). Investigating ahuja–

orlin's large neighbourhood search approach for examination timetabling. *OR Spectrum*, 29(2):351–372.

Abdullah, S., Ahmadi, S., Burke, E. K., Dror, M., and McCollum, B. (2007b). A tabu-based large neighbourhood search methodology for the capacitated examination timetabling problem. *Journal of the Operational Research Society*, 58(11):1494–1502.

Abdullah, S. and Alzaqebah, M. (2013). A hybrid self-adaptive bees algorithm for examination timetabling problems. *Applied Soft Computing*, 13(8):3608–3620.

Abdullah, S., Burke, E. K., and McCollum, B. (2005). An investigation of variable neighbourhood search for university course timetabling. In Kendall, G., Lei, L., and Pinedo, M., editors, *In proceedings of the 2nd Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA 2005), 18 -21 July 2005, New York, USA*, pages 413–427. Paper.

Abdullah, S. and Turabieh, H. (2012). On the use of multi neighbourhood structures within a tabu-based memetic approach to university timetabling problems. *Information Sciences*, 191:146–168.

Abdullah, S., Turabieh, H., and McCollum, B. (2009). A hybridization of electromagnetic-like mechanism and great deluge for examination timetabling problems. In Blesa, M. J., Blum, C., Di Gaspero, L., Roli, A., Sampels, M., and Schaerf, A., editors, *Hybrid Metaheuristics*, pages 60–72, Berlin, Heidelberg. Springer Berlin Heidelberg.

Abido, M. A. (2002). Optimal power flow using tabu search algorithm. *Electric Power Components and Systems*, 30(5):469–483.

Abuhamdah, A. and Ayob, M. (2010a). Adaptive randomized descent algorithm for solving course timetabling problems. *International Journal of Physical Sciences*, 5(16):2516–2522.

Abuhamdah, A. and Ayob, M. (2010b). Average late acceptance randomized descent algorithm for solving course timetabling problems. In *Information Technology (ITSim), 2010 International Symposium in*, volume 2, pages 748–753. IEEE.

Adamidis, P. and Arapakis, P. (1999). Evolutionary algorithms in lecture timetabling. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1145–1151 Vol. 2.

Adnan, F. A., Ab Saad, S., Yahya, Z. R., and Wan Muhamad, W. Z. A. (2018). Genetic algorithm method in examination timetabling problem: A survey. In Yacob, N. A., Mohd Noor, N. A., Mohd Yunus, N. Y., Lob Yussof, R., and Zakaria, S. A. K. Y., editors, *Regional Conference on Science, Technology and Social Sciences (RCSTSS 2016)*, pages 901–907, Singapore. Springer, Singapore.

Ahandani, M. A. and Vakil-Baghmisheh, M.-T. (2011). Examination timetabling using a hill climbing with combined neighbourhood structure. *Computer and Knowledge Engineering (ICCKE). 2011 1st International eConference on*, pages 13–14.

Al-Betar, M. A., Khader, A. T., and Doush, I. A. (2014). Memetic techniques for examination timetabling. *Annals of Operations Research*, 218(1):23–50.

Al Jadaan, O., Rajamani, L., and Rao, C. (2008). Improved selection operator for ga. *Journal of Theoretical & Applied Information Technology*, 4(4).

Al-Yakoob, S. M., Sherali, H. D., and Al-Jazzaf, M. (2010). A mixed-integer mathematical modeling approach to exam timetabling. *Computational Management Science*, 7(1):19–46.

Aladağ, Ç. and Hocaoğlu, G. (2007). A tabu search algorithm to solve a course timetabling problem. *Hacettepe Journal of Mathematics and Statistics Volume*, 36(1):53–64.

Aldeeb, B., Al-Betar, M., Abdelmajeed, A., Younes, M., Alkenani, M., Alomoush, W., Alissa, K., and Alqahtani, M. (2019). A comprehensive review of uncapacitated university examination timetabling problem. *International Journal of Applied Engineering Research*, 14:4524–4547.

Aldeeb, B. A., Md Norwawi, N., Al-Betar, M. A., and Jali, M. Z. (2015). Intelligent examination timetabling system using hybrid intelligent water drops algorithm. In *5th International Conference on Computing and Informatics (ICOCI) 2015.*

Aldeeb, B. A., Norwawi, N. M., Al-Betar, M. A., and Jali, M. Z. B. (2014). Solving university examination timetabling problem using intelligent water drops algorithm. In *International Conference on Swarm, Evolutionary, and Memetic Computing*, pages 187–200. Springer.

Algethami, H., Pinheiro, R. L., and Landa-Silva, D. (2016). A genetic algorithm for a workforce scheduling and routing problem. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 927–934. IEEE.

Alsuwaylimi, A. and Fieldsend, J. (2019). A new initialisation method for examination timetabling heuristics. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1636–1643.

Altıntas, C., Asta, S., Ozcan, E., and Yigit, T. (2014). A self-generating memetic algorithm for examination timetabling. In 10^{th} *International Conference of the Practice and Theory of Automated Timetabling.*

Alvarez-Valdes, R., Crespo, E., and Tamarit, J. M. (2002). Design and implementation of a course scheduling system using tabu search. *European Journal of Operational Research*, 137(3):512–523.

Alves, S. S., Oliveira, S. A., and Neto, A. R. R. (2017). A recursive genetic algorithm-based approach for educational timetabling problems. In *Designing with Computational Intelligence*, pages 161–175. Springer.

Alzaqebah, M. and Abdullah, S. (2011). Hybrid artificial bee colony search algorithm based on disruptive selection for examination timetabling problems. In *International Conference on Combinatorial Optimization and Applications*, pages 31–45. Springer.

Alzaqebah, M. and Abdullah, S. (2014). An adaptive artificial bee colony and late-acceptance hill-climbing algorithm for examination timetabling. *Journal of Scheduling*, 17(3):249–262.

Alzaqebah, M. and Abdullah, S. (2015). Hybrid bee colony optimization for examination timetabling problems. *Computers & Operations Research*, 54(C):142–154.

Amaral, P. and Pais, T. C. (2016). Compromise ratio with weighting functions in a tabu search multi-criteria approach to examination timetabling. *Computers & Operations Research*, 72:160–174.

Ansari, A. and Bakar, A. A. (2014). A comparative study of three artificial intelligence techniques: Genetic algorithm, neural network, and fuzzy logic, on scheduling problem. In *2014 4th International Conference on Artificial Intelligence with Applications in Engineering and Technology*, pages 31–36. IEEE.

Appleby, J. S., Blake, D. V., and Newman, E. A. (1961). Techniques for Producing School Timetables on a Computer and their Application to other Scheduling Problems. *The Computer Journal*, 3(4):237–245.

ARANI, T. and Lotfi, V. (1989). A three phased approach to final exam scheduling. *IIE Transactions*, 21(1):86–96.

Asmuni, H. (2008). *Fuzzy methodologies for automated university timetabling solution construction and evaluation.* PhD thesis, University of Nottingham.

Asmuni, H., Burke, E. K., and Garibaldi, J. M. (2005a). Fuzzy multiple heuristic ordering for course timetabling. In *Proceedings of the 5th United Kingdom workshop on computational intelligence (UKCI 2005)*, pages 302–309.

Asmuni, H., Burke, E. K., Garibaldi, J. M., and McCollum, B. (2005b). Fuzzy multiple heuristic orderings for examination timetabling. In Burke, E. and Trick, M., editors, *Practice and Theory of Automated Timetabling V*, pages 334–353, Berlin, Heidelberg. Springer Berlin Heidelberg.

Asmuni, H., Burke, E. K., Garibaldi, J. M., and McCollum, B. (2006). A novel fuzzy approach to evaluate the quality of examination timetabling. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 327–346. Springer.

Asmuni, H., Burke, E. K., Garibaldi, J. M., McCollum, B., and Parkes, A. J. (2009). An investigation of fuzzy multiple heuristic orderings in the construction of university examination timetables. *Computers & Operations Research*, 36(4):981–1001.

Atsuta, M., Nonobe, K., and Ibaraki, T. (2008). ITC-2007 Track2: an approach using general CSP solver. In *Proceedings of the 7^{th} International conference on the Practice and Theory of Automated Timetabling (PATAT 2008)*, pages 19–22, Montreal, Canada.

Auf'm Hofe, H. M. (2001). Solving rostering tasks by generic methods for constraint optimization. *International Journal of Foundations of Computer Science*, 12(05):671–693.

Avanthay, C., Hertz, A., and Zufferey, N. (2003). A variable neighborhood search for graph coloring. *European Journal of Operational Research*, 151(2):379–388.

Aycan, E. and Ayav, T. (2009). Solving the course scheduling problem using simulated annealing. In *Advance Computing Conference, 2009. IACC 2009. IEEE International*, pages 462–466. IEEE.

Ayob, M., Abdullah, S., and Malik, A. (2007a). A practical examination timetabling problem at the universiti kebangsaan malaysia. *International Journal of Computer Science and Network Security*, 7(9):198–204.

Ayob, M., Malik, A. M. A., Abdullah, S., Hamdan, A. R., Kendall, G., and Qu, R. (2007b). Solving a practical examination timetabling problem: A case study. In Gervasi, O. and Gavrilova, M. L., editors, *Computational Science and Its Applications – ICCSA 2007*, pages 611–624, Berlin, Heidelberg. Springer Berlin Heidelberg.

Azimi, Z. N. (2005). Hybrid heuristics for examination timetabling problem. *Applied Mathematics and Computation*, 163(2):705–733.

Babaei, H., Karimpour, J., and Hadidi, A. (2015a). A survey of approaches for university course timetabling problem. *Computers & Industrial Engineering*, 86:43–59. Applications of Computational Intelligence and Fuzzy Logic to Manufacturing and Service Systems.

Babaei, H., Karimpour, J., and Hadidi, A. (2015b). A survey of approaches for university course timetabling problem. *Computers & Industrial Engineering*, 86:43–59.

Bai, R., Burke, E. K., Gendreau, M., and Kendall, G. (2007). A simulated annealing hyper-heuristic: Adaptive heuristic selection for different vehicle routing problems. In *Proc. of the 3rd Multidisciplinary Int. Conf. on Scheduling: Theory and Applications, Paris, France, August*, pages 28–31.

Balakrishnan, N. (1991). Examination scheduling: a computerized application. *Omega*, 19(1):37–41.

Bardadym, V. A. (1995). Computer-aided school and university timetabling: The new wave. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 22–45. Springer.

Battistutta, M., Schaerf, A., and Urli, T. (2017). Feature-based tuning of single-stage simulated annealing for examination timetabling. *Annals of Operations Research*, 252(2):239–254.

Beheshti, Z. and Shamsuddin, S. M. (2013). A review of population-based meta-heuristic algorithm. *International Journal of Advances in Soft Computing and Its Applications*, 5:1–35.

Behrooz, F., Mariun, N., Marhaban, M. H., Mohd Radzi, M. A., and Ramli, A. R. (2018). Review of control techniques for hvac systems—nonlinearity approaches based on fuzzy cognitive maps. *Energies*, 11(3).

Beligiannis, G. N., Moschopoulos, C. N., Kaperonis, G. P., and Likothanassis, S. D. (2008). Applying evolutionary computation to the school timetabling problem: The greek case. *Computers & Operations Research*, 35(4):1265–1280.

Bellio, R., Ceschia, S., Di Gaspero, L., and Schaerf, A. (2021). Two-stage multi-neighborhood simulated annealing for uncapacitated examination timetabling. *Computers & Operations Research*, 132:105300.

Bilgin, B., Özcan, E., and Korkmaz, E. E. (2006). An experimental study on hyper-heuristics and exam timetabling. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 394–412. Springer.

Birbil, Ş. İ. and Fang, S.-C. (2003). An electromagnetism-like mechanism for global optimization. *Journal of global optimization*, 25(3):263–282.

Blickle, T. and Thiele, L. (1995). A mathematical analysis of tournament selection. In *Proceedings of the* 6th *International Conference on Genetic Algorithms*, page 9–16, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Blum, C., Correia, S., Dorigo, M., Paechter, B., Rossi-Doria, O., and Snoek, M. (2002a). A ga evolving instructions for a timetable builder. In Burke, E. and De Causmaecker, P., editors, *Proceedings of the* 4th *International Conference on the Practice and Theory of Automated Timetabling (PATAT 2002)*, pages 120–123. KaHo Sint-Lieven. Imported from HMI.

Blum, C., Puchinger, J., Raidl, G. R., and Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151.

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *CM Computing Surveys (CSUR)*, 35(3):268–308.

Blum, C. and Roli, A. (2008). *Hybrid Metaheuristics: An Introduction*, pages 1–30. Springer Berlin Heidelberg, Berlin, Heidelberg.

Blum, C., Sampels, M., and Zlochin, M. (2002b). On a particularity in model-based search. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 35–42. Morgan Kaufmann Publishers Inc.

Boizumault, P., Delon, Y., and Peridy, L. (1996). Constraint logic programming for examination timetabling. *The Journal of Logic Programming*, 26(2):217–233.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm intelligence : from natural* *to artificial systems*. Santa Fe Institute studies in the sciences of complexity. Oxford University Press, New York.

Bonutti, A., De Cesco, F., Di Gaspero, L., and Schaerf, A. (2012). Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results. *Annals of Operations Research*, 194(1):59–70.

Bosch, R. and Trick, M. (2005). *Integer Programming*, pages 69–95. Springer US, Boston, MA.

Boufflet, J. P. and Nègre, S. (1996). Three methods used to solve an examination timetable problem. In Burke, E. and Ross, P., editors, *Practice and Theory of Automated Timetabling*, pages 325–344, Berlin, Heidelberg. Springer Berlin Heidelberg.

Brabazon, A. and O'Neill, M. (2006). *Biologically inspired algorithms for financial modelling*. Springer Science & Business Media.

Brailsford, S. C., Potts, C. N., and Smith, B. M. (1999). Constraint satisfaction problems: Algorithms and applications. *European Journal of Operational Research*, 119(3):557–581.

Brélaz, D. (1979). New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256.

Brélaz, D. (1979). New methods to color the vertices of a graph. *Communications of the ACM*.

Broder, S. (1964). Final examination scheduling. *Commun. ACM*, 7(8):494–498.

Btissam, D. and Abounacer, R. (2017). Multi-objective examination timetabling problem: Modeling and resolution using a based ε-constraint method. *IJCSNS*, 17(4):192.

Bullnheimer, B. (1997). An examination scheduling model to maximize students' study time. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 78–91. Springer.

Bulut, H., Ergüt, M., Asil, V., and Bokor, R. H. (2004). Numerical solution of a viscous incompressible flow problem through an orifice by adomian decomposition method. *Applied Mathematics and Computation*, 153(3):733–741.

Burke, E., Bykov, Y., Newall, J., and Petrović, S. (2003a). A time-predefined approach to course timetabling. *Yugoslav Journal of Operations Research*, 13(2):139–151.

Burke, E., Bykov, Y., Newall, J., and Petrovic, S. (2004a). A time-predefined local search approach to exam timetabling problems. *IIE Transactions*, 36(6):509–528.

Burke, E., Bykov, Y., and Petrovic, S. (2001). A multicriteria approach to examination timetabling. In Burke, E. and Erben, W., editors, *Practice and Theory of Automated Timetabling III*, pages 118–131, Berlin, Heidelberg. Springer Berlin Heidelberg.

Burke, E. and Carter, M. (1998). *Practice and Theory of Automated Timetabling II: Second International Conference, PATAT'97, Toronto, Canada, August 20-22, 1997, Selected Papers*, volume 2. Springer Science & Business Media.

Burke, E., De Causmaecker, P., Petrovic, S., and Berghe, G. V. (2003b). Variable neighborhood search for nurse rostering problems. In *Metaheuristics: computer decision-making*, pages 153–172. Springer.

Burke, E., De Werra, D., and Kingston, J. (2004b). Applications to timetabling. In Gross, J. and Yellen, J., editors, *Handbook of graph theory*, pages 445–474. Chapman Hall/CRC Press.

Burke, E., Eckersley, A., McCollum, B., Petrovic, S., and Qu, R. (2010a). Hybrid variable neighbourhood approaches to university exam timetabling. *European Journal of Operational Research*, 206(1):46–53.

Burke, E., Eckersley, A., McCollum, B., Petrovic, S., and Qu, R. (2010b). Hybrid variable neighbourhood approaches to university exam timetabling. *European Journal of Operational Research*, 206(1):46–53.

Burke, E., Elliman, D., Ford, P., and Weare, R. (1996). Examination timetabling in british universities: A survey. In Burke, E. and Ross, P., editors, *Practice and Theory of Automated Timetabling*, pages 76–90, Berlin, Heidelberg. Springer Berlin Heidelberg.

Burke, E., Elliman, D., and Weare, R. (1995a). Specialised recombinative operators for timetabling problems. In Fogarty, T. C., editor, *Evolutionary Computing*, pages 75–85, Berlin, Heidelberg. Springer Berlin Heidelberg.

Burke, E. and Erben, W. (2003). *Practice and Theory of Automated Timetabling III: Third International Conference, PATAT 2000 Konstanz, Germany, August 16-18, 2000 Selected Papers*, volume 2079. Springer.

Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., and Schulenburg, S. (2003c). Hyper-heuristics: An emerging direction in modern search technology. In *Handbook of meta-heuristics*, pages 457–474. Springer.

Burke, E. and Newall, J. (1997). Investigating the benefits of utilising problem specific heuristics within a memetic timetabling algorithm. *Workin Paper NOTTCS-TR-97-6, dept. of Computer Science, University of Nottingham, UK.*

Burke, E. and Trick, M. (2005). *Practice and Theory of Automated Timetabling V: 5th International Conference, PATAT 2004, Pittsburgh, PA, USA, August 18-20, 2004, Revised Selected Papers*, volume 3616. Springer.

Burke, E. K. and Bykov, Y. (2006). Solving exam timetabling problems with the flex-deluge algorithm. In *Proceedings of PATAT*, volume 2006, pages 370–372. Citeseer.

Burke, E. K. and Bykov, Y. (2008). A late acceptance strategy in hill-climbing for examination timetabling problems. In *Proceedings of the 7th International conference on the*

Practice and Theory of Automated Timetabling (PATAT 2008), pages 1–7, Montreal, Canada.

Burke, E. K. and Bykov, Y. (2016). An adaptive flex-deluge approach to university exam timetabling. *INFORMS Journal on Computing*, 28(4):781–794.

Burke, E. K. and Bykov, Y. (2017). The late acceptance hill-climbing heuristic. *European Journal of Operational Research*, 258(1):70–78.

Burke, E. K., De Causmaecker, P., Berghe, G. V., and Van Landeghem, H. (2004c). The state of the art of nurse rostering. *Journal of Scheduling*, 7(6):441–499.

Burke, E. K., Elliman, D., and Weare, R. (1994a). A genetic algorithm based university timetabling system. In *Proceedings of the 2^{nd} east-west international conference on computer technologies in education*, volume 1, pages 35–40.

Burke, E. K., Elliman, D., and Weare, R. F. (1995b). A hybrid genetic algorithm for highly constrained timetabling problems. In Eshelman, L. J., editor, *Proceedings of the 6^{th} International Conference on Genetic Algorithms, Pittsburgh, PA, USA, July 15-19, 1995*, pages 605–610. Morgan Kaufmann.

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

Burke, E. K. and Erben, W. (2001). *Practice and Theory of Automated Timetabling III*. Springer, Berlin, Heidelberg.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., and Qu, R. (2013). Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724.

Burke, E. K., Gustafson, S., and Kendall, G. (2004d). Diversity in genetic programming: an analysis of measures and correlation with fitness. *IEEE Transactions on Evolutionary Computation*, 8(1):47–62.

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., and Qu, R. (2009). A survey of hyper-heuristics. *Computer Science Technical Report No. NOTTCS-TR-SUB-0906241418-2747, School of Computer Science and Information Technology, University of Nottingham*.

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., and Woodward, J. R. (2010c). *A Classification of Hyper-heuristic Approaches*, pages 449–468. Springer US, Boston, MA.

Burke, E. K., Kendall, G., et al. (2005a). *Search methodologies*. Springer.

Burke, E. K., Kendall, G., Mısır, M., and Özcan, E. (2012a). Monte carlo hyper-heuristics for examination timetabling. *Annals of Operations Research*, 196(1):73–90.

Burke, E. K., Kendall, G., and Soubeiga, E. (2003d). A tabu-search hyperheuristic for timetabling and rostering. *Journal of heuristics*, 9(6):451–470.

Burke, E. K., McCollum, B., McMullan, P., and Parkes, A. J. (2008). Multi-objective aspects of the examination timetabling competition track. In *Proceedings of PATAT*, pages 3119–3126. Citeseer.

Burke, E. K., McCollum, B., Meisels, A., Petrovic, S., and Qu, R. (2007). A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, 176(1):177 – 192.

Burke, E. K. and Newall, J. P. (1999). A multistage evolutionary algorithm for the timetable problem. *IEEE transactions on evolutionary computation*, 3(1):63–74.

Burke, E. K. and Newall, J. P. (2003). Enhancing timetable solutions with local search methods. In Burke, E. and De Causmaecker, P., editors, *Practice and Theory of Automated Timetabling IV*, pages 195–206, Berlin, Heidelberg. Springer Berlin Heidelberg.

Burke, E. K. and Newall, J. P. (2004). Solving examination timetabling problems through adaption of heuristic orderings. *Annals of Operations Research*, 129(1-4):107–134.

Burke, E. K., Newall, J. P., and Weare, R. F. (1995c). A memetic algorithm for university exam timetabling. In *international conference on the practice and theory of automated timetabling*, pages 241–250. Springer.

Burke, E. K., Newall, J. P., and Weare, R. F. (1998a). Initialization strategies and diversity in evolutionary timetabling. *Evolutionary computation*, 6(1):81–103.

Burke, E. K., Newall, J. P., and Weare, R. F. (1998b). A simple heuristically guided search for the timetable problem. In *Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems, ICSC Academic Press, Nottingham.*

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

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

Burke, E. K., Pham, N., Qu, R., and Yellen, J. (2012b). Linear combinations of heuristics for examination timetabling. *Annals of Operations Research*, 194(1):89–109.

Burke, E. K., Qu, R., and Soghier, A. (2014). Adaptive selection of heuristics for improving exam timetables. *Annals of Operations Research*, 218(1):129–145.

Burke, E. K., Silva, J. D. L., and Soubeiga, E. (2005b). Multi-objective hyper-heuristic approaches for space allocation and timetabling. In *Metaheuristics: Progress as Real Problem Solvers*, pages 129–158. Springer.

Burke, E. K. and Silva, J. L. (2005). The design of memetic algorithms for scheduling and timetabling problems. In *Recent Advances in Memetic Algorithms*, pages 289–311. Springer.

Bykov, Y. and Petrovic, S. (2013). An initial study of a novel step counting hill climbing heuristic applied to timetabling problems. In *Proceedings of 6th Multidisciplinary International Scheduling Conference (MISTA 2013).*

Bykov, Y. and Petrovic, S. (2016). A step counting hill climbing algorithm applied to university examination timetabling. *Journal of Scheduling*, 19(4):479–492.

Caldeira, J. and Rosa, A. C. (1997). School timetabling using genetic search. *Proceedings of the* 2^{nd} *International Conference on the Practice and Theory of Automated Timetabling, Toronto*, pages 115–122.

Cantu-Paz, E. (1999). *Designing Efficient and Accurate Parallel Genetic Algorithms (Parallel Algorithms).* PhD thesis, University of Illinois at Urbana-Champaign, USA.

Caramia, M., Dell'Olmo, P., and Italiano, G. F. (2001). New algorithms for examination timetabling. In Näher, S. and Wagner, D., editors, *Algorithm Engineering*, pages 230–241, Berlin, Heidelberg. Springer Berlin Heidelberg.

Caramia, M., Dell'Olmo, P., and Italiano, G. F. (2008). Novel local-search-based approaches to university examination timetabling. *INFORMS Journal on Computing*, 20(1):86–99.

Carter, M. (1983). *A decomposition algorithm for practical timetabling problems.* Department of Industrial Engineering, University of Toronto.

Carter, M. (2001). Timetabling. *Encyclopedia of operations research and management science*, pages 833–836.

Carter, M. W. (1986). A survey of practical applications of examination timetabling algorithms. *Operations Research*, 34(2):193–202.

Carter, M. W. and Johnson, D. (2001). Extended clique initialisation in examination timetabling. *Journal of the operational research society*, 52(5):538–544.

Carter, M. W. and Laporte, G. (1996). Recent developments in practical examination timetabling. In Burke, E. and Ross, P., editors, *Practice and Theory of Automated Timetabling*, pages 1–21, Berlin, Heidelberg. Springer Berlin Heidelberg.

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

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

Casey, S. and Thompson, J. (2002). Grasping the examination scheduling problem. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 232–244. Springer.

Černỳ, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45(1):41–51.

Chen, X. and Bushnell, M. (1996). *Efficient Branch and Bound Search with Application to Computer-Aided Design*, volume 4. Kluwer Academic Publishers.

Cheong, C., Tan, K., and Veeravalli, B. (2009). A multi-objective evolutionary algorithm for examination timetabling. *Journal of Scheduling*, 12:121–146.

Cheong, C. Y., Tan, K. C., and Veeravalli, B. (2007). Solving the exam timetabling problem via a multi-objective evolutionary algorithm-a more general approach. In *Computational Intelligence in Scheduling, 2007. SCIS'07. IEEE Symposium on*, pages 165–172. IEEE.

Cheraitia, M. and Haddadi, S. (2016). Simulated annealing for the uncapacitated exam scheduling problem. *International Journal of Metaheuristics*, 5(2):156–170.

Cheraitia, M., Haddadi, S., and Salhi, A. (2017). Hybridizing plant propagation and local search for uncapacitated exam scheduling problems. *International Journal of of Services and Operations Management.*

Chu, S.-C., Chen, Y.-T., and Ho, J.-H. (2006). Timetable scheduling using particle swarm optimization. In *Innovative Computing, Information and Control, 2006. ICICIC'06. First International Conference on*, volume 3, pages 324–327. IEEE.

Clerc, M. (2008). Initialisations for particle swarm optimisation. *Online at http://clerc. maurice. free. fr/pso.*

Clerc, M. and Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE transactions on Evolutionary Computation*, 6(1):58–73.

Coello, C., Veldhuizen, D., and Lamont, G. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems.* Springer.

Cole, A. (1964). The preparation of examination timetables using a small-store computer. *The Computer Journal*, 7(2):117–121.

Colijn, A. W. and Layfield, C. (1995). Conflict reduction in examination schedules. In *1995). Proceedings of the 1st International Conference on the Practice and Theory of Automated Timetabling. 30th Aug-1st Sep*, pages 297–307.

Colorni, A., Dorigo, M., and Maniezzo, V. (1991). Genetic algorithms and highly constrained problems: The time-table case. In Schwefel, H.-P. and Männer, R., editors, *Parallel Problem Solving from Nature*, pages 55–59, Berlin, Heidelberg. Springer Berlin Heidelberg.

Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of* *the Third Annual ACM Symposium on Theory of Computing*, STOC '71, page 151–158, New York, NY, USA. Association for Computing Machinery.

Cooper, T. B. and Kingston, J. H. (1995). The complexity of timetable construction problems. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 281–295. Springer.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition.

Corne, D., Fang, H.-L., Mellish, C. S., and Corne, D. (1993). *Solving the modular exam scheduling problem with genetic algorithms*. Department of Artificial Intelligence, University of Edinburgh.

Corne, D. and Ross, P. (1995a). Peckish initialisation strategies for evolutionary timetabling. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 227–240. Springer.

Corne, D. and Ross, P. (1995b). Some combinatorial landscapes on which a genetic algorithm outperforms other stochastic iterative methods. In *AISB Workshop on Evolutionary Computing*, pages 1–13. Springer.

Corne, D., Ross, P., and Fang, H.-L. (1994a). Evolutionary timetabling: Practice, prospects and work in progress. In *In Proceedings of the UK Planning and Scheduling SIG Workshop, Strathclyde*.

Corne, D., Ross, P., and Fang, H.-L. (1994b). Fast practical evolutionary timetabling. In *AISB Workshop on Evolutionary Computing*, pages 250–263. Springer.

Corne, D. W., Jerram, N. R., Knowles, J. D., and Oates, M. J. (2001). Pesa-ii: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, GECCO'01, page 283–290, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Corne, D. W., Knowles, J. D., and Oates, M. J. (2000). The pareto envelope-based selection algorithm for multiobjective optimization. In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J. J., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature PPSN VI*, pages 839–848, Berlin, Heidelberg. Springer Berlin Heidelberg.

Corr, P. H., McCollum, B., McGreevy, M., and McMullan, P. (2006a). A new neural network based construction heuristic for the examination timetabling problem. In *Parallel Problem Solving from Nature-PPSN IX*, pages 392–401. Springer.

Corr, P. H., McCollum, B., McGreevy, M. A. J., and McMullan, P. (2006b). A new neural network based construction heuristic for the examination timetabling problem. In Runarsson, T. P., Beyer, H.-G., Burke, E., Merelo-Guervós, J. J., Whitley, L. D., and Yao, X., editors, *Parallel Problem Solving from Nature - PPSN IX*, pages 392–401, Berlin, Heidelberg. Springer Berlin Heidelberg.

Costa, D. (1994). A tabu search algorithm for computing an operational timetable. *European Journal of Operational Research*, 76(1):98–110.

Costa, D. and Hertz, A. (1997). Ants can colour graphs. *Journal of the operational research society*, 48(3):295–305.

Côté, P., Wong, T., and Sabourin, R. (2005). A hybrid multi-objective evolutionary algorithm for the uncapacitated exam proximity problem. In Burke, E. and Trick, M., editors, *Practice and Theory of Automated Timetabling V*, pages 294–312, Berlin, Heidelberg. Springer Berlin Heidelberg.

Cowling, P., Kendall, G., and Soubeiga, E. (2000). A hyperheuristic approach to scheduling a sales summit. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 176–190. Springer.

David, P. (1998). A constraint-based approach for examination timetabling using local repair techniques. In Burke, E. and Carter, M., editors, *Practice and Theory of Automated Timetabling II*, pages 169–186, Berlin, Heidelberg. Springer Berlin Heidelberg.

Davis, L. (1991). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York.

De Causmaecker, P., Demeester, P., and Berghe, G. V. (2009). A decomposed metaheuristic approach for a real-world university timetabling problem. *European Journal of Operational Research*, 195(1):307–318.

De Giovanni, L., Massi, G., and Pezzella, F. (2013). An adaptive genetic algorithm for large-size open stack problems. *International Journal of Production Research*, 51(3):682–697.

De Smet, G. (2008). ITC2007-examination track. In *Practice and Theory of Automated Timetabling (PATAT 2008), Montreal*, pages 19–22.

de Werra, D. (1985). An introduction to timetabling. *European Journal of Operational Research*, 19(2):151–162.

de Werra, D. (1997). The combinatorics of timetabling. *European Journal of Operational Research*, 96(3):504 – 513.

Dean, T. and Boddy, M. (1988). An analysis of time-dependent planning. In *Proceedings of the Seventh AAAI National Conference on Artificial Intelligence*, volume 88, page 49–54. AAAI Press.

Deb, K. and Deb, K. (2014). *Multi-objective Optimization*, pages 403–449. Springer US, Boston, MA.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Demeester, P., Bilgin, B., Causmaecker, P., and Berghe, G. (2012). A hyperheuristic approach to examination timetabling problems: Benchmarks and a new problem from practice. *Journal of Scheduling*, 15(1):83–103.

Deng, G.-F. and Lin, W.-T. (2011). Ant colony optimization-based algorithm for airline crew scheduling problem. *Expert Systems with Applications*, 38(5):5787–5793.

D'Eramo, C., Nuara, A., and Restelli, M. (2016). Estimating the maximum expected value through gaussian approximation. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1032–1040. JMLR.org.

Dhande, A., Ladhake, S., and Dhande, S. (2012). Genetic algoritham- an effective approach to solve real application problem. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2.

Di Gaspero, L. (2002). Recolour, shake and kick: A recipe for the examination timetabling problem. In *Proceedings of the fourth international conference on the practice and theory of automated timetabling, Gent, Belgium*, pages 404–407.

Di Gaspero, L., McCollum, B., and Schaerf, A. (2007). The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3). Technical report, Citeseer.

Di Gaspero, L. and Schaerf, A. (2000). Tabu search techniques for examination timetabling. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 104–117. Springer.

Dong, T., Qi, X., Zhang, Q., Li, W., and Xiong, L. (2019). Overview on vision-based 3d object recognition methods. In Zhao, Y., Barnes, N., Chen, B., Westermann, R., Kong, X., and Lin, C., editors, *Image and Graphics*, pages 243–254, Cham. Springer International Publishing.

Dorigo, M., Birattari, M., and Stutzle, T. (2006). Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39.

Dorigo, M. and Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2):243 – 278.

Dorigo, M., Maniezzo, V., Colorni, A., Dorigo, M., Dorigo, M., Maniezzo, V., Maniezzo, V., Colorni, A., and Colorni, A. (1991). Positive feedback as a search strategy. Technical report, Technical Report No. 91-016, Politecnico di Milano, Italy.

Dowsland, K., Pugh, N., and Thompson, J. (2002). Examination timetabling with ants (abstract). In *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT2002)*, pages 397 – 399.

Dowsland, K. A. and Thompson, J. M. (2005). Ant colony optimization for the examination scheduling problem. *Journal of the Operational Research Society*, 56(4):426–438.

Dowsland, K. A. and Thompson, J. M. (2012). Simulated annealing. In *Handbook of natural computing*, pages 1623–1655. Springer.

Dozier, G., Bowen, J., and Bahler, D. (1994). Solving small and large scale constraint satisfaction problems using a heuristic-based microgenetic algorithm. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 306–311. IEEE.

Dréo, J., Pétrowski, A., Siarry, P., and Taillard, E. (2006). *Metaheuristics for hard optimization: methods and case studies.* Springer Science & Business Media.

Drools, T. (2020). Drools planner user guide.

Duan, Q., Gupta, V. K., and Sorooshian, S. (1993). Shuffled complex evolution approach for effective and efficient global minimization. *Journal of optimization theory and applications*, 76(3):501–521.

Dubois-Lacoste, J., López-Ibáñez, M., and Stützle, T. (2015). Anytime pareto local search. *European Journal of Operational Research*, 243(2):369–385.

Dueck, G. (1993). New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104(1):86 – 92.

Duong, T.-A. and Lam, K.-H. (2004). Combining constraint programming and simulated annealing on university exam timetabling. In *RIVF*, pages 205–210. Citeseer.

Easton, K., Nemhauser, G., and Trick, M. (2004). Sports scheduling. In Leung, J. Y., editor, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, pages 1–17, Boca Raton, Florida. Chapman and Hall/CRC.

Eiben, A. E. and Schippers, C. A. (1998). On evolutionary exploration and exploitation. *Fundamenta Informaticae*, 35:35–50.

El-Sherbiny, M. M., Zeineldin, R. A., and El-Dhshan, A. M. (2015). Genetic algorithm for solving course timetable problems. *International Journal of Computer Applications*, 124(10).

Eley, M. (2007). Ant algorithms for the exam timetabling problem. In Burke, E. K. and Rudová, H., editors, *Practice and Theory of Automated Timetabling VI*, pages 364–382, Berlin, Heidelberg. Springer Berlin Heidelberg.

Epitropakis, M. G. and Burke, E. K. (2018). Hyper-heuristics. *Handbook of Heuristics*, pages 1–57.

Erben, W. (2000). A grouping genetic algorithm for graph colouring and exam timetabling. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 132–156. Springer.

Erben, W. and Keppler, J. (1995). A genetic algorithm solving a weekly course-timetabling problem. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 198–211. Springer.

Ergül, A. (1995). Ga-based examination scheduling experience at middle east technical university. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 212–226. Springer.

Fang, H.-L. (1994). *Genetic algorithms in timetabling and scheduling.* PhD thesis, University of Edinburgh.

Feng, X., Lee, Y., and Moon, I. (2017). An integer program and a hybrid genetic algorithm for the university timetabling problem. *Optimization Methods and Software*, 32(3):625–649.

Fernandes, C., Caldeira, J. a. P., Melicio, F., and Rosa, A. (1999a). Evolutionary algorithm for school timetabling. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 2*, GECCO'99, page 1777, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Fernandes, C., Caldeira, J. a. P., Melicio, F., and Rosa, A. (1999b). High school weekly timetabling by evolutionary algorithms. In *Proceedings of the 1999 ACM Symposium on Applied Computing*, SAC '99, page 344–350, New York, NY, USA. Association for Computing Machinery.

Festa, P. (2014). A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems. In *2014* 16th *International Conference on Transparent Optical Networks (ICTON)*, pages 1–20.

Fidanova, S., Roeva, O., and Luque, G. (2019). *Ant Colony Optimization Algorithm for Workforce Planning: Influence of the Algorithm Parameters*, pages 119–128. Springer International Publishing, Cham.

Fleszar, K. and Hindi, K. S. (2004). Solving the resource-constrained project scheduling problem by a variable neighbourhood search. *European Journal of Operational Research*, 155(2):402–413.

Fong, C. W., Asmuni, H., and McCollum, B. (2015). A hybrid swarm-based approach to university timetabling. *IEEE Transactions on Evolutionary Computation*, 19(6):870–884.

Fonseca, C. M. M. d. (1995). *Multiobjective genetic algorithms with application to control engineering problems.* PhD thesis, University of Sheffield.

Fouskakis, D. and Draper, D. (2002). Stochastic optimization: a review. *International Statistical Review*, 70(3):315–349.

Foxley, E. and Lockyer, K. (1968). The construction of examination timetables by computer. *The Computer Journal*, 11(3):264–268.

Frausto-Solís, J. and Alonso-Pecina, F. (2008). A hybrid simulated annealing-tabu search algorithm for post enrollment course timetabling. In *Proceeding of the* 7^{th} *International Conference on the Practice and Theory of Automated Timetabling PATAT*, volume 8.

Frausto-Solis, J., Mora-Vargas, J., Larre, M., and Luis Gomez-Ramos, J. (2006). A new genetic algorithm for the university timetablingproblem using forced diversity. *WSEAS Transactions on Systems*, 5(10):2398–2403.

Freuder, E. C. and Wallace, M. (2005). *Constraint Programming*, pages 239–272. Springer US, Boston, MA.

Gaetano, J. (2018). Holm-bonferroni sequential correction: An excel calculator (1.3).

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, United States.

Garey, M. R., Johnson, D. S., and Stockmeyer, L. (1974). Some simplified np-complete problems. In *Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 47–63. ACM.

Garza-Santisteban, F., Amaya, I., Cruz-Duarte, J., Ortiz-Bayliss, J. C., Özcan, E., and Terashima-Marín, H. (2020). Exploring problem state transformations to enhance hyper-heuristics for the job-shop scheduling problem. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8.

Gashgari, R., Alhashimi, L., Obaid, R., and Palaniswamy, T. (2018). A survey on exam scheduling techniques. In 1^{st} *International Conference on Computer Applications and Information Security, ICCAIS 2018*, pages 1–5.

Ghatei, S., Khajei, R., Maman, M., and Meybodi, M. (2012). A modified pso using great deluge algorithm for optimization. *Journal of Basic and Applied Scientific Research*, 2:1362–1367.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549.

Glover, F. (1989). Tabu search - part i. *ORSA Journal on computing*, 1(3):190–206.

Glover, F. W. and Laguna, M. (1993). Tabu search. In (Ed.), C. R., editor, *Modern Heuristic Techniques for Combinatorial Optimization*, pages 60–150, Oxford. Blackwell Publishers.

Glover, F. W. and Laguna, M. (1997). *Tabu Search Tabu Search*, volume 408. Kluwer Academic Publishers, Boston, 1 edition.

Gogos, C., Alefragis, P., and Housos, E. (2008). A multi-staged algorithmic process for the solution of the examination timetabling problem. *Practice and Theory of Automated Timetabling (PATAT 2008), Montreal*, pages 19–22.

Gogos, C., Alefragis, P., and Housos, E. (2012). An improved multi-staged algorithmic process for the solution of the examination timetabling problem. *Annals of Operations Research*, 194(1):203–221.

Gogos, C., Goulas, G., Alefragis, P., and Housos, E. (2009). Pursuit of better results for the examination timetabling problem using grid resources. In *Computational Intelligence in Scheduling, 2009. CI-Sched'09. IEEE Symposium on*, pages 48–53. IEEE.

Gogos, C., Goulas, G., Alefragis, P., Kolonias, V., and Housos, E. (2010). Distributed scatter search for the examination timetabling problem. In *PATAT 2010 Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling.*

Gonsalves, T. and Oishi, R. (2015). Artificial immune algorithm for exams timetable. *Journal of Information Sciences and Computing Technologies*, 4(2):287–296.

Gotlieb, C. C. (1962). The construction of class-teacher time-tables. In *IFIP Congress*, pages 73–77.

Grass, J. and Zilberstein, S. (1996). Anytime algorithm development tools. *SIGART Bull.*, 7(2):20–27.

Guliashki, V., Toshev, H., and Korsemov, C. (2009). Survey of evolutionary algorithms used in multiobjective optimization. *Problems of engineering cybernetics and robotics*, 60(1):42–54.

Guntsch, M. and Middendorf, M. (2002). A population based approach for aco. In Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., and Raidl, G. R., editors, *Applications of Evolutionary Computing*, pages 72–81, Berlin, Heidelberg. Springer Berlin Heidelberg.

Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European journal of operational research*, 130(3):449–467.

Hart, W. E., Krasnogor, N., and Smith, J. E. (2004). *Recent Advances in Memetic Algorithms*, volume 166. Springer, Berlin, Heidelberg.

Haykin, S. (1999). A comprehensive foundation: Neural networks.

Hertz, A. (1991). Tabu search for large scale timetabling problems. *European Journal of Operational Research*, 54(1):39–47.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* MIT press.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics.*

Hoos, H. H. and Stützle, T. (2004). *Stochastic local search: Foundations and applications.* Elsevier.

Innet, S. (2013). A noval approach of genetic algorithm for solving examination timetabling problems: A case study of thai universities. In *Communications and Information Technologies (ISCIT), 2013 13th International Symposium on*, pages 233–237. IEEE.

Ishak, S., Lee, L., and Ibragimov, G. (2016). Hybrid genetic algorithm for university examination timetabling problem. *MALAYSIAN JOURNAL OF MATHEMATICAL SCIENCES*, 10(2):145–178.

Islam, T., Shahriar, Z., Perves, M. A., and Hasan, M. (2016). University timetable generator using tabu search. *Journal of Computer and Communications*, 4(16):28.

Jaddi, N. S. and Abdullah, S. (2013). An interactive rough set attribute reduction using great deluge algorithm. In Zaman, H. B., Robinson, P., Olivier, P., Shih, T. K., and Velastin, S., editors, *Advances in Visual Informatics*, pages 285–299, Cham. Springer International Publishing.

Jansson, C. and Knüppel, O. (1995). A branch and bound algorithm for bound constrained optimization problems without derivatives. *Journal of Global Optimization*, 7(3):297–331.

Jat, S. N. and Yang, S. (2008). A memetic algorithm for the university course timetabling problem. In *Tools with Artificial Intelligence, 2008. ICTAI'08. 20th IEEE International Conference on*, volume 1, pages 427–433. IEEE.

Jat, S. N. and Yang, S. (2011). A hybrid genetic algorithm and tabu search approach for post enrolment course timetabling. *Journal of Scheduling*, 14(6):617–637.

Jha, S. K. (2014). Exam timetabling problem using genetic algorithm. *International Journal of Research in Engineering and Technology*, 3(5):649–654.

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

Joslin, D. E. and Clements, D. P. (1999). 'squeaky wheel' optimization. *Journal of Artificial Intelligence Research*, 10:353–373.

Juang, Y.-S., Lin, S.-S., and Kao, H.-P. (2007). An adaptive scheduling system with genetic algorithms for arranging employee training programs. *Expert Systems with Applications*, 33(3):642–651.

June, T. L., Obit, J. H., Leau, Y.-B., and Bolongkikit, J. (2019a). Implementation of constraint programming and simulated annealing for examination timetabling problem. In *Computational Science and Technology*, pages 175–184. Springer.

June, T. L., Obit, J. H., Leau, Y. B., and Bolongkikit, J. (2019b). Implementation of constraint programming and simulated annealing for examination timetabling problem. In *Lecture Notes in Electrical Engineering*, volume 481, pages 175–183. Springer, Singapore.

Kahar, M. and Kendall, G. (2010). The examination timetabling problem at Universiti

Malaysia Pahang: Comparison of a constructive heuristic with an existing software solution. *European Journal of Operational Research*, 207(2):557–565.

Kanit, R., Ozkan, O., and Gunduz, M. (2009). Effects of project size and resource constraints on project duration through priority rule-base heuristics. *Artificial Intelligence Review*, 32(1):115–123.

Karp, R. M. (1972). *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA.

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

Kendall, G. and Hussin, N. M. (2005b). A tabu search hyper-heuristic approach to the examination timetabling problem at the mara university of technology. In Burke, E. and Trick, M., editors, *Practice and Theory of Automated Timetabling V*, pages 270–293, Berlin, Heidelberg. Springer Berlin Heidelberg.

Kendall, G. and Li, J. (2008). Combining examinations to accelerate timetable construction. In *Proceedings of The 7th International Conference on the Practice and Theory of Automated Timetabling, Montreal*.

Khader, A. T., Belaton, B., et al. (2011). A practical rough sets analysis in real-world examination timetabling problem instances. In *Computer Networks and Intelligent Computing*, pages 21–30. Springer.

Khair, A. F., Makhtar, M., Mazlan, M., Mohamed, M. A., and Rahman, M. N. A. (2018). Solving examination timetabling problem in unisza using ant colony optimization. *International Journal of Engineering & Technology*, 7(2.15):132–135.

Khair, A. F., Makhtar, M., Mazlan, M., Mohamed, M. A., and Rahman, M. N. A. (2019). An ant colony algorithm for universiti sultan zainal abidin examination timetabling problem. *Indonesian Journal of Electrical Engineering and Computer Science*, 13(1):191–198.

Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., et al. (1983). Optimization by simulated annealing. *science*, 220(4598):671–680.

Knowles, J. D. and Corne, D. W. (2000). Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172.

Knuth, D. E. (1973). *The Art of Computer Programming, Volume III: Sorting and Searching*. Addison-Wesley.

Kochetov, Y. (2016). Formulation space search approach for the teacher/class timetabling problem. *Yugoslav Journal of Operations Research*, 18(1).

Konak, A., Coit, D. W., and Smith, A. E. (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety*, 91(9):992–1007. Special Issue - Genetic Algorithms and Reliability.

Kristiansen, S. and Stidsen, T. (2013). *A Comprehensive Study of Educational Timetabling - a Survey*. Number 8.2013 in DTU Management Engineering Report. DTU Management Engineering.

Kumar, R. et al. (2012). Blending roulette wheel selection & rank selection in genetic algorithms. *International Journal of Machine Learning and Computing*, 2(4):365–370.

Kusumawardani, D., Muklason, A., and Supoyo, V. A. (2019). Examination timetabling automation and optimization using greedy-simulated annealing hyper-heuristics algorithm. In *2019 12th International Conference on Information & Communication Technology and System (ICTS)*, pages 1–6. IEEE.

Laporte, G. and Desroches, S. (1984). Examination timetabling by computer. *Computers and Operations Research*, 11(4):351–360.

Le Huédé, F., Grabisch, M., Labreuche, C., and Savéant, P. (2006). Mcs—a new algorithm for multicriteria optimisation in constraint programming. *Annals of Operations Research*, 147(1):143–174.

Leake, D. B. (1996). *Case-Based Reasoning: Experiences, Lessons & Future Directions.* MIT Press, Cambridge, Massachusetts, United States, 1^{st} edition.

Lei, Y. and Shi, J. (2017). A nnia scheme for timetabling problems. *Journal of Optimization*, 2017.

Leite, N., Fernandes, C. M., Melício, F., and Rosa, A. C. (2018). A cellular memetic algorithm for the examination timetabling problem. *Computers & Operations Research*, 94:118 – 138.

Leite, N., Melício, F., and Rosa, A. C. (2016). A shuffled complex evolution algorithm for the examination timetabling problem. In Merelo, J. J., Rosa, A., Cadenas, J. M., Dourado, A., Madani, K., and Filipe, J., editors, *Computational Intelligence*, pages 151–168, Cham. Springer International Publishing.

Leite, N., Melício, F., and Rosa, A. C. (2019a). A fast simulated annealing algorithm for the examination timetabling problem. *Expert Systems with Applications*, 122:137–151.

Leite, N., Melício, F., and Rosa, A. C. (2019b). A fast simulated annealing algorithm for the examination timetabling problem. *Expert Systems with Applications*, 122:137 – 151.

Lewis, R. (2008). A survey of metaheuristic-based techniques for university timetabling problems. *OR Spectrum*, 30:167–190.

Lewis, R. (2012). A time-dependent metaheuristic algorithm for post enrolment-based course timetabling. *Annals of Operations Research*, 194(1):273–289.

Lewis, R. and Paechter, B. (2004). New crossover operators for timetabling with evolutionary algorithms. In *5th International Conference on Recent Advances in Soft Computing, Nottingham, UK*, volume 5, pages 189–195.

Lewis, R. and Paechter, B. (2005). Application of the grouping genetic algorithm to university course timetabling. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 144–153. Springer.

Lewis, R., Paechter, B., and Mccollum, B. (2007). Post enrolment based course timetabling: A description of the problem model used for track two of the second international timetabling competition. *Cardiff University, Cardiff Business School, Accounting and Finance Section, Cardiff Accounting and Finance Working Papers.*

Lewis, R. and Thompson, J. (2011). On the application of graph colouring techniques in round-robin sports scheduling. *Computers & Operations Research*, 38(1):190–204.

Li, J., Bai, R., Shen, Y., and Qu, R. (2015). Search with evolutionary ruin and stochastic rebuild: A theoretic framework and a case study on exam timetabling. *European Journal of Operational Research*, 242(3):798–806.

Li, J. and Kwan, R. S. (2003). A fuzzy genetic algorithm for driver scheduling. *European Journal of Operational Research*, 147(2):334–344. Fuzzy Sets in Scheduling and Planning.

Ludbrook, J. (1998). Multiple comparison procedures updated. *Clinical and Experimental Pharmacology and Physiology*, 25(12):1032–1037.

Luke, S. (2009). *Essentials of metaheuristics*, volume 113. Lulu Raleigh.

Lutuksin, T. and Pongcharoen, P. (2010). Best-worst ant colony system parameter investigation by using experimental design and analysis for course timetabling problem. In *Computer and Network Technology (ICCNT), 2010 Second International Conference on*, pages 467–471. IEEE.

Maaranen, H., Miettinen, K., and Mäkelä, M. (2004). Quasi-random initial population for genetic algorithms. *Computers & Mathematics with Applications*, 47(12):1885–1895.

Malim, M. R., Khader, A. T., and Mustafa, A. (2006). Artificial immune algorithms for university timetabling. In *Proceedings of the 6th international conference on practice and theory of automated timetabling*, pages 234–245. Brno, Czech Republic.

Mandal, A. K. and Kahar, M. (2015). Solving examination timetabling problem using partial exam assignment with hill climbing search. In *2015 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, pages 84–89. IEEE.

Mandal, A. K., Kahar, M. N. M., and Kendall, G. (2020). Addressing examination timetabling problem using a partial exams approach in constructive and improvement. *Computation*, 8(2):46.

Mann, H. B. and Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*.

Marie-Sainte, S. L. (2017). A new hybrid particle swarm optimization algorithm for real-world university examination timetabling problem. *2017 Computing Conference*, pages 157–163.

Martins, S. L. and Ribeiro, C. C. (2006). *Metaheuristics and Applications to Optimisation Problems in Telecommunications*, pages 103–128. Springer US, Boston, MA.

Massoodian, S. and Esteki, A. (2008). A hybrid genetic algorithm for curriculum based course timetabling. In *7th International Conference on the Practice and Theory of Automated Timetabling, PATAT 2008*.

Mavrovouniotis, M. (2013). Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors. *Applied Soft Computing*, 13:4023–4037.

Mavrovouniotis, M. and Yang, S. (2013). Adapting the pheromone evaporation rate in dynamic routing problems. In Esparcia-Alcázar, A. I., editor, *Applications of Evolutionary Computation*, pages 606–615, Berlin, Heidelberg. Springer Berlin Heidelberg.

McCollum, B. (2007). A perspective on bridging the gap between theory and practice in university timetabling. In Burke, E. K. and Rudová, H., editors, *Practice and Theory of Automated Timetabling VI*, pages 3–23, Berlin, Heidelberg. Springer Berlin Heidelberg.

McCollum, B., Ahmadi, S., and Barone, R. (2003). Perturbation based hyper-heuristic for examination timetabling problems. In *The 1st Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA), ; Conference date: 01-08-2003 Through 01-08-2003*, pages 155–171.

McCollum, B., McMullan, M., Burke, E., Parkes, A., and Qu, R. (2007). The second international timetabling competition: examination timetabling track (technical report:qub/ieee/tech/itc2007/exam/v4.0/17). *Electrical Engineering and Computer Science, Queens University, Belfast, UK, September.*

Mccollum, B., McMullan, P., Paechter, B., Lewis, R., Schaerf, A., Di Gaspero, L., Parkes, A., Qu, R., and Burke, E. (2010). Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing*, 22:120–130.

McCollum, B., McMullan, P., Parkes, A., Burke, E., and Qu, R. (2012a). A new model for automated examination timetabling. *Annals of Operations Research*, 194(1):291–315.

McCollum, B., McMullan, P., Parkes, A. J., Burke, E. K., and Abdullah, S. (2009). An extended great deluge approach to the examination timetabling problem. *Proceedings of the 4th Multidisciplinary International Scheduling: Theory and Applications 2009 (MISTA 2009)*, pages 424–434.

McCollum, B., McMullan, P., Parkes, A. J., Burke, E. K., and Qu, R. (2012b). A new model for automated examination timetabling. *Annals of Operations Research*, 194(1):291–315.

McMullan, P. (2007). An extended implementation of the great deluge algorithm for course timetabling. In Shi, Y., van Albada, G. D., Dongarra, J., and Sloot, P. M. A., editors, *Computational Science – ICCS 2007*, pages 538–545, Berlin, Heidelberg. Springer Berlin Heidelberg.

Mehta, N. (1982). Computer-based examination management system. *Journal of Educational Technology Systems*, 11(2):185–198.

Mehta, N. K. (1981). The application of a graph coloring method to an examination scheduling problem. *Interfaces*, 11(5):57–65.

Melício, F., Caldeira, J. P., and Rosa, A. (2004). Two neighbourhood approaches to the timetabling problem. *Proceedings of the practice and theory of automated timetabling (PATAT'04)*, pages 267–282.

Merlot, L. T. G., Boland, N., Hughes, B. D., and Stuckey, P. J. (2003). A hybrid algorithm for the examination timetabling problem. In Burke, E. and De Causmaecker, P., editors, *Practice and Theory of Automated Timetabling IV*, pages 207–231, Berlin, Heidelberg. Springer Berlin Heidelberg.

Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & operations research*, 24(11):1097–1100.

Mohammed, M. A., Ghani, M. K. A., Mostafa, O. I. O. S. A., Ahmad, M. S., Ibrahim, D. A., and Burhanuddin, M. (2017). A review of genetic algorithm application in examination timetabling problem. *Journal of Engineering and Applied Sciences*, 12(20):5166–5181.

Mohmad Kahar, M. and Kendall, G. (2015). A great deluge algorithm for a real-world examination timetabling problem. *Journal of the Operational Research Society*, 66(1):116–133.

Momani, S. and Odibat, Z. (2006). Analytical solution of a time-fractional navier–stokes equation by adomian decomposition method. *Applied Mathematics and Computation*, 177(2):488–494.

Montana, D., Brinn, M., Moore, S., and Bidwell, G. (1998). Genetic algorithms for complex, real-time scheduling. In *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, volume 3, pages 2213–2218. IEEE.

Mühlenthaler, M. (2015). Fairness in academic course timetabling. In *Fairness in Academic Course Timetabling*, pages 75–105. Springer.

Muklason, A. (2017). *Hyper-heuristics and fairness in examination timetabling problems.* PhD thesis, University of Nottingham.

Muklason, A., Parkes, A. J., Özcan, E., McCollum, B., and McMullan, P. (2017). Fairness in examination timetabling: Student preferences and extended formulations. *Applied Soft Computing*, 55:302 – 318.

Müller, T. (2005). *Constraint-based timetabling.* PhD thesis, Charles University in Prague,Faculty of Mathematics and Physics.

Müller, T. (2008). Itc2007 solver description: A hybrid approach. *Annals of Operations Research*, 172:429–446.

Müller, T., Barták, R., Rudová, H., et al. (2004). Conflict-based statistics. In *J. Gottlieb, D. Landa Silva, N. Musliu, and E. Soubeiga, editors, EU/ME Workshop on Design and Evaluation of Advanced Hybrid Meta-Heuristics. University of Nottingham.*

Müller, T., Rudová, H., and Müllerová, Z. (2018). University course timetabling and International Timetabling Competition 2019. In Burke, E. K., Di Gaspero, L., McCollum, B., Musliu, N., and Özcan, E., editors, *Proceedings of the 12th International Conference on the Practice and Theory of Automated Timetabling (PATAT-2018)*, pages 5–31.

Mumford, C. L. (2010). A multiobjective framework for heavily constrained examination timetabling problems. *Annals of Operations Research*, 180(1):3–31.

Mushi, A. R. (2006). Tabu search heuristic for university course timetabling problem. *African Journal of Science and Technology*, 7(1):34–40.

Mutingi, M. and Mbohwa, C. (2017). Multi-criterion examination timetabling: A fuzzy grouping genetic algorithm approach. In *Grouping Genetic Algorithms*, pages 161–182. Springer.

Myszkowski, P. B. and Norberciak, M. (2003). Evolutionary algorithms for timetable problems. In *Annales UMCS, Sectio Informatica.* Citeseer.

Nahas, N., Khatab, A., Ait-Kadi, D., and Nourelfath, M. (2008). Extended great deluge algorithm for the imperfect preventive maintenance optimization of multi-state systems. *Reliability Engineering & System Safety*, 93(11):1658–1672.

Nalepa, J. and Blocho, M. (2016). Adaptive memetic algorithm for minimizing distance in the vehicle routing problem with time windows. *Soft Computing*, 20(6):2309–2327.

Naseem, S. and Shengxiang, J. (2009). A guided search genetic algorithm for the university course timetabling problem. *Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA 2009).*

Neri, F. and Cotta, C. (2012). Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2:1–14.

Nguyen, P. T. M., Passow, B. N., and Yang, Y. (2016). Improving anytime behavior for traffic signal control optimization based on NSGA-II and local search. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 4611–4618. IEEE.

Norgren, E. and Jonasson, J. (2016). Investigating a genetic algorithm-simulated annealing hybrid applied to university course timetabling problem: A comparative study between simulated annealing initialized with genetic algorithm, genetic algorithm and simulated annealing.

Nurcahyadi, T. and Blum, C. (2021). Adding negative learning to ant colony optimization: A comprehensive study. *Mathematics*, 9(4).

Obaid, O. I., Ahmad, M., Mostafa, S. A., and Mohammed, M. A. (2012). Comparing performance of genetic algorithm with varying crossover in solving examination timetabling problem. *J. Emerg. Trends Comput. Inf. Sci*, 3(10):1427–1434.

Osaba, E., Carballedo, R., Diaz, F., Onieva, E., De La Iglesia, I., and Perallos, A. (2014). Crossover versus mutation: a comparative analysis of the evolutionary strategy of genetic algorithms applied to combinatorial optimization problems. *The Scientific World Journal*, 2014.

Osaba, E., Carballedo, R., Díaz, F., and Perallos, A. (2013). Analysis of the suitability of using blind crossover operators in genetic algorithms for solving routing problems. In *2013 IEEE 8th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, pages 17–22. IEEE.

Osman, I. H. and Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operations Research*, 63:513–623.

Othman, N. S. and Mashhod, F. (2012). Graph colouring and clustering heuristic approach for minimizing examination duration: A case study. *IBIMA Business Review*, 2012:1.

OuYang, Y. and Chen, Y. (2010). Research on automated timetabling algorithm for make-up examination and final clear examination. In *Computer Science and Education (ICCSE), 2010 5th International Conference on*, pages 570–573. IEEE.

Özcan, E., Burke, E. K., Di Gaspero, L., McCollum, B., and Schaerf, A. (2019). The practice and theory of automated timetabling (2016). *Annals of Operations Research*, 275(1):1–2.

Özcan, E., Parkes, A. J., and Alkan, A. (2012). The interleaved constructive memetic algorithm and its application to timetabling. *Computers & Operations Research*, 39(10):2310–2322.

Paechter, B., Cumming, A., Luchian, H., and Petriuc, M. (1994). Two solutions to the general timetable problem using evolutionary methods. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, pages 300–305. IEEE.

Paechter, B., Cumming, A., Norman, M. G., and Luchian, H. (1996). Extensions to a memetic timetabling system. In Burke, E. and Ross, P., editors, *Practice and Theory of Automated Timetabling*, pages 251–265, Berlin, Heidelberg. Springer Berlin Heidelberg.

Paechter, B., Gambardella, L. M., and Rossi-Doria, O. (2002). International timetabling competition, web page. *URL: http://www. idsia. ch/Files/ttcomp2002.*

Pais, T. C. and Amaral, P. (2012). Managing the tabu list length using a fuzzy inference system: an application to examination timetabling. *Annals of Operations Research*, 194(1):341–363.

Pais, T. C. and Burke, E. (2010). Choquet integral for combining heuristic values for exam timetabling problem. In *Proceedings of the 8ˆth International Conference on the Practice and Theory of Automated Timetabling, (PATAT 2010)*, pages 305–320.

Paquete, L. and Stützle, T. (2002). An experimental investigation of iterated local search for coloring graphs. In *Workshops on Applications of Evolutionary Computation*, pages 122–131. Springer.

Paquete, L. F. and Fonseca, C. M. (2001). A study of examination timetabling with multi-objective evolutionary algorithms. In *Proceedings of the 4th Metaheuristics International Conference (MIC 2001)*, pages 149–154.

Parkes, A. J. and Ozcan, E. (2010). Properties of yeditepe examination timetabling benchmark instances. In *Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling*, pages 531–534.

Peck, J. E. L. and Williams, M. R. (1966). Algorithm 286: Examination scheduling. *Communications of the ACM*, 9(6):433–434.

Pei, W., Huayu, G., Zheqi, Z., and Meibo, L. (2019). A novel hybrid firefly algorithm for global optimization. In *2019 IEEE* 4^{th} *International Conference on Computer and Communication Systems (ICCCS)*, pages 164–168.

Pelikan, M. (2010). Genetic algorithms. *Wiley Encyclopedia of Operations Research and Management Science.*

Pérez, J. A. M., Moreno-Vega, J. M., and Martın, I. R. (2003). Variable neighborhood tabu search and its application to the median cycle problem. *European Journal of Operational Research*, 151(2):365–378.

Pérez-Peló, S., Sánchez-Oro, J., and Duarte, A. (2019). Detecting weak points in networks using variable neighborhood search. In Sifaleras, A., Salhi, S., and Brimberg, J., editors, *Variable Neighborhood Search*, pages 141–151, Cham. Springer International Publishing.

Petrovic, S. and Burke, E. (2004). University timetabling. handbook of scheduling: Algorithms, models and performance analysis, chapter 45.

Petrovic, S. and Bykov, Y. (2002). A multiobjective optimisation technique for exam timetabling based on trajectories. In *PATAT*, pages 181–194. Springer.

Petrovic, S., Patel, V., and Young, Y. (2005). University timetabling with fuzzy constraints. practice and theory of automated timetabling v. *Lecture Notes in Computer Science*, 3616.

Petrovski, A., Brownlee, A., and McCall, J. (2005). Statistical optimisation and tuning of ga factors. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 758–764.

Pillay, N. (2008). A developmental approach to the examination timetabling problem. *Practice and Theory of Automated Timetabling (PATAT 2008)*, pages 19–22.

Pillay, N. (2012). Evolving hyper-heuristics for the uncapacitated examination timetabling problem. *Journal of the Operational Research Society*, 63(1):47–58.

Pillay, N. and Banzhaf, W. (2010). An informed genetic algorithm for the examination timetabling problem. *Applied Soft Computing*, 10(2):457–467.

Pillay, N. and Özcan, E. (2019). Automated generation of constructive ordering heuristics for educational timetabling. *Annals of Operations Research*, 275(1):181–208.

Pongcharoen, P., Promtet, W., Yenradee, P., and Hicks, C. (2008). Stochastic optimisation timetabling tool for university course scheduling. *International Journal of Production Economics*, 112(2):903–918.

Porto-Pazos, A. B., Veiguela, N., Mesejo, P., Navarrete, M., Alvarellos, A., Ibáñez, O., Pazos, A., and Araque, A. (2011). Artificial astrocytes improve neural network performance. *PloS one*, 6(4):e19109.

Post, G., Di Gaspero, L., Kingston, J. H., McCollum, B., and Schaerf, A. (2016). The third international timetabling competition. *Annals of Operations Research*, 239(1):69–75.

Prida Romero, B. (1982). Examination scheduling in a large engineering school: A computer-assisted participative procedure. *Interfaces*, 12(2):17–24.

Qu, R. and Burke, E. K. (2007). Adaptive decomposition and construction for examination timetabling problems. *Proceedings of the 3rd Multidisciplinary International Scheduling: Theory and Applications*, pages 418–425.

Qu, R. and Burke, E. K. (2009). Hybridizations within a graph-based hyper-heuristic framework for university timetabling problems. *Journal of the Operational Research Society.*

Qu, R., Burke, E. K., and McCollum, B. (2009a). Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems. *European Journal of Operational Research*, 198(2):392–404.

Qu, R., Burke, E. K., McCollum, B., Merlot, L. T. G., and Lee, S. Y. (2009b). A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling*, 12:55–89.

Raghavjee, R. and Pillay, N. (2013). A study of genetic algorithms to solve the school timetabling problem. In Castro, F., Gelbukh, A., and González, M., editors, *Advances in Soft Computing and Its Applications*, pages 64–80, Berlin, Heidelberg. Springer Berlin Heidelberg.

Rahim, S. K. N. A., Bargiela, A., and Qu, R. (2013). Hill climbing versus genetic algorithm optimization in solving the examination timetabling problem. In *ICORES 2013 - Proceedings of the 2nd International Conference on Operations Research and Enterprise Systems*, pages 43–52.

Rahman, S. A., Bargiela, A., Burke, E. K., Özcan, E., and McCollum, B. (2009). Construction of examination timetables based on ordering heuristics. In *2009* 24th *International Symposium on Computer and Information Sciences, ISCIS 2009*, pages 680–685.

Rahnamayan, S., Tizhoosh, H. R., and Salama, M. M. (2007). Quasi-oppositional differential evolution. In *2007 IEEE Congress on Evolutionary Computation*, pages 2229–2236.

Raidl, G. R., Puchinger, J., and Blum, C. (2010). Metaheuristic hybrids. In *Handbook of metaheuristics*, pages 469–496. Springer.

Rajah, C. and Pillay, N. (2019). A structure-based partial solution search for the examination timetabling problem. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 81–86. IEEE.

Redl, T. A. (2009). On using graph coloring to create university timetables with essential and preferential conditions. In *Advances in Marketing, Management and Finances, Proceedings of the 3rd International Conference on Management, Marketing and Finances (International Conference on Computational and Information Sciences*, pages 162–167.

Reeves, C. R. and Wright, C. C. (1995). Epistasis in genetic algorithms: An experimental design perspective. In *Proc. of the* 6th *International Conference on Genetic Algorithms, (pp 217–224*, pages 217–224. Morgan Kaufmann.

Reis, L. P. and Oliveira, E. (1999). Constraint logic programming using set variables for solving timetabling problems. In *12th international conference on applications of Prolog*.

Ross, P., Corne, D., and Fang, H.-L. (1994). Improving evolutionary timetabling with delta evaluation and directed mutation. In Davidor, Y., Schwefel, H.-P., and Männer, R., editors, *Parallel Problem Solving from Nature — PPSN III*, pages 556–565, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ross, P., Corne, D., and Terashima-Marín, H. (1995). The phase-transition niche for evolutionary algorithms in timetabling. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 309–324. Springer.

Ross, P., Hart, E., and Corne, D. (1997). Some observations about ga-based exam timetabling. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 115–129. Springer.

Ross, P., Hart, E., and Corne, D. (2003). Genetic algorithms and timetabling. In *Advances in evolutionary computing*, pages 755–771. Springer.

Ross, P., Marín-Blázquez, J. G., and Hart, E. (2004). Hyper-heuristics applied to class and exam timetabling problems. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 1691–1698. IEEE.

Rossi-Doria, O., Blum, C., Knowles, J., Sampels, M., Socha, K., and Paechter, B. (2002). A local search for the timetabling problem. In *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling, PATAT*, pages 124–127.

Rothlauf, F. (2002). *Representations for Genetic and Evolutionary Algorithms*, pages 9–30. Physica-Verlag HD, Heidelberg.

Rozaimee, A., Shafee, A. N., Hadi, N. A. A., and Mohamed, M. A. (2017). A framework for university's final exam timetable allocation using genetic algorithm. *World Applied Sciences Journal*, 35(7):1210–1215.

Rudolph, G. (1994). Convergence of non-elitist strategies. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, pages 63–66 vol.1.

Rudová, H. and Murray, K. (2002). University course timetabling with soft constraints. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 310–328. Springer.

Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach.* Prentice Hall Press, 3rd edition.

Sabar, N. R. and Ayob, M. (2009). Examination timetabling using scatter search hyper-heuristic. In *Data Mining and Optimization, 2009. DMO'09. 2nd Conference on*, pages 127–131. IEEE.

Sabar, N. R., Ayob, M., and Kendall, G. (2009a). Tabu exponential monte-carlo with counter heuristic for examination timetabling. In *Computational Intelligence in Scheduling, 2009. CI-Sched'09. IEEE Symposium on*, pages 90–94. IEEE.

Sabar, N. R., Ayob, M., Kendall, G., and Qu, R. (2009b). Roulette wheel graph colouring for solving examination timetabling problems. In *International Conference on Combinatorial Optimization and Applications*, pages 463–470. Springer.

Sabar, N. R., Ayob, M., Qu, R., and Kendall, G. (2012). A graph coloring constructive hyper-heuristic for examination timetabling problems. *Applied Intelligence*, 37(1):1–11.

Saharan, S. and Kumar, R. (2016). Graph coloring based optimized algorithm for resource utilization in examination scheduling. *Applied Mathematics & Information Sciences*, 10(3):1193–1201.

Sani, H. and Yabo, M. (2016). Solving timetabling problems using genetic algorithm technique. *International Journal of Computer Applications*, 134(15).

Schaerf, A. (1999). A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):87–127.

Schaerf, A. and Di Gaspero, L. (2001). Local search techniques for educational timetabling problems. In *Proceedings of the 6th International Symposium on Operations Research in Slovenia (SOR-01)*.

Selim, S. Z. and Alsultan, K. (1991). A simulated annealing algorithm for the clustering problem. *Pattern Recognition*, 24(10):1003–1008.

Selvi, V. and Umarani, R. (2010). Comparative analysis of ant colony and particle swarm optimization techniques. *International Journal of Computer Applications*, 5:1–6.

Shaker, K. and Abdullah, S. (2009). Incorporating great deluge approach with kempe chain neighbourhood structure for curriculum-based course timetabling problems. In *2009 2^{nd} Conference on Data Mining and Optimization, DMO 2009*, pages 149–153.

Sheibani, K. (2002). An evolutionary approach for the examination timetabling problems. In *2002). Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling. 21st-23rd August*, pages 387–396. Citeseer.

Shih, W.-K. and Liu, J. W. (1995). Algorithms for scheduling imprecise computations with timing constraints to minimize maximum error. *IEEE Transactions on Computers*, 44(3):466–471.

Sigl, B., Golub, M., and Mornar, V. (2003). Solving timetable scheduling problem using genetic algorithms. In *Information Technology Interfaces, 2003. ITI 2003. Proceedings of the 25th International Conference on*, pages 519–524. IEEE.

Silva, J. D. L., Burke, E. K., and Petrovic, S. (2004). An introduction to multiobjective metaheuristics for scheduling and timetabling. *Metaheuristics for multiobjective optimisation*, pages 91–129.

Sin, E. S. and Kham, N. S. M. (2012). Hyper heuristic based on great deluge and its variants for exam timetabling problem. *International Journal of Artificial Intelligence & Applications*, 3(1):149.

Skiundefinedcim, C. C. and Golden, B. L. (1983). Optimization by simulated annealing: A preliminary computational study for the tsp. In *Proceedings of the 15^{th} Conference on Winter Simulation - Volume 2*, WSC ’83, page 523–535. IEEE Press.

Socha, K., Sampels, M., and Manfrin, M. (2003). Ant algorithms for the university course timetabling problem with regard to the state-of-the-art. In *Applications of Evolutionary Computing*, pages 334–345, Berlin, Heidelberg. Springer Berlin Heidelberg.

Soghier, A. (2012). Novel hyper-heuristic approaches in exam timetabling. *The University of Nottingham.*

Soria-Alcaraz, J. A., Ochoa, G., Swan, J., Carpio, M., Puga, H., and Burke, E. K. (2014). Effective learning hyper-heuristics for the course timetabling problem. *European Journal of Operational Research*, 238(1):77–86.

Soria-Alcaraz, J. A., Özcan, E., Swan, J., Kendall, G., and Carpio, M. (2016). Iterated local search using an add and delete hyper-heuristic for university course timetabling. *Applied Soft Computing*, 40:581–593.

Soria-Alcaraz Jorge, A., Martín, C., Héctor, P., and Sotelo-Figueroa Marco, A. (2012). Application of a parallel computational approach in the design methodology for the course timetabling problem. In *PATAT 2012 - Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling.*

Stefansson, H., Sigmarsdottir, S., Jensson, P., and Shah, N. (2011). Discrete and continuous time representations and mathematical models for large production scheduling problems: A case study from the pharmaceutical industry. *European Journal of Operational Research*, 215(2):383–392.

Sultan, A. B. M., Mahmod, R., Sulaiman, M. N., Bakar, M., et al. (2008). Selecting quality initial random seed for metaheuristic approaches: a case of timetabling problem. *Int J Comput Internet Manag*, 16(1):8.

Sumathi, S., Hamsapriya, T., and Surekha, P. (2008). *Evolutionary intelligence: an introduction to theory and applications with Matlab.* Springer Science & Business Media.

Sutar, S. R. and Bichkar, R. S. (2012). University timetabling based on hard constraints using genetic algorithm. *International Journal of Computer Applications*, 42(15):1–7.

Talbi, E.-G. (2009). *Metaheuristics: from design to implementation.* John Wiley & Sons, Ltd.

Tam, V. and Ting, D. (2003). Combining the min-conflicts and look-forward heuristics to effectively solve a set of hard university timetabling problems. In *Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on*, pages 492–496. IEEE.

Tan, K. C., Khor, E. F., Lee, T. H., and Sathikannan, R. (2003). An evolutionary algorithm with advanced goal and priority specification for multi-objective optimization. *Journal of Artificial Intelligence Research*, 18:183–215.

Teich, J. (2001). Pareto-front exploration with uncertain objectives. In Zitzler, E., Thiele, L., Deb, K., Coello Coello, C. A., and Corne, D., editors, *Evolutionary Multi-Criterion Optimization*, pages 314–328, Berlin, Heidelberg. Springer Berlin Heidelberg.

Tein, L. H. and Ramli, R. (2010). Recent advancements of nurse scheduling models and a potential path. In *Proc. 6th IMT-GT Conference on Mathematics, Statistics and its Applications (ICMSA 2010)*, pages 395–409.

Teodorović, D. and Lučić, P. (1998). A fuzzy set theory approach to the aircrew rostering problem. *Fuzzy Sets and Systems*, 95(3):261–271.

Teoh, C. K., Wibowo, A., and Ngadiman, M. S. (2015). Review of state of the art for metaheuristic techniques in academic scheduling problems. *Artificial Intelligence Review*, 44(1):1–21.

Terashima-Marín, H. (1998). *Combinations of GAs and CSP strategies for solving examination timetabling problems*. PhD thesis, Instituto Tecnológico y de Estudios Superiores de Monterrey.

Terashima-Marín, H., Ross, P., and Valenzuela-Rendón, M. (1999a). Application of the hardness theory when solving the timetabling problem with genetic algorithms. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 1, pages 604–611. IEEE.

Terashima-Marín, H., Ross, P., and Valenzuela-Rendón, M. (1999b). Evolution of constraint satisfaction strategies in examination timetabling. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*, pages 635–642. Morgan Kaufmann Publishers Inc.

Thomas, J. J., Khader, A. T., Belaton, B., and Ken, C. C. (2012). Integrated problem solving steering framework on clash reconciliation strategies for university examination timetabling problem. In *International Conference on Neural Information Processing*, pages 297–304. Springer.

Thomas, J. J., Khader, A. T., Belaton, B., and Leow, A. (2011). Exploration of rough sets analysis in real-world examination timetabling problem instances. In *International Conference in Swarm Intelligence*, pages 173–182. Springer.

Thompson, J. and Dowsland, K. A. (1996a). General cooling schedules for a simulated annealing based timetabling system. In Burke, E. and Ross, P., editors, *Practice and Theory of Automated Timetabling*, pages 345–363, Berlin, Heidelberg. Springer Berlin Heidelberg.

Thompson, J. M. and Dowsland, K. A. (1996b). Variants of simulated annealing for the examination timetabling problem. *Annals of Operations research*, 63(1):105–128.

Thompson, J. M. and Dowsland, K. A. (1998). A robust simulated annealing based examination timetabling system. *Computers & Operations Research*, 25(7-8):637–648.

Ting, T. O., Yang, X.-S., Cheng, S., and Huang, K. (2015). *Hybrid Metaheuristic Algorithms: Past, Present, and Future*, pages 71–83. Springer International Publishing, Cham.

Trick, M. A. (2011). *Sports Scheduling*, pages 489–508. Springer New York, New York, NY.

Tuga, M., Berretta, R., and Mendes, A. (2007). A hybrid simulated annealing with kempe chain neighborhood for the university timetabling problem. In *2007 International Conference on Computer and Information Science*, pages 400–405, Los Alamitos, CA, USA. IEEE Computer Society.

Turabieh, H. and Abdullah, S. (2011a). A hybrid fish swarm optimisation algorithm for solving examination timetabling problems. In *International Conference on Learning and Intelligent Optimization*, pages 539–551. Springer.

Turabieh, H. and Abdullah, S. (2011b). An integrated hybrid approach to the examination timetabling problem. *Omega*, 39(6):598–607.

Ülker, Ö., Özcan, E., and Korkmaz, E. E. (2006). Linear linkage encoding in grouping problems: applications on graph coloring and timetabling. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 347–363. Springer.

Umbarkar, A. J. and Sheth, P. D. (2015). Crossover operators in genetic algorithms: a review. *ICTACT journal on soft computing*, 6(1).

Van Bulck, D., Goossens, D., Belien, J., and Davari, M. (2021). The fifth international timetabling competition (itc 2021): Sports timetabling. In *MathSport International 2021*, pages 117–122. University of Reading.

Črepinšek, M., Liu, S.-H., and Mernik, M. (2013). Exploration and exploitation in evolutionary algorithms: A survey. *ACM computing surveys (CSUR)*, 45(3).

Voß, S., Martello, S., Osman, I. H., and Roucairol, C. (2012). *Meta-heuristics: Advances and trends in local search paradigms for optimization.* Springer Science & Business Media.

Wazwaz, A.-M. (2005). Adomian decomposition method for a reliable treatment of the emden–fowler equation. *Applied Mathematics and Computation*, 161(2):543–560.

Weitz, R. and Lakshminarayanan, S. (1997). An empirical comparison of heuristic and graph theoretic methods for creating maximally diverse groups, vlsi design, and exam scheduling. *Omega*, 25(4):473–482.

Welsh, D. J. A. and Powell, M. B. (1967). An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10(1):85–86.

White, G. M. and Xie, B. S. (2000). Examination timetables and tabu search with longer-term memory. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 85–103. Springer.

Whitley, D. et al. (1995). Genetic algorithms and neural networks. *Genetic algorithms in engineering and computer science*, 3:191–201.

Wilcoxon, F. (1950). Some rapid approximate statistical procedures. *Annals of the New York Academy of Sciences.*

Wilke, P. and Ostler, J. (2008). Solving the school timetabling problem using tabu search, simulated annealing, genetic and branch & bound algorithms. In Burke Edmund, G. M., editor, *PATAT 2008 Proceedings of the* 7^{th} *International Conference on the Practice and Theory of Automated Timetabling*, pages 1–4.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

Wong, T., Bigras, P., and De Kelper, B. (2005). A multi-neighborhood and multi-operator strategy for the uncapacitated exam proximity problem. In *2005 IEEE International Conference on Systems, Man and Cybernetics*, volume 4, pages 3810–3816. IEEE.

Wong, T., Cote, P., and Gely, P. (2002). Final exam timetabling: a practical approach. In *IEEE CCECE2002. Canadian Conference on Electrical and Computer Engineering. Conference Proceedings (Cat. No.02CH37373)*, volume 2, pages 726–731.

Wong, T., Côté, P., and Sabourin, R. (2004). A hybrid moea for the capacitated exam proximity problem. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 1495–1501. IEEE.

Wong, Y.-Y., Lee, K.-H., Leung, K.-S., and Ho, C.-W. (2003). A novel approach in parameter adaptation and diversity maintenance for genetic algorithms. *Soft Computing*, 7(8):506–515.

Wood, D. (1968). A system for computing university examination timetables. *The Computer Journal*, 11(1):41–47.

Wren, A. (1996). Scheduling, timetabling and rostering — a special relationship? In Burke, E. and Ross, P., editors, *Practice and Theory of Automated Timetabling*, pages 46–75, Berlin, Heidelberg. Springer Berlin Heidelberg.

Yang, S. and Jat, S. N. (2011). Genetic algorithms with guided and local search strategies for university course timetabling. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(1):93–106.

Yang, X.-s. (2010). *Nature-Inspired Metaheuristic Algorithms*. Luniver Press.

Yang, Y. and Petrovic, S. (2005). A novel similarity measure for heuristic selection in examination timetabling. In Burke, E. and Trick, M., editors, *Practice and Theory of Automated Timetabling V*, pages 247–269, Berlin, Heidelberg. Springer Berlin Heidelberg.

Zadeh, L. (1975). The concept of a linguistic variable and its application to approximate reasoning—i,ii, and iii. *Information Sciences*, 8;8;9(3):199–249;301–357;43–80.

Zadeh, L. A. (1965). Fuzzy sets. *Information and control*, 8(3):338–353.

Zhang, J. (2011). Comparative study of several intelligent algorithms for knapsack problem. *Procedia Environmental Sciences*, 11:163–168. 2011 2^{nd} International Conference on Challenges in Environmental Science and Computer Engineering (CESCE 2011).

Zhang, W., Xing, Z., Wang, G., and Wittenburg, L. (2003). An analysis and application of distributed constraint satisfaction and optimization algorithms in sensor networks. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-agent Systems*, volume 2, pages 185–192.

Zhong, J.-H., Shen, M., Zhang, J., Chung, H. S.-H., Shi, Y.-H., and Li, Y. (2013). A differential evolution algorithm with dual populations for solving periodic railway timetable scheduling problem. *IEEE Transactions on Evolutionary Computation*, 17(4):512–527.

Zilberstein, S. (1996). Using anytime algorithms in intelligent systems. *AI magazine*, 17(3):73–73.

Zimmermann, H.-J. (1996). *Fuzzy Set Theory—and Its Applications (3rd Ed.).* Kluwer Academic Publishers, USA.

Zitzler, E., Laumanns, M., and Thiele, L. (2001). Spea2: Improving the strength pareto evolutionary algorithm. *TIK-Report*, 103.

Zitzler, E. and Thiele, L. (1998). An evolutionary algorithm for multiobjective optimization: the strength pareto approach. *TIK-Report*, 43.

Zolfaghari, S. and Liang, M. (1999). Jointly solving the group scheduling and machining speed selection problems: A hybrid tabu search and simulated annealing approach. *International Journal of Production Research*, 37(10):2377–2397.